ESCUELA DE DOCTORADO Y ESTUDIOS DE POSGRADO

Trabajo de Fin de Máster

Aprendizaje automático en el diseño de un detector de estrés a partir de señales biomédicas

Titulación: Máster en Ingeniería Industrial

Estudiante: Jorge Hernández Vidal

Tutora: Rosa María Aguilar Chinea

Septiembre 2019


Universidad
de La Laguna

Índice General


Universidad
de La Laguna

Universidad de La Laguna

Memoria

Aprendizaje automático en el diseño de un detector de estrés a partir de señales biomédicas

1. Memoria

1.1. Abstract

Automatic learning in the design of a stress detector from biomedical signals.

In this project the use of artificial neural networks to elaborate a stress detector is investigated. Specifically, we describe the development of a project to determine a classifier model of a system based on supervised automatic learning algorithms. We use the data of 15 subjects provided by the Siegen university and Robert Bosch Gmbh in the development, and we study how to work with this data and the results obtained. This study is designed on Python programming language, which has libraries that facilitate this type of investigation.

Índice

1.2. Introducción

La presente investigación trata sobre el diseño de una red neuronal artificial para la detección de estrés a partir de señales biomédicas obtenidas mediante *wearables*, en este caso son una pulsera y un medidor de pecho. Las Redes Neuronales Artificiales (RNA) se enmarcan dentro de un campo de la inteligencia artificial, en definitiva, son "un modelo matemático de una red neuronal biológica formado por la unión de unidades más simples llamadas perceptrón" [2].

Debido a la creciente aparición de enfermedades provocadas por el estrés en la sociedad europea actual y gracias a diversos estudios médicos se han podido identificar estos tipos:

- Riesgos estrés prenatal aumenta riesgo enfermedades psiquiátricas. [1]
- Enfermedades cardiovasculares:
    - Ataque cardiaco.
    - Insuficiencia cardiaca.
    - Accidente cerebrovascular [17].
- Ansiedad [6].

El estrés es una reacción de alerta y activación ante una situación a la que no se puede atender si no se incrementa la actividad cognitiva, fisiológica y conductual. Las reacciones de estrés suelen acarrear diferentes respuestas emocionales, especialmente de ansiedad, que es una emoción natural que comprende las reacciones que tienen los seres humanos ante la amenaza de un resultado negativo o incierto [6].

Debido a los efectos negativos del estrés cobra relevancia su detección, para poder tomar medidas adecuadas a tiempo y evitar sus efectos negativos. Por ello, utilizando los datos de 15 sujetos facilitados por Robert Bosch GmbH[1] y University Siegen[2], para la detección de estrés y diversión, se ha realizado una investigación sobre diferentes tipos de redes neuronales artificiales con la finalidad de diagnosticar estrés a partir de señales biomédicas.

[1] Compañía alemana iniciada en 1886 por Robert Bosch
[2] http://www.uni-siegen.de/start/

El proyecto describe en la sección 1.3 los objetivos del mismo, en la sección 1.4 el estado del arte actual en este campo de estudio. La metodología empleada para el desarrollo de las diferentes pruebas de la investigación para el diseño de las RNA esta descrito en el apartado 1.5. Los resultados se muestran en el 1.6 y las conclusiones obtenidas en el 1.7. Los códigos desarrollados se encuentran en los Apéndices de la sección 2.

1.3. Objetivos

- Llevar a cabo una revisión bibliográfica sobre los métodos actuales utilizados para la detección de estrés.
- Realizar diferentes diseños de las redes neuronales artificiales para la realización de un detector de estrés a partir de señales biomédicas.
- Llevar a cabo el entrenamiento y la prueba de los diferentes modelos, métodos y comparar los resultados para saber cuál sería la opción más adecuada.

1.4. Estado del arte

Actualmente se investigan diferentes formas de diagnosticar el estrés, por ejemplo, la detección facial que se basa principalmente en los movimientos asimétricos de cejas y boca [18]. También existen estudios sobre el estrés en ambientes laborales que se detecta utilizando sensores ECG (electrocardiógrafo) y GSR (Respuesta galvánica de la piel), este último da una medida de la conductividad eléctrica de la piel. Estas medidas permiten clasificar situaciones de estrés y no estrés [21]. Así mismo existen diferentes estudios de como poder provocar estrés en condiciones de laboratorio: como el TSST (Prueba de estrés social) y Cold pressor test (prueba de presión fría), entre otros [5]. A la hora de detectar estrés algunos de las mediciones más utilizadas para ello serían:

- Actividad cardiaca.
- Actividad electrodérmica.
- Actividad cerebral.
- Información del habla.

Existe una investigación llevada a cabo en la Universidad de Siegen[20] que estudia distintos métodos para la detección de estrés en diferentes sujetos obteniendo dos conjuntos de datos para cada sujeto, uno a través de una banda pectoral la cual tiene los siguientes sensores:

- ACC: acelerómetro.
- RESP: Respiración.
- ECG: Electrocardiógrafo.
- EDA: Actividad electrodérmica.

- EMG: Electromiografía, es un procedimiento de diagnóstico que se utiliza para evaluar la salud de los músculos y las células nerviosas que los controlan [7].
- TEMP: Temperatura.

Y el otro conjunto de datos a través de una pulsera que lleva estos otros sensores:

- ACC: acelerómetro.
- BVP: Pulso sanguíneo.
- EDA: Actividad electrodérmica.
- TEMP: Temperatura.

Esta investigación utiliza diversos métodos para la detección del estrés y otros estados afectivos como la diversión y el estado base. Además, cuentan con estados de meditación que los usan para volver de cualquiera de los estados de estrés o diversión al estado base. Este estudio facilita los datos utilizados en este TFM [20].

Los métodos empleados por los investigadores de la Universidad de Siegen fueron:

- Árbol de decisión.
- Random Forest.
- AdaBoost.
- LDA (Análisis de discriminación lineal).
- kNN (Vecino cercano).

Obteniendo resultados para la detección de estrés y no estrés de hasta un 92%-93% para el detector de pecho y del 87%-88% para el de pulsera. Sin embargo, a la hora de detectar diversión su precisión baja, siendo de un 75%-76% para la pulsera y el detector de pecho.

1.5. Metodología

EL diseño de la RNA del detector de estrés de esta investigación se ha realizado utilizando Python[3] 3, debido a las grandes facilidades que ofrece a la hora de trabajar con redes neuronales a través del entorno del notebook de Jupiter[4] gracias a la flexibilidad que da a la hora del uso de las librerías la ejecución del código y la presentación del documento de salida.

Anaconda Navigator es una interfaz de usuario gráfica de escritorio incluida en Anaconda que le permite iniciar aplicaciones y administrar fácilmente paquetes, entornos y canales de conda sin la necesidad de usar comandos de línea de comandos [15].

LAS LIBRERÍAS UTILIZADAS EN ESTA INVESTIGACIÓN	
KERAS	Se trata una API[5] de redes neuronales de alto nivel, escrita en Python y capaz de ejecutarse sobre TensorFlow [8].
TENSORFLOW	Es una plataforma de código abierto de extremo a extremo para el aprendizaje automático [14].
SCIKIT-LEARN	Librería de herramientas simples y eficientes para minería y análisis de datos [13].
MATPLOTLIB	Es una librería de trazado 2D de Python para producir graficas en una variedad de formatos y entornos interactivos en todas las plataformas [9].
PICKLE	Se trata de un módulo que implementa protocolos binarios para serializar y deserializar una estructura de objeto Python [12].
PANDAS	Es una librería que proporciona estructuras de datos y herramientas de análisis de datos de alto rendimiento para el lenguaje de programación Python [11].
NUMPY	Es un paquete para la computación científica con Python, permite trabajar con matrices de forma sencilla, algebra lineal, entre otras funciones [10].

Tabla 1 Finalidad de las librerías utilizadas

[3] https://www.python.org/
[4] https://jupyter.org/
[5] Application Programming Interface

Las librerías descritas anteriormente (ver tabla 1) ofrecen las herramientas necesarias, según la función que permita desempeñar, para diseñar las redes neuronales artificiales. Estas son modelos computacionales que se basan en el funcionamiento del cerebro humano. No se conocen perfectamente los mecanismos del cerebro, aunque se sabe que está formado por un gran número de neuronas 10^{12}, funcionando de forma paralela y que producen diferentes niveles de activación en neuronas adyacentes. De igual modo, las RNA están formadas por una cierta cantidad de unidades básicas de procesamiento llamadas perceptrones o neuronas, conectadas según ciertas reglas, donde cada conexión tiene asociado un peso [3].

Cuando se diseña una red neuronal se deben establecer las siguientes características:

- La cantidad de neuronas en cada capa.
- La cantidad de capas.
- La función de activación optimizador y pérdidas o función de coste.
- El tipo de conexión entre neuronas, en este caso se recomienda que sea hacia delante para el reconocimiento de patrones.
- El algoritmo de entrenamiento para conseguir que la red aprenda, este puede ser supervisado o no supervisado. En la presente investigación es supervisado puesto que conocemos la salida a la cual queremos que se adapte la red [3].

Las neuronas artificiales o perceptrones están formadas por:

- Conjunto de entradas X (ver figura 1): Representan las entradas de la neurona.
- Pesos sinápticos W (ver figura 1): Cada entrada tiene un peso que se va ajustando de forma automática a medida que la red neuronal va aprendiendo.
- Función de agregación, Σ (ver figura 1): Suma todas las entradas ponderadas por sus pesos.

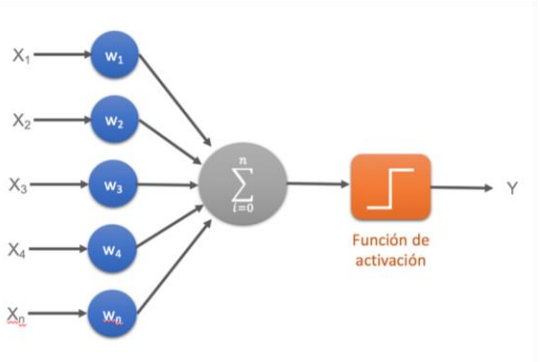


Figura 1 Imagen perceptrón [19]

Además, se debe definir la función de activación, la función de coste y el optimizador. La función de activación mantiene el conjunto de valores de salida en un rango determinado, la función de coste o *loss* trata de determinar el error entre el valor obtenido y el valor real, para poder entrenar la RNA. Por último, el optimizador facilita el gradiente en el que se ha de avanzar durante el entrenamiento de la red, es decir, da la dirección hacia donde debe ir la función, pues fija donde tiene la ratio de aumento más pronunciado [19].

Finalmente se define lo que es un ensemble, puesto que se hacen pruebas con el mismo en esta investigación. A la hora de trabajar con un ensemble se realiza el entrenamiento de dos o más RNA con diferentes conjuntos de datos para las mismas salidas, es decir, el conjunto principal de datos se divide en diferentes subconjuntos con distintas variables en cada uno. Posteriormente, con los resultados de ambas redes, se obtiene un valor final utilizando las salidas que han sido obtenidas de las diferentes redes, mediante algún algoritmo matemático u otra red neuronal [4].

1.5.1. Análisis de datos

En la presente investigación se utilizan los datos de 15 sujetos facilitados por Robert Bosch GmbH y University Siegen, como se ha comentado anteriormente. Los datos se encuentran en ficheros pkl, se trata de un objeto Python almacenado directamente en un archivo o cadena, ordenados con sus etiquetas según sea el detector de pecho o muñeca y por el tipo de sensor y separados en diferentes archivos según el sujeto.

En las figuras 2 a 12 se representan gráficamente para el sujeto 2 los datos que se obtienen de cada uno de los sensores, para el detector pectoral, el de pulsera y sus estados:

- Detector Pectoral:
  - Acelerómetro o ACC:

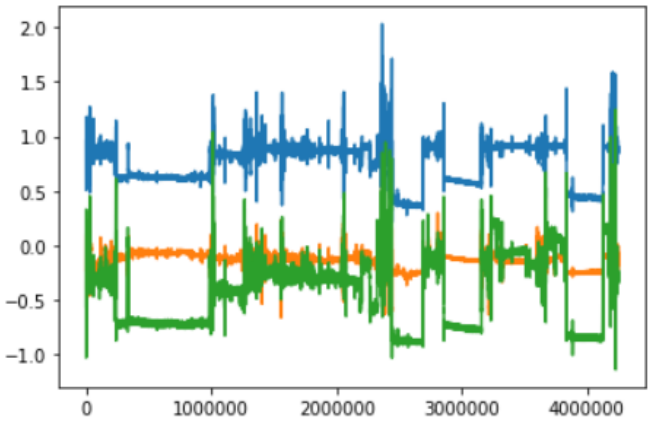


Figura 2 Señal acelerómetro Detector Pectoral

  - Electrocardiograma o ECG:

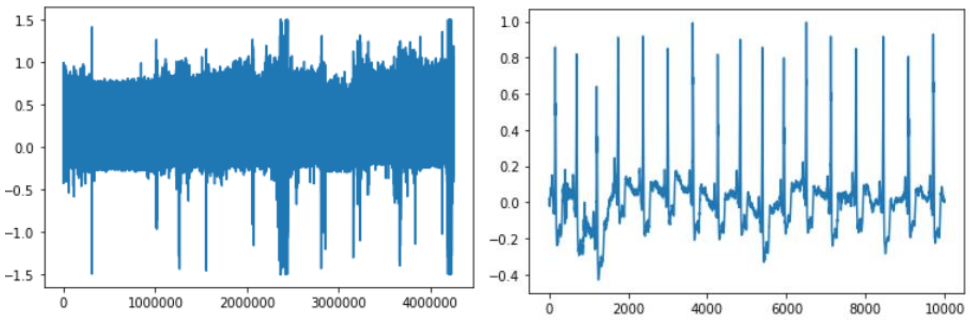


Figura 3 Señal electrocardiograma Detector Pectoral

  - Electromiografía o EMG:

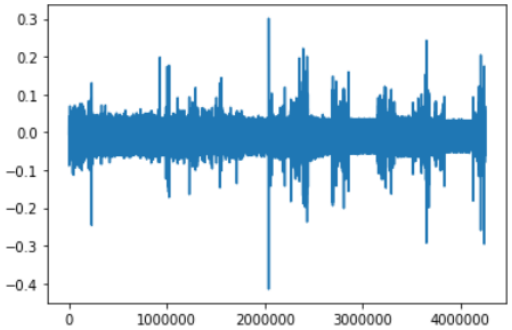


Figura 4 Señal Electromiografía Detector Pectoral

- Actividad Electrodérmica o EDA:

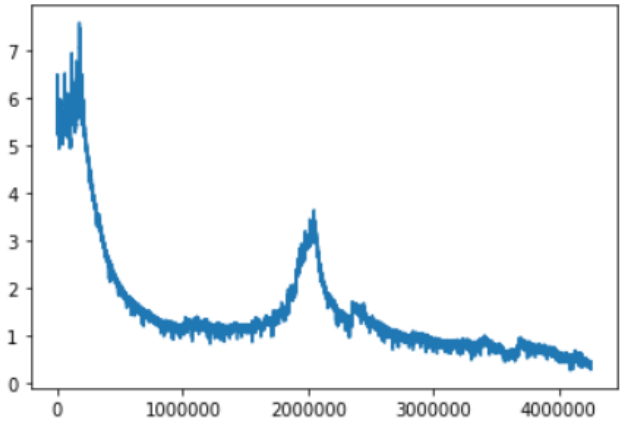


Figura 5 Señal Electrodérmica Detector Pectoral

- Temperatura:

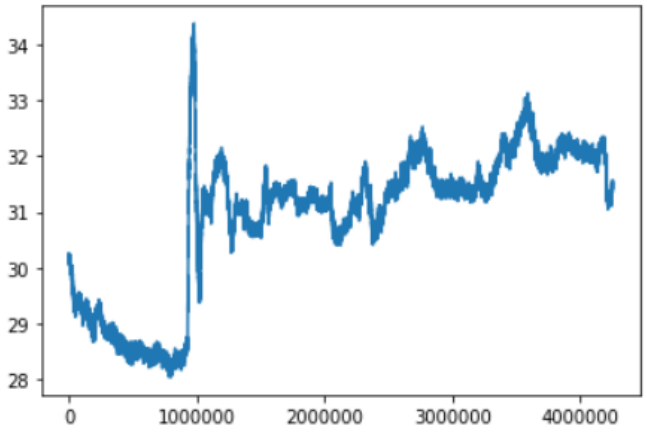


Figura 6 Señal de Temperatura Detector Pectoral

- Respiración:

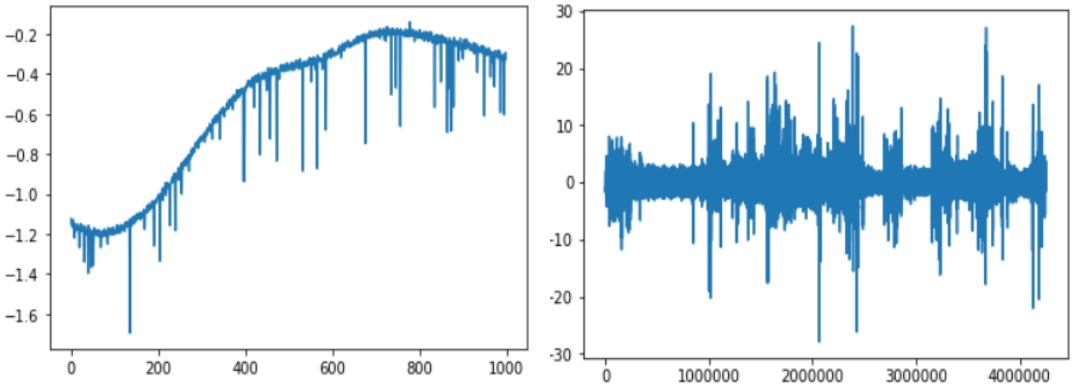


Figura 7 Señal Respiración Detector Pectoral

- Detector de Pulsera:

  - Acelerómetro o ACC:

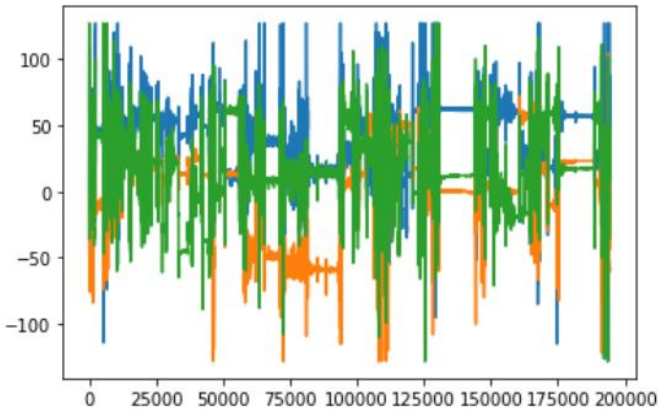


Figura 8 Señal acelerómetro Detector de pulsera

  - Pulso sanguíneo o BVP:

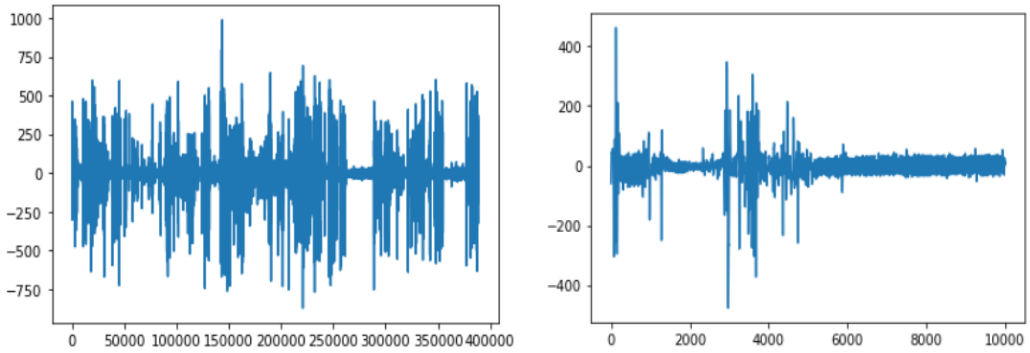


Figura 9 Señal de pulso sanguíneo Detector de pulsera

  - Actividad Electrodérmica o EDA:

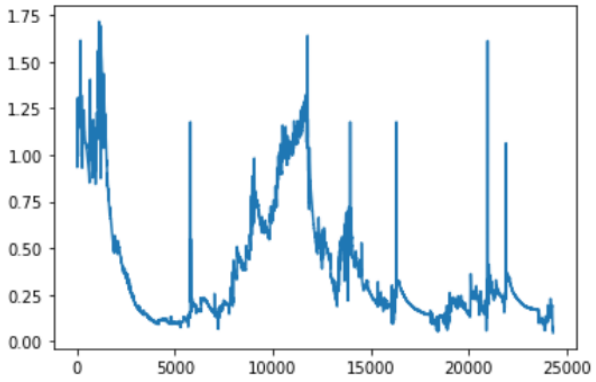


Figura 10 Señal de Actividad Electrodérmica Detector de pulsera

- Temperatura:

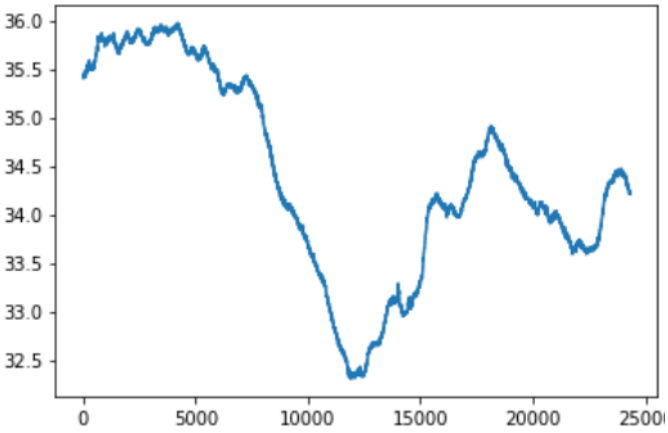


Figura 11 Señal Temperatura Detector de Pulsera

- Estados (ver tabla 2):

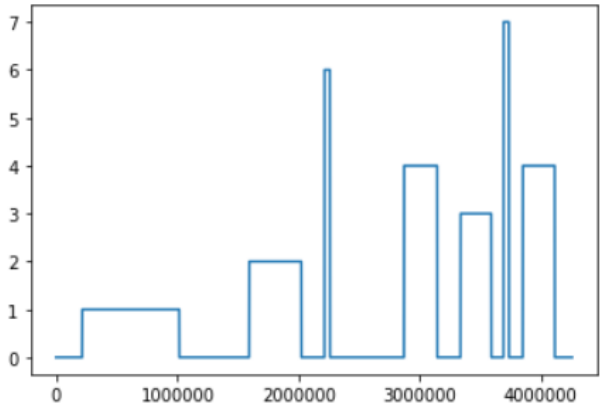


Figura 12 Estados del Sujeto

1.5.2. Flujograma del proceso

A continuación, vemos un flujograma de los pasos que sigue el código de la RNA:

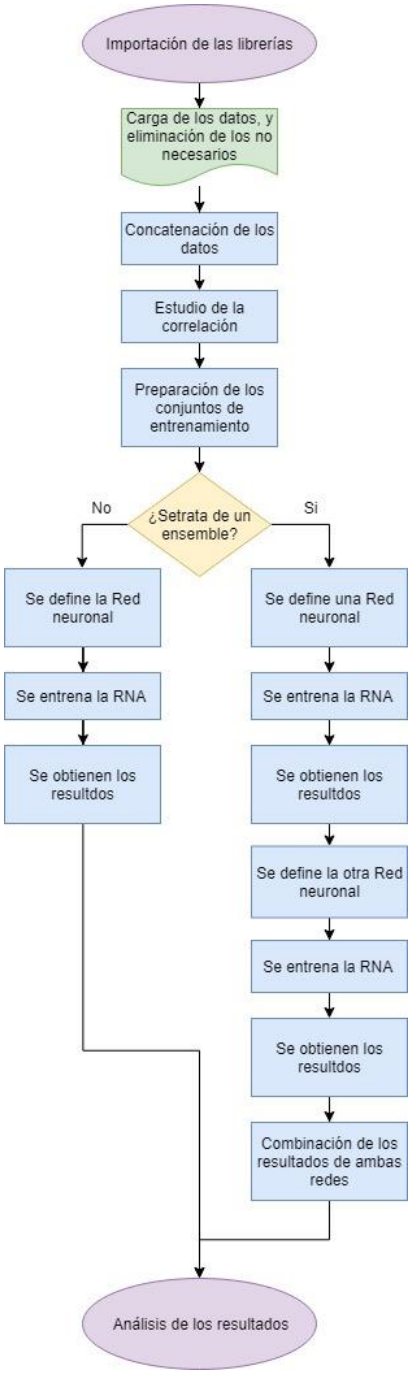


Figura 13 Flujograma del código de la RNA

1.5.3. RNA Detector Pectoral

La carga de los datos se realiza utilizando la librería pickle, posteriormente se guardan en variables para que se puedan unir en una matriz.

```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A2l=data['label']
```

Figura 14 Carga de los datos para el detector pectoral

Para cumplimentar la información se añade otra columna con la edad del sujeto, toda la información se concatena en una matriz en la que se descartan los datos que no son de interés para el entrenamiento de la RNA, puesto que no aportan información sobre estados de la investigación.

```
A2a=27*(np.ones((len(a),1)))
A2=np.concatenate((a,b,c,d,e,f,A2a),axis=1)
B1 = np.asmatrix(A2l)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

Figura 15 Creación de la matriz y eliminación de datos no utilizables

La descripción de los estados es:

ESTADOS	DESCRIPCIÓN
0	No definido / transitorio.
1	Línea de base.
2	Estrés.
3	Diversión.
4	Meditación.
5/6/7	Debe ignorarse en este conjunto de datos.

Tabla 2 Estados definidos en la investigación de referencia [16]

Se descartan los datos para salidas con valores superiores a 4 e inferiores a 1. La elevada cantidad de datos empleados conlleva un gran consumo de memoria RAM. Por esa razón, sirve para aliviar los requerimientos sobre este recurso y evitar posibles errores en la ejecución del código.

Una vez generada la matriz de datos de cada sujeto se unen:

```
MAT=np.concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT15),axis=0)
```

Figura 16 Concatenación de las matrices de los sujetos

Se estudia la correlación de los datos de los diferentes sensores para determinar si existe algún dato que no sea de utilidad porque aporta la misma información o muy similar que otro sensor, es decir, que su correlación sea cercana a uno (ver tabla 3).

```
data=pandas.DataFrame(MAT)
corr = data.corr()
corr.style.background_gradient(cmap='coolwarm')
```

Figura 17 Generación matriz de correlación

	0	1	2	3	4	5	6	7	8	9
0	1	-0.116371	0.111886	-0.0997043	4.41357e-05	-0.00957082	0.0941815	0.265322	-0.00163408	0.00518957
1	-0.116371	1	0.141815	0.736804	-0.000777993	0.0285744	0.0597632	0.176475	-0.00258757	-0.220018
2	0.111886	0.141815	1	0.15978	0.00130894	0.00706299	0.107432	0.287241	-0.00249185	-0.490428
3	-0.0997043	0.736804	0.15978	1	-0.000655872	0.0183223	0.000482257	0.113601	0.00185039	-0.0848492
4	4.41357e-05	-0.000777993	0.00130894	-0.000655872	1	-0.14177	-0.000186393	0.00384364	0.0064098	7.70639e-05
5	-0.00957082	0.0285744	0.00706299	0.0183223	-0.14177	1	-0.00713845	0.0676265	0.00390984	0.00842724
6	0.0941815	0.0597632	0.107432	0.000482257	-0.000186393	-0.00713845	1	0.149413	-0.000231788	-0.115606
7	0.265322	0.176475	0.287241	0.113601	0.00384364	0.0676265	0.149413	1	0.00795273	-0.066754
8	-0.00163408	-0.00258757	-0.00249185	0.00185039	0.0064098	0.00390984	-0.000231788	0.00795273	1	0.000472801
9	0.00518957	-0.220018	-0.490428	-0.0848492	7.70639e-05	0.00842724	-0.115606	-0.066754	0.000472801	1

Tabla 3 Correlación de las entradas frente a las salidas

Como se observa en la tabla 3, existe una correlación elevada entre las columnas 1 y 3 que se corresponden con señales del mismo sensor el acelerómetro, por lo tanto, no se procede a descartar la información de ninguno de los sensores. Puesto que, a la hora de descartar señales, sería de mayor relevancia aquella que permita eliminar los datos de un sensor en su totalidad, significando que este no aporta información puesto que sería la misma que da otro sensor. Como igualmente se dispondría de las señales recopiladas por el acelerómetro no se procede a su eliminación. Se procede a dividir los datos en los conjuntos de validación y entrenamiento:

```
X = MAT[:, 1:8]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y, test_size=validation_size, random_state=seed)
```

Figura 18 Separación de los conjuntos de entrenamiento y test con una relación 80-20 y de manera aleatoria [3]

Los valores de las columnas en la matriz X varían cuando se usan los datos del medidor de pecho o de la pulsera, debido a la cantidad de sensores de cada uno. Una vez se tienen ambos conjuntos de datos se define la RNA, la figura 18 es un ejemplo de una de las redes utilizadas en la investigación.

```
model = Sequential([
    Dense(128, input_shape=(9,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
    Dense(60),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

Figura 19 Diseño de una de las redes neuronales utilizadas

Los optimizadores con los que se han realizado prueba son:

- Adadelta.
- Nadam.
- Rmsprop.

Para el caso de las funciones de coste:

- mean_squared_logarithmic_error.
- mean_absolute_error.
- Hinge.

Los activadores utilizados son:

- Selu.
- Elu.
- Capas selu y elu.
- Linear.

Una vez definida la red se lleva a cabo su entrenamiento, según sea el detector de pecho o de pulsera variará la extensión del entrenamiento. Debido a la cantidad de datos

disponibles de cada detector, se varían los entrenamientos tanto en el Batch[6] como los Epochs[7], así como la cantidad de veces que llevamos a cabo el entrenamiento:

```
NUM_EPOCHS =5
BATCH_SIZE = 200

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS, validation_split=0.1)
```

Figura 20 Entrenamiento de la RNA

Cuando se completa el entrenamiento se calculan los resultados para el conjunto de validación (ver figura 20), y se redondean los valores obtenidos (ver figura 21), puesto que, la salida tiene que ser un número entero.

```
Y_test = model.predict(X_val).flatten()
```

Figura 21 Cálculo de las salidas para el conjunto de validación

```
Y_test =np.round(Y_test, 0)
```

Figura 22 Función para el redondeo de las salidas

La evaluación de los resultados se realiza de dos maneras: el error cuadrático y el número de aciertos. Pero se debe tener en cuenta que el error cuadrático no es una medida tan relevante en el caso de esta investigación, puesto que no interesa de la misma manera saber cuánto se aproxima a los valores sino cuantos ha acertado.

```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

Figura 23 Cálculo del error cuadrático

```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

Figura 24 Cálculo del porcentaje de aciertos

[6] Tamaño de los lotes de procesamiento.
[7] Número de veces que se entrena la red en ese entrenamiento.

Se genera una gráfica que sirve para detectar el tipo de errores que comete en la red.

```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```

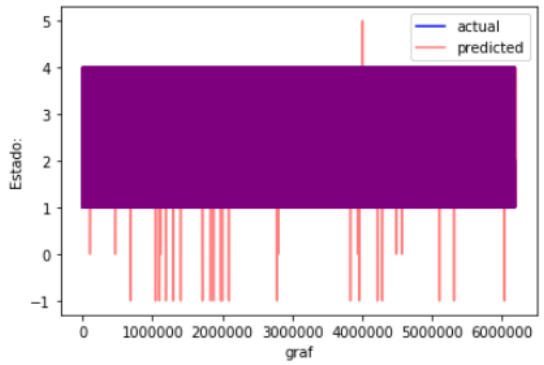


Figura 25 Gráfico en la que se muestran las salidas reales frente a las predichas

Pudiendo observar si los errores se encuentran entre los márgenes de las salidas o si al contrario son estados que no existen para nuestra red neuronal, nótese que a ella solo se le han descrito salidas entre 1 y 4.

1.5.4. RNA Detector de Pulsera

Para el caso del medidor de pulsera los sensores trabajan a diferentes frecuencias, por lo tanto, dependiendo de la frecuencia de cada uno existen momentos en los que se reciben valores de unos sensores, pero de otros no. Debido a esto se descartan esos valores extra y se utilizan los valores que se dan en los mismos instantes de tiempo. Es decir, se eliminan los datos recibidos de los sensores que trabajan a mayor frecuencia porque sus valores no varían mucho en esos pequeños instantes de tiempo, hasta que los sensores de menor frecuencia vuelvan a dar un valor.

La carga de los datos del detector de pulsera se realiza de la misma forma que con el detector pectoral:

```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
```

Figura 26 Carga de datos para el medidor de pecho

Al tratar los datos del detector de pulsera se calcula qué sensor es el que tiene la mínima cantidad de datos, es decir el de menor frecuencia.

```
mind=min(len(a),len(b),len(c),len(d),len(A2l))
```

Figura 27 Cálculo del valor mínimo de datos que se tienen

Con ello se calcula la diferencia de las frecuencias respecto a la mínima:

```
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
```

Figura 28 Relación de frecuencias de los diferentes sensores

Para llevar a cabo el código de la manera más general posible, se usa el valor de la relación entre las frecuencias y se crea un nuevo vector que se queda con el valor correspondiente a la menor frecuencia, creando un nuevo vector con datos a la menor frecuencia (ver figura 28). Se realiza para todas las entradas, se unen los datos y finalmente

se descartan los correspondientes a las salidas 0,5,6,7 como se hacía con el medidor de pecho como se aprecia en la figura 13.

```
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
```

Figura 29 Creación de arrays con los datos a la frecuencia de la menor de los sensores

El resto de pasos a aplicar en la metodología del detector de pulsera es igual a como se trabaja con el detector pectoral. Menos para la correlación puesto que se trata de sensores diferentes, por lo que su estudio varía.

	0	1	2	3	4	5	6	7
0	1	0.0129906	0.0970228	0.0768107	0.000704121	-0.028306	-0.339708	-0.0033121
1	0.0129906	1	-0.10602	0.112981	-0.000290653	0.182445	0.105206	0.225818
2	0.0970228	-0.10602	1	0.121402	0.00209529	-0.177823	0.0207045	-0.05912
3	0.0768107	0.112981	0.121402	1	0.000764396	-0.0207728	-0.0986843	-0.04957
4	0.000704121	-0.000290653	0.00209529	0.000764396	1	-0.000327508	-0.00232309	-0.00235096
5	-0.028306	0.182445	-0.177823	-0.0207728	-0.000327508	1	0.210019	0.0993003
6	-0.339708	0.105206	0.0207045	-0.0986843	-0.00232309	0.210019	1	0.32277
7	-0.0033121	0.225818	-0.05912	-0.04957	-0.00235096	0.0993003	0.32277	1

Tabla 4 Correlación de los datos del detector de pulsera

Se determina a partir de los valores de la tabla 4 que no existe ninguna correlación elevada entre los datos que se obtienen de los diferentes sensores del detector de pulsera.

1.5.5. RNA Ensemble

En esta investigación se utiliza un ensemble con los datos del medidor de pulsera. Por lo que el refinado de los datos se realiza de la misma forma, la principal diferencia data en crear dos conjuntos de datos después de haber definido los conjuntos de validación y entrenamiento como en la figura 16.

Se generan dos conjuntos, denominados A (formado por X_trainA y X_valA) y B (formado por X_trainB y X_valB) para los entrenamientos de las dos redes del ensemble. Para el conjunto A no se tiene en cuenta el ritmo cardiaco y para el conjunto B no se trabaja con la actividad electrodérmica.

```
X_trainA= np.delete(X_train,3 , 1)
X_trainB= np.delete(X_train,4 , 1)
X_valA= np.delete(X_val,3, 1)
X_valB= np.delete(X_val,4 , 1)
```

Figura 30 Generación de los conjuntos de datos para las dos RNA del ensemble

Se ha realizado el mismo procedimiento para las dos redes neuronales artificiales, que coincide con el que ha sido utilizado con las otras RNA, calculando los resultados obtenidos de ambas redes. Para obtener las salidas del ensemble se combinan usando el mínimo, el valor medio y el máximo entre ambas salidas.

```
YtA = np.asmatrix(Y_testA)
YtB = np.asmatrix(Y_testB)
ense1=np.zeros(len(Y_testA))
ense2=np.zeros(len(Y_testA))
ense3=np.zeros(len(Y_testA))
Yt=np.concatenate((YtA.T,YtB.T),axis=1)
for i in range(len(Yt)):
    ense1[i]=np.min(Yt[i,:])
    ense2[i]=np.max(Yt[i,:])
    ense3[i]=np.mean(Yt[i,:])

ense3 =np.round(ense3, 0)
```

Figura 31 Calculo de las salidas del ensemble

Para la media, además se redondea la salida porque tiene que ser un valor entero. Finalmente se obtienen el error de ajuste y el número de aciertos para cada una de las salidas y se comparan con los que se han obtenido antes de realizar el ensemble.

1.6. Resultados

Los resultados calculados según los tipos de ejecución se encuentran representados en las tablas 5 a 8, se puede comprobar en los apéndices 1 a 32, para cada una de las ejecuciones, los resultados que han sido obtenidos. Para cada tabla se especifica el tipo de red, activador, la función de coste, el número de apéndice en el que se recoge la salida de la ejecución del código, el error cuadrático(R2) y porcentaje de acierto(F1).

DETECTOR PECTORAL

RED	Activador	Función de coste	Apéndice	R2	F1
ELU 5 CAPAS	Adadelta	MAE[8]	1	99,18%	99,40%
ELU Y SELU 5 CAPAS	Adadelta	MAE	2	98,79%	99,12%
SELU 5 CAPAS	Adadelta	MAE	3	98,32%	98,78%
LINEAR 5 CAPAS	Adadelta	MAE	4	-442%	27,26%
ELU 5 CAPAS	Nadam	MAE	5	99,00%	99,28%
ELU 5 CAPAS	Rmsprop	MAE	6	98,53%	98,94%
ELU 5 CAPAS	Adadelta	MSLE[9]	7	99,28%	99,28%
ELU 5 CAPAS	Adadelta	Hinge[10]	8	-726,61	14,44%
ELU 7 CAPAS	Adadelta	MAE	9	99,08%	99,43%
ELU 4 CAPAS	Adadelta	MAE	10	98,93%	99,08%

Tabla 5 Resultados RNA Detector Pectoral

[8] Error absoluto medio
[9] Error logarítmico cuadrado medio
[10] Error de máximo excedente

DETECTOR DE PULSERA (CON MEDITACIÓN)

ENTRENAMIENTOS	Red	Activador	Función de coste	Apéndice	R2	F1
3	Elu 4 capas	Adadelta	MAE	11	95,41%	96,53%
	Elu 5 capas	Adadelta	MAE	12	95,17%	97,28%
	Elu 7 capas	Adadelta	MAE	13	91,56%	95,13%
2	Elu 4 capas	Adadelta	MAE	14	95,63%	97,46%
	Elu 5 capas	Adadelta	MAE	15	95,30%	97,22%
	Elu 7 capas	Adadelta	MAE	16	93,81%	96,75%
1	Elu 4 capas	Adadelta	MAE	17	94,73%	96,11%
	Elu 5 capas	Adadelta	MAE	18	96,32%	97,61%
	Elu 7 capas	Adadelta	MAE	19	95,57%	97,26%
2	Elu y Selu 5 capas	Adadelta	MAE	20	96,00%	97,78%

Tabla 6 Resultados detector de pulsera con meditación

DETECTOR DE PULSERA (SIN MEDITACIÓN)

ENTRENAMIENTOS	Red	Activador	Función de coste	Apéndice	R2	F1
2	Elu y Selu 5 capas	Adadelta	MAE	21	89,23%	96,91%
3	Elu 4 capas	Adadelta	MAE	22	94.82%	98,17%
	Elu 5 capas	Adadelta	MAE	23	93,43%	97,75%
	Elu 7 capas	Adadelta	MAE	24	85,73%	95,78%
2	Elu 4 capas	Adadelta	MAE	25	94,50%	98,11%
	Elu 5 capas	Adadelta	MAE	26	93,73%	98,14%
	Elu 7 capas	Adadelta	MAE	27	90,39%	97,31%
1	Elu 4 capas	Adadelta	MAE	28	89,11%	96,55%
	Elu 5 capas	Adadelta	MAE	29	89,35%	97,30%
	Elu 7 capas	Adadelta	MAE	30	91,13%	96,78%

Tabla 7 Resultados detector de pulsera sin meditación

ENSEMBLE

CONJUNTO O TIPO DE ENSEMBLE	Red	Activador	Función de coste	Apéndice	R2	F1
CONJUNTO A	Elu 5 capas	Adadelta	MAE	31	94,55%	97,15%
CONJUNTO B	Elu 5 capas	Adadelta	MAE	31	92,37%	94,30%
MINIMO	Elu 5 capas	Adadelta	MAE	31	93,05%	95,60%
MAXIMO	Elu 5 capas	Adadelta	MAE	31	93,87%	95,86%
MEDIA	Elu 5 capas	Adadelta	MAE	31	94,86%	95,56%
CONJUNTO A	Elu 4 capas	Adadelta	MAE	32	96,23%	97,18%
CONJUNTO B	Elu 4 capas	Adadelta	MAE	32	93,34%	95,20%
MINIMO	Elu 4 capas	Adadelta	MAE	32	94,14%	95,47%
MAXIMO	Elu 4 capas	Adadelta	MAE	32	95,47%	96,91%
MEDIA	Elu 4 capas	Adadelta	MAE	32	96,05%	96,50%

Tabla 8 Resultados de los ensemble para el detector de pulsera

1.7. Conclusiones

En la presente investigación se estudian los datos de 15 sujetos, a los que se les ha provocado estrés mediante test destinados para tal efecto, afecto y meditación, con el fin de estudiar sus señales biomédicas en esos casos. Para ello, se utilizan dos detectores, una banda pectoral y una pulsera, con sensores específicos.

El objetivo principal de la investigación ha sido diseñar redes neuronales artificiales para la realización de un detector de estrés a partir de señales biomédicas y el análisis de sus resultados. Por la gran cantidad de datos con los que se trabaja, los requerimientos respecto a la RAM necesaria son elevados, así mismo, para poder trabajar de una forma más eficiente con ellos se utiliza la GPU además de la CPU evitando un consumo elevado en la misma a la hora de entrenar la red neuronal.

En la evaluación de los resultados se utilizan los métodos de: Ajuste de los resultados, error cuadrático, y la tasa de acierto, sin embargo, para esta aplicación el primero no tiene tanta relevancia, puesto que, al tratarse de un clasificador con los estados de salida definidos saber cuántas veces se ha acertado con la salida correspondiente indica mejor como está trabajando la RNA. Así mismo, se pueden comparar los resultados obtenidos con los de la investigación de Schmidt, P., Reiss, A., Duerichen, R., Marberger, C., y Van Laerhoven, K. Introducing WESAD, a Multimodal Dataset for Wearable Stress and Affect Detection[20] en la que se estudia para el mismo conjunto de sujetos pero utilizando otros métodos.

En el desarrollo de las RNA se ha estudiado en un principio como se comportaba con diferentes tipos de redes neuronales, mostrando algunas que no son válidas, para el detector de pecho, que se encuentran en los apéndices 4 y 8.

Si se analizan los resultados obtenidos se puede determinar que las mejores configuraciones de red para este propósito son de tipo ELU aunque las redes SELU también trabajan de forma eficiente, así como la combinación de ambas. Porque se trata de tipos de neuronas similares, donde la SELU solo añade un escalar al tipo ELU. En cuanto a la función de coste vemos como el error medio absoluto y el error medio logarítmico cuadrado trabajan de manera correcta. Para el optimizador ocurre lo mismo vemos como tanto el optimizador Nadam como el Adadelta consiguen buenos resultados, pero algo mejores en el caso del segundo.

En el detector de pulsera se estudia cómo puede influir el largo del entrenamiento en los resultados de diferentes redes neuronales con diferentes complejidades, pudiendo determinarse que con los entrenamientos más extensos y redes más complejas no se consiguen los mejores resultados. También, se estudia excluyendo el estado de la meditación para comparar mejor los resultados con los del estudio de Robert Bosch GmbH y University Siegen. Por otra parte, se ha estudiado la posible mejora de los resultados con el uso de ensembles. Sin embargo, no se consigue una mejora de los resultados obtenidos con una simple red o respecto a los que se consiguen con uno de los conjuntos empleados para el ensemble. Aunque, en el caso del ajuste de los resultados si se produce una mejora en el primero de ellos.

Si comparamos los resultados obtenidos con los del otro estudio vemos que ellos estudian varios supuestos como se muestra en el estado del arte del presente proyecto. Ellos estudian la detección de estrés y no estrés solamente obteniendo hasta un 92%-93% de acierto para el detector de pecho y del 87%-88% para el de pulsera. En esta investigación no se ha estudiado solo la detección de estrés y no estrés puesto que con los demás estados ya obtenemos resultados mayores al 99% para el detector de pecho y del 97-98% para el de pulsera. Cuando se incluyen estados de afecto en el estudio de Robert Bosch GmbH y University Siegen su precisión es de un 75%-76% para la pulsera y el detector de pecho, mientras que en esta investigación también se está detectando el estado de meditación consiguiendo los resultados anteriormente mencionados.

Frente a futuras investigaciones existen algunos campos en los que se podría ampliar la investigación:

- Utilizar ensemble en los cuales se aproveche la salida de una de las redes como entrada de la siguiente o con otros métodos para combinar los resultados
- En el caso del detector de pulsera se podría replicar o interpolar los datos para los sensores de menores frecuencias, en lugar de lo comentado en el apartado 1.5.2, para evitar de reducir la cantidad de datos utilizados como se realiza en esta investigación aumentarla para mejorar los resultados.

1.8. Referencias Bibliográficas

[1] Abbott, P. W., Gumusoglu, S. B., Bittle, J., Beversdorf, D. Q., & Stevens, H. E. 2018. Prenatal stress and genetic risk: How prenatal stress interacts with genetics to alter risk for psychiatric illness. *Psychoneuroendocrinology*, *90*, pp 9-21.

[2] Aguilar, R., Torres, J., & Martín, C. 2019. Aprendizaje Automático en la Identificación de Sistemas. Un Caso de Estudio en la Predicción de la Generación Eléctrica de un Parque Eólico. *Revista Iberoamericana de Automática e Informática industrial, 16(1)*, pp 114-127. doi: https://doi.org/10.4995/riai.2018.9421

[3] Aguilar, R. & Muñoz, V. 2018. Control Inteligente. Entorno Institucional 2017/2018. Universidad de La Laguna.

[4] Brownlee, J. 2018. *Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python*. Machine Learning Mastery.

[5] Can, Y. S., Arnrich, B., & Ersoy, C. 2019. Stress detection in daily life scenarios using smart phones and wearable sensors: A survey. *Journal of biomedical informatics*, 103139.

[6] Definición de ansiedad y estrés. Sociedad Española para el Estudio de la Ansiedad y el Estrés- SEAS. Disponible en la URL: http://www.ansiedadyestres.org/salud

[7] Definición de Electromiografia. Mayo Clinic. Disponible en la URL: https://www.mayoclinic.org/es-es/tests-procedures/emg/about/pac-20393913

[8] Definición de la librería Keras. Keras Documentation. Disponible en la URL: https://keras.io/

[9] Definición de la librería Matplotlib. Matplotlib. Disponible en la URL: https://matplotlib.org/

[10] Definición de la librería NumPy. NumPy.org. Disponible en la URL: https://numpy.org/

[11] Definición de la librería Pandas. Pandas. Disponible en la URL: https://pandas.pydata.org/

[12] Definición de la librería Pickle. Python software Fundation. Disponible en la URL: https://docs.python.org/3/library/pickle.html

[13] Definición de la librería Scikit-learn. Scikit-learn. Disponible en la URL: https://scikit-learn.org/stable/index.html

[14] Definición de la librería TensorFlow. TensorFlow. Disponible en la URL: https://www.tensorflow.org/

[15] Descripción de Anaconda Navigator. Anaconda, Inc. Disponible en la URL: https://anaconda.org/anaconda/anaconda-navigator

[16] Descripción de los datos de los sujetos. Schmidt, P., Reiss, A., Duerichen, R., Marberger, C., & Van Laerhoven, K. Disponible en la URL: https://archive.ics.uci.edu/ml/datasets/WESAD+%28Wearable+Stress+and+Affect+Detection%29

[17] Johnson, J. V., & Hall, E. M. 1988. Job strain, work place social support, and cardiovascular disease: a cross-sectional study of a random sample of the Swedish working population. American journal of public health, 78(10), pp 1336-1342.

[18] Metaxas, D., Venkataraman, S., & Vogler, C. 2004, June. Image-based stress recognition using a model-based dynamic face tracking system. In *International Conference on Computational Science*, pp. 813-821. Springer, Berlin, Heidelberg.

[19] Resumen de redes neuronales artificiales. Diego Calvo Data Scientist Proyect Manager. Disponible en la URL: http://www.diegocalvo.es/definicion-de-red-neuronal/

[20] Schmidt, P., Reiss, A., Duerichen, R., Marberger, C., & Van Laerhoven, K. 2018, October. Introducing WESAD, a Multimodal Dataset for Wearable Stress and Affect Detection. In *Proceedings of the 2018 on International Conference on Multimodal Interaction*, pp. 400-408. ACM.

[21] Sriramprakash, S., Prasanna, V. D., & Murthy, O. R. 2017. Stress detection in working people. *Procedia computer science, 115*, pp. 359-366.

Universidad de La Laguna

Apéndices

Aprendizaje automático en el diseño de un detector de estrés a partir de señales biomédicas

2. Apéndices

Índice de apéndices

Chest_elux5_Adadel_mae_Final

August 13, 2019

1 Chest data

Librerias:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import sys
import os
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Chest

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A2l=data['label']
A2a=27*(np.ones((len(a),1)))
A2=np.concatenate((a,b,c,d,e,f,A2a),axis=1)
B1 = np.asmatrix(A2l)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
```

```
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>=4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A3l=data['label']
A3a=25*(np.ones((len(a),1)))
A3=np.concatenate((a,b,c,d,e,f,A3a),axis=1)
B2 = np.asmatrix(A3l)
B2=B2.T
MAT2=np.concatenate((B2,A3),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A4l=data['label']
A4a=35*(np.ones((len(a),1)))
A4=np.concatenate((a,b,c,d,e,f,A4a),axis=1)
B3 = np.asmatrix(A4l)
B3=B3.T
MAT3=np.concatenate((B3,A4),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A5l=data['label']
A5a=27*(np.ones((len(a),1)))
A5=np.concatenate((a,b,c,d,e,f,A5a),axis=1)
B4 = np.asmatrix(A5l)
B4=B4.T
MAT4=np.concatenate((B4,A5),axis=1)
```

```
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A6l=data['label']
A6a=28*(np.ones((len(a),1)))
A6=np.concatenate((a,b,c,d,e,f,A6a),axis=1)
B5 = np.asmatrix(A6l)
B5=B5.T
MAT5=np.concatenate((B5,A6),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A7l=data['label']
A7a=27*(np.ones((len(a),1)))
A7=np.concatenate((a,b,c,d,e,f,A7a),axis=1)
B6 = np.asmatrix(A7l)
B6=B6.T
MAT6=np.concatenate((B6,A7),axis=1)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A8l=data['label']
A8a=26*(np.ones((len(a),1)))
A8=np.concatenate((a,b,c,d,e,f,A8a),axis=1)
B7 = np.asmatrix(A8l)
B7=B7.T
MAT7 = np.concatenate((B7,A8),axis=1)
```

```
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A9l=data['label']
A9a=28*(np.ones((len(a),1)))
A9=np.concatenate((a,b,c,d,e,f,A9a),axis=1)
B8 = np.asmatrix(A9l)
B8=B8.T
MAT8 = np.concatenate((B8,A9),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A10l=data['label']
A10a=26*(np.ones((len(a),1)))
A10=np.concatenate((a,b,c,d,e,f,A10a),axis=1)
B9 = np.asmatrix(A10l)
B9=B9.T
MAT9 = np.concatenate((B9,A10),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A11l=data['label']
A11a=28*(np.ones((len(a),1)))
A11=np.concatenate((a,b,c,d,e,f,A11a),axis=1)
B10 = np.asmatrix(A11l)
B10=B10.T
MAT10 = np.concatenate((B10,A11),axis=1)
```

```
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A13l=data['label']
A13a=24*(np.ones((len(a),1)))
A13=np.concatenate((a,b,c,d,e,f,A13a),axis=1)
B11 = np.asmatrix(A13l)
B11=B11.T
MAT11 = np.concatenate((B11,A13),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A14l=data['label']
A14a=24*(np.ones((len(a),1)))
A14=np.concatenate((a,b,c,d,e,f,A14a),axis=1)
B12 = np.asmatrix(A14l)
B12=B12.T
MAT12 = np.concatenate((B12,A14),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A15l=data['label']
A15a=29*(np.ones((len(a),1)))
A15=np.concatenate((a,b,c,d,e,f,A15a),axis=1)
B13 = np.asmatrix(A15l)
B13=B13.T
MAT13 = np.concatenate((B13,A15),axis=1)
```

```
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A16l=data['label']
A16a=29*(np.ones((len(a),1)))
A16=np.concatenate((a,b,c,d,e,f,A16a),axis=1)
B14 = np.asmatrix(A16l)
B14=B14.T
MAT14 = np.concatenate((B14,A16),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>4), 0)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A17l=data['label']
A17a=29*(np.ones((len(a),1)))
A17=np.concatenate((a,b,c,d,e,f,A17a),axis=1)
B15 = np.asmatrix(A17l)
B15=B15.T
MAT15 = np.concatenate((B15,A17),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Comprobamos su correlación:

[18]:
```
data=pandas.DataFrame(MAT)
corr = data.corr()
corr.style.background_gradient(cmap='coolwarm')
```

[18]:
```
<pandas.io.formats.style.Styler at 0x23894efab70>
```

Dividimos en los conjuntos de validación y entrenamiento:

[19]:
```
X = MAT[:, 1:10]
Y = MAT[:, 0]
```

```
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)

#scaler = StandardScaler()
#X_train = scaler.fit_transform(X_train)
#X_val = scaler.fit_transform(X_val)
```

Definimos la arquitectura de la red:

[20]:
```
model = Sequential([
    Dense(128, input_shape=(9,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
    Dense(48),
    Activation('elu'),
    Dense(24),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

Entrenamos la red:

[21]:
```
NUM_EPOCHS =5
BATCH_SIZE = 200

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.1)
```

```
Train on 22271762 samples, validate on 2474641 samples
Epoch 1/5
22271762/22271762 [==============================] - 385s 17us/step - loss:
0.1249 - val_loss: 0.0589
Epoch 2/5
22271762/22271762 [==============================] - 352s 16us/step - loss:
0.0467 - val_loss: 0.0392
Epoch 3/5
22271762/22271762 [==============================] - 354s 16us/step - loss:
0.0381 - val_loss: 0.0323
Epoch 4/5
22271762/22271762 [==============================] - 352s 16us/step - loss:
0.0342 - val_loss: 0.0335
Epoch 5/5
22271762/22271762 [==============================] - 358s 16us/step - loss:
0.0320 - val_loss: 0.0267
```

[22]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[23]:
```
Y_test =np.round(Y_test, 0)
```

Mostramos una parte de los resultados:

[24]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

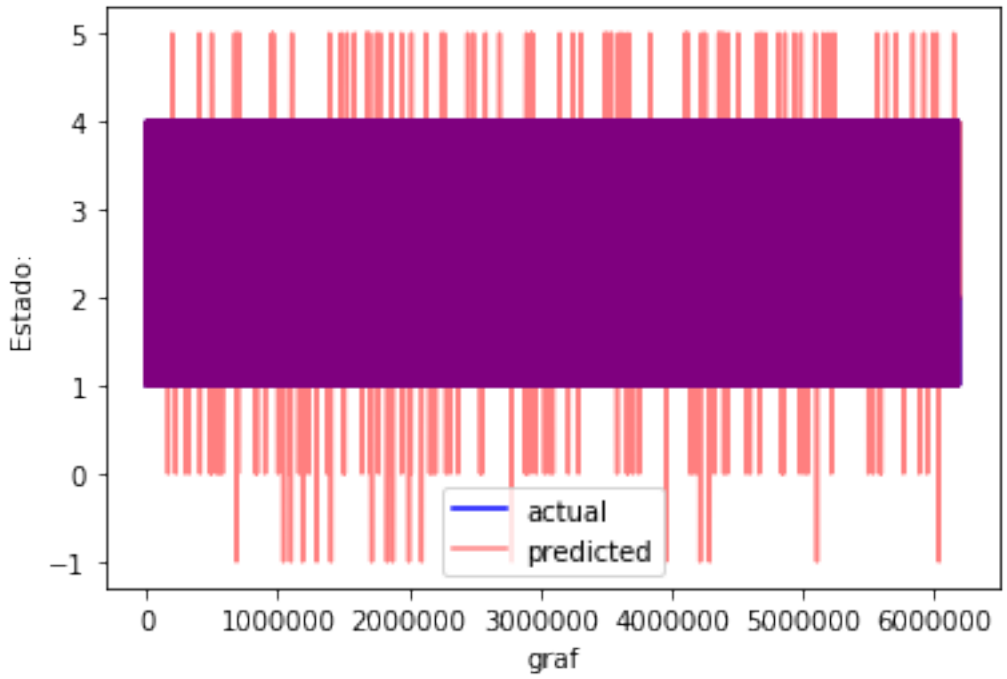
```
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 4., predicted:4.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

Evaluamos los resultados:

[25]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.9918323255486744
```

[26]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[27]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[27]:
```
0.994040184586011
```

[ ]:
```

```

Chest_eluselux5_Adadel_mae_Final

August 13, 2019

1 Chest data

Librerias:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import sys
import os
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Chest

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A2l=data['label']
A2a=27*(np.ones((len(a),1)))
A2=np.concatenate((a,b,c,d,e,f,A2a),axis=1)
B1 = np.asmatrix(A2l)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
```

```
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>=4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A3l=data['label']
A3a=25*(np.ones((len(a),1)))
A3=np.concatenate((a,b,c,d,e,f,A3a),axis=1)
B2 = np.asmatrix(A3l)
B2=B2.T
MAT2=np.concatenate((B2,A3),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A4l=data['label']
A4a=35*(np.ones((len(a),1)))
A4=np.concatenate((a,b,c,d,e,f,A4a),axis=1)
B3 = np.asmatrix(A4l)
B3=B3.T
MAT3=np.concatenate((B3,A4),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A5l=data['label']
A5a=27*(np.ones((len(a),1)))
A5=np.concatenate((a,b,c,d,e,f,A5a),axis=1)
B4 = np.asmatrix(A5l)
B4=B4.T
MAT4=np.concatenate((B4,A5),axis=1)
```

```
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A6l=data['label']
A6a=28*(np.ones((len(a),1)))
A6=np.concatenate((a,b,c,d,e,f,A6a),axis=1)
B5 = np.asmatrix(A6l)
B5=B5.T
MAT5=np.concatenate((B5,A6),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A7l=data['label']
A7a=27*(np.ones((len(a),1)))
A7=np.concatenate((a,b,c,d,e,f,A7a),axis=1)
B6 = np.asmatrix(A7l)
B6=B6.T
MAT6=np.concatenate((B6,A7),axis=1)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A8l=data['label']
A8a=26*(np.ones((len(a),1)))
A8=np.concatenate((a,b,c,d,e,f,A8a),axis=1)
B7 = np.asmatrix(A8l)
B7=B7.T
MAT7 = np.concatenate((B7,A8),axis=1)
```

```
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A9l=data['label']
A9a=28*(np.ones((len(a),1)))
A9=np.concatenate((a,b,c,d,e,f,A9a),axis=1)
B8 = np.asmatrix(A9l)
B8=B8.T
MAT8 = np.concatenate((B8,A9),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A10l=data['label']
A10a=26*(np.ones((len(a),1)))
A10=np.concatenate((a,b,c,d,e,f,A10a),axis=1)
B9 = np.asmatrix(A10l)
B9=B9.T
MAT9 = np.concatenate((B9,A10),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A11l=data['label']
A11a=28*(np.ones((len(a),1)))
A11=np.concatenate((a,b,c,d,e,f,A11a),axis=1)
B10 = np.asmatrix(A11l)
B10=B10.T
MAT10 = np.concatenate((B10,A11),axis=1)
```

```
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A13l=data['label']
A13a=24*(np.ones((len(a),1)))
A13=np.concatenate((a,b,c,d,e,f,A13a),axis=1)
B11 = np.asmatrix(A13l)
B11=B11.T
MAT11 = np.concatenate((B11,A13),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A14l=data['label']
A14a=24*(np.ones((len(a),1)))
A14=np.concatenate((a,b,c,d,e,f,A14a),axis=1)
B12 = np.asmatrix(A14l)
B12=B12.T
MAT12 = np.concatenate((B12,A14),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A15l=data['label']
A15a=29*(np.ones((len(a),1)))
A15=np.concatenate((a,b,c,d,e,f,A15a),axis=1)
B13 = np.asmatrix(A15l)
B13=B13.T
MAT13 = np.concatenate((B13,A15),axis=1)
```

```
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A16l=data['label']
A16a=29*(np.ones((len(a),1)))
A16=np.concatenate((a,b,c,d,e,f,A16a),axis=1)
B14 = np.asmatrix(A16l)
B14=B14.T
MAT14 = np.concatenate((B14,A16),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>4), 0)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A17l=data['label']
A17a=29*(np.ones((len(a),1)))
A17=np.concatenate((a,b,c,d,e,f,A17a),axis=1)
B15 = np.asmatrix(A17l)
B15=B15.T
MAT15 = np.concatenate((B15,A17),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Comprobamos su correlación:

[18]:
```
data=pandas.DataFrame(MAT)
corr = data.corr()
corr.style.background_gradient(cmap='coolwarm')
```

[18]:
```
<pandas.io.formats.style.Styler at 0x2328189ada0>
```

Dividimos en los conjuntos de validación y entrenamiento:

[19]:
```
X = MAT[:, 1:10]
Y = MAT[:, 0]
```

```
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)

#scaler = StandardScaler()
#X_train = scaler.fit_transform(X_train)
#X_val = scaler.fit_transform(X_val)
```

Definimos la arquitectura de la red:

[20]:
```
model = Sequential([
    Dense(128, input_shape=(9,)),
    Activation('elu'),
    Dense(70),
    Activation('selu'),
    Dense(48),
    Activation('elu'),
    Dense(24),
    Activation('selu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

Entrenamos la red:

[21]:
```
NUM_EPOCHS =5
BATCH_SIZE = 200

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.1)
```

```
Train on 22271762 samples, validate on 2474641 samples
Epoch 1/5
22271762/22271762 [==============================] - 395s 18us/step - loss:
0.1354 - val_loss: 0.0537
Epoch 2/5
22271762/22271762 [==============================] - 393s 18us/step - loss:
0.0519 - val_loss: 0.0373
Epoch 3/5
22271762/22271762 [==============================] - 388s 17us/step - loss:
0.0429 - val_loss: 0.0374
Epoch 4/5
22271762/22271762 [==============================] - 387s 17us/step - loss:
0.0384 - val_loss: 0.0350
Epoch 5/5
22271762/22271762 [==============================] - 389s 17us/step - loss:
0.0357 - val_loss: 0.0332
```

[22]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[23]:
```
Y_test =np.round(Y_test, 0)
```

Mostramos una parte de los resultados:

[24]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

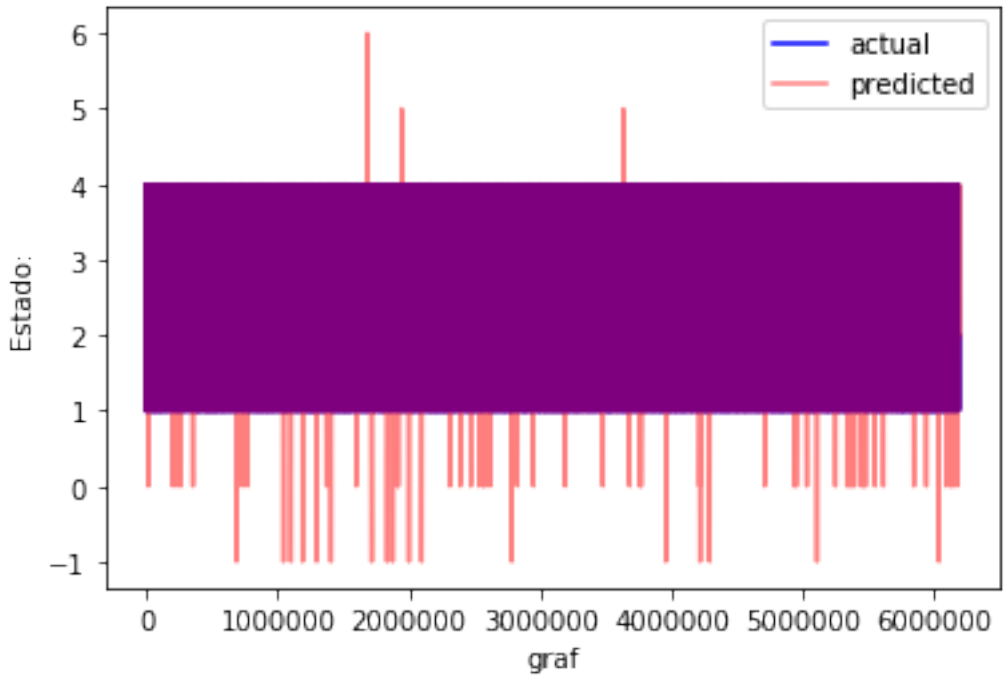
```
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 4., predicted:4.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

Evaluamos los resultados:

[25]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.9879145895694361
```

[26]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[27]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[27]:
```
0.9912582046264177
```

[ ]:
```
```

Chest_selux5_Adadel_mae_Final

August 16, 2019

1 Chest data

Librerias:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Chest

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A2l=data['label']
A2a=27*(np.ones((len(a),1)))
A2=np.concatenate((a,b,c,d,e,f,A2a),axis=1)
B1 = np.asmatrix(A2l)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>=4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A3l=data['label']
A3a=25*(np.ones((len(a),1)))
A3=np.concatenate((a,b,c,d,e,f,A3a),axis=1)
B2 = np.asmatrix(A3l)
B2=B2.T
MAT2=np.concatenate((B2,A3),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A4l=data['label']
A4a=35*(np.ones((len(a),1)))
A4=np.concatenate((a,b,c,d,e,f,A4a),axis=1)
B3 = np.asmatrix(A4l)
B3=B3.T
MAT3=np.concatenate((B3,A4),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A5l=data['label']
A5a=27*(np.ones((len(a),1)))
A5=np.concatenate((a,b,c,d,e,f,A5a),axis=1)
B4 = np.asmatrix(A5l)
B4=B4.T
MAT4=np.concatenate((B4,A5),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A6l=data['label']
A6a=28*(np.ones((len(a),1)))
A6=np.concatenate((a,b,c,d,e,f,A6a),axis=1)
B5 = np.asmatrix(A6l)
B5=B5.T
MAT5=np.concatenate((B5,A6),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A7l=data['label']
A7a=27*(np.ones((len(a),1)))
A7=np.concatenate((a,b,c,d,e,f,A7a),axis=1)
B6 = np.asmatrix(A7l)
B6=B6.T
MAT6=np.concatenate((B6,A7),axis=1)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A8l=data['label']
A8a=26*(np.ones((len(a),1)))
A8=np.concatenate((a,b,c,d,e,f,A8a),axis=1)
B7 = np.asmatrix(A8l)
B7=B7.T
MAT7 = np.concatenate((B7,A8),axis=1)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A9l=data['label']
A9a=28*(np.ones((len(a),1)))
A9=np.concatenate((a,b,c,d,e,f,A9a),axis=1)
B8 = np.asmatrix(A9l)
B8=B8.T
MAT8 = np.concatenate((B8,A9),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A10l=data['label']
A10a=26*(np.ones((len(a),1)))
A10=np.concatenate((a,b,c,d,e,f,A10a),axis=1)
B9 = np.asmatrix(A10l)
B9=B9.T
MAT9 = np.concatenate((B9,A10),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A11l=data['label']
A11a=28*(np.ones((len(a),1)))
A11=np.concatenate((a,b,c,d,e,f,A11a),axis=1)
B10 = np.asmatrix(A11l)
B10=B10.T
MAT10 = np.concatenate((B10,A11),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A13l=data['label']
A13a=24*(np.ones((len(a),1)))
A13=np.concatenate((a,b,c,d,e,f,A13a),axis=1)
B11 = np.asmatrix(A13l)
B11=B11.T
MAT11 = np.concatenate((B11,A13),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A14l=data['label']
A14a=24*(np.ones((len(a),1)))
A14=np.concatenate((a,b,c,d,e,f,A14a),axis=1)
B12 = np.asmatrix(A14l)
B12=B12.T
MAT12 = np.concatenate((B12,A14),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A15l=data['label']
A15a=29*(np.ones((len(a),1)))
A15=np.concatenate((a,b,c,d,e,f,A15a),axis=1)
B13 = np.asmatrix(A15l)
B13=B13.T
MAT13 = np.concatenate((B13,A15),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A16l=data['label']
A16a=29*(np.ones((len(a),1)))
A16=np.concatenate((a,b,c,d,e,f,A16a),axis=1)
B14 = np.asmatrix(A16l)
B14=B14.T
MAT14 = np.concatenate((B14,A16),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>4), 0)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A17l=data['label']
A17a=29*(np.ones((len(a),1)))
A17=np.concatenate((a,b,c,d,e,f,A17a),axis=1)
B15 = np.asmatrix(A17l)
B15=B15.T
MAT15 = np.concatenate((B15,A17),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Comprobamos su correlación:

[18]:
```
data=pandas.DataFrame(MAT)
corr = data.corr()
corr.style.background_gradient(cmap='coolwarm')
```

[18]:
```
<pandas.io.formats.style.Styler at 0x1a78ff3d710>
```

Dividimos en los conjuntos de validación y entrenamiento:

[19]:
```
X = MAT[:, 1:10]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
```

```
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)

#scaler = StandardScaler()
#X_train = scaler.fit_transform(X_train)
#X_val = scaler.fit_transform(X_val)
```

Definimos la arquitectura de la red:

[20]:
```
model = Sequential([
    Dense(128, input_shape=(9,)),
    Activation('elu'),
    Dense(70),
    Activation('selu'),
    Dense(48),
    Activation('elu'),
    Dense(24),
    Activation('selu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

Entrenamos la red:

[21]:
```
NUM_EPOCHS =5
BATCH_SIZE = 200

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.1)
```

```
Train on 22271762 samples, validate on 2474641 samples
Epoch 1/5
22271762/22271762 [==============================] - 386s 17us/step - loss:
0.1355 - val_loss: 0.0587
Epoch 2/5
22271762/22271762 [==============================] - 389s 17us/step - loss:
0.0495 - val_loss: 0.0405
Epoch 3/5
22271762/22271762 [==============================] - 375s 17us/step - loss:
0.0410 - val_loss: 0.0409
Epoch 4/5
22271762/22271762 [==============================] - 372s 17us/step - loss:
0.0370 - val_loss: 0.0349
Epoch 5/5
22271762/22271762 [==============================] - 373s 17us/step - loss:
0.0348 - val_loss: 0.0380
```

[22]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[23]:
```
Y_test =np.round(Y_test, 0)
```

Mostramos una parte de los resultados:

[24]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

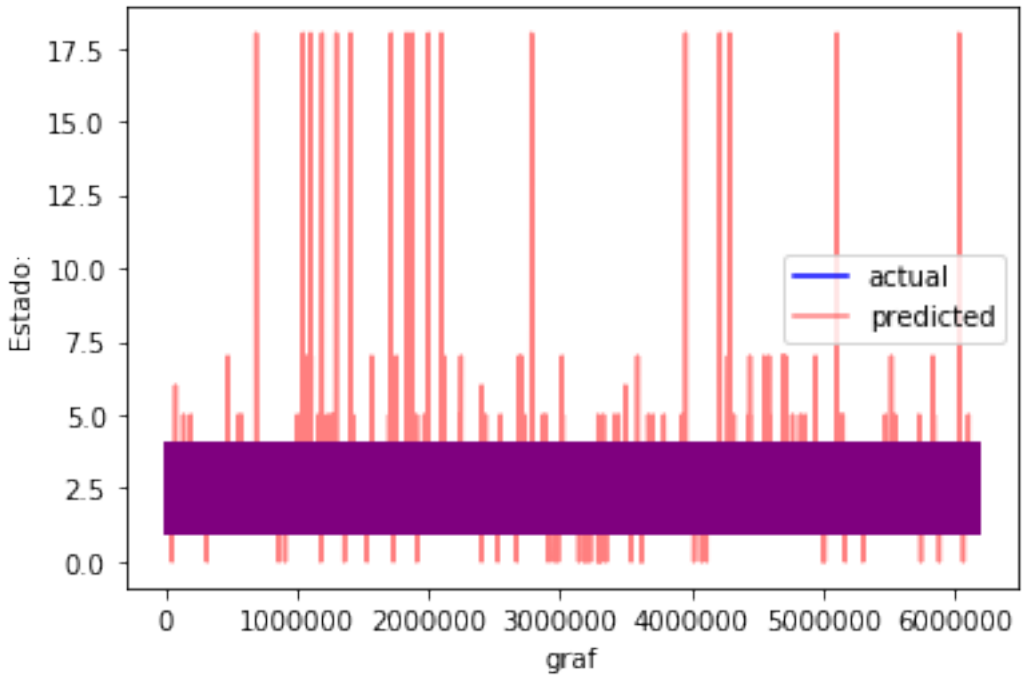
```
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 4., predicted:4.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

Evaluamos los resultados:

[25]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.9832446518938989
```

[26]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[27]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[27]:
```
0.987896099974768
```

[ ]:
```

```

Chest_linearx5_Adadel_mae_Final

August 16, 2019

1 Chest data

Librerias:

[2]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Chest

Cargamos los datos:

[3]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A2l=data['label']
A2a=27*(np.ones((len(a),1)))
A2=np.concatenate((a,b,c,d,e,f,A2a),axis=1)
B1 = np.asmatrix(A2l)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>=4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A3l=data['label']
A3a=25*(np.ones((len(a),1)))
A3=np.concatenate((a,b,c,d,e,f,A3a),axis=1)
B2 = np.asmatrix(A3l)
B2=B2.T
MAT2=np.concatenate((B2,A3),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A4l=data['label']
A4a=35*(np.ones((len(a),1)))
A4=np.concatenate((a,b,c,d,e,f,A4a),axis=1)
B3 = np.asmatrix(A4l)
B3=B3.T
MAT3=np.concatenate((B3,A4),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A5l=data['label']
A5a=27*(np.ones((len(a),1)))
A5=np.concatenate((a,b,c,d,e,f,A5a),axis=1)
B4 = np.asmatrix(A5l)
B4=B4.T
MAT4=np.concatenate((B4,A5),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A6l=data['label']
A6a=28*(np.ones((len(a),1)))
A6=np.concatenate((a,b,c,d,e,f,A6a),axis=1)
B5 = np.asmatrix(A6l)
B5=B5.T
MAT5=np.concatenate((B5,A6),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A7l=data['label']
A7a=27*(np.ones((len(a),1)))
A7=np.concatenate((a,b,c,d,e,f,A7a),axis=1)
B6 = np.asmatrix(A7l)
B6=B6.T
MAT6=np.concatenate((B6,A7),axis=1)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A8l=data['label']
A8a=26*(np.ones((len(a),1)))
A8=np.concatenate((a,b,c,d,e,f,A8a),axis=1)
B7 = np.asmatrix(A8l)
B7=B7.T
MAT7 = np.concatenate((B7,A8),axis=1)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A9l=data['label']
A9a=28*(np.ones((len(a),1)))
A9=np.concatenate((a,b,c,d,e,f,A9a),axis=1)
B8 = np.asmatrix(A9l)
B8=B8.T
MAT8 = np.concatenate((B8,A9),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A10l=data['label']
A10a=26*(np.ones((len(a),1)))
A10=np.concatenate((a,b,c,d,e,f,A10a),axis=1)
B9 = np.asmatrix(A10l)
B9=B9.T
MAT9 = np.concatenate((B9,A10),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A11l=data['label']
A11a=28*(np.ones((len(a),1)))
A11=np.concatenate((a,b,c,d,e,f,A11a),axis=1)
B10 = np.asmatrix(A11l)
B10=B10.T
MAT10 = np.concatenate((B10,A11),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A13l=data['label']
A13a=24*(np.ones((len(a),1)))
A13=np.concatenate((a,b,c,d,e,f,A13a),axis=1)
B11 = np.asmatrix(A13l)
B11=B11.T
MAT11 = np.concatenate((B11,A13),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A14l=data['label']
A14a=24*(np.ones((len(a),1)))
A14=np.concatenate((a,b,c,d,e,f,A14a),axis=1)
B12 = np.asmatrix(A14l)
B12=B12.T
MAT12 = np.concatenate((B12,A14),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A15l=data['label']
A15a=29*(np.ones((len(a),1)))
A15=np.concatenate((a,b,c,d,e,f,A15a),axis=1)
B13 = np.asmatrix(A15l)
B13=B13.T
MAT13 = np.concatenate((B13,A15),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A16l=data['label']
A16a=29*(np.ones((len(a),1)))
A16=np.concatenate((a,b,c,d,e,f,A16a),axis=1)
B14 = np.asmatrix(A16l)
B14=B14.T
MAT14 = np.concatenate((B14,A16),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>4), 0)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[17]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A17l=data['label']
A17a=29*(np.ones((len(a),1)))
A17=np.concatenate((a,b,c,d,e,f,A17a),axis=1)
B15 = np.asmatrix(A17l)
B15=B15.T
MAT15 = np.concatenate((B15,A17),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[18]:
```
MAT=np.
↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Comprobamos su correlación:

[19]:
```
data=pandas.DataFrame(MAT)
corr = data.corr()
corr.style.background_gradient(cmap='coolwarm')
```

[19]:
```
<pandas.io.formats.style.Styler at 0x2b10db7ab00>
```

Dividimos en los conjuntos de validación y entrenamiento:

[20]:
```
X = MAT[:, 1:10]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
```

```
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)

#scaler = StandardScaler()
#X_train = scaler.fit_transform(X_train)
#X_val = scaler.fit_transform(X_val)
```

Definimos la arquitectura de la red:

[35]:
```
model = Sequential([
    Dense(128, input_shape=(9,)),
    Activation('linear'),
    Dense(70),
    Activation('linear'),
    Dense(48),
    Activation('linear'),
    Dense(24),
    Activation('linear'),
    Dense(1),
    Activation('linear'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

Entrenamos la red:

[36]:
```
NUM_EPOCHS =5
BATCH_SIZE = 200

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.1)
```

```
Train on 22271762 samples, validate on 2474641 samples
Epoch 1/5
22271762/22271762 [==============================] - 359s 16us/step - loss:
0.9948 - val_loss: 0.9842
Epoch 2/5
22271762/22271762 [==============================] - 357s 16us/step - loss:
0.9873 - val_loss: 0.9841
Epoch 3/5
22271762/22271762 [==============================] - 356s 16us/step - loss:
0.9866 - val_loss: 0.9852
Epoch 4/5
22271762/22271762 [==============================] - 356s 16us/step - loss:
0.9863 - val_loss: 0.9839
Epoch 5/5
22271762/22271762 [==============================] - 355s 16us/step - loss:
0.9860 - val_loss: 0.9840
```

[37]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[38]:
```
Y_test =np.round(Y_test, 0)
```

Mostramos una parte de los resultados:

[39]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

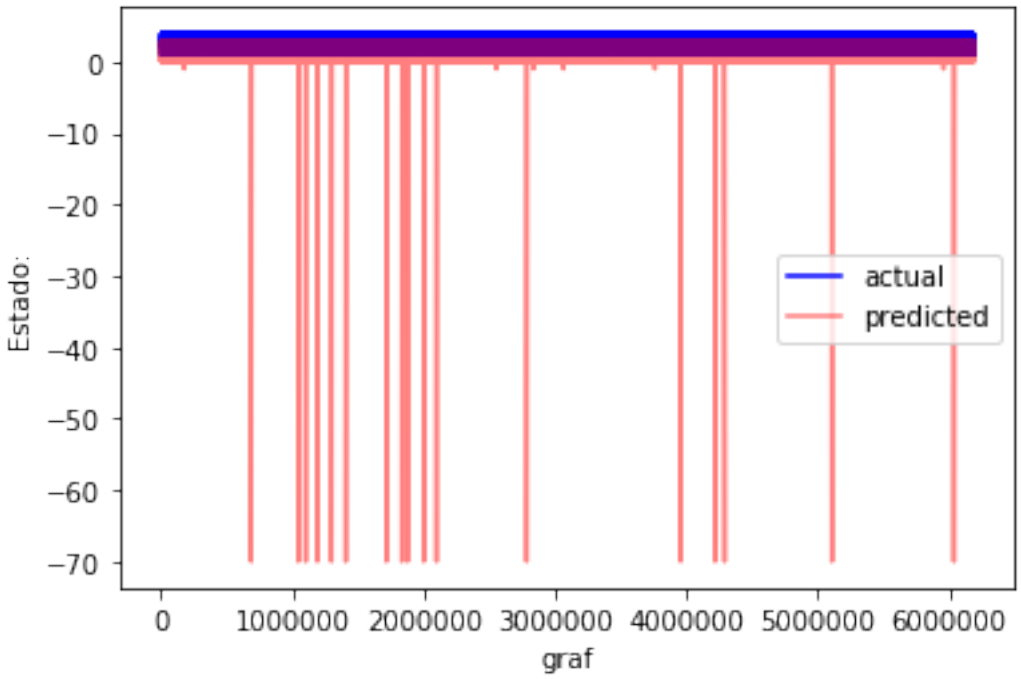
```
Estado: 1., predicted:2.
Estado: 4., predicted:2.
Estado: 4., predicted:2.
Estado: 3., predicted:2.
Estado: 4., predicted:2.
Estado: 3., predicted:2.
Estado: 1., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 4., predicted:2.
Estado: 4., predicted:3.
Estado: 1., predicted:2.
Estado: 2., predicted:3.
Estado: 1., predicted:1.
Estado: 4., predicted:2.
Estado: 1., predicted:2.
Estado: 4., predicted:3.
Estado: 2., predicted:2.
Estado: 4., predicted:3.
Estado: 1., predicted:2.
Estado: 4., predicted:2.
Estado: 1., predicted:2.
Estado: 3., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 4., predicted:3.
Estado: 1., predicted:3.
Estado: 1., predicted:2.
```

Evaluamos los resultados:

[40]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
-4.4242401679490415
```

[41]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[42]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[42]: 0.27255855032513004

[ ]:

Chest_elux5_Nadam_mae_Final

August 16, 2019

1 Chest data

Librerias:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import sys
import os
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Chest

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A2l=data['label']
A2a=27*(np.ones((len(a),1)))
A2=np.concatenate((a,b,c,d,e,f,A2a),axis=1)
B1 = np.asmatrix(A2l)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
```

```
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>=4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A3l=data['label']
A3a=25*(np.ones((len(a),1)))
A3=np.concatenate((a,b,c,d,e,f,A3a),axis=1)
B2 = np.asmatrix(A3l)
B2=B2.T
MAT2=np.concatenate((B2,A3),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A4l=data['label']
A4a=35*(np.ones((len(a),1)))
A4=np.concatenate((a,b,c,d,e,f,A4a),axis=1)
B3 = np.asmatrix(A4l)
B3=B3.T
MAT3=np.concatenate((B3,A4),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A5l=data['label']
A5a=27*(np.ones((len(a),1)))
A5=np.concatenate((a,b,c,d,e,f,A5a),axis=1)
B4 = np.asmatrix(A5l)
B4=B4.T
MAT4=np.concatenate((B4,A5),axis=1)
```

```
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A6l=data['label']
A6a=28*(np.ones((len(a),1)))
A6=np.concatenate((a,b,c,d,e,f,A6a),axis=1)
B5 = np.asmatrix(A6l)
B5=B5.T
MAT5=np.concatenate((B5,A6),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A7l=data['label']
A7a=27*(np.ones((len(a),1)))
A7=np.concatenate((a,b,c,d,e,f,A7a),axis=1)
B6 = np.asmatrix(A7l)
B6=B6.T
MAT6=np.concatenate((B6,A7),axis=1)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A8l=data['label']
A8a=26*(np.ones((len(a),1)))
A8=np.concatenate((a,b,c,d,e,f,A8a),axis=1)
B7 = np.asmatrix(A8l)
B7=B7.T
MAT7 = np.concatenate((B7,A8),axis=1)
```

```
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A9l=data['label']
A9a=28*(np.ones((len(a),1)))
A9=np.concatenate((a,b,c,d,e,f,A9a),axis=1)
B8 = np.asmatrix(A9l)
B8=B8.T
MAT8 = np.concatenate((B8,A9),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A10l=data['label']
A10a=26*(np.ones((len(a),1)))
A10=np.concatenate((a,b,c,d,e,f,A10a),axis=1)
B9 = np.asmatrix(A10l)
B9=B9.T
MAT9 = np.concatenate((B9,A10),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A11l=data['label']
A11a=28*(np.ones((len(a),1)))
A11=np.concatenate((a,b,c,d,e,f,A11a),axis=1)
B10 = np.asmatrix(A11l)
B10=B10.T
MAT10 = np.concatenate((B10,A11),axis=1)
```

```
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A13l=data['label']
A13a=24*(np.ones((len(a),1)))
A13=np.concatenate((a,b,c,d,e,f,A13a),axis=1)
B11 = np.asmatrix(A13l)
B11=B11.T
MAT11 = np.concatenate((B11,A13),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A14l=data['label']
A14a=24*(np.ones((len(a),1)))
A14=np.concatenate((a,b,c,d,e,f,A14a),axis=1)
B12 = np.asmatrix(A14l)
B12=B12.T
MAT12 = np.concatenate((B12,A14),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A15l=data['label']
A15a=29*(np.ones((len(a),1)))
A15=np.concatenate((a,b,c,d,e,f,A15a),axis=1)
B13 = np.asmatrix(A15l)
B13=B13.T
MAT13 = np.concatenate((B13,A15),axis=1)
```

```
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A16l=data['label']
A16a=29*(np.ones((len(a),1)))
A16=np.concatenate((a,b,c,d,e,f,A16a),axis=1)
B14 = np.asmatrix(A16l)
B14=B14.T
MAT14 = np.concatenate((B14,A16),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>4), 0)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A17l=data['label']
A17a=29*(np.ones((len(a),1)))
A17=np.concatenate((a,b,c,d,e,f,A17a),axis=1)
B15 = np.asmatrix(A17l)
B15=B15.T
MAT15 = np.concatenate((B15,A17),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Comprobamos su correlación:

[18]:
```
data=pandas.DataFrame(MAT)
corr = data.corr()
corr.style.background_gradient(cmap='coolwarm')
```

[18]:
```
<pandas.io.formats.style.Styler at 0x146a47da358>
```

Dividimos en los conjuntos de validación y entrenamiento:

[19]:
```
X = MAT[:, 1:10]
Y = MAT[:, 0]
```

```
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)

#scaler = StandardScaler()
#X_train = scaler.fit_transform(X_train)
#X_val = scaler.fit_transform(X_val)
```

Definimos la arquitectura de la red:

[20]:
```
model = Sequential([
    Dense(128, input_shape=(9,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
    Dense(48),
    Activation('elu'),
    Dense(24),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Nadam',loss='mae')
```

Entrenamos la red:

[21]:
```
NUM_EPOCHS =5
BATCH_SIZE = 200

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.1)
```

```
Train on 22271762 samples, validate on 2474641 samples
Epoch 1/5
22271762/22271762 [==============================] - 386s 17us/step - loss:
0.0701 - val_loss: 0.0450
Epoch 2/5
22271762/22271762 [==============================] - 381s 17us/step - loss:
0.0261 - val_loss: 0.0210
Epoch 3/5
22271762/22271762 [==============================] - 381s 17us/step - loss:
0.0224 - val_loss: 0.0187
Epoch 4/5
22271762/22271762 [==============================] - 382s 17us/step - loss:
0.0208 - val_loss: 0.0391
Epoch 5/5
22271762/22271762 [==============================] - 382s 17us/step - loss:
0.0200 - val_loss: 0.0152
```

[22]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[23]:
```
Y_test =np.round(Y_test, 0)
```

Mostramos una parte de los resultados:

[24]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

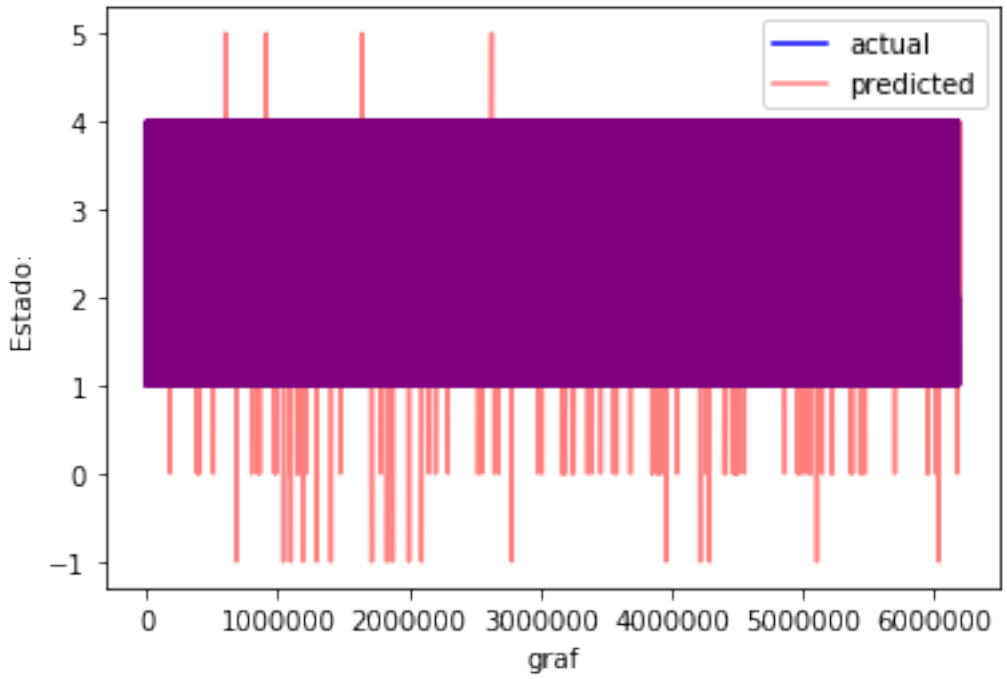
```
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 4., predicted:4.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

Evaluamos los resultados:

[25]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.9899588276899651
```

[26]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[27]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[27]:
```
0.9928455059571484
```

[ ]:
```
```

Chest_elux5_Rmsprop_mae_Final

August 16, 2019

1 Chest data

Librerias:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import sys
import os
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Chest

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A2l=data['label']
A2a=27*(np.ones((len(a),1)))
A2=np.concatenate((a,b,c,d,e,f,A2a),axis=1)
B1 = np.asmatrix(A2l)
B1=B1.T
```

```
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>=4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A3l=data['label']
A3a=25*(np.ones((len(a),1)))
A3=np.concatenate((a,b,c,d,e,f,A3a),axis=1)
B2 = np.asmatrix(A3l)
B2=B2.T
MAT2=np.concatenate((B2,A3),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A4l=data['label']
A4a=35*(np.ones((len(a),1)))
A4=np.concatenate((a,b,c,d,e,f,A4a),axis=1)
B3 = np.asmatrix(A4l)
B3=B3.T
MAT3=np.concatenate((B3,A4),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A5l=data['label']
A5a=27*(np.ones((len(a),1)))
A5=np.concatenate((a,b,c,d,e,f,A5a),axis=1)
B4 = np.asmatrix(A5l)
B4=B4.T
```

```
MAT4=np.concatenate((B4,A5),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A6l=data['label']
A6a=28*(np.ones((len(a),1)))
A6=np.concatenate((a,b,c,d,e,f,A6a),axis=1)
B5 = np.asmatrix(A6l)
B5=B5.T
MAT5=np.concatenate((B5,A6),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A7l=data['label']
A7a=27*(np.ones((len(a),1)))
A7=np.concatenate((a,b,c,d,e,f,A7a),axis=1)
B6 = np.asmatrix(A7l)
B6=B6.T
MAT6=np.concatenate((B6,A7),axis=1)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A8l=data['label']
A8a=26*(np.ones((len(a),1)))
A8=np.concatenate((a,b,c,d,e,f,A8a),axis=1)
B7 = np.asmatrix(A8l)
B7=B7.T
```

```
MAT7 = np.concatenate((B7,A8),axis=1)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A9l=data['label']
A9a=28*(np.ones((len(a),1)))
A9=np.concatenate((a,b,c,d,e,f,A9a),axis=1)
B8 = np.asmatrix(A9l)
B8=B8.T
MAT8 = np.concatenate((B8,A9),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A10l=data['label']
A10a=26*(np.ones((len(a),1)))
A10=np.concatenate((a,b,c,d,e,f,A10a),axis=1)
B9 = np.asmatrix(A10l)
B9=B9.T
MAT9 = np.concatenate((B9,A10),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A11l=data['label']
A11a=28*(np.ones((len(a),1)))
A11=np.concatenate((a,b,c,d,e,f,A11a),axis=1)
B10 = np.asmatrix(A11l)
B10=B10.T
```

```
MAT10 = np.concatenate((B10,A11),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A13l=data['label']
A13a=24*(np.ones((len(a),1)))
A13=np.concatenate((a,b,c,d,e,f,A13a),axis=1)
B11 = np.asmatrix(A13l)
B11=B11.T
MAT11 = np.concatenate((B11,A13),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A14l=data['label']
A14a=24*(np.ones((len(a),1)))
A14=np.concatenate((a,b,c,d,e,f,A14a),axis=1)
B12 = np.asmatrix(A14l)
B12=B12.T
MAT12 = np.concatenate((B12,A14),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A15l=data['label']
A15a=29*(np.ones((len(a),1)))
A15=np.concatenate((a,b,c,d,e,f,A15a),axis=1)
B13 = np.asmatrix(A15l)
B13=B13.T
```

```
MAT13 = np.concatenate((B13,A15),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A16l=data['label']
A16a=29*(np.ones((len(a),1)))
A16=np.concatenate((a,b,c,d,e,f,A16a),axis=1)
B14 = np.asmatrix(A16l)
B14=B14.T
MAT14 = np.concatenate((B14,A16),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>4), 0)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A17l=data['label']
A17a=29*(np.ones((len(a),1)))
A17=np.concatenate((a,b,c,d,e,f,A17a),axis=1)
B15 = np.asmatrix(A17l)
B15=B15.T
MAT15 = np.concatenate((B15,A17),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Comprobamos su correlación:

[18]:
```
data=pandas.DataFrame(MAT)
corr = data.corr()
corr.style.background_gradient(cmap='coolwarm')
```

[18]:
```
<pandas.io.formats.style.Styler at 0x266292a8cc0>
```

Dividimos en los conjuntos de validación y entrenamiento:

[19]:
```
X = MAT[:, 1:10]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)

#scaler = StandardScaler()
#X_train = scaler.fit_transform(X_train)
#X_val = scaler.fit_transform(X_val)
```

Definimos la arquitectura de la red:

[20]:
```
model = Sequential([
    Dense(128, input_shape=(9,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
    Dense(48),
    Activation('elu'),
    Dense(24),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Rmsprop',loss='mae')
```

Entrenamos la red:

[21]:
```
NUM_EPOCHS =5
BATCH_SIZE = 200

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.1)
```

```
Train on 22271762 samples, validate on 2474641 samples
Epoch 1/5
22271762/22271762 [==============================] - 308s 14us/step - loss:
0.1342 - val_loss: 0.0470
Epoch 2/5
22271762/22271762 [==============================] - 303s 14us/step - loss:
0.0492 - val_loss: 0.0521
Epoch 3/5
22271762/22271762 [==============================] - 304s 14us/step - loss:
0.0410 - val_loss: 0.0373
Epoch 4/5
22271762/22271762 [==============================] - 302s 14us/step - loss:
0.0389 - val_loss: 0.0327
Epoch 5/5
```

```
22271762/22271762 [==============================] - 300s 13us/step - loss:
0.0383 - val_loss: 0.0287
```

[22]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[23]:
```
Y_test =np.round(Y_test, 0)
```

Mostramos una parte de los resultados:

[24]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

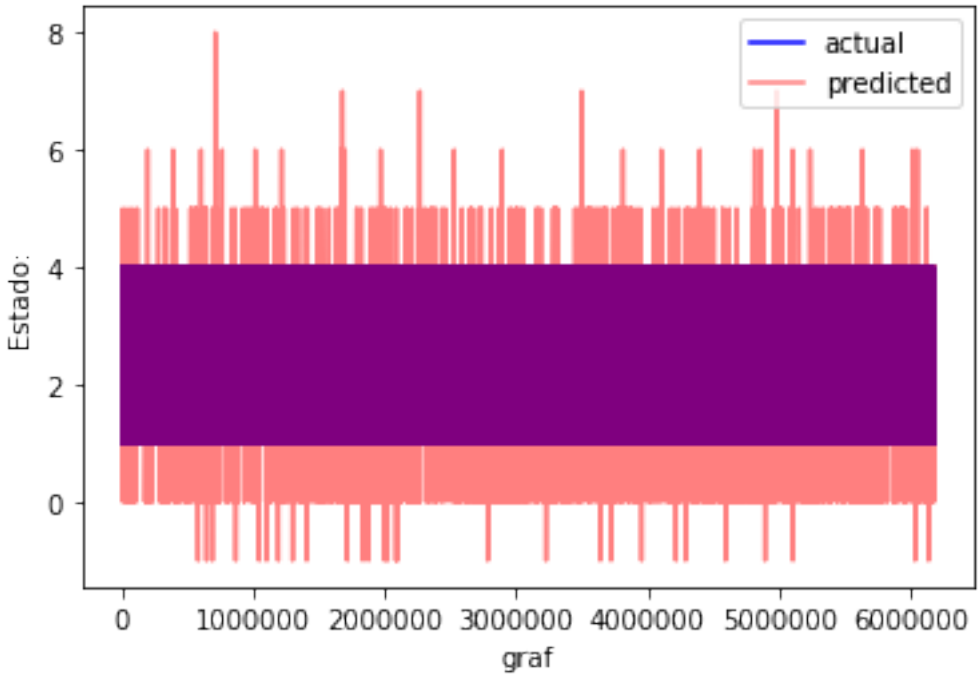
```
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 4., predicted:4.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

Evaluamos los resultados:

[25]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.9853921976325933
```

[26]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[27]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[27]:
```
0.9894395969612393
```

[ ]:
```
```

Chest_elux5_Adadel_msle_Final

August 19, 2019

1 Chest data

Librerias:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import sys
import os
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Chest

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A2l=data['label']
A2a=27*(np.ones((len(a),1)))
A2=np.concatenate((a,b,c,d,e,f,A2a),axis=1)
B1 = np.asmatrix(A2l)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
```

```
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>=4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A3l=data['label']
A3a=25*(np.ones((len(a),1)))
A3=np.concatenate((a,b,c,d,e,f,A3a),axis=1)
B2 = np.asmatrix(A3l)
B2=B2.T
MAT2=np.concatenate((B2,A3),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A4l=data['label']
A4a=35*(np.ones((len(a),1)))
A4=np.concatenate((a,b,c,d,e,f,A4a),axis=1)
B3 = np.asmatrix(A4l)
B3=B3.T
MAT3=np.concatenate((B3,A4),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A5l=data['label']
A5a=27*(np.ones((len(a),1)))
A5=np.concatenate((a,b,c,d,e,f,A5a),axis=1)
B4 = np.asmatrix(A5l)
B4=B4.T
MAT4=np.concatenate((B4,A5),axis=1)
```

```
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A6l=data['label']
A6a=28*(np.ones((len(a),1)))
A6=np.concatenate((a,b,c,d,e,f,A6a),axis=1)
B5 = np.asmatrix(A6l)
B5=B5.T
MAT5=np.concatenate((B5,A6),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A7l=data['label']
A7a=27*(np.ones((len(a),1)))
A7=np.concatenate((a,b,c,d,e,f,A7a),axis=1)
B6 = np.asmatrix(A7l)
B6=B6.T
MAT6=np.concatenate((B6,A7),axis=1)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A8l=data['label']
A8a=26*(np.ones((len(a),1)))
A8=np.concatenate((a,b,c,d,e,f,A8a),axis=1)
B7 = np.asmatrix(A8l)
B7=B7.T
MAT7 = np.concatenate((B7,A8),axis=1)
```

```
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A9l=data['label']
A9a=28*(np.ones((len(a),1)))
A9=np.concatenate((a,b,c,d,e,f,A9a),axis=1)
B8 = np.asmatrix(A9l)
B8=B8.T
MAT8 = np.concatenate((B8,A9),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A10l=data['label']
A10a=26*(np.ones((len(a),1)))
A10=np.concatenate((a,b,c,d,e,f,A10a),axis=1)
B9 = np.asmatrix(A10l)
B9=B9.T
MAT9 = np.concatenate((B9,A10),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A11l=data['label']
A11a=28*(np.ones((len(a),1)))
A11=np.concatenate((a,b,c,d,e,f,A11a),axis=1)
B10 = np.asmatrix(A11l)
B10=B10.T
MAT10 = np.concatenate((B10,A11),axis=1)
```

```
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A13l=data['label']
A13a=24*(np.ones((len(a),1)))
A13=np.concatenate((a,b,c,d,e,f,A13a),axis=1)
B11 = np.asmatrix(A13l)
B11=B11.T
MAT11 = np.concatenate((B11,A13),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A14l=data['label']
A14a=24*(np.ones((len(a),1)))
A14=np.concatenate((a,b,c,d,e,f,A14a),axis=1)
B12 = np.asmatrix(A14l)
B12=B12.T
MAT12 = np.concatenate((B12,A14),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A15l=data['label']
A15a=29*(np.ones((len(a),1)))
A15=np.concatenate((a,b,c,d,e,f,A15a),axis=1)
B13 = np.asmatrix(A15l)
B13=B13.T
MAT13 = np.concatenate((B13,A15),axis=1)
```

```
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A16l=data['label']
A16a=29*(np.ones((len(a),1)))
A16=np.concatenate((a,b,c,d,e,f,A16a),axis=1)
B14 = np.asmatrix(A16l)
B14=B14.T
MAT14 = np.concatenate((B14,A16),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>4), 0)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A17l=data['label']
A17a=29*(np.ones((len(a),1)))
A17=np.concatenate((a,b,c,d,e,f,A17a),axis=1)
B15 = np.asmatrix(A17l)
B15=B15.T
MAT15 = np.concatenate((B15,A17),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Comprobamos su correlación:

[18]:
```
data=pandas.DataFrame(MAT)
corr = data.corr()
corr.style.background_gradient(cmap='coolwarm')
```

[18]:
```
<pandas.io.formats.style.Styler at 0x1faeb361a58>
```

Dividimos en los conjuntos de validación y entrenamiento:

[19]:
```
X = MAT[:, 1:10]
Y = MAT[:, 0]
```

```
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)

#scaler = StandardScaler()
#X_train = scaler.fit_transform(X_train)
#X_val = scaler.fit_transform(X_val)
```

Definimos la arquitectura de la red:

[20]:
```
model = Sequential([
    Dense(128, input_shape=(9,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
    Dense(48),
    Activation('elu'),
    Dense(24),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='msle')
```

Entrenamos la red:

[21]:
```
NUM_EPOCHS =5
BATCH_SIZE = 200

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.1)
```

```
Train on 22271762 samples, validate on 2474641 samples
Epoch 1/5
22271762/22271762 [==============================] - 367s 16us/step - loss:
0.0117 - val_loss: 0.0020
Epoch 2/5
22271762/22271762 [==============================] - 362s 16us/step - loss:
0.0027 - val_loss: 0.0128
Epoch 3/5
22271762/22271762 [==============================] - 365s 16us/step - loss:
0.0021 - val_loss: 0.0037
Epoch 4/5
22271762/22271762 [==============================] - 363s 16us/step - loss:
0.0018 - val_loss: 0.0021
Epoch 5/5
22271762/22271762 [==============================] - 364s 16us/step - loss:
0.0015 - val_loss: 8.3507e-04
```

[22]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[23]:
```
Y_test =np.round(Y_test, 0)
```

Mostramos una parte de los resultados:

[24]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

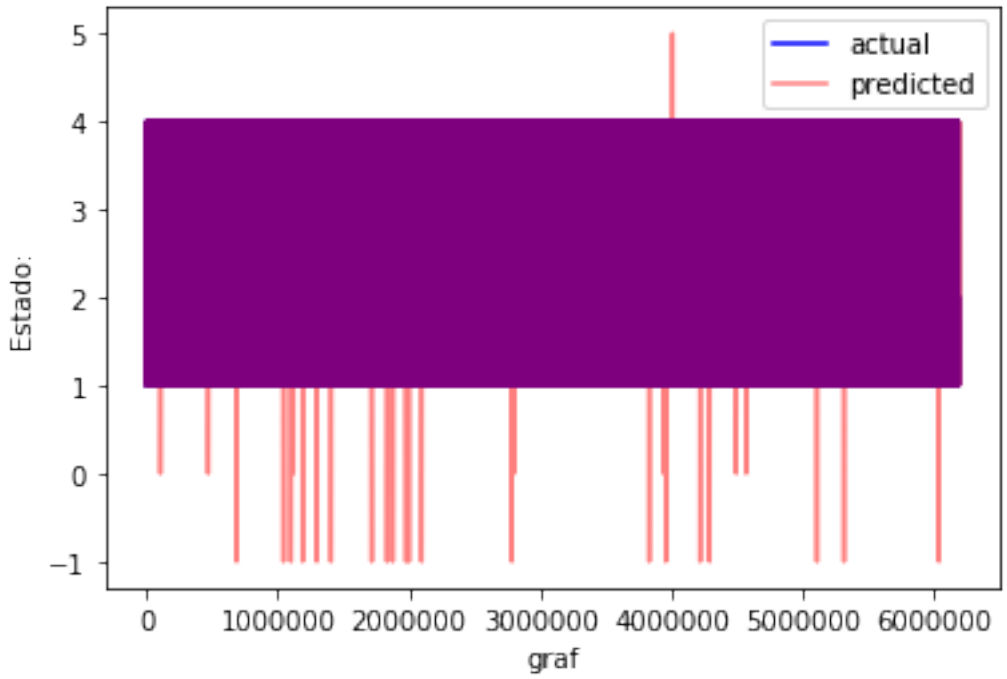
```
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 4., predicted:4.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

Evaluamos los resultados:

[25]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.9928146995809906
```

[26]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[27]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[27]:
```
0.9928419498849207
```

[ ]:
```
```

Chest_elux5_Adadel_hinge_Final

August 19, 2019

1 Chest data

Librerias:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import sys
import os
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Chest

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A2l=data['label']
A2a=27*(np.ones((len(a),1)))
A2=np.concatenate((a,b,c,d,e,f,A2a),axis=1)
B1 = np.asmatrix(A2l)
B1=B1.T
```

```
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>=4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A3l=data['label']
A3a=25*(np.ones((len(a),1)))
A3=np.concatenate((a,b,c,d,e,f,A3a),axis=1)
B2 = np.asmatrix(A3l)
B2=B2.T
MAT2=np.concatenate((B2,A3),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A4l=data['label']
A4a=35*(np.ones((len(a),1)))
A4=np.concatenate((a,b,c,d,e,f,A4a),axis=1)
B3 = np.asmatrix(A4l)
B3=B3.T
MAT3=np.concatenate((B3,A4),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A5l=data['label']
A5a=27*(np.ones((len(a),1)))
A5=np.concatenate((a,b,c,d,e,f,A5a),axis=1)
B4 = np.asmatrix(A5l)
B4=B4.T
```

```
MAT4=np.concatenate((B4,A5),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A6l=data['label']
A6a=28*(np.ones((len(a),1)))
A6=np.concatenate((a,b,c,d,e,f,A6a),axis=1)
B5 = np.asmatrix(A6l)
B5=B5.T
MAT5=np.concatenate((B5,A6),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A7l=data['label']
A7a=27*(np.ones((len(a),1)))
A7=np.concatenate((a,b,c,d,e,f,A7a),axis=1)
B6 = np.asmatrix(A7l)
B6=B6.T
MAT6=np.concatenate((B6,A7),axis=1)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A8l=data['label']
A8a=26*(np.ones((len(a),1)))
A8=np.concatenate((a,b,c,d,e,f,A8a),axis=1)
B7 = np.asmatrix(A8l)
B7=B7.T
```

```
MAT7 = np.concatenate((B7,A8),axis=1)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A9l=data['label']
A9a=28*(np.ones((len(a),1)))
A9=np.concatenate((a,b,c,d,e,f,A9a),axis=1)
B8 = np.asmatrix(A9l)
B8=B8.T
MAT8 = np.concatenate((B8,A9),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A10l=data['label']
A10a=26*(np.ones((len(a),1)))
A10=np.concatenate((a,b,c,d,e,f,A10a),axis=1)
B9 = np.asmatrix(A10l)
B9=B9.T
MAT9 = np.concatenate((B9,A10),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A11l=data['label']
A11a=28*(np.ones((len(a),1)))
A11=np.concatenate((a,b,c,d,e,f,A11a),axis=1)
B10 = np.asmatrix(A11l)
B10=B10.T
```

```
MAT10 = np.concatenate((B10,A11),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A13l=data['label']
A13a=24*(np.ones((len(a),1)))
A13=np.concatenate((a,b,c,d,e,f,A13a),axis=1)
B11 = np.asmatrix(A13l)
B11=B11.T
MAT11 = np.concatenate((B11,A13),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A14l=data['label']
A14a=24*(np.ones((len(a),1)))
A14=np.concatenate((a,b,c,d,e,f,A14a),axis=1)
B12 = np.asmatrix(A14l)
B12=B12.T
MAT12 = np.concatenate((B12,A14),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A15l=data['label']
A15a=29*(np.ones((len(a),1)))
A15=np.concatenate((a,b,c,d,e,f,A15a),axis=1)
B13 = np.asmatrix(A15l)
B13=B13.T
```

```
MAT13 = np.concatenate((B13,A15),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A16l=data['label']
A16a=29*(np.ones((len(a),1)))
A16=np.concatenate((a,b,c,d,e,f,A16a),axis=1)
B14 = np.asmatrix(A16l)
B14=B14.T
MAT14 = np.concatenate((B14,A16),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>4), 0)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A17l=data['label']
A17a=29*(np.ones((len(a),1)))
A17=np.concatenate((a,b,c,d,e,f,A17a),axis=1)
B15 = np.asmatrix(A17l)
B15=B15.T
MAT15 = np.concatenate((B15,A17),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Comprobamos su correlación:

[18]:
```
data=pandas.DataFrame(MAT)
corr = data.corr()
corr.style.background_gradient(cmap='coolwarm')
```

[18]:
```
<pandas.io.formats.style.Styler at 0x1921d7b8ac8>
```

Dividimos en los conjuntos de validación y entrenamiento:

[19]:
```
X = MAT[:, 1:10]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)

#scaler = StandardScaler()
#X_train = scaler.fit_transform(X_train)
#X_val = scaler.fit_transform(X_val)
```

Definimos la arquitectura de la red:

[20]:
```
model = Sequential([
    Dense(128, input_shape=(9,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
    Dense(48),
    Activation('elu'),
    Dense(24),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='hinge')
```

Entrenamos la red:

[21]:
```
NUM_EPOCHS =5
BATCH_SIZE = 200

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.1)
```

```
Train on 22271762 samples, validate on 2474641 samples
Epoch 1/5
22271762/22271762 [==============================] - 358s 16us/step - loss:
3.3479e-05 - val_loss: 9.6984e-06
Epoch 2/5
22271762/22271762 [==============================] - 354s 16us/step - loss:
7.8126e-06 - val_loss: 9.6984e-06
Epoch 3/5
22271762/22271762 [==============================] - 354s 16us/step - loss:
7.8126e-06 - val_loss: 9.6984e-06
Epoch 4/5
22271762/22271762 [==============================] - 355s 16us/step - loss:
7.8126e-06 - val_loss: 9.6984e-06
Epoch 5/5
```

```
22271762/22271762 [==============================] - 376s 17us/step - loss:
7.8126e-06 - val_loss: 9.6984e-06
```

[22]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[23]:
```
Y_test =np.round(Y_test, 0)
```

Mostramos una parte de los resultados:

[24]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

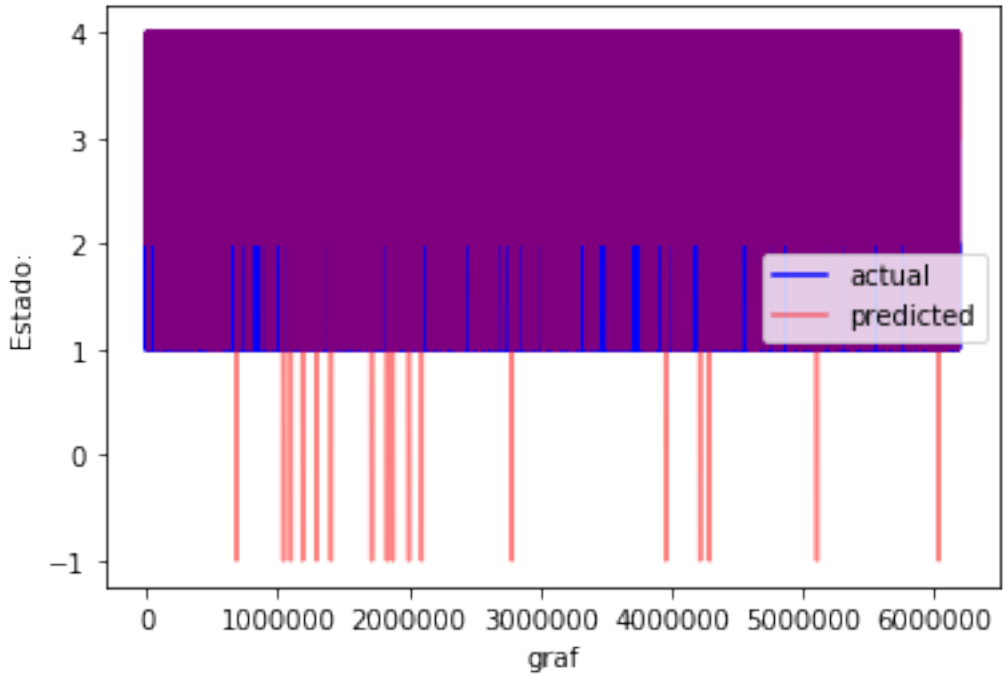
```
Estado: 1., predicted:2.
Estado: 4., predicted:3.
Estado: 4., predicted:4.
Estado: 3., predicted:3.
Estado: 4., predicted:3.
Estado: 3., predicted:2.
Estado: 1., predicted:3.
Estado: 2., predicted:3.
Estado: 2., predicted:3.
Estado: 1., predicted:3.
Estado: 3., predicted:3.
Estado: 4., predicted:3.
Estado: 4., predicted:3.
Estado: 1., predicted:4.
Estado: 2., predicted:3.
Estado: 1., predicted:3.
Estado: 4., predicted:3.
Estado: 1., predicted:3.
Estado: 4., predicted:3.
Estado: 2., predicted:3.
Estado: 4., predicted:3.
Estado: 1., predicted:2.
Estado: 4., predicted:3.
Estado: 1., predicted:3.
Estado: 3., predicted:3.
Estado: 2., predicted:3.
Estado: 2., predicted:3.
Estado: 4., predicted:3.
Estado: 1., predicted:3.
Estado: 1., predicted:3.
```

Evaluamos los resultados:

[25]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
-7.246078264733157
```

[26]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[27]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[27]:
```
0.1443957675628346
```

[ ]:
```

```

Chest_elux7_Adadel_mae_Final

August 13, 2019

1 Chest data

Librerias:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import sys
import os
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Chest

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A2l=data['label']
A2a=27*(np.ones((len(a),1)))
A2=np.concatenate((a,b,c,d,e,f,A2a),axis=1)
B1 = np.asmatrix(A2l)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
```

```
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>=4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A3l=data['label']
A3a=25*(np.ones((len(a),1)))
A3=np.concatenate((a,b,c,d,e,f,A3a),axis=1)
B2 = np.asmatrix(A3l)
B2=B2.T
MAT2=np.concatenate((B2,A3),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A4l=data['label']
A4a=35*(np.ones((len(a),1)))
A4=np.concatenate((a,b,c,d,e,f,A4a),axis=1)
B3 = np.asmatrix(A4l)
B3=B3.T
MAT3=np.concatenate((B3,A4),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A5l=data['label']
A5a=27*(np.ones((len(a),1)))
A5=np.concatenate((a,b,c,d,e,f,A5a),axis=1)
B4 = np.asmatrix(A5l)
B4=B4.T
MAT4=np.concatenate((B4,A5),axis=1)
```

```
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A6l=data['label']
A6a=28*(np.ones((len(a),1)))
A6=np.concatenate((a,b,c,d,e,f,A6a),axis=1)
B5 = np.asmatrix(A6l)
B5=B5.T
MAT5=np.concatenate((B5,A6),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A7l=data['label']
A7a=27*(np.ones((len(a),1)))
A7=np.concatenate((a,b,c,d,e,f,A7a),axis=1)
B6 = np.asmatrix(A7l)
B6=B6.T
MAT6=np.concatenate((B6,A7),axis=1)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A8l=data['label']
A8a=26*(np.ones((len(a),1)))
A8=np.concatenate((a,b,c,d,e,f,A8a),axis=1)
B7 = np.asmatrix(A8l)
B7=B7.T
MAT7 = np.concatenate((B7,A8),axis=1)
```

```
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A9l=data['label']
A9a=28*(np.ones((len(a),1)))
A9=np.concatenate((a,b,c,d,e,f,A9a),axis=1)
B8 = np.asmatrix(A9l)
B8=B8.T
MAT8 = np.concatenate((B8,A9),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A10l=data['label']
A10a=26*(np.ones((len(a),1)))
A10=np.concatenate((a,b,c,d,e,f,A10a),axis=1)
B9 = np.asmatrix(A10l)
B9=B9.T
MAT9 = np.concatenate((B9,A10),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A11l=data['label']
A11a=28*(np.ones((len(a),1)))
A11=np.concatenate((a,b,c,d,e,f,A11a),axis=1)
B10 = np.asmatrix(A11l)
B10=B10.T
MAT10 = np.concatenate((B10,A11),axis=1)
```

```
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A13l=data['label']
A13a=24*(np.ones((len(a),1)))
A13=np.concatenate((a,b,c,d,e,f,A13a),axis=1)
B11 = np.asmatrix(A13l)
B11=B11.T
MAT11 = np.concatenate((B11,A13),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A14l=data['label']
A14a=24*(np.ones((len(a),1)))
A14=np.concatenate((a,b,c,d,e,f,A14a),axis=1)
B12 = np.asmatrix(A14l)
B12=B12.T
MAT12 = np.concatenate((B12,A14),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A15l=data['label']
A15a=29*(np.ones((len(a),1)))
A15=np.concatenate((a,b,c,d,e,f,A15a),axis=1)
B13 = np.asmatrix(A15l)
B13=B13.T
MAT13 = np.concatenate((B13,A15),axis=1)
```

```
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A16l=data['label']
A16a=29*(np.ones((len(a),1)))
A16=np.concatenate((a,b,c,d,e,f,A16a),axis=1)
B14 = np.asmatrix(A16l)
B14=B14.T
MAT14 = np.concatenate((B14,A16),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>4), 0)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A17l=data['label']
A17a=29*(np.ones((len(a),1)))
A17=np.concatenate((a,b,c,d,e,f,A17a),axis=1)
B15 = np.asmatrix(A17l)
B15=B15.T
MAT15 = np.concatenate((B15,A17),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Comprobamos su correlación:

[18]:
```
data=pandas.DataFrame(MAT)
corr = data.corr()
corr.style.background_gradient(cmap='coolwarm')
```

[18]:
```
<pandas.io.formats.style.Styler at 0x264ffc03c18>
```

Dividimos en los conjuntos de validación y entrenamiento:

[19]:
```
X = MAT[:, 1:10]
Y = MAT[:, 0]
```

```
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)
#scaler = StandardScaler()
#X_train = scaler.fit_transform(X_train)
#X_val = scaler.fit_transform(X_val)
```

Definimos la arquitectura de la red:

[20]:
```
model = Sequential([
    Dense(128, input_shape=(9,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
    Dense(60),
    Activation('elu'),
    Dense(40),
    Activation('elu'),
    Dense(40),
    Activation('elu'),
    Dense(30),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

Entrenamos la red:

[21]:
```
NUM_EPOCHS =5
BATCH_SIZE = 200

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.1)
```

```
Train on 22271762 samples, validate on 2474641 samples
Epoch 1/5
22271762/22271762 [==============================] - 466s 21us/step - loss:
0.0954 - val_loss: 0.0439
Epoch 2/5
22271762/22271762 [==============================] - 458s 21us/step - loss:
0.0340 - val_loss: 0.0309
Epoch 3/5
22271762/22271762 [==============================] - 458s 21us/step - loss:
0.0296 - val_loss: 0.0266
Epoch 4/5
22271762/22271762 [==============================] - 461s 21us/step - loss:
0.0276 - val_loss: 0.0229
```

```
Epoch 5/5
22271762/22271762 [==============================] - 462s 21us/step - loss:
0.0262 - val_loss: 0.0216
```

[22]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados pùesto que deben de ser enteros:

[23]:
```
Y_test =np.round(Y_test, 0)
```

Mostramos una parte de los resultados:

[24]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

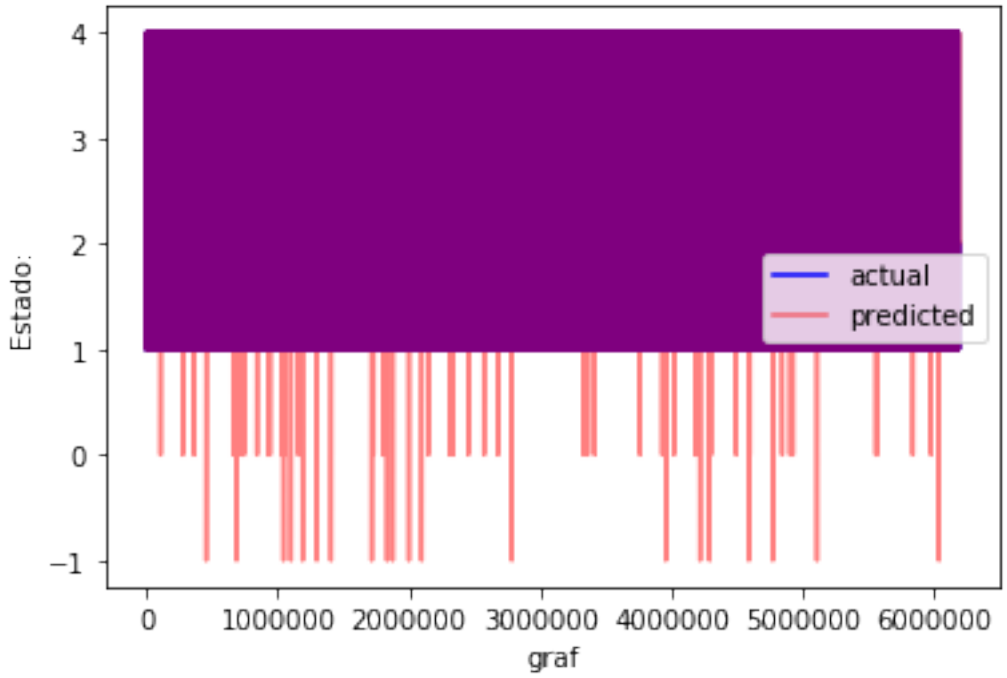
```
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 4., predicted:4.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

Evaluamos los resultados:

[25]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.9908086594460415
```

[26]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[27]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[27]:
```
0.9943419658064259
```

[ ]:
```
```

Chest_elux4_Adadel_mae_Final

August 13, 2019

1 Chest data

Librerias:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import sys
import os
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Chest

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A2l=data['label']
A2a=27*(np.ones((len(a),1)))
A2=np.concatenate((a,b,c,d,e,f,A2a),axis=1)
B1 = np.asmatrix(A2l)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
```

```
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>=4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A3l=data['label']
A3a=25*(np.ones((len(a),1)))
A3=np.concatenate((a,b,c,d,e,f,A3a),axis=1)
B2 = np.asmatrix(A3l)
B2=B2.T
MAT2=np.concatenate((B2,A3),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A4l=data['label']
A4a=35*(np.ones((len(a),1)))
A4=np.concatenate((a,b,c,d,e,f,A4a),axis=1)
B3 = np.asmatrix(A4l)
B3=B3.T
MAT3=np.concatenate((B3,A4),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A5l=data['label']
A5a=27*(np.ones((len(a),1)))
A5=np.concatenate((a,b,c,d,e,f,A5a),axis=1)
B4 = np.asmatrix(A5l)
B4=B4.T
MAT4=np.concatenate((B4,A5),axis=1)
```

```
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A6l=data['label']
A6a=28*(np.ones((len(a),1)))
A6=np.concatenate((a,b,c,d,e,f,A6a),axis=1)
B5 = np.asmatrix(A6l)
B5=B5.T
MAT5=np.concatenate((B5,A6),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A7l=data['label']
A7a=27*(np.ones((len(a),1)))
A7=np.concatenate((a,b,c,d,e,f,A7a),axis=1)
B6 = np.asmatrix(A7l)
B6=B6.T
MAT6=np.concatenate((B6,A7),axis=1)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A8l=data['label']
A8a=26*(np.ones((len(a),1)))
A8=np.concatenate((a,b,c,d,e,f,A8a),axis=1)
B7 = np.asmatrix(A8l)
B7=B7.T
MAT7 = np.concatenate((B7,A8),axis=1)
```

```
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A9l=data['label']
A9a=28*(np.ones((len(a),1)))
A9=np.concatenate((a,b,c,d,e,f,A9a),axis=1)
B8 = np.asmatrix(A9l)
B8=B8.T
MAT8 = np.concatenate((B8,A9),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A10l=data['label']
A10a=26*(np.ones((len(a),1)))
A10=np.concatenate((a,b,c,d,e,f,A10a),axis=1)
B9 = np.asmatrix(A10l)
B9=B9.T
MAT9 = np.concatenate((B9,A10),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A11l=data['label']
A11a=28*(np.ones((len(a),1)))
A11=np.concatenate((a,b,c,d,e,f,A11a),axis=1)
B10 = np.asmatrix(A11l)
B10=B10.T
MAT10 = np.concatenate((B10,A11),axis=1)
```

```
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A13l=data['label']
A13a=24*(np.ones((len(a),1)))
A13=np.concatenate((a,b,c,d,e,f,A13a),axis=1)
B11 = np.asmatrix(A13l)
B11=B11.T
MAT11 = np.concatenate((B11,A13),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A14l=data['label']
A14a=24*(np.ones((len(a),1)))
A14=np.concatenate((a,b,c,d,e,f,A14a),axis=1)
B12 = np.asmatrix(A14l)
B12=B12.T
MAT12 = np.concatenate((B12,A14),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A15l=data['label']
A15a=29*(np.ones((len(a),1)))
A15=np.concatenate((a,b,c,d,e,f,A15a),axis=1)
B13 = np.asmatrix(A15l)
B13=B13.T
MAT13 = np.concatenate((B13,A15),axis=1)
```

```
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A16l=data['label']
A16a=29*(np.ones((len(a),1)))
A16=np.concatenate((a,b,c,d,e,f,A16a),axis=1)
B14 = np.asmatrix(A16l)
B14=B14.T
MAT14 = np.concatenate((B14,A16),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>4), 0)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['chest']['ACC']
b=data['signal']['chest']['ECG']
c=data['signal']['chest']['EMG']
d=data['signal']['chest']['EDA']
e=data['signal']['chest']['Temp']
f=data['signal']['chest']['Resp']
A17l=data['label']
A17a=29*(np.ones((len(a),1)))
A17=np.concatenate((a,b,c,d,e,f,A17a),axis=1)
B15 = np.asmatrix(A17l)
B15=B15.T
MAT15 = np.concatenate((B15,A17),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Comprobamos su correlación:

[18]:
```
data=pandas.DataFrame(MAT)
corr = data.corr()
corr.style.background_gradient(cmap='coolwarm')
```

[18]:
```
<pandas.io.formats.style.Styler at 0x1ab9003dba8>
```

Dividimos en los conjuntos de validación y entrenamiento:

[22]:
```
X = MAT[:, 1:10]
Y = MAT[:, 0]
```

```
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)
#scaler = StandardScaler()
#X_train = scaler.fit_transform(X_train)
#X_val = scaler.fit_transform(X_val)
```

Definimos la arquitectura de la red:

[26]:
```
model = Sequential([
    Dense(128, input_shape=(9,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
    Dense(60),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

Entrenamos la red:

[27]:
```
NUM_EPOCHS =5
BATCH_SIZE = 200

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.1)
```

```
Train on 22271762 samples, validate on 2474641 samples
Epoch 1/5
22271762/22271762 [==============================] - 329s 15us/step - loss:
0.1802 - val_loss: 0.0869
Epoch 2/5
22271762/22271762 [==============================] - 330s 15us/step - loss:
0.0735 - val_loss: 0.0642
Epoch 3/5
22271762/22271762 [==============================] - 322s 14us/step - loss:
0.0602 - val_loss: 0.0464
Epoch 4/5
22271762/22271762 [==============================] - 316s 14us/step - loss:
0.0535 - val_loss: 0.0594
Epoch 5/5
22271762/22271762 [==============================] - 331s 15us/step - loss:
0.0488 - val_loss: 0.0485
```

[28]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[29]:
```
Y_test =np.round(Y_test, 0)
```

Mostramos una parte de los resultados:

[30]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

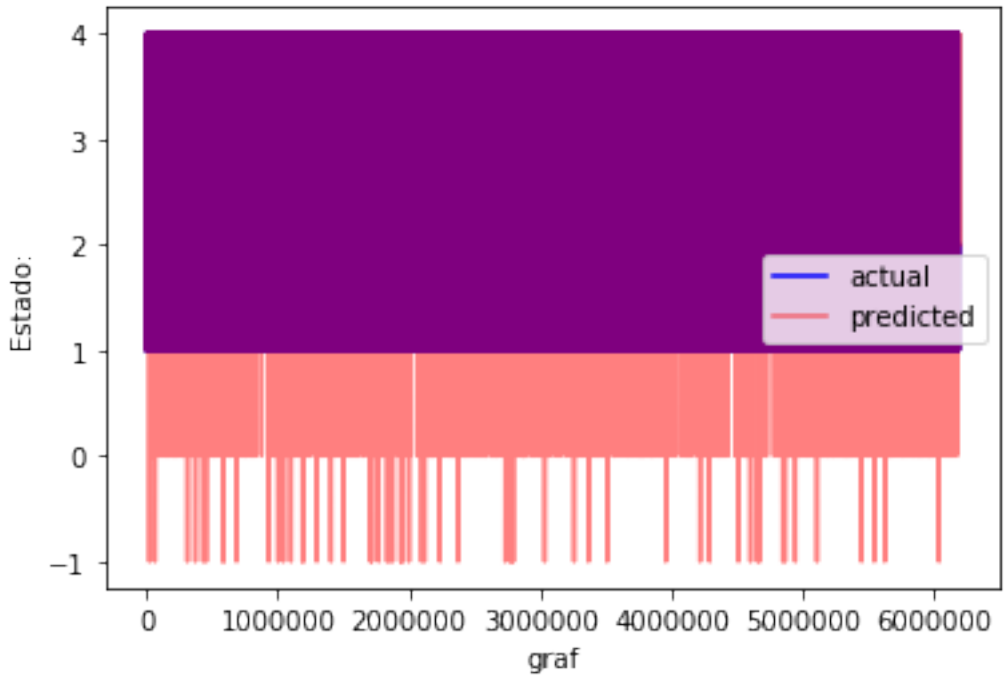
```
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 4., predicted:4.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

Evaluamos los resultados:

[31]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.9892761203009343
```

[32]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[33]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[33]:
```
0.9907550204061972
```

[ ]:
```

```

Wrist_elux4_mae_Adadel_final

August 30, 2019

1 Wrist Data

Librerías:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Wrist

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
```

```
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
```

```
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT2=np.concatenate((B1,A2),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
```

```
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=25*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT3=np.concatenate((B1,A2),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
```

```
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=35*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT4=np.concatenate((B1,A2),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
```

```
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT5=np.concatenate((B1,A2),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT6=np.concatenate((B1,A2),axis=1)
```

```
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
```

```
B1 = np.asmatrix(E)
B1=B1.T
MAT7=np.concatenate((B1,A2),axis=1)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
```

```
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT8=np.concatenate((B1,A2),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
```

```
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT9=np.concatenate((B1,A2),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
```

```
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT10=np.concatenate((B1,A2),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
```

```
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT11=np.concatenate((B1,A2),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
```

```
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT12=np.concatenate((B1,A2),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
```

```
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT13=np.concatenate((B1,A2),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
```

```
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=24*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT14=np.concatenate((B1,A2),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>4), 0)
MAT14= np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
```

```
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=29*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT15=np.concatenate((B1,A2),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
 ↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Dividimos en los conjuntos de validación y entrenamiento:

[18]:
```
X = MAT[:, 1:10]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)
```

Definimos la arquitectura de la red:

[19]:
```
model = Sequential([
    Dense(128, input_shape=(7,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
```

```
    Dense(60),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

```
WARNING: Logging before flag parsing goes to stderr.
W0830 18:43:38.173150 21680 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:74: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

W0830 18:43:38.218029 21680 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:517: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

W0830 18:43:38.230027 21680 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:4138: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

W0830 18:43:38.287892 21680 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-packages\keras\optimizers.py:790:
The name tf.train.Optimizer is deprecated. Please use
tf.compat.v1.train.Optimizer instead.
```

Entrenamos la red:

[20]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
W0830 18:43:38.476339 21680 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:986: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.

W0830 18:43:38.482322 21680 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:973: The name tf.assign is
deprecated. Please use tf.compat.v1.assign instead.
```

```
Train on 115092 samples, validate on 28773 samples
Epoch 1/100
115092/115092 [==============================] - 20s 172us/step - loss: 0.5812 -
val_loss: 0.4073
Epoch 2/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.3536 -
val_loss: 0.3165
Epoch 3/100
115092/115092 [==============================] - 17s 145us/step - loss: 0.2866 -
val_loss: 0.2833
Epoch 4/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.2494 -
val_loss: 0.3436
Epoch 5/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.2250 -
val_loss: 0.2282
Epoch 6/100
115092/115092 [==============================] - 17s 144us/step - loss: 0.2052 -
val_loss: 0.1929
Epoch 7/100
115092/115092 [==============================] - 17s 147us/step - loss: 0.1894 -
val_loss: 0.2017
Epoch 8/100
115092/115092 [==============================] - 17s 146us/step - loss: 0.1784 -
val_loss: 0.1831
Epoch 9/100
115092/115092 [==============================] - 16s 137us/step - loss: 0.1686 -
val_loss: 0.1819
Epoch 10/100
115092/115092 [==============================] - 16s 138us/step - loss: 0.1580 -
val_loss: 0.1538
Epoch 11/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.1504 -
val_loss: 0.1528
Epoch 12/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.1441 -
val_loss: 0.1321
Epoch 13/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.1389 -
val_loss: 0.1535
Epoch 14/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.1344 -
val_loss: 0.1532
Epoch 15/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.1312 -
val_loss: 0.1380
Epoch 16/100
115092/115092 [==============================] - 17s 145us/step - loss: 0.1277 -
```

```
val_loss: 0.1224
Epoch 17/100
115092/115092 [==============================] - 16s 137us/step - loss: 0.1253 -
val_loss: 0.1261
Epoch 18/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1230 -
val_loss: 0.1179
Epoch 19/100
115092/115092 [==============================] - 17s 144us/step - loss: 0.1208 -
val_loss: 0.1232
Epoch 20/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1185 -
val_loss: 0.1284
Epoch 21/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1163 -
val_loss: 0.1255
Epoch 22/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1146 -
val_loss: 0.1445
Epoch 23/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1131 -
val_loss: 0.1094
Epoch 24/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1117 -
val_loss: 0.1152
Epoch 25/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1098 -
val_loss: 0.1301
Epoch 26/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1077 -
val_loss: 0.1117
Epoch 27/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.1063 -
val_loss: 0.1067
Epoch 28/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1045 -
val_loss: 0.1212
Epoch 29/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.1033 -
val_loss: 0.1202
Epoch 30/100
115092/115092 [==============================] - 17s 144us/step - loss: 0.1018 -
val_loss: 0.1172
Epoch 31/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1013 -
val_loss: 0.1151
Epoch 32/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1005 -
```

```
val_loss: 0.1122
Epoch 33/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0990 -
val_loss: 0.1118
Epoch 34/100
115092/115092 [==============================] - 17s 143us/step - loss: 0.0981 -
val_loss: 0.1119
Epoch 35/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0970 -
val_loss: 0.1033
Epoch 36/100
115092/115092 [==============================] - 17s 144us/step - loss: 0.0965 -
val_loss: 0.1110
Epoch 37/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0963 -
val_loss: 0.1015
Epoch 38/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0952 -
val_loss: 0.1066
Epoch 39/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0945 -
val_loss: 0.0912
Epoch 40/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0935 -
val_loss: 0.1089
Epoch 41/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0926 -
val_loss: 0.0947
Epoch 42/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0923 -
val_loss: 0.1089
Epoch 43/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0919 -
val_loss: 0.0998
Epoch 44/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0905 -
val_loss: 0.0956
Epoch 45/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0903 -
val_loss: 0.0940
Epoch 46/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0896 -
val_loss: 0.0944
Epoch 47/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0891 -
val_loss: 0.1151
Epoch 48/100
115092/115092 [==============================] - 17s 143us/step - loss: 0.0889 -
```

```
val_loss: 0.0930
Epoch 49/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0883 -
val_loss: 0.1023
Epoch 50/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0879 -
val_loss: 0.0896
Epoch 51/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0876 -
val_loss: 0.0967
Epoch 52/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0870 -
val_loss: 0.0813
Epoch 53/100
115092/115092 [==============================] - 17s 144us/step - loss: 0.0869 -
val_loss: 0.1002
Epoch 54/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0860 -
val_loss: 0.0872
Epoch 55/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0857 -
val_loss: 0.0865
Epoch 56/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0856 -
val_loss: 0.0881
Epoch 57/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0848 -
val_loss: 0.0999
Epoch 58/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0843 -
val_loss: 0.0879
Epoch 59/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0834 -
val_loss: 0.1008
Epoch 60/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0837 -
val_loss: 0.0929
Epoch 61/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0832 -
val_loss: 0.0933
Epoch 62/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0830 -
val_loss: 0.0816
Epoch 63/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0821 -
val_loss: 0.0925
Epoch 64/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0819 -
```

```
val_loss: 0.1027
Epoch 65/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0817 -
val_loss: 0.0785
Epoch 66/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0813 -
val_loss: 0.0972
Epoch 67/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0811 -
val_loss: 0.0925
Epoch 68/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0807 -
val_loss: 0.0849
Epoch 69/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0804 -
val_loss: 0.0926
Epoch 70/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0805 -
val_loss: 0.0949
Epoch 71/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0797 -
val_loss: 0.0780
Epoch 72/100
115092/115092 [==============================] - 17s 144us/step - loss: 0.0793 -
val_loss: 0.0815
Epoch 73/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0791 -
val_loss: 0.0853
Epoch 74/100
115092/115092 [==============================] - 17s 143us/step - loss: 0.0788 -
val_loss: 0.0920
Epoch 75/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0784 -
val_loss: 0.0776
Epoch 76/100
115092/115092 [==============================] - 17s 144us/step - loss: 0.0787 -
val_loss: 0.0939
Epoch 77/100
115092/115092 [==============================] - 17s 147us/step - loss: 0.0783 -
val_loss: 0.0944
Epoch 78/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.0776 -
val_loss: 0.0852
Epoch 79/100
115092/115092 [==============================] - 17s 147us/step - loss: 0.0777 -
val_loss: 0.0940
Epoch 80/100
115092/115092 [==============================] - 17s 151us/step - loss: 0.0772 -
```

```
val_loss: 0.0819
Epoch 81/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0772 -
val_loss: 0.0717
Epoch 82/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0768 -
val_loss: 0.1034
Epoch 83/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0766 -
val_loss: 0.0871
Epoch 84/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0765 -
val_loss: 0.0743
Epoch 85/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.0759 -
val_loss: 0.0854
Epoch 86/100
115092/115092 [==============================] - 16s 138us/step - loss: 0.0753 -
val_loss: 0.0716
Epoch 87/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.0753 -
val_loss: 0.0804
Epoch 88/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0754 -
val_loss: 0.0880
Epoch 89/100
115092/115092 [==============================] - 17s 150us/step - loss: 0.0749 -
val_loss: 0.0889
Epoch 90/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0746 -
val_loss: 0.0814
Epoch 91/100
115092/115092 [==============================] - 17s 144us/step - loss: 0.0746 -
val_loss: 0.0799
Epoch 92/100
115092/115092 [==============================] - 15s 134us/step - loss: 0.0742 -
val_loss: 0.0739
Epoch 93/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0735 -
val_loss: 0.0780
Epoch 94/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0735 -
val_loss: 0.0731
Epoch 95/100
115092/115092 [==============================] - 16s 137us/step - loss: 0.0735 -
val_loss: 0.0914
Epoch 96/100
115092/115092 [==============================] - 16s 137us/step - loss: 0.0735 -
```

```
val_loss: 0.0852
Epoch 97/100
115092/115092 [==============================] - 16s 138us/step - loss: 0.0730 -
val_loss: 0.0790
Epoch 98/100
115092/115092 [==============================] - 16s 137us/step - loss: 0.0725 -
val_loss: 0.0786
Epoch 99/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.0727 -
val_loss: 0.0780
Epoch 100/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0727 -
val_loss: 0.0880
```

[21]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
Train on 115092 samples, validate on 28773 samples
Epoch 1/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.0727 -
val_loss: 0.0755
Epoch 2/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0720 -
val_loss: 0.0659
Epoch 3/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0725 -
val_loss: 0.0786
Epoch 4/100
115092/115092 [==============================] - 17s 146us/step - loss: 0.0718 -
val_loss: 0.0770
Epoch 5/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0719 -
val_loss: 0.0659
Epoch 6/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0718 -
val_loss: 0.0738
Epoch 7/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0715 -
val_loss: 0.0833
Epoch 8/100
115092/115092 [==============================] - 16s 136us/step - loss: 0.0714 -
val_loss: 0.0682
Epoch 9/100
115092/115092 [==============================] - 16s 136us/step - loss: 0.0712 -
```

```
val_loss: 0.0770
Epoch 10/100
115092/115092 [==============================] - 17s 145us/step - loss: 0.0707 -
val_loss: 0.0658
Epoch 11/100
115092/115092 [==============================] - 18s 153us/step - loss: 0.0712 -
val_loss: 0.0657
Epoch 12/100
115092/115092 [==============================] - 18s 153us/step - loss: 0.0706 -
val_loss: 0.0877
Epoch 13/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0710 -
val_loss: 0.0706
Epoch 14/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0704 -
val_loss: 0.0815
Epoch 15/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0707 -
val_loss: 0.0800
Epoch 16/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0700 -
val_loss: 0.0781
Epoch 17/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0698 -
val_loss: 0.0713
Epoch 18/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0696 -
val_loss: 0.0776
Epoch 19/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0696 -
val_loss: 0.0718
Epoch 20/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0696 -
val_loss: 0.1008
Epoch 21/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0703 -
val_loss: 0.0745
Epoch 22/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0698 -
val_loss: 0.0753
Epoch 23/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0691 -
val_loss: 0.0699
Epoch 24/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0691 -
val_loss: 0.0790
Epoch 25/100
115092/115092 [==============================] - 17s 146us/step - loss: 0.0693 -
```

```
val_loss: 0.0781
Epoch 26/100
115092/115092 [==============================] - 17s 144us/step - loss: 0.0692 -
val_loss: 0.0720
Epoch 27/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0691 -
val_loss: 0.0818
Epoch 28/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0692 -
val_loss: 0.0674
Epoch 29/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0689 -
val_loss: 0.0828
Epoch 30/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0684 -
val_loss: 0.0773
Epoch 31/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0688 -
val_loss: 0.0681
Epoch 32/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0685 -
val_loss: 0.0773
Epoch 33/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0685 -
val_loss: 0.0786
Epoch 34/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0683 -
val_loss: 0.0691
Epoch 35/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0679 -
val_loss: 0.0753
Epoch 36/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0686 -
val_loss: 0.0779
Epoch 37/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0680 -
val_loss: 0.0764
Epoch 38/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0678 -
val_loss: 0.0751
Epoch 39/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0682 -
val_loss: 0.0707
Epoch 40/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0679 -
val_loss: 0.0850
Epoch 41/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0680 -
```

```
val_loss: 0.0692
Epoch 42/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0678 -
val_loss: 0.0691
Epoch 43/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0681 -
val_loss: 0.0722
Epoch 44/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0679 -
val_loss: 0.0851
Epoch 45/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0676 -
val_loss: 0.0679
Epoch 46/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0677 -
val_loss: 0.0751
Epoch 47/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0681 -
val_loss: 0.0749
Epoch 48/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0674 -
val_loss: 0.0715
Epoch 49/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0674 -
val_loss: 0.0800
Epoch 50/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0678 -
val_loss: 0.0692
Epoch 51/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0675 -
val_loss: 0.0659
Epoch 52/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0673 -
val_loss: 0.0714
Epoch 53/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0671 -
val_loss: 0.0723
Epoch 54/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0669 -
val_loss: 0.0799
Epoch 55/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0673 -
val_loss: 0.0746
Epoch 56/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0674 -
val_loss: 0.0755
Epoch 57/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0669 -
```

```
val_loss: 0.0684
Epoch 58/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0671 -
val_loss: 0.0712
Epoch 59/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0671 -
val_loss: 0.0980
Epoch 60/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0681 -
val_loss: 0.0724
Epoch 61/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0664 -
val_loss: 0.0689
Epoch 62/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0666 -
val_loss: 0.0717
Epoch 63/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0666 -
val_loss: 0.0723
Epoch 64/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0665 -
val_loss: 0.0686
Epoch 65/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0665 -
val_loss: 0.0807
Epoch 66/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0668 -
val_loss: 0.0730
Epoch 67/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0659 -
val_loss: 0.0699
Epoch 68/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0662 -
val_loss: 0.0647
Epoch 69/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0663 -
val_loss: 0.0875
Epoch 70/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0662 -
val_loss: 0.0762
Epoch 71/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0659 -
val_loss: 0.0839
Epoch 72/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0663 -
val_loss: 0.0820
Epoch 73/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0667 -
```

```
val_loss: 0.0857
Epoch 74/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0665 -
val_loss: 0.0885
Epoch 75/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0678 -
val_loss: 0.0728
Epoch 76/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0660 -
val_loss: 0.0758
Epoch 77/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0666 -
val_loss: 0.0700
Epoch 78/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0662 -
val_loss: 0.0662
Epoch 79/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0662 -
val_loss: 0.0756
Epoch 80/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0658 -
val_loss: 0.0763
Epoch 81/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0658 -
val_loss: 0.0742
Epoch 82/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0659 -
val_loss: 0.0763
Epoch 83/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0656 -
val_loss: 0.0716
Epoch 84/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0662 -
val_loss: 0.1091
Epoch 85/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0662 -
val_loss: 0.0720
Epoch 86/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0661 -
val_loss: 0.0699
Epoch 87/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0657 -
val_loss: 0.0800
Epoch 88/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0660 -
val_loss: 0.0790
Epoch 89/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0660 -
```

```
val_loss: 0.0766
Epoch 90/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0660 -
val_loss: 0.0741
Epoch 91/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0665 -
val_loss: 0.0719
Epoch 92/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0663 -
val_loss: 0.0667
Epoch 93/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0666 -
val_loss: 0.0730
Epoch 94/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0663 -
val_loss: 0.0763
Epoch 95/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0664 -
val_loss: 0.0751
Epoch 96/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0660 -
val_loss: 0.0761
Epoch 97/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0663 -
val_loss: 0.0723
Epoch 98/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0661 -
val_loss: 0.0738
Epoch 99/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0664 -
val_loss: 0.0674
Epoch 100/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0662 -
val_loss: 0.0724
```

[22]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
Train on 115092 samples, validate on 28773 samples
Epoch 1/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0662 -
val_loss: 0.0662
Epoch 2/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0664 -
```

```
val_loss: 0.0647
Epoch 3/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0665 -
val_loss: 0.0848
Epoch 4/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0664 -
val_loss: 0.0666
Epoch 5/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0662 -
val_loss: 0.0698
Epoch 6/100
115092/115092 [==============================] - 17s 144us/step - loss: 0.0663 -
val_loss: 0.0654
Epoch 7/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0665 -
val_loss: 0.0749
Epoch 8/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0657 -
val_loss: 0.0792
Epoch 9/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0659 -
val_loss: 0.0680
Epoch 10/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0661 -
val_loss: 0.0754
Epoch 11/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0654 -
val_loss: 0.0657
Epoch 12/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0658 -
val_loss: 0.0761
Epoch 13/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0659 -
val_loss: 0.0725
Epoch 14/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0657 -
val_loss: 0.0711
Epoch 15/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0657 -
val_loss: 0.0703
Epoch 16/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0658 -
val_loss: 0.0733
Epoch 17/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0663 -
val_loss: 0.0803
Epoch 18/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0661 -
```

```
val_loss: 0.0839
Epoch 19/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0659 -
val_loss: 0.0749
Epoch 20/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0650 -
val_loss: 0.0621
Epoch 21/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0654 -
val_loss: 0.0782
Epoch 22/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0657 -
val_loss: 0.0725
Epoch 23/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0651 -
val_loss: 0.0633
Epoch 24/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0654 -
val_loss: 0.0960
Epoch 25/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0646 -
val_loss: 0.0695
Epoch 26/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0647 -
val_loss: 0.0813
Epoch 27/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0653 -
val_loss: 0.0950
Epoch 28/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0658 -
val_loss: 0.0713
Epoch 29/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0642 -
val_loss: 0.0779
Epoch 30/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0650 -
val_loss: 0.0667
Epoch 31/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0646 -
val_loss: 0.0648
Epoch 32/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0648 -
val_loss: 0.0813
Epoch 33/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0652 -
val_loss: 0.0734
Epoch 34/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0649 -
```

```
val_loss: 0.0620
Epoch 35/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0649 -
val_loss: 0.0692
Epoch 36/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0648 -
val_loss: 0.0922
Epoch 37/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0649 -
val_loss: 0.0707
Epoch 38/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0648 -
val_loss: 0.0660
Epoch 39/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0644 -
val_loss: 0.0779
Epoch 40/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0641 -
val_loss: 0.0688
Epoch 41/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0647 -
val_loss: 0.0748
Epoch 42/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0651 -
val_loss: 0.0719
Epoch 43/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0646 -
val_loss: 0.0651
Epoch 44/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0645 -
val_loss: 0.0638
Epoch 45/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0646 -
val_loss: 0.0840
Epoch 46/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0646 -
val_loss: 0.0659
Epoch 47/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0644 -
val_loss: 0.0785
Epoch 48/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0647 -
val_loss: 0.0758
Epoch 49/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0643 -
val_loss: 0.0724
Epoch 50/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0644 -
```

```
val_loss: 0.0696
Epoch 51/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0645 -
val_loss: 0.0751
Epoch 52/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0645 -
val_loss: 0.0777
Epoch 53/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0645 -
val_loss: 0.0579
Epoch 54/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0642 -
val_loss: 0.0730
Epoch 55/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0643 -
val_loss: 0.0729
Epoch 56/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0643 -
val_loss: 0.0599
Epoch 57/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0645 -
val_loss: 0.0661
Epoch 58/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0643 -
val_loss: 0.0649
Epoch 59/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0645 -
val_loss: 0.0762
Epoch 60/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0641 -
val_loss: 0.0634
Epoch 61/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0639 -
val_loss: 0.0649
Epoch 62/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0641 -
val_loss: 0.0636
Epoch 63/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0638 -
val_loss: 0.0849
Epoch 64/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0638 -
val_loss: 0.0658
Epoch 65/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0643 -
val_loss: 0.0789
Epoch 66/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0647 -
```

```
val_loss: 0.0846
Epoch 67/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0634 -
val_loss: 0.0661
Epoch 68/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0642 -
val_loss: 0.0784
Epoch 69/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0638 -
val_loss: 0.0753
Epoch 70/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0645 -
val_loss: 0.0755
Epoch 71/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0643 -
val_loss: 0.0540
Epoch 72/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0637 -
val_loss: 0.0709
Epoch 73/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0640 -
val_loss: 0.0779
Epoch 74/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0639 -
val_loss: 0.0666
Epoch 75/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0640 -
val_loss: 0.0778
Epoch 76/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0635 -
val_loss: 0.0688
Epoch 77/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0636 -
val_loss: 0.0704
Epoch 78/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0634 -
val_loss: 0.0636
Epoch 79/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0639 -
val_loss: 0.0781
Epoch 80/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0636 -
val_loss: 0.0579
Epoch 81/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0632 -
val_loss: 0.0871
Epoch 82/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0635 -
```

```
val_loss: 0.0635
Epoch 83/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0647 -
val_loss: 0.0672
Epoch 84/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0676 -
val_loss: 0.0718
Epoch 85/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0683 -
val_loss: 0.0606
Epoch 86/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0679 -
val_loss: 0.0653
Epoch 87/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0680 -
val_loss: 0.0686
Epoch 88/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0679 -
val_loss: 0.0797
Epoch 89/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0679 -
val_loss: 0.0836
Epoch 90/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0692 -
val_loss: 0.0804
Epoch 91/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0675 -
val_loss: 0.0765
Epoch 92/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0678 -
val_loss: 0.0776
Epoch 93/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0680 -
val_loss: 0.0785
Epoch 94/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0674 -
val_loss: 0.0651
Epoch 95/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0677 -
val_loss: 0.0651
Epoch 96/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0680 -
val_loss: 0.0665
Epoch 97/100
115092/115092 [==============================] - 17s 144us/step - loss: 0.0681 -
val_loss: 0.0601
Epoch 98/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0677 -
```

```
val_loss: 0.0711
Epoch 99/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0672 -
val_loss: 0.0821
Epoch 100/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0673 -
val_loss: 0.0752
```

[23]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[24]:
```
Y_test =np.round(Y_test, 0)
```

[25]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

```
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
```

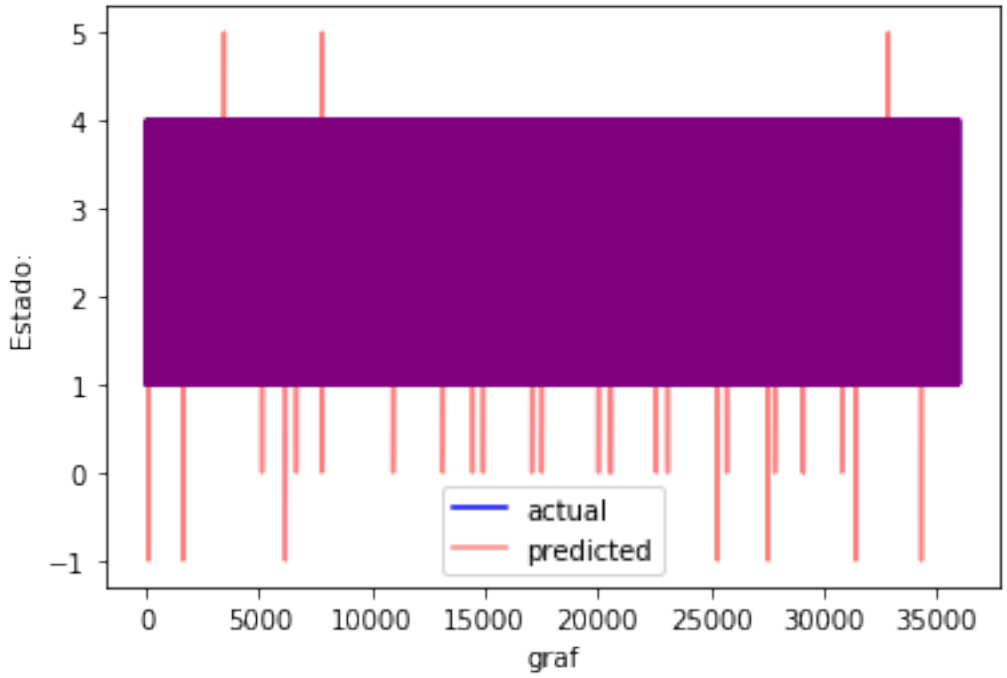
```
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

Evaluamos los resultados:

[26]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.9541467724414469
```

[27]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[28]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[28]:
```
0.9653293296633024
```

[ ]:

Wrist_elux5_mae_Adadel_final

August 31, 2019

1 Wrist Data

Librerías:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Wrist

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
```

```
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
```

```
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT2=np.concatenate((B1,A2),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
```

```
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=25*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT3=np.concatenate((B1,A2),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
```

```
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=35*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT4=np.concatenate((B1,A2),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
```

```
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT5=np.concatenate((B1,A2),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT6=np.concatenate((B1,A2),axis=1)
```

```
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
```

```
B1 = np.asmatrix(E)
B1=B1.T
MAT7=np.concatenate((B1,A2),axis=1)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
```

```
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT8=np.concatenate((B1,A2),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
```

```
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT9=np.concatenate((B1,A2),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
```

```
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT10=np.concatenate((B1,A2),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
```

```
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT11=np.concatenate((B1,A2),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
```

```
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT12=np.concatenate((B1,A2),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
```

```
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT13=np.concatenate((B1,A2),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
```

```
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=24*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT14=np.concatenate((B1,A2),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>4), 0)
MAT14= np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
```

```
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=29*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT15=np.concatenate((B1,A2),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
 ↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Dividimos en los conjuntos de validación y entrenamiento:

[18]:
```
X = MAT[:, 1:8]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)
```

Definimos la arquitectura de la red:

[19]:
```
model = Sequential([
    Dense(128, input_shape=(7,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
```

```
    Dense(48),
    Activation('elu'),
    Dense(24),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

```
WARNING: Logging before flag parsing goes to stderr.
W0830 22:51:34.426502 17120 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:74: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

W0830 22:51:34.441463 17120 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:517: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

W0830 22:51:34.444489 17120 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:4138: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

W0830 22:51:34.512310 17120 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-packages\keras\optimizers.py:790:
The name tf.train.Optimizer is deprecated. Please use
tf.compat.v1.train.Optimizer instead.
```

Entrenamos la red:

[20]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
W0830 22:51:34.731726 17120 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:986: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.

W0830 22:51:34.737670 17120 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:973: The name tf.assign is
```

```
deprecated. Please use tf.compat.v1.assign instead.


Train on 115092 samples, validate on 28773 samples
Epoch 1/100
115092/115092 [==============================] - 20s 171us/step - loss: 0.5629 -
val_loss: 0.3969
Epoch 2/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.3315 -
val_loss: 0.3110
Epoch 3/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.2613 -
val_loss: 0.2899
Epoch 4/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.2206 -
val_loss: 0.2127
Epoch 5/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.1829 -
val_loss: 0.1696
Epoch 6/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.1569 -
val_loss: 0.1894
Epoch 7/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.1428 -
val_loss: 0.1326
Epoch 8/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.1322 -
val_loss: 0.1323
Epoch 9/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1244 -
val_loss: 0.1215
Epoch 10/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.1185 -
val_loss: 0.1247
Epoch 11/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.1134 -
val_loss: 0.1225
Epoch 12/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.1086 -
val_loss: 0.1062
Epoch 13/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.1059 -
val_loss: 0.0967
Epoch 14/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1022 -
val_loss: 0.1032
Epoch 15/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0991 -
```

```
val_loss: 0.1046
Epoch 16/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0959 -
val_loss: 0.0932
Epoch 17/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0943 -
val_loss: 0.1007
Epoch 18/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0919 -
val_loss: 0.0844
Epoch 19/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0899 -
val_loss: 0.0984
Epoch 20/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0876 -
val_loss: 0.0970
Epoch 21/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0869 -
val_loss: 0.0834
Epoch 22/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0855 -
val_loss: 0.0897
Epoch 23/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0836 -
val_loss: 0.0982
Epoch 24/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0832 -
val_loss: 0.0945
Epoch 25/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0822 -
val_loss: 0.0856
Epoch 26/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0809 -
val_loss: 0.1015
Epoch 27/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0804 -
val_loss: 0.0858
Epoch 28/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0795 -
val_loss: 0.0951
Epoch 29/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0787 -
val_loss: 0.0860
Epoch 30/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0775 -
val_loss: 0.0863
Epoch 31/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0769 -
```

```
val_loss: 0.0776
Epoch 32/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0760 -
val_loss: 0.0799
Epoch 33/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0749 -
val_loss: 0.0933
Epoch 34/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0743 -
val_loss: 0.0801
Epoch 35/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0741 -
val_loss: 0.0930
Epoch 36/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0728 -
val_loss: 0.0772
Epoch 37/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0726 -
val_loss: 0.0738
Epoch 38/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0719 -
val_loss: 0.0755
Epoch 39/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0716 -
val_loss: 0.0680
Epoch 40/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0708 -
val_loss: 0.0716
Epoch 41/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0701 -
val_loss: 0.0724
Epoch 42/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0696 -
val_loss: 0.0789
Epoch 43/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0698 -
val_loss: 0.0892
Epoch 44/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0691 -
val_loss: 0.0810
Epoch 45/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0693 -
val_loss: 0.0697
Epoch 46/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0686 -
val_loss: 0.0835
Epoch 47/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0678 -
```

```
val_loss: 0.0854
Epoch 48/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0676 -
val_loss: 0.0765
Epoch 49/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0673 -
val_loss: 0.0762
Epoch 50/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0669 -
val_loss: 0.0743
Epoch 51/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0663 -
val_loss: 0.0709
Epoch 52/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0663 -
val_loss: 0.0797
Epoch 53/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0660 -
val_loss: 0.0837
Epoch 54/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0656 -
val_loss: 0.0666
Epoch 55/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0653 -
val_loss: 0.0606
Epoch 56/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0649 -
val_loss: 0.0767
Epoch 57/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0652 -
val_loss: 0.0636
Epoch 58/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0651 -
val_loss: 0.0658
Epoch 59/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0648 -
val_loss: 0.0853
Epoch 60/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0644 -
val_loss: 0.0894
Epoch 61/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0645 -
val_loss: 0.0799
Epoch 62/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0644 -
val_loss: 0.0707
Epoch 63/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0646 -
```

```
val_loss: 0.0642
Epoch 64/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0641 -
val_loss: 0.0676
Epoch 65/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0639 -
val_loss: 0.0661
Epoch 66/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0639 -
val_loss: 0.0724
Epoch 67/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0639 -
val_loss: 0.0696
Epoch 68/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0639 -
val_loss: 0.0721
Epoch 69/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0626 -
val_loss: 0.0708
Epoch 70/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0629 -
val_loss: 0.0707
Epoch 71/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0628 -
val_loss: 0.0659
Epoch 72/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0622 -
val_loss: 0.0653
Epoch 73/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0617 -
val_loss: 0.0713
Epoch 74/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0617 -
val_loss: 0.0676
Epoch 75/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0620 -
val_loss: 0.0653
Epoch 76/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0611 -
val_loss: 0.0642
Epoch 77/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0619 -
val_loss: 0.0602
Epoch 78/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0613 -
val_loss: 0.0712
Epoch 79/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0609 -
```

```
val_loss: 0.0685
Epoch 80/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0618 -
val_loss: 0.0664
Epoch 81/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0611 -
val_loss: 0.0594
Epoch 82/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0601 -
val_loss: 0.0631
Epoch 83/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0600 -
val_loss: 0.0553
Epoch 84/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0591 -
val_loss: 0.0569
Epoch 85/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0589 -
val_loss: 0.0620
Epoch 86/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0586 -
val_loss: 0.0585
Epoch 87/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0585 -
val_loss: 0.0600
Epoch 88/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0579 -
val_loss: 0.0591
Epoch 89/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0574 -
val_loss: 0.0576
Epoch 90/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0565 -
val_loss: 0.0704
Epoch 91/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0566 -
val_loss: 0.0640
Epoch 92/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0570 -
val_loss: 0.0582
Epoch 93/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0567 -
val_loss: 0.0584
Epoch 94/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0570 -
val_loss: 0.0573
Epoch 95/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0557 -
```

```
val_loss: 0.0597
Epoch 96/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0564 -
val_loss: 0.0705
Epoch 97/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0574 -
val_loss: 0.0580
Epoch 98/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0564 -
val_loss: 0.0592
Epoch 99/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0556 -
val_loss: 0.0544
Epoch 100/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0561 -
val_loss: 0.0568
```

[21]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
Train on 115092 samples, validate on 28773 samples
Epoch 1/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0569 -
val_loss: 0.0709
Epoch 2/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0566 -
val_loss: 0.0691
Epoch 3/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0558 -
val_loss: 0.0654
Epoch 4/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0561 -
val_loss: 0.0601
Epoch 5/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0569 -
val_loss: 0.0776
Epoch 6/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0554 -
val_loss: 0.0708
Epoch 7/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0545 -
val_loss: 0.0543
Epoch 8/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0552 -
```

```
val_loss: 0.0540
Epoch 9/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0559 -
val_loss: 0.0591
Epoch 10/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0556 -
val_loss: 0.0504
Epoch 11/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0559 -
val_loss: 0.0630
Epoch 12/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0556 -
val_loss: 0.0540
Epoch 13/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0553 -
val_loss: 0.0629
Epoch 14/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0559 -
val_loss: 0.0680
Epoch 15/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0555 -
val_loss: 0.0677
Epoch 16/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0553 -
val_loss: 0.0539
Epoch 17/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0556 -
val_loss: 0.0599
Epoch 18/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0562 -
val_loss: 0.0547
Epoch 19/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0556 -
val_loss: 0.0749
Epoch 20/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0559 -
val_loss: 0.0737
Epoch 21/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0555 -
val_loss: 0.0759
Epoch 22/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0573 -
val_loss: 0.0695
Epoch 23/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0574 -
val_loss: 0.0621
Epoch 24/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0570 -
```

```
val_loss: 0.0491
Epoch 25/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0565 -
val_loss: 0.0705
Epoch 26/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0571 -
val_loss: 0.0560
Epoch 27/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0574 -
val_loss: 0.0605
Epoch 28/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0575 -
val_loss: 0.0553
Epoch 29/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0578 -
val_loss: 0.0779
Epoch 30/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0583 -
val_loss: 0.0634
Epoch 31/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0592 -
val_loss: 0.0540
Epoch 32/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0591 -
val_loss: 0.0577
Epoch 33/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0595 -
val_loss: 0.0611
Epoch 34/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0580 -
val_loss: 0.0574
Epoch 35/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0579 -
val_loss: 0.0625
Epoch 36/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0582 -
val_loss: 0.0577
Epoch 37/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0576 -
val_loss: 0.0534
Epoch 38/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0591 -
val_loss: 0.0596
Epoch 39/100
115092/115092 [==============================] - 18s 153us/step - loss: 0.0580 -
val_loss: 0.0701
Epoch 40/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0596 -
```

```
val_loss: 0.0536
Epoch 41/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0589 -
val_loss: 0.0558
Epoch 42/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0589 -
val_loss: 0.0620
Epoch 43/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0585 -
val_loss: 0.0697
Epoch 44/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0587 -
val_loss: 0.0554
Epoch 45/100
115092/115092 [==============================] - 19s 166us/step - loss: 0.0591 -
val_loss: 0.0573
Epoch 46/100
115092/115092 [==============================] - 20s 171us/step - loss: 0.0579 -
val_loss: 0.0597
Epoch 47/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0581 -
val_loss: 0.0777
Epoch 48/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0597 -
val_loss: 0.0685
Epoch 49/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0585 -
val_loss: 0.0632
Epoch 50/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0588 -
val_loss: 0.0586
Epoch 51/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0596 -
val_loss: 0.0915
Epoch 52/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0587 -
val_loss: 0.0656
Epoch 53/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0599 -
val_loss: 0.0749
Epoch 54/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0599 -
val_loss: 0.0656
Epoch 55/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0592 -
val_loss: 0.0671
Epoch 56/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0595 -
```

```
val_loss: 0.0707
Epoch 57/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0596 -
val_loss: 0.0692
Epoch 58/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0602 -
val_loss: 0.0717
Epoch 59/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0606 -
val_loss: 0.0563
Epoch 60/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0649 -
val_loss: 0.0803
Epoch 61/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0613 -
val_loss: 0.0759
Epoch 62/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0615 -
val_loss: 0.0520
Epoch 63/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0604 -
val_loss: 0.0754
Epoch 64/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0618 -
val_loss: 0.0565
Epoch 65/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0613 -
val_loss: 0.0574
Epoch 66/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0621 -
val_loss: 0.0754
Epoch 67/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0634 -
val_loss: 0.0723
Epoch 68/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0628 -
val_loss: 0.0612
Epoch 69/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0635 -
val_loss: 0.0680
Epoch 70/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0626 -
val_loss: 0.0780
Epoch 71/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0622 -
val_loss: 0.0535
Epoch 72/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0636 -
```

```
val_loss: 0.1095
Epoch 73/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0646 -
val_loss: 0.0662
Epoch 74/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0639 -
val_loss: 0.0753
Epoch 75/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0638 -
val_loss: 0.0591
Epoch 76/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0652 -
val_loss: 0.0593
Epoch 77/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0643 -
val_loss: 0.0672
Epoch 78/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0647 -
val_loss: 0.0684
Epoch 79/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0661 -
val_loss: 0.0624
Epoch 80/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0668 -
val_loss: 0.0630
Epoch 81/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0670 -
val_loss: 0.0635
Epoch 82/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0685 -
val_loss: 0.0704
Epoch 83/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0679 -
val_loss: 0.0650
Epoch 84/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0679 -
val_loss: 0.0724
Epoch 85/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0685 -
val_loss: 0.0625
Epoch 86/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0709 -
val_loss: 0.0824
Epoch 87/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0710 -
val_loss: 0.0640
Epoch 88/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0688 -
```

```
val_loss: 0.0977
Epoch 89/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0696 -
val_loss: 0.0748
Epoch 90/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0685 -
val_loss: 0.0641
Epoch 91/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0680 -
val_loss: 0.0624
Epoch 92/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0691 -
val_loss: 0.0696
Epoch 93/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0693 -
val_loss: 0.0590
Epoch 94/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0684 -
val_loss: 0.0657
Epoch 95/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0667 -
val_loss: 0.0588
Epoch 96/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0687 -
val_loss: 0.0709
Epoch 97/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0676 -
val_loss: 0.0742
Epoch 98/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0659 -
val_loss: 0.0608
Epoch 99/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0644 -
val_loss: 0.0690
Epoch 100/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0628 -
val_loss: 0.0744
```

[22]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
Train on 115092 samples, validate on 28773 samples
Epoch 1/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0622 -
```

```
val_loss: 0.0584
Epoch 2/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0623 -
val_loss: 0.0598
Epoch 3/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0614 -
val_loss: 0.0793
Epoch 4/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0621 -
val_loss: 0.0586
Epoch 5/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0597 -
val_loss: 0.0719
Epoch 6/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0609 -
val_loss: 0.0579
Epoch 7/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0606 -
val_loss: 0.0585
Epoch 8/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0610 -
val_loss: 0.0759
Epoch 9/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0609 -
val_loss: 0.0531
Epoch 10/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0601 -
val_loss: 0.0642
Epoch 11/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0605 -
val_loss: 0.0702
Epoch 12/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0619 -
val_loss: 0.0602
Epoch 13/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0616 -
val_loss: 0.0649
Epoch 14/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0617 -
val_loss: 0.0651
Epoch 15/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0609 -
val_loss: 0.0554
Epoch 16/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0618 -
val_loss: 0.0619
Epoch 17/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0604 -
```

```
val_loss: 0.0718
Epoch 18/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0613 -
val_loss: 0.0547
Epoch 19/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0612 -
val_loss: 0.0829
Epoch 20/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0610 -
val_loss: 0.0641
Epoch 21/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0613 -
val_loss: 0.0602
Epoch 22/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0613 -
val_loss: 0.0793
Epoch 23/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0610 -
val_loss: 0.0654
Epoch 24/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0610 -
val_loss: 0.0536
Epoch 25/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0612 -
val_loss: 0.0649
Epoch 26/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0611 -
val_loss: 0.0678
Epoch 27/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0615 -
val_loss: 0.0720
Epoch 28/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0619 -
val_loss: 0.0651
Epoch 29/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0631 -
val_loss: 0.0916
Epoch 30/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0628 -
val_loss: 0.0565
Epoch 31/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0623 -
val_loss: 0.0706
Epoch 32/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0637 -
val_loss: 0.0776
Epoch 33/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0630 -
```

```
val_loss: 0.0728
Epoch 34/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0620 -
val_loss: 0.0669
Epoch 35/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0627 -
val_loss: 0.0622
Epoch 36/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0633 -
val_loss: 0.0769
Epoch 37/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0643 -
val_loss: 0.0680
Epoch 38/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0643 -
val_loss: 0.0722
Epoch 39/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0656 -
val_loss: 0.0717
Epoch 40/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0643 -
val_loss: 0.0761
Epoch 41/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0657 -
val_loss: 0.0587
Epoch 42/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0644 -
val_loss: 0.0707
Epoch 43/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0630 -
val_loss: 0.0678
Epoch 44/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0623 -
val_loss: 0.0615
Epoch 45/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0617 -
val_loss: 0.0882
Epoch 46/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0625 -
val_loss: 0.0797
Epoch 47/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0623 -
val_loss: 0.0528
Epoch 48/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0623 -
val_loss: 0.0647
Epoch 49/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0628 -
```

```
val_loss: 0.0685
Epoch 50/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0639 -
val_loss: 0.0569
Epoch 51/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0624 -
val_loss: 0.0644
Epoch 52/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0634 -
val_loss: 0.0712
Epoch 53/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0639 -
val_loss: 0.0781
Epoch 54/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0626 -
val_loss: 0.0586
Epoch 55/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0623 -
val_loss: 0.0685
Epoch 56/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0627 -
val_loss: 0.0706
Epoch 57/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0634 -
val_loss: 0.0557
Epoch 58/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0628 -
val_loss: 0.0715
Epoch 59/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0632 -
val_loss: 0.0631
Epoch 60/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0629 -
val_loss: 0.0556
Epoch 61/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0628 -
val_loss: 0.0673
Epoch 62/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0622 -
val_loss: 0.0668
Epoch 63/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0634 -
val_loss: 0.0490
Epoch 64/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0633 -
val_loss: 0.0520
Epoch 65/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0635 -
```

```
val_loss: 0.0686
Epoch 66/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0631 -
val_loss: 0.0618
Epoch 67/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0624 -
val_loss: 0.0618
Epoch 68/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0632 -
val_loss: 0.0662
Epoch 69/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0639 -
val_loss: 0.1032
Epoch 70/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0632 -
val_loss: 0.0737
Epoch 71/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0633 -
val_loss: 0.0617
Epoch 72/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0653 -
val_loss: 0.0673
Epoch 73/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0632 -
val_loss: 0.0663
Epoch 74/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0619 -
val_loss: 0.0771
Epoch 75/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0638 -
val_loss: 0.0676
Epoch 76/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0641 -
val_loss: 0.0677
Epoch 77/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0653 -
val_loss: 0.0725
Epoch 78/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0644 -
val_loss: 0.0919
Epoch 79/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0637 -
val_loss: 0.0586
Epoch 80/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0643 -
val_loss: 0.0725
Epoch 81/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0639 -
```

```
val_loss: 0.0594
Epoch 82/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0647 -
val_loss: 0.0603
Epoch 83/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0637 -
val_loss: 0.0787
Epoch 84/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0647 -
val_loss: 0.0715
Epoch 85/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0654 -
val_loss: 0.0559
Epoch 86/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0662 -
val_loss: 0.0671
Epoch 87/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0649 -
val_loss: 0.0600
Epoch 88/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0666 -
val_loss: 0.0629
Epoch 89/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0662 -
val_loss: 0.0666
Epoch 90/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0653 -
val_loss: 0.0658
Epoch 91/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0667 -
val_loss: 0.0631
Epoch 92/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0674 -
val_loss: 0.0673
Epoch 93/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0661 -
val_loss: 0.0660
Epoch 94/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0652 -
val_loss: 0.0785
Epoch 95/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0665 -
val_loss: 0.0691
Epoch 96/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0676 -
val_loss: 0.0784
Epoch 97/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0670 -
```

```
val_loss: 0.0623
Epoch 98/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0681 -
val_loss: 0.0660
Epoch 99/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0669 -
val_loss: 0.0861
Epoch 100/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0664 -
val_loss: 0.0569
```

[23]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[24]:
```
Y_test =np.round(Y_test, 0)
```

[25]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

```
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:3.
Estado: 1., predicted:1.
Estado: 2., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:2.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
```

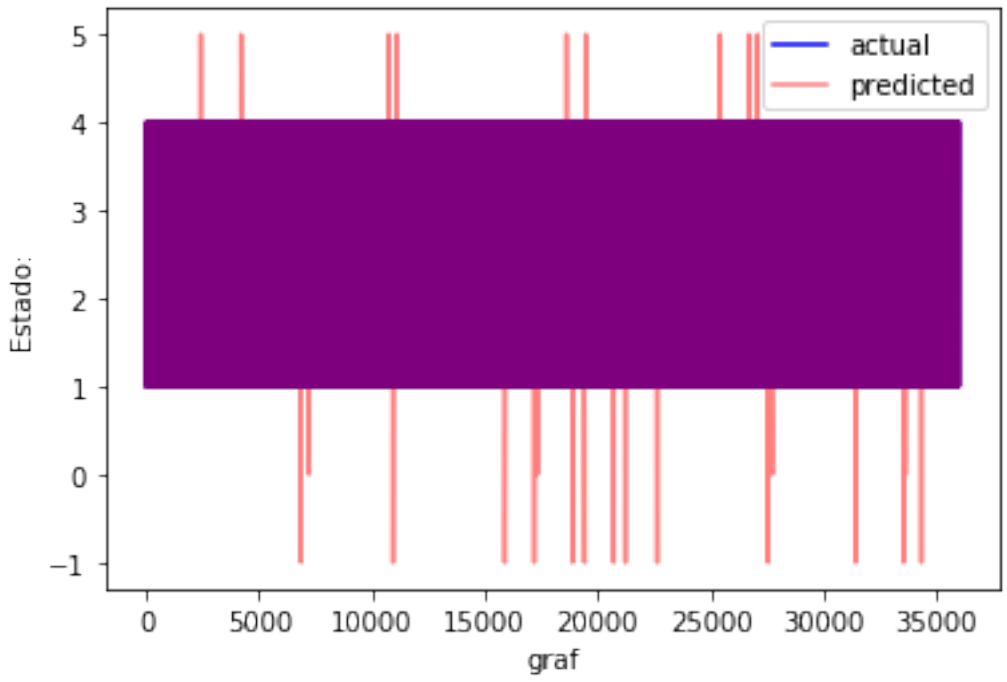
```
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

Evaluamos los resultados:

[26]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.9516759810555495
```

[27]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[28]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
```

```
accur/len(Y_val)
```

[28]:
```
0.972836210971168
```

[ ]:
```

```

Wrist_elux7_mae_Adadel_final

August 31, 2019

1 Wrist Data

Librerías:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Wrist

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
```

```
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
```

```
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT2=np.concatenate((B1,A2),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
```

```
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=25*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT3=np.concatenate((B1,A2),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
```

```
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=35*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT4=np.concatenate((B1,A2),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
```

```
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT5=np.concatenate((B1,A2),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT6=np.concatenate((B1,A2),axis=1)
```

```
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
```

```
B1 = np.asmatrix(E)
B1=B1.T
MAT7=np.concatenate((B1,A2),axis=1)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
```

```
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT8=np.concatenate((B1,A2),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
```

```
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT9=np.concatenate((B1,A2),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
```

```
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT10=np.concatenate((B1,A2),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
```

```
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT11=np.concatenate((B1,A2),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
```

```
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT12=np.concatenate((B1,A2),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
```

```
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT13=np.concatenate((B1,A2),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
```

```
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=24*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT14=np.concatenate((B1,A2),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>4), 0)
MAT14= np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
```

```
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=29*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT15=np.concatenate((B1,A2),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
 ↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Dividimos en los conjuntos de validación y entrenamiento:

[18]:
```
X = MAT[:, 1:8]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)
```

Definimos la arquitectura de la red:

[19]:
```
model = Sequential([
    Dense(128, input_shape=(7,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
```

```
    Dense(60),
    Activation('elu'),
    Dense(40),
    Activation('elu'),
    Dense(40),
    Activation('elu'),
    Dense(30),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

```
WARNING: Logging before flag parsing goes to stderr.
W0831 00:46:19.774099 23100 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:74: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

W0831 00:46:19.787102 23100 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:517: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

W0831 00:46:19.792051 23100 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:4138: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

W0831 00:46:19.878854 23100 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-packages\keras\optimizers.py:790:
The name tf.train.Optimizer is deprecated. Please use
tf.compat.v1.train.Optimizer instead.
```

Entrenamos la red:

[20]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
W0831 00:46:20.109243 23100 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:986: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.
```

```
W0831 00:46:20.114234 23100 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:973: The name tf.assign is
deprecated. Please use tf.compat.v1.assign instead.


Train on 115092 samples, validate on 28773 samples
Epoch 1/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.5183 -
val_loss: 0.4381
Epoch 2/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.2869 -
val_loss: 0.2696
Epoch 3/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.2129 -
val_loss: 0.2185
Epoch 4/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.1772 -
val_loss: 0.1802
Epoch 5/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.1586 -
val_loss: 0.1495
Epoch 6/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.1457 -
val_loss: 0.1383
Epoch 7/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.1347 -
val_loss: 0.1748
Epoch 8/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.1256 -
val_loss: 0.1191
Epoch 9/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.1188 -
val_loss: 0.1282
Epoch 10/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.1119 -
val_loss: 0.1088
Epoch 11/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.1081 -
val_loss: 0.1141
Epoch 12/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.1041 -
val_loss: 0.1042
Epoch 13/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.1008 -
val_loss: 0.0996
Epoch 14/100
```

```
115092/115092 [==============================] - 22s 192us/step - loss: 0.0977 -
val_loss: 0.0960
Epoch 15/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0960 -
val_loss: 0.0977
Epoch 16/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0918 -
val_loss: 0.0951
Epoch 17/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0898 -
val_loss: 0.0944
Epoch 18/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0875 -
val_loss: 0.0898
Epoch 19/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0858 -
val_loss: 0.0973
Epoch 20/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0844 -
val_loss: 0.0989
Epoch 21/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0832 -
val_loss: 0.0805
Epoch 22/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0802 -
val_loss: 0.0866
Epoch 23/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0801 -
val_loss: 0.1035
Epoch 24/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0784 -
val_loss: 0.0833
Epoch 25/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0776 -
val_loss: 0.0740
Epoch 26/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0758 -
val_loss: 0.0776
Epoch 27/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0750 -
val_loss: 0.0778
Epoch 28/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0740 -
val_loss: 0.0714
Epoch 29/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0734 -
val_loss: 0.0677
Epoch 30/100
```

```
115092/115092 [==============================] - 22s 192us/step - loss: 0.0722 -
val_loss: 0.0755
Epoch 31/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0724 -
val_loss: 0.0715
Epoch 32/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0721 -
val_loss: 0.0683
Epoch 33/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0707 -
val_loss: 0.0714
Epoch 34/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0699 -
val_loss: 0.0687
Epoch 35/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0697 -
val_loss: 0.0747
Epoch 36/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0696 -
val_loss: 0.0877
Epoch 37/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0674 -
val_loss: 0.0756
Epoch 38/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0673 -
val_loss: 0.0876
Epoch 39/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0675 -
val_loss: 0.0631
Epoch 40/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0667 -
val_loss: 0.0614
Epoch 41/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0668 -
val_loss: 0.0824
Epoch 42/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0649 -
val_loss: 0.0661
Epoch 43/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0654 -
val_loss: 0.0787
Epoch 44/100
115092/115092 [==============================] - 22s 194us/step - loss: 0.0649 -
val_loss: 0.0639
Epoch 45/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0654 -
val_loss: 0.0677
Epoch 46/100
```

```
115092/115092 [==============================] - 22s 192us/step - loss: 0.0634 -
val_loss: 0.0656
Epoch 47/100
115092/115092 [==============================] - 22s 194us/step - loss: 0.0645 -
val_loss: 0.0629
Epoch 48/100
115092/115092 [==============================] - 22s 195us/step - loss: 0.0635 -
val_loss: 0.0674
Epoch 49/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0628 -
val_loss: 0.0792
Epoch 50/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0626 -
val_loss: 0.0760
Epoch 51/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0636 -
val_loss: 0.0686
Epoch 52/100
115092/115092 [==============================] - 23s 201us/step - loss: 0.0628 -
val_loss: 0.0622
Epoch 53/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0622 -
val_loss: 0.0691
Epoch 54/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0620 -
val_loss: 0.0633
Epoch 55/100
115092/115092 [==============================] - 23s 196us/step - loss: 0.0619 -
val_loss: 0.0662
Epoch 56/100
115092/115092 [==============================] - 22s 194us/step - loss: 0.0619 -
val_loss: 0.0612
Epoch 57/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0622 -
val_loss: 0.0691
Epoch 58/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0613 -
val_loss: 0.0628
Epoch 59/100
115092/115092 [==============================] - 22s 194us/step - loss: 0.0611 -
val_loss: 0.0692
Epoch 60/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0606 -
val_loss: 0.0506
Epoch 61/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0590 -
val_loss: 0.0532
Epoch 62/100
```

```
115092/115092 [==============================] - 22s 192us/step - loss: 0.0593 -
val_loss: 0.0498
Epoch 63/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0577 -
val_loss: 0.0592
Epoch 64/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0583 -
val_loss: 0.0627
Epoch 65/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0577 -
val_loss: 0.0554
Epoch 66/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0579 -
val_loss: 0.0701
Epoch 67/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0578 -
val_loss: 0.0495
Epoch 68/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0568 -
val_loss: 0.0721
Epoch 69/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0570 -
val_loss: 0.0638
Epoch 70/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0564 -
val_loss: 0.0488
Epoch 71/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0551 -
val_loss: 0.0616
Epoch 72/100
115092/115092 [==============================] - 23s 196us/step - loss: 0.0550 -
val_loss: 0.0628
Epoch 73/100
115092/115092 [==============================] - 22s 194us/step - loss: 0.0546 -
val_loss: 0.0740
Epoch 74/100
115092/115092 [==============================] - 22s 194us/step - loss: 0.0552 -
val_loss: 0.0514
Epoch 75/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0547 -
val_loss: 0.0717
Epoch 76/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0549 -
val_loss: 0.0581
Epoch 77/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0542 -
val_loss: 0.0584
Epoch 78/100
```

```
115092/115092 [==============================] - 22s 190us/step - loss: 0.0540 -
val_loss: 0.0552
Epoch 79/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0539 -
val_loss: 0.0490
Epoch 80/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0536 -
val_loss: 0.0548
Epoch 81/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0554 -
val_loss: 0.0814
Epoch 82/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0531 -
val_loss: 0.0466
Epoch 83/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0523 -
val_loss: 0.0554
Epoch 84/100
115092/115092 [==============================] - 22s 195us/step - loss: 0.0523 -
val_loss: 0.0667
Epoch 85/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0530 -
val_loss: 0.0490
Epoch 86/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0527 -
val_loss: 0.0430
Epoch 87/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0518 -
val_loss: 0.0628
Epoch 88/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0519 -
val_loss: 0.0573
Epoch 89/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0515 -
val_loss: 0.0787
Epoch 90/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0523 -
val_loss: 0.0501
Epoch 91/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0537 -
val_loss: 0.0658
Epoch 92/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0535 -
val_loss: 0.0506
Epoch 93/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0533 -
val_loss: 0.0442
Epoch 94/100
```

```
115092/115092 [==============================] - 22s 192us/step - loss: 0.0526 -
val_loss: 0.0538
Epoch 95/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0524 -
val_loss: 0.0548
Epoch 96/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0525 -
val_loss: 0.0408
Epoch 97/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0534 -
val_loss: 0.0519
Epoch 98/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0531 -
val_loss: 0.0427
Epoch 99/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0533 -
val_loss: 0.0569
Epoch 100/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0538 -
val_loss: 0.0492
```

[21]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
Train on 115092 samples, validate on 28773 samples
Epoch 1/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0540 -
val_loss: 0.0414
Epoch 2/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0531 -
val_loss: 0.0520
Epoch 3/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0537 -
val_loss: 0.0562
Epoch 4/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0521 -
val_loss: 0.0743
Epoch 5/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0519 -
val_loss: 0.0530
Epoch 6/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0511 -
val_loss: 0.0559
Epoch 7/100
```

```
115092/115092 [==============================] - 22s 191us/step - loss: 0.0516 -
val_loss: 0.0486
Epoch 8/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0521 -
val_loss: 0.0679
Epoch 9/100
115092/115092 [==============================] - 22s 195us/step - loss: 0.0520 -
val_loss: 0.0599
Epoch 10/100
115092/115092 [==============================] - 22s 195us/step - loss: 0.0529 -
val_loss: 0.0803
Epoch 11/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0526 -
val_loss: 0.0646
Epoch 12/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0520 -
val_loss: 0.0553
Epoch 13/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0538 -
val_loss: 0.0447
Epoch 14/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0529 -
val_loss: 0.0504
Epoch 15/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0536 -
val_loss: 0.0406
Epoch 16/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0543 -
val_loss: 0.0481
Epoch 17/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0530 -
val_loss: 0.0551
Epoch 18/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0537 -
val_loss: 0.0534
Epoch 19/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0525 -
val_loss: 0.0501
Epoch 20/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0537 -
val_loss: 0.0705
Epoch 21/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0533 -
val_loss: 0.0525
Epoch 22/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0534 -
val_loss: 0.0500
Epoch 23/100
```

```
115092/115092 [==============================] - 22s 193us/step - loss: 0.0543 -
val_loss: 0.0533
Epoch 24/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0551 -
val_loss: 0.0579
Epoch 25/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0547 -
val_loss: 0.0514
Epoch 26/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0583 -
val_loss: 0.0657
Epoch 27/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0567 -
val_loss: 0.0763
Epoch 28/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0555 -
val_loss: 0.0633
Epoch 29/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0592 -
val_loss: 0.0685
Epoch 30/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0620 -
val_loss: 0.0708
Epoch 31/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0623 -
val_loss: 0.0806
Epoch 32/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0597 -
val_loss: 0.0523
Epoch 33/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0613 -
val_loss: 0.0626
Epoch 34/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0606 -
val_loss: 0.0625
Epoch 35/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0620 -
val_loss: 0.0602
Epoch 36/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0633 -
val_loss: 0.0739
Epoch 37/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0627 -
val_loss: 0.0572
Epoch 38/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0671 -
val_loss: 0.0733
Epoch 39/100
```

```
115092/115092 [==============================] - 22s 192us/step - loss: 0.0677 -
val_loss: 0.0644
Epoch 40/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0664 -
val_loss: 0.0594
Epoch 41/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0663 -
val_loss: 0.0612
Epoch 42/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0658 -
val_loss: 0.0685
Epoch 43/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0662 -
val_loss: 0.0749
Epoch 44/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0642 -
val_loss: 0.0772
Epoch 45/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0657 -
val_loss: 0.0909
Epoch 46/100
115092/115092 [==============================] - 22s 194us/step - loss: 0.0667 -
val_loss: 0.0570
Epoch 47/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0663 -
val_loss: 0.0519
Epoch 48/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0649 -
val_loss: 0.0745
Epoch 49/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0650 -
val_loss: 0.0917
Epoch 50/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0690 -
val_loss: 0.0578
Epoch 51/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0782 -
val_loss: 0.0890
Epoch 52/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0779 -
val_loss: 0.0705
Epoch 53/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0802 -
val_loss: 0.0758
Epoch 54/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0837 -
val_loss: 0.0879
Epoch 55/100
```

```
115092/115092 [==============================] - 22s 192us/step - loss: 0.0916 -
val_loss: 0.0964
Epoch 56/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0896 -
val_loss: 0.1102
Epoch 57/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0786 -
val_loss: 0.0844
Epoch 58/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0796 -
val_loss: 0.0861
Epoch 59/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0768 -
val_loss: 0.0698
Epoch 60/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0770 -
val_loss: 0.0848
Epoch 61/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0760 -
val_loss: 0.0728
Epoch 62/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0763 -
val_loss: 0.0796
Epoch 63/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0790 -
val_loss: 0.1093
Epoch 64/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0752 -
val_loss: 0.0744
Epoch 65/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0770 -
val_loss: 0.0781
Epoch 66/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0787 -
val_loss: 0.0919
Epoch 67/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0787 -
val_loss: 0.0853
Epoch 68/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0801 -
val_loss: 0.0589
Epoch 69/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0775 -
val_loss: 0.0626
Epoch 70/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0797 -
val_loss: 0.0960
Epoch 71/100
```

```
115092/115092 [==============================] - 22s 192us/step - loss: 0.0807 -
val_loss: 0.0732
Epoch 72/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0812 -
val_loss: 0.0786
Epoch 73/100
115092/115092 [==============================] - 22s 194us/step - loss: 0.0779 -
val_loss: 0.0965
Epoch 74/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0822 -
val_loss: 0.1047
Epoch 75/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0813 -
val_loss: 0.0824
Epoch 76/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0827 -
val_loss: 0.1222
Epoch 77/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0830 -
val_loss: 0.0840
Epoch 78/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0862 -
val_loss: 0.0817
Epoch 79/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0866 -
val_loss: 0.1006
Epoch 80/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0817 -
val_loss: 0.0875
Epoch 81/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0796 -
val_loss: 0.0893
Epoch 82/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0791 -
val_loss: 0.0991
Epoch 83/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0782 -
val_loss: 0.1013
Epoch 84/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0803 -
val_loss: 0.0949
Epoch 85/100
115092/115092 [==============================] - 22s 195us/step - loss: 0.0810 -
val_loss: 0.0896
Epoch 86/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0804 -
val_loss: 0.0693
Epoch 87/100
```

```
115092/115092 [==============================] - 22s 188us/step - loss: 0.0765 -
val_loss: 0.0856
Epoch 88/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0773 -
val_loss: 0.0886
Epoch 89/100
115092/115092 [==============================] - 22s 187us/step - loss: 0.0747 -
val_loss: 0.0846
Epoch 90/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0751 -
val_loss: 0.0798
Epoch 91/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0761 -
val_loss: 0.0589
Epoch 92/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0771 -
val_loss: 0.0972
Epoch 93/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0752 -
val_loss: 0.0732
Epoch 94/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0740 -
val_loss: 0.0742
Epoch 95/100
115092/115092 [==============================] - 23s 201us/step - loss: 0.0724 -
val_loss: 0.0756
Epoch 96/100
115092/115092 [==============================] - 23s 196us/step - loss: 0.0730 -
val_loss: 0.0868
Epoch 97/100
115092/115092 [==============================] - 24s 205us/step - loss: 0.0719 -
val_loss: 0.0725
Epoch 98/100
115092/115092 [==============================] - 24s 206us/step - loss: 0.0732 -
val_loss: 0.0841
Epoch 99/100
115092/115092 [==============================] - 23s 203us/step - loss: 0.0751 -
val_loss: 0.0688
Epoch 100/100
115092/115092 [==============================] - 23s 202us/step - loss: 0.0760 -
val_loss: 0.0699
```

[22]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
Train on 115092 samples, validate on 28773 samples
Epoch 1/100
115092/115092 [==============================] - 22s 195us/step - loss: 0.0727 -
val_loss: 0.0812
Epoch 2/100
115092/115092 [==============================] - 22s 195us/step - loss: 0.0737 -
val_loss: 0.1181
Epoch 3/100
115092/115092 [==============================] - 22s 194us/step - loss: 0.0764 -
val_loss: 0.0860
Epoch 4/100
115092/115092 [==============================] - 22s 194us/step - loss: 0.0739 -
val_loss: 0.0678
Epoch 5/100
115092/115092 [==============================] - 22s 195us/step - loss: 0.0739 -
val_loss: 0.0711
Epoch 6/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0722 -
val_loss: 0.0813
Epoch 7/100
115092/115092 [==============================] - 23s 200us/step - loss: 0.0719 -
val_loss: 0.0804
Epoch 8/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0720 -
val_loss: 0.0655
Epoch 9/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0742 -
val_loss: 0.0601
Epoch 10/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0737 -
val_loss: 0.0818
Epoch 11/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0756 -
val_loss: 0.0926
Epoch 12/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0758 -
val_loss: 0.0714
Epoch 13/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0713 -
val_loss: 0.0701
Epoch 14/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0693 -
val_loss: 0.0711
Epoch 15/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0687 -
val_loss: 0.0642
Epoch 16/100
115092/115092 [==============================] - 23s 196us/step - loss: 0.0703 -
```

```
val_loss: 0.0780
Epoch 17/100
115092/115092 [==============================] - 23s 196us/step - loss: 0.0701 -
val_loss: 0.0735
Epoch 18/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0695 -
val_loss: 0.0585
Epoch 19/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0717 -
val_loss: 0.0697
Epoch 20/100
115092/115092 [==============================] - 23s 196us/step - loss: 0.0707 -
val_loss: 0.0789
Epoch 21/100
115092/115092 [==============================] - 22s 195us/step - loss: 0.0733 -
val_loss: 0.1511
Epoch 22/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0721 -
val_loss: 0.0790
Epoch 23/100
115092/115092 [==============================] - 23s 200us/step - loss: 0.0709 -
val_loss: 0.0829
Epoch 24/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0707 -
val_loss: 0.0685
Epoch 25/100
115092/115092 [==============================] - 23s 200us/step - loss: 0.0716 -
val_loss: 0.0736
Epoch 26/100
115092/115092 [==============================] - 23s 200us/step - loss: 0.0721 -
val_loss: 0.0740
Epoch 27/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0748 -
val_loss: 0.0728
Epoch 28/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0707 -
val_loss: 0.0929
Epoch 29/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0733 -
val_loss: 0.0890
Epoch 30/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0752 -
val_loss: 0.0850
Epoch 31/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0710 -
val_loss: 0.0595
Epoch 32/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0707 -
```

```
val_loss: 0.0747
Epoch 33/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0728 -
val_loss: 0.0675
Epoch 34/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0719 -
val_loss: 0.0697
Epoch 35/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0740 -
val_loss: 0.0795
Epoch 36/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0710 -
val_loss: 0.0683
Epoch 37/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0731 -
val_loss: 0.0781
Epoch 38/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0745 -
val_loss: 0.0718
Epoch 39/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0734 -
val_loss: 0.0787
Epoch 40/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0752 -
val_loss: 0.0794
Epoch 41/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0722 -
val_loss: 0.0714
Epoch 42/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0721 -
val_loss: 0.0701
Epoch 43/100
115092/115092 [==============================] - 23s 200us/step - loss: 0.0740 -
val_loss: 0.0860
Epoch 44/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0772 -
val_loss: 0.0869
Epoch 45/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0796 -
val_loss: 0.0860
Epoch 46/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0786 -
val_loss: 0.0884
Epoch 47/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0796 -
val_loss: 0.0814
Epoch 48/100
115092/115092 [==============================] - 23s 200us/step - loss: 0.0785 -
```

```
val_loss: 0.0787
Epoch 49/100
115092/115092 [==============================] - 23s 201us/step - loss: 0.0803 -
val_loss: 0.0678
Epoch 50/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0808 -
val_loss: 0.0814
Epoch 51/100
115092/115092 [==============================] - 23s 201us/step - loss: 0.0829 -
val_loss: 0.1045
Epoch 52/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0829 -
val_loss: 0.0662
Epoch 53/100
115092/115092 [==============================] - 23s 200us/step - loss: 0.0835 -
val_loss: 0.0787
Epoch 54/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0785 -
val_loss: 0.0783
Epoch 55/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0816 -
val_loss: 0.0749
Epoch 56/100
115092/115092 [==============================] - 23s 200us/step - loss: 0.0838 -
val_loss: 0.0772
Epoch 57/100
115092/115092 [==============================] - 23s 200us/step - loss: 0.0857 -
val_loss: 0.0802
Epoch 58/100
115092/115092 [==============================] - 23s 200us/step - loss: 0.0831 -
val_loss: 0.0699
Epoch 59/100
115092/115092 [==============================] - 23s 200us/step - loss: 0.0870 -
val_loss: 0.1185
Epoch 60/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0892 -
val_loss: 0.0802
Epoch 61/100
115092/115092 [==============================] - 23s 202us/step - loss: 0.0927 -
val_loss: 0.1162
Epoch 62/100
115092/115092 [==============================] - 23s 201us/step - loss: 0.0927 -
val_loss: 0.0962
Epoch 63/100
115092/115092 [==============================] - 23s 200us/step - loss: 0.0884 -
val_loss: 0.0918
Epoch 64/100
115092/115092 [==============================] - 23s 201us/step - loss: 0.0964 -
```

```
val_loss: 0.0854
Epoch 65/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.1056 -
val_loss: 0.1364
Epoch 66/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0972 -
val_loss: 0.0928
Epoch 67/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0973 -
val_loss: 0.1081
Epoch 68/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0907 -
val_loss: 0.0814
Epoch 69/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0923 -
val_loss: 0.0876
Epoch 70/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0964 -
val_loss: 0.1051
Epoch 71/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0923 -
val_loss: 0.0969
Epoch 72/100
115092/115092 [==============================] - 23s 200us/step - loss: 0.1034 -
val_loss: 0.1105
Epoch 73/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.1003 -
val_loss: 0.0949
Epoch 74/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.1005 -
val_loss: 0.1198
Epoch 75/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.1024 -
val_loss: 0.1012
Epoch 76/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0997 -
val_loss: 0.0986
Epoch 77/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0978 -
val_loss: 0.0970
Epoch 78/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0920 -
val_loss: 0.1229
Epoch 79/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0928 -
val_loss: 0.0985
Epoch 80/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0922 -
```

```
val_loss: 0.0995
Epoch 81/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0944 -
val_loss: 0.1065
Epoch 82/100
115092/115092 [==============================] - 23s 200us/step - loss: 0.0926 -
val_loss: 0.0971
Epoch 83/100
115092/115092 [==============================] - 23s 200us/step - loss: 0.0893 -
val_loss: 0.0885
Epoch 84/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0881 -
val_loss: 0.0918
Epoch 85/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0884 -
val_loss: 0.0824
Epoch 86/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0887 -
val_loss: 0.0928
Epoch 87/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0911 -
val_loss: 0.0817
Epoch 88/100
115092/115092 [==============================] - 23s 200us/step - loss: 0.0887 -
val_loss: 0.0821
Epoch 89/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0865 -
val_loss: 0.0840
Epoch 90/100
115092/115092 [==============================] - 23s 201us/step - loss: 0.0858 -
val_loss: 0.0842
Epoch 91/100
115092/115092 [==============================] - 23s 200us/step - loss: 0.0874 -
val_loss: 0.1015
Epoch 92/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0904 -
val_loss: 0.0797
Epoch 93/100
115092/115092 [==============================] - 23s 200us/step - loss: 0.0868 -
val_loss: 0.0750
Epoch 94/100
115092/115092 [==============================] - 23s 200us/step - loss: 0.0852 -
val_loss: 0.0682
Epoch 95/100
115092/115092 [==============================] - 23s 200us/step - loss: 0.0821 -
val_loss: 0.0874
Epoch 96/100
115092/115092 [==============================] - 23s 200us/step - loss: 0.0892 -
```

```
val_loss: 0.0916
Epoch 97/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0858 -
val_loss: 0.1160
Epoch 98/100
115092/115092 [==============================] - 23s 200us/step - loss: 0.0886 -
val_loss: 0.0850
Epoch 99/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0847 -
val_loss: 0.0831
Epoch 100/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0887 -
val_loss: 0.0837
```

[23]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[24]:
```
Y_test =np.round(Y_test, 0)
```

[25]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

```
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:2.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 2., predicted:2.
```

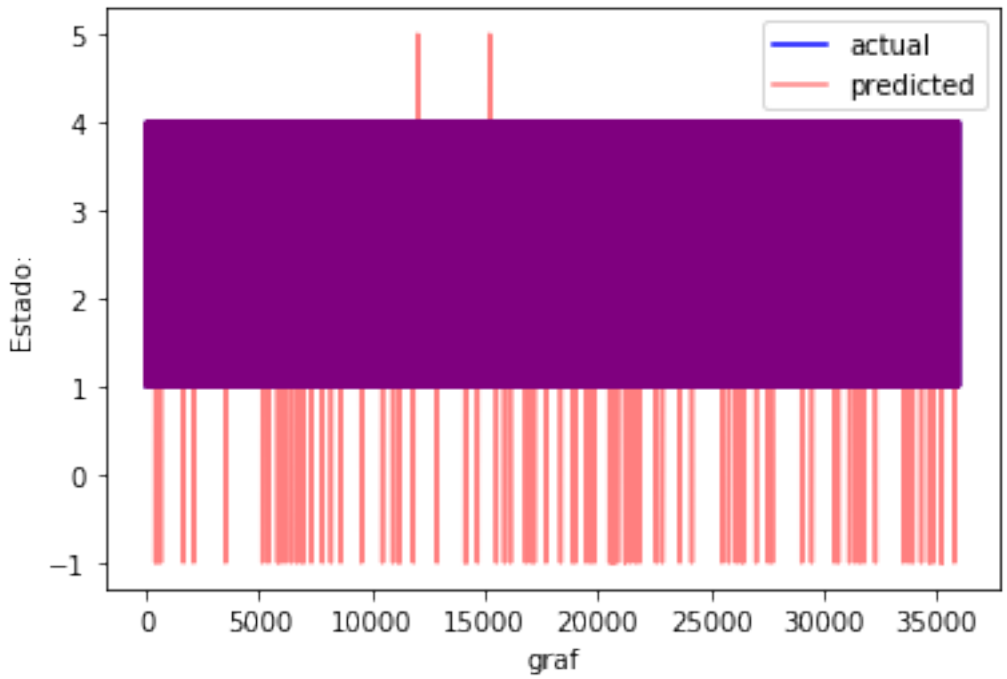
```
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

Evaluamos los resultados:

[26]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.9156114264749748
```

[27]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[28]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[28]:
```
0.9513164845552868
```

[ ]:
```

```

Wrist_elux4_mae_Adadel_final

August 30, 2019

1 Wrist Data

Librerías:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Wrist

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
```

```
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
```

```
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT2=np.concatenate((B1,A2),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
```

```
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=25*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT3=np.concatenate((B1,A2),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
```

```
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=35*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT4=np.concatenate((B1,A2),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
```

```
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT5=np.concatenate((B1,A2),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT6=np.concatenate((B1,A2),axis=1)
```

```
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
```

```
B1 = np.asmatrix(E)
B1=B1.T
MAT7=np.concatenate((B1,A2),axis=1)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
```

```
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT8=np.concatenate((B1,A2),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
```

```
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT9=np.concatenate((B1,A2),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
```

```
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT10=np.concatenate((B1,A2),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
```

```
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT11=np.concatenate((B1,A2),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
```

```
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT12=np.concatenate((B1,A2),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
```

```
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT13=np.concatenate((B1,A2),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
```

```
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=24*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT14=np.concatenate((B1,A2),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>4), 0)
MAT14= np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
```

```
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=29*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT15=np.concatenate((B1,A2),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
 ↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Dividimos en los conjuntos de validación y entrenamiento:

[18]:
```
X = MAT[:, 1:10]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)
```

Definimos la arquitectura de la red:

[19]:
```
model = Sequential([
    Dense(128, input_shape=(7,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
```

```
    Dense(60),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

```
WARNING: Logging before flag parsing goes to stderr.
W0830 02:57:15.235106 19500 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:74: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

W0830 02:57:15.248070 19500 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:517: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

W0830 02:57:15.252061 19500 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:4138: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

W0830 02:57:15.303957 19500 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-packages\keras\optimizers.py:790:
The name tf.train.Optimizer is deprecated. Please use
tf.compat.v1.train.Optimizer instead.
```

Entrenamos la red:

[20]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
W0830 02:57:15.474464 19500 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:986: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.

W0830 02:57:15.479451 19500 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:973: The name tf.assign is
deprecated. Please use tf.compat.v1.assign instead.
```

```
Train on 115092 samples, validate on 28773 samples
Epoch 1/100
115092/115092 [==============================] - 18s 153us/step - loss: 0.5981 -
val_loss: 0.4853
Epoch 2/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.3578 -
val_loss: 0.3916
Epoch 3/100
115092/115092 [==============================] - 17s 144us/step - loss: 0.2837 -
val_loss: 0.2953
Epoch 4/100
115092/115092 [==============================] - 17s 143us/step - loss: 0.2476 -
val_loss: 0.2450
Epoch 5/100
115092/115092 [==============================] - 17s 143us/step - loss: 0.2249 -
val_loss: 0.2368
Epoch 6/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.2082 -
val_loss: 0.2215
Epoch 7/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.1959 -
val_loss: 0.2067
Epoch 8/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.1851 -
val_loss: 0.2045
Epoch 9/100
115092/115092 [==============================] - 17s 143us/step - loss: 0.1767 -
val_loss: 0.1896
Epoch 10/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.1685 -
val_loss: 0.1614
Epoch 11/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.1611 -
val_loss: 0.1587
Epoch 12/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.1553 -
val_loss: 0.1551
Epoch 13/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1491 -
val_loss: 0.1609
Epoch 14/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1442 -
val_loss: 0.1488
Epoch 15/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1403 -
val_loss: 0.1376
Epoch 16/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.1367 -
```

```
val_loss: 0.1417
Epoch 17/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.1333 -
val_loss: 0.1326
Epoch 18/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1302 -
val_loss: 0.1405
Epoch 19/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1280 -
val_loss: 0.1306
Epoch 20/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1257 -
val_loss: 0.1420
Epoch 21/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1232 -
val_loss: 0.1345
Epoch 22/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.1212 -
val_loss: 0.1275
Epoch 23/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.1195 -
val_loss: 0.1238
Epoch 24/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.1181 -
val_loss: 0.1179
Epoch 25/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1161 -
val_loss: 0.1359
Epoch 26/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1146 -
val_loss: 0.1151
Epoch 27/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1133 -
val_loss: 0.1248
Epoch 28/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1118 -
val_loss: 0.1161
Epoch 29/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1108 -
val_loss: 0.1091
Epoch 30/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.1091 -
val_loss: 0.1273
Epoch 31/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.1082 -
val_loss: 0.1074
Epoch 32/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.1073 -
```

```
val_loss: 0.1139
Epoch 33/100
115092/115092 [==============================] - 17s 150us/step - loss: 0.1061 -
val_loss: 0.1155
Epoch 34/100
115092/115092 [==============================] - 17s 150us/step - loss: 0.1054 -
val_loss: 0.1239
Epoch 35/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1044 -
val_loss: 0.1176
Epoch 36/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1034 -
val_loss: 0.1070
Epoch 37/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.1029 -
val_loss: 0.1007
Epoch 38/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.1019 -
val_loss: 0.1068
Epoch 39/100
115092/115092 [==============================] - 17s 144us/step - loss: 0.1012 -
val_loss: 0.1040
Epoch 40/100
115092/115092 [==============================] - 17s 144us/step - loss: 0.0999 -
val_loss: 0.1152
Epoch 41/100
115092/115092 [==============================] - 17s 144us/step - loss: 0.0994 -
val_loss: 0.1103
Epoch 42/100
115092/115092 [==============================] - 17s 144us/step - loss: 0.0988 -
val_loss: 0.0938
Epoch 43/100
115092/115092 [==============================] - 16s 143us/step - loss: 0.0977 -
val_loss: 0.1051
Epoch 44/100
115092/115092 [==============================] - 17s 144us/step - loss: 0.0966 -
val_loss: 0.0984
Epoch 45/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0968 -
val_loss: 0.0995
Epoch 46/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0966 -
val_loss: 0.1114
Epoch 47/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0954 -
val_loss: 0.0865
Epoch 48/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0949 -
```

```
val_loss: 0.1311
Epoch 49/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0944 -
val_loss: 0.0965
Epoch 50/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0948 -
val_loss: 0.1125
Epoch 51/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0939 -
val_loss: 0.0943
Epoch 52/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0936 -
val_loss: 0.1040
Epoch 53/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0928 -
val_loss: 0.0948
Epoch 54/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0929 -
val_loss: 0.1133
Epoch 55/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0924 -
val_loss: 0.1039
Epoch 56/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0919 -
val_loss: 0.0919
Epoch 57/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0913 -
val_loss: 0.0947
Epoch 58/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0913 -
val_loss: 0.0878
Epoch 59/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0908 -
val_loss: 0.0959
Epoch 60/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0906 -
val_loss: 0.0934
Epoch 61/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0907 -
val_loss: 0.0970
Epoch 62/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0897 -
val_loss: 0.0790
Epoch 63/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0893 -
val_loss: 0.0900
Epoch 64/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0890 -
```

```
val_loss: 0.0902
Epoch 65/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0893 -
val_loss: 0.0993
Epoch 66/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0890 -
val_loss: 0.0909
Epoch 67/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0879 -
val_loss: 0.0962
Epoch 68/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0879 -
val_loss: 0.0826
Epoch 69/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0878 -
val_loss: 0.0959
Epoch 70/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0867 -
val_loss: 0.0899
Epoch 71/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0870 -
val_loss: 0.0879
Epoch 72/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0869 -
val_loss: 0.0857
Epoch 73/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0864 -
val_loss: 0.0885
Epoch 74/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0859 -
val_loss: 0.0884
Epoch 75/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0860 -
val_loss: 0.0903
Epoch 76/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0854 -
val_loss: 0.0837
Epoch 77/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0852 -
val_loss: 0.0872
Epoch 78/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0852 -
val_loss: 0.0799
Epoch 79/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0851 -
val_loss: 0.0975
Epoch 80/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0846 -
```

```
val_loss: 0.0803
Epoch 81/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0840 -
val_loss: 0.0840
Epoch 82/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0843 -
val_loss: 0.0860
Epoch 83/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0837 -
val_loss: 0.0811
Epoch 84/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0830 -
val_loss: 0.0721
Epoch 85/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0840 -
val_loss: 0.0874
Epoch 86/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0833 -
val_loss: 0.0930
Epoch 87/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0836 -
val_loss: 0.0794
Epoch 88/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0833 -
val_loss: 0.0780
Epoch 89/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0826 -
val_loss: 0.0867
Epoch 90/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0826 -
val_loss: 0.0909
Epoch 91/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0823 -
val_loss: 0.0825
Epoch 92/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0823 -
val_loss: 0.0823
Epoch 93/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0817 -
val_loss: 0.0848
Epoch 94/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0812 -
val_loss: 0.0904
Epoch 95/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0807 -
val_loss: 0.0983
Epoch 96/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0815 -
```

```
val_loss: 0.0928
Epoch 97/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0802 -
val_loss: 0.0954
Epoch 98/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0806 -
val_loss: 0.0882
Epoch 99/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0802 -
val_loss: 0.0773
Epoch 100/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0801 -
val_loss: 0.0824
```

[21]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
Train on 115092 samples, validate on 28773 samples
Epoch 1/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0797 -
val_loss: 0.0837
Epoch 2/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0796 -
val_loss: 0.0805
Epoch 3/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0796 -
val_loss: 0.0796
Epoch 4/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0800 -
val_loss: 0.0793
Epoch 5/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0799 -
val_loss: 0.0932
Epoch 6/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0795 -
val_loss: 0.0765
Epoch 7/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0796 -
val_loss: 0.0906
Epoch 8/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0791 -
val_loss: 0.0793
Epoch 9/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0791 -
```

```
val_loss: 0.0794
Epoch 10/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0790 -
val_loss: 0.0821
Epoch 11/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0793 -
val_loss: 0.0758
Epoch 12/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0787 -
val_loss: 0.0963
Epoch 13/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0787 -
val_loss: 0.0784
Epoch 14/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0788 -
val_loss: 0.0819
Epoch 15/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0788 -
val_loss: 0.0764
Epoch 16/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0781 -
val_loss: 0.0880
Epoch 17/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0778 -
val_loss: 0.0879
Epoch 18/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0783 -
val_loss: 0.0751
Epoch 19/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0778 -
val_loss: 0.0687
Epoch 20/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0776 -
val_loss: 0.0806
Epoch 21/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0780 -
val_loss: 0.0817
Epoch 22/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0775 -
val_loss: 0.0864
Epoch 23/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0772 -
val_loss: 0.0793
Epoch 24/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0778 -
val_loss: 0.0832
Epoch 25/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0772 -
```

```
val_loss: 0.0813
Epoch 26/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0767 -
val_loss: 0.0736
Epoch 27/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0772 -
val_loss: 0.0860
Epoch 28/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0771 -
val_loss: 0.0801
Epoch 29/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0770 -
val_loss: 0.0991
Epoch 30/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0771 -
val_loss: 0.1036
Epoch 31/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0761 -
val_loss: 0.0950
Epoch 32/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0762 -
val_loss: 0.1001
Epoch 33/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0765 -
val_loss: 0.0761
Epoch 34/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0761 -
val_loss: 0.0809
Epoch 35/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0761 -
val_loss: 0.0677
Epoch 36/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0764 -
val_loss: 0.0716
Epoch 37/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0764 -
val_loss: 0.0765
Epoch 38/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0764 -
val_loss: 0.0763
Epoch 39/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0766 -
val_loss: 0.0829
Epoch 40/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0757 -
val_loss: 0.0925
Epoch 41/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0760 -
```

```
val_loss: 0.0829
Epoch 42/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0758 -
val_loss: 0.1000
Epoch 43/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0759 -
val_loss: 0.0802
Epoch 44/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0755 -
val_loss: 0.0799
Epoch 45/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0759 -
val_loss: 0.0824
Epoch 46/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0760 -
val_loss: 0.0730
Epoch 47/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0749 -
val_loss: 0.0848
Epoch 48/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0756 -
val_loss: 0.0853
Epoch 49/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0745 -
val_loss: 0.0825
Epoch 50/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0743 -
val_loss: 0.0824
Epoch 51/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0746 -
val_loss: 0.0699
Epoch 52/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0749 -
val_loss: 0.0866
Epoch 53/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0743 -
val_loss: 0.0974
Epoch 54/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0740 -
val_loss: 0.0912
Epoch 55/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.0748 -
val_loss: 0.0906
Epoch 56/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.0750 -
val_loss: 0.0716
Epoch 57/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.0749 -
```

```
val_loss: 0.0729
Epoch 58/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0747 -
val_loss: 0.0753
Epoch 59/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0746 -
val_loss: 0.0700
Epoch 60/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.0737 -
val_loss: 0.0751
Epoch 61/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.0737 -
val_loss: 0.1002
Epoch 62/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.0742 -
val_loss: 0.0688
Epoch 63/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0740 -
val_loss: 0.0772
Epoch 64/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.0736 -
val_loss: 0.0728
Epoch 65/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0741 -
val_loss: 0.0880
Epoch 66/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0738 -
val_loss: 0.0909
Epoch 67/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.0732 -
val_loss: 0.0800
Epoch 68/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0728 -
val_loss: 0.0775
Epoch 69/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0732 -
val_loss: 0.0679
Epoch 70/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.0729 -
val_loss: 0.0850
Epoch 71/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.0728 -
val_loss: 0.0708
Epoch 72/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0725 -
val_loss: 0.0747
Epoch 73/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0738 -
```

```
val_loss: 0.0854
Epoch 74/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.0728 -
val_loss: 0.0832
Epoch 75/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.0736 -
val_loss: 0.0875
Epoch 76/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.0729 -
val_loss: 0.0771
Epoch 77/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0725 -
val_loss: 0.0878
Epoch 78/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.0728 -
val_loss: 0.0902
Epoch 79/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.0725 -
val_loss: 0.0839
Epoch 80/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.0728 -
val_loss: 0.0644
Epoch 81/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.0729 -
val_loss: 0.0902
Epoch 82/100
115092/115092 [==============================] - 16s 139us/step - loss: 0.0728 -
val_loss: 0.0809
Epoch 83/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0730 -
val_loss: 0.0822
Epoch 84/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0726 -
val_loss: 0.1080
Epoch 85/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0721 -
val_loss: 0.0805
Epoch 86/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0723 -
val_loss: 0.0691
Epoch 87/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0723 -
val_loss: 0.0856
Epoch 88/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0722 -
val_loss: 0.0666
Epoch 89/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0726 -
```

```
val_loss: 0.0805
Epoch 90/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0720 -
val_loss: 0.0878
Epoch 91/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0720 -
val_loss: 0.0819
Epoch 92/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0716 -
val_loss: 0.1019
Epoch 93/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0716 -
val_loss: 0.0771
Epoch 94/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0720 -
val_loss: 0.0651
Epoch 95/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0714 -
val_loss: 0.0921
Epoch 96/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0714 -
val_loss: 0.0632
Epoch 97/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0712 -
val_loss: 0.0749
Epoch 98/100
115092/115092 [==============================] - 16s 140us/step - loss: 0.0712 -
val_loss: 0.0757
Epoch 99/100
115092/115092 [==============================] - 16s 141us/step - loss: 0.0711 -
val_loss: 0.0767
Epoch 100/100
115092/115092 [==============================] - 16s 142us/step - loss: 0.0712 -
val_loss: 0.0709
```

[22]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[23]:
```
Y_test =np.round(Y_test, 0)
```

[24]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

```
Estado: 1., predicted:1.
```

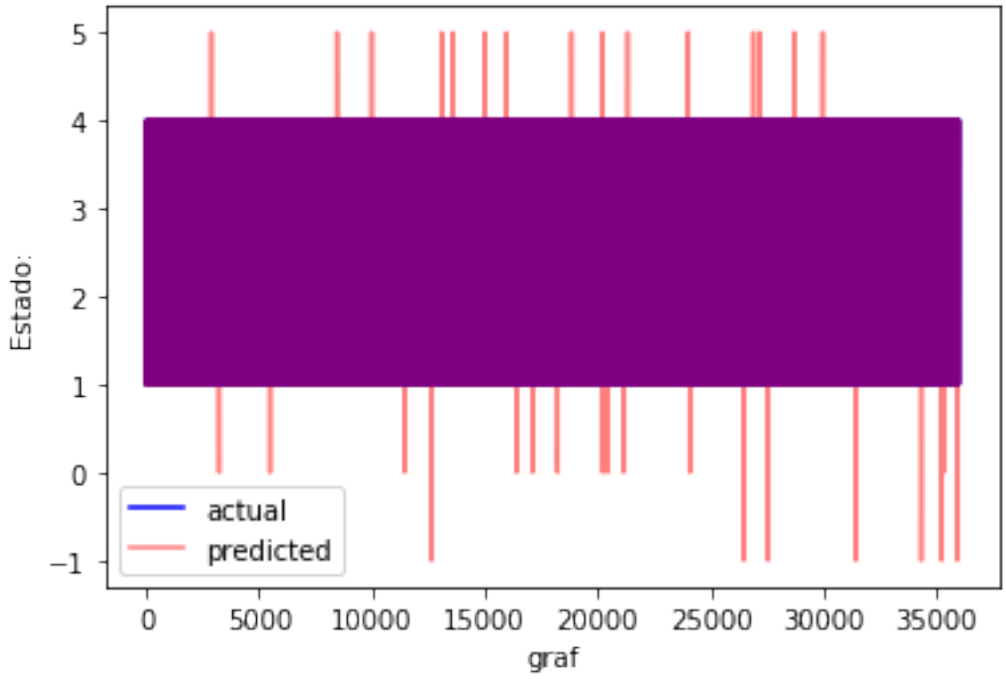
```
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:4.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

Evaluamos los resultados:

[25]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.9563134630940406
```

[26]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[27]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[27]:
```
0.9746156198737732
```

[ ]:
```

```

Wrist_elux5_mae_Adadel_final

August 30, 2019

1 Wrist Data

Librerías:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Wrist

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
```

```
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
```

```
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT2=np.concatenate((B1,A2),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
```

```
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=25*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT3=np.concatenate((B1,A2),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
```

```
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=35*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT4=np.concatenate((B1,A2),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
```

```
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT5=np.concatenate((B1,A2),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT6=np.concatenate((B1,A2),axis=1)
```

```
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
```

```
B1 = np.asmatrix(E)
B1=B1.T
MAT7=np.concatenate((B1,A2),axis=1)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
```

```
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT8=np.concatenate((B1,A2),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
```

```
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT9=np.concatenate((B1,A2),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
```

```
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT10=np.concatenate((B1,A2),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
```

```
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT11=np.concatenate((B1,A2),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
```

```
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT12=np.concatenate((B1,A2),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
```

```
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT13=np.concatenate((B1,A2),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
```

```
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=24*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT14=np.concatenate((B1,A2),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>4), 0)
MAT14= np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
```

```
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=29*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT15=np.concatenate((B1,A2),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
 ↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Dividimos en los conjuntos de validación y entrenamiento:

[18]:
```
X = MAT[:, 1:8]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)
```

Definimos la arquitectura de la red:

[19]:
```
model = Sequential([
    Dense(128, input_shape=(7,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
```

```
    Dense(48),
    Activation('elu'),
    Dense(24),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

```
WARNING: Logging before flag parsing goes to stderr.
W0830 03:54:14.022293  2160 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:74: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

W0830 03:54:14.035257  2160 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:517: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

W0830 03:54:14.038250  2160 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:4138: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

W0830 03:54:14.099122  2160 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-packages\keras\optimizers.py:790:
The name tf.train.Optimizer is deprecated. Please use
tf.compat.v1.train.Optimizer instead.
```

Entrenamos la red:

[20]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
W0830 03:54:14.282597  2160 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:986: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.

W0830 03:54:14.287618  2160 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:973: The name tf.assign is
```

```
deprecated. Please use tf.compat.v1.assign instead.


Train on 115092 samples, validate on 28773 samples
Epoch 1/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.5602 -
val_loss: 0.3840
Epoch 2/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.3263 -
val_loss: 0.3050
Epoch 3/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.2474 -
val_loss: 0.2530
Epoch 4/100
115092/115092 [==============================] - 17s 147us/step - loss: 0.2056 -
val_loss: 0.1918
Epoch 5/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.1809 -
val_loss: 0.1983
Epoch 6/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.1629 -
val_loss: 0.1572
Epoch 7/100
115092/115092 [==============================] - 17s 147us/step - loss: 0.1488 -
val_loss: 0.1333
Epoch 8/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.1381 -
val_loss: 0.1644
Epoch 9/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.1294 -
val_loss: 0.1237
Epoch 10/100
115092/115092 [==============================] - 17s 147us/step - loss: 0.1212 -
val_loss: 0.1229
Epoch 11/100
115092/115092 [==============================] - 17s 147us/step - loss: 0.1151 -
val_loss: 0.1237
Epoch 12/100
115092/115092 [==============================] - 17s 147us/step - loss: 0.1108 -
val_loss: 0.1039
Epoch 13/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.1070 -
val_loss: 0.1179
Epoch 14/100
115092/115092 [==============================] - 17s 147us/step - loss: 0.1043 -
val_loss: 0.1132
Epoch 15/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.1016 -
```

```
val_loss: 0.1195
Epoch 16/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0992 -
val_loss: 0.0958
Epoch 17/100
115092/115092 [==============================] - 17s 147us/step - loss: 0.0967 -
val_loss: 0.1008
Epoch 18/100
115092/115092 [==============================] - 17s 147us/step - loss: 0.0940 -
val_loss: 0.0958
Epoch 19/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0922 -
val_loss: 0.0937
Epoch 20/100
115092/115092 [==============================] - 17s 147us/step - loss: 0.0906 -
val_loss: 0.0863
Epoch 21/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0891 -
val_loss: 0.0955
Epoch 22/100
115092/115092 [==============================] - 17s 150us/step - loss: 0.0866 -
val_loss: 0.0945
Epoch 23/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0851 -
val_loss: 0.0992
Epoch 24/100
115092/115092 [==============================] - 17s 150us/step - loss: 0.0842 -
val_loss: 0.0842
Epoch 25/100
115092/115092 [==============================] - 17s 151us/step - loss: 0.0832 -
val_loss: 0.0779
Epoch 26/100
115092/115092 [==============================] - 17s 150us/step - loss: 0.0820 -
val_loss: 0.0835
Epoch 27/100
115092/115092 [==============================] - 17s 150us/step - loss: 0.0802 -
val_loss: 0.0794
Epoch 28/100
115092/115092 [==============================] - 17s 150us/step - loss: 0.0796 -
val_loss: 0.0862
Epoch 29/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0783 -
val_loss: 0.0811
Epoch 30/100
115092/115092 [==============================] - 17s 151us/step - loss: 0.0773 -
val_loss: 0.0861
Epoch 31/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0765 -
```

```
val_loss: 0.0766
Epoch 32/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0753 -
val_loss: 0.0825
Epoch 33/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0750 -
val_loss: 0.0814
Epoch 34/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0742 -
val_loss: 0.0780
Epoch 35/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0739 -
val_loss: 0.0733
Epoch 36/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0728 -
val_loss: 0.0835
Epoch 37/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0719 -
val_loss: 0.0794
Epoch 38/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0716 -
val_loss: 0.0684
Epoch 39/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0705 -
val_loss: 0.0664
Epoch 40/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0699 -
val_loss: 0.0733
Epoch 41/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0695 -
val_loss: 0.0664
Epoch 42/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0684 -
val_loss: 0.0692
Epoch 43/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0679 -
val_loss: 0.0770
Epoch 44/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0678 -
val_loss: 0.0744
Epoch 45/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0662 -
val_loss: 0.0679
Epoch 46/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0665 -
val_loss: 0.0746
Epoch 47/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0657 -
```

```
val_loss: 0.0679
Epoch 48/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0658 -
val_loss: 0.0716
Epoch 49/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0645 -
val_loss: 0.0652
Epoch 50/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0645 -
val_loss: 0.0657
Epoch 51/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0636 -
val_loss: 0.0739
Epoch 52/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0631 -
val_loss: 0.0647
Epoch 53/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0630 -
val_loss: 0.0574
Epoch 54/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0614 -
val_loss: 0.0680
Epoch 55/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0623 -
val_loss: 0.0770
Epoch 56/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0619 -
val_loss: 0.0688
Epoch 57/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0612 -
val_loss: 0.0705
Epoch 58/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0607 -
val_loss: 0.0710
Epoch 59/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0609 -
val_loss: 0.0608
Epoch 60/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0602 -
val_loss: 0.0642
Epoch 61/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0597 -
val_loss: 0.0708
Epoch 62/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0596 -
val_loss: 0.0682
Epoch 63/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0597 -
```

```
val_loss: 0.0752
Epoch 64/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0593 -
val_loss: 0.0508
Epoch 65/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0591 -
val_loss: 0.0642
Epoch 66/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0591 -
val_loss: 0.0578
Epoch 67/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0586 -
val_loss: 0.0770
Epoch 68/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0590 -
val_loss: 0.0637
Epoch 69/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0594 -
val_loss: 0.0693
Epoch 70/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0589 -
val_loss: 0.0557
Epoch 71/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0590 -
val_loss: 0.0630
Epoch 72/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0589 -
val_loss: 0.0676
Epoch 73/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0586 -
val_loss: 0.0659
Epoch 74/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0587 -
val_loss: 0.0720
Epoch 75/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0583 -
val_loss: 0.0552
Epoch 76/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0581 -
val_loss: 0.0598
Epoch 77/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0591 -
val_loss: 0.0625
Epoch 78/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0583 -
val_loss: 0.0569
Epoch 79/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0585 -
```

```
val_loss: 0.0590
Epoch 80/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0579 -
val_loss: 0.0606
Epoch 81/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0588 -
val_loss: 0.0603
Epoch 82/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0579 -
val_loss: 0.0524
Epoch 83/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0581 -
val_loss: 0.0503
Epoch 84/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0589 -
val_loss: 0.0539
Epoch 85/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0592 -
val_loss: 0.0807
Epoch 86/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0588 -
val_loss: 0.0827
Epoch 87/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0587 -
val_loss: 0.0531
Epoch 88/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0584 -
val_loss: 0.0628
Epoch 89/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0586 -
val_loss: 0.0596
Epoch 90/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0585 -
val_loss: 0.0838
Epoch 91/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0611 -
val_loss: 0.0542
Epoch 92/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0589 -
val_loss: 0.0629
Epoch 93/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0582 -
val_loss: 0.0521
Epoch 94/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0581 -
val_loss: 0.0612
Epoch 95/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0577 -
```

```
val_loss: 0.0614
Epoch 96/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0580 -
val_loss: 0.0480
Epoch 97/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0568 -
val_loss: 0.0571
Epoch 98/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0567 -
val_loss: 0.0641
Epoch 99/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0566 -
val_loss: 0.0664
Epoch 100/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0568 -
val_loss: 0.0740
```

[21]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
Train on 115092 samples, validate on 28773 samples
Epoch 1/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0565 -
val_loss: 0.0680
Epoch 2/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0565 -
val_loss: 0.0575
Epoch 3/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0559 -
val_loss: 0.0643
Epoch 4/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0560 -
val_loss: 0.0527
Epoch 5/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0563 -
val_loss: 0.0554
Epoch 6/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0568 -
val_loss: 0.0583
Epoch 7/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0564 -
val_loss: 0.0547
Epoch 8/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0566 -
```

```
val_loss: 0.0531
Epoch 9/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0563 -
val_loss: 0.0563
Epoch 10/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0561 -
val_loss: 0.0626
Epoch 11/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0563 -
val_loss: 0.0681
Epoch 12/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0553 -
val_loss: 0.0688
Epoch 13/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0563 -
val_loss: 0.0758
Epoch 14/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0559 -
val_loss: 0.0578
Epoch 15/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0555 -
val_loss: 0.0578
Epoch 16/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0560 -
val_loss: 0.0690
Epoch 17/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0556 -
val_loss: 0.0574
Epoch 18/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0557 -
val_loss: 0.0524
Epoch 19/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0548 -
val_loss: 0.0586
Epoch 20/100
115092/115092 [==============================] - 18s 153us/step - loss: 0.0545 -
val_loss: 0.0523
Epoch 21/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0542 -
val_loss: 0.0524
Epoch 22/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0544 -
val_loss: 0.0478
Epoch 23/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0549 -
val_loss: 0.0727
Epoch 24/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0542 -
```

```
val_loss: 0.0519
Epoch 25/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0532 -
val_loss: 0.0471
Epoch 26/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0532 -
val_loss: 0.0558
Epoch 27/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0536 -
val_loss: 0.0575
Epoch 28/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0528 -
val_loss: 0.0562
Epoch 29/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0535 -
val_loss: 0.0565
Epoch 30/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0531 -
val_loss: 0.0539
Epoch 31/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0536 -
val_loss: 0.0456
Epoch 32/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0537 -
val_loss: 0.0623
Epoch 33/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0534 -
val_loss: 0.0597
Epoch 34/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0530 -
val_loss: 0.0563
Epoch 35/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0521 -
val_loss: 0.0543
Epoch 36/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0526 -
val_loss: 0.0552
Epoch 37/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0513 -
val_loss: 0.0542
Epoch 38/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0516 -
val_loss: 0.0495
Epoch 39/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0516 -
val_loss: 0.0596
Epoch 40/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0515 -
```

```
val_loss: 0.0639
Epoch 41/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0529 -
val_loss: 0.0657
Epoch 42/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0523 -
val_loss: 0.0691
Epoch 43/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0513 -
val_loss: 0.0578
Epoch 44/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0529 -
val_loss: 0.0604
Epoch 45/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0532 -
val_loss: 0.0531
Epoch 46/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0526 -
val_loss: 0.0546
Epoch 47/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0522 -
val_loss: 0.0504
Epoch 48/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0532 -
val_loss: 0.0509
Epoch 49/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0529 -
val_loss: 0.0514
Epoch 50/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0535 -
val_loss: 0.0684
Epoch 51/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0538 -
val_loss: 0.0458
Epoch 52/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0540 -
val_loss: 0.0534
Epoch 53/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0542 -
val_loss: 0.0659
Epoch 54/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0543 -
val_loss: 0.0577
Epoch 55/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0543 -
val_loss: 0.0458
Epoch 56/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0550 -
```

```
val_loss: 0.0544
Epoch 57/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0554 -
val_loss: 0.0469
Epoch 58/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0548 -
val_loss: 0.0557
Epoch 59/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0552 -
val_loss: 0.0582
Epoch 60/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0550 -
val_loss: 0.0472
Epoch 61/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0564 -
val_loss: 0.0575
Epoch 62/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0567 -
val_loss: 0.0543
Epoch 63/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0555 -
val_loss: 0.0552
Epoch 64/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0571 -
val_loss: 0.0553
Epoch 65/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0561 -
val_loss: 0.0582
Epoch 66/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0575 -
val_loss: 0.0684
Epoch 67/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0578 -
val_loss: 0.0560
Epoch 68/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0578 -
val_loss: 0.0629
Epoch 69/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0589 -
val_loss: 0.0807
Epoch 70/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0585 -
val_loss: 0.0635
Epoch 71/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0585 -
val_loss: 0.0523
Epoch 72/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0599 -
```

```
val_loss: 0.0562
Epoch 73/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0588 -
val_loss: 0.0645
Epoch 74/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0598 -
val_loss: 0.0562
Epoch 75/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0595 -
val_loss: 0.0623
Epoch 76/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0605 -
val_loss: 0.0698
Epoch 77/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0603 -
val_loss: 0.0646
Epoch 78/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0612 -
val_loss: 0.0666
Epoch 79/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0610 -
val_loss: 0.0628
Epoch 80/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0623 -
val_loss: 0.0603
Epoch 81/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0622 -
val_loss: 0.0658
Epoch 82/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0619 -
val_loss: 0.0685
Epoch 83/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0651 -
val_loss: 0.0595
Epoch 84/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0645 -
val_loss: 0.0588
Epoch 85/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0646 -
val_loss: 0.0637
Epoch 86/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0621 -
val_loss: 0.0624
Epoch 87/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0617 -
val_loss: 0.0700
Epoch 88/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0623 -
```

```
val_loss: 0.0793
Epoch 89/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0633 -
val_loss: 0.0640
Epoch 90/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0634 -
val_loss: 0.0619
Epoch 91/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0619 -
val_loss: 0.0783
Epoch 92/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0630 -
val_loss: 0.0699
Epoch 93/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0632 -
val_loss: 0.0626
Epoch 94/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0624 -
val_loss: 0.0636
Epoch 95/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0624 -
val_loss: 0.0585
Epoch 96/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.0627 -
val_loss: 0.0878
Epoch 97/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0620 -
val_loss: 0.0707
Epoch 98/100
115092/115092 [==============================] - 18s 157us/step - loss: 0.0615 -
val_loss: 0.0537
Epoch 99/100
115092/115092 [==============================] - 18s 156us/step - loss: 0.0627 -
val_loss: 0.0623
Epoch 100/100
115092/115092 [==============================] - 18s 154us/step - loss: 0.0624 -
val_loss: 0.0573
```

[22]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[23]:
```
Y_test =np.round(Y_test, 0)
```

[24]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
```

```
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
↪array2string(prediction))
```

```
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:2.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

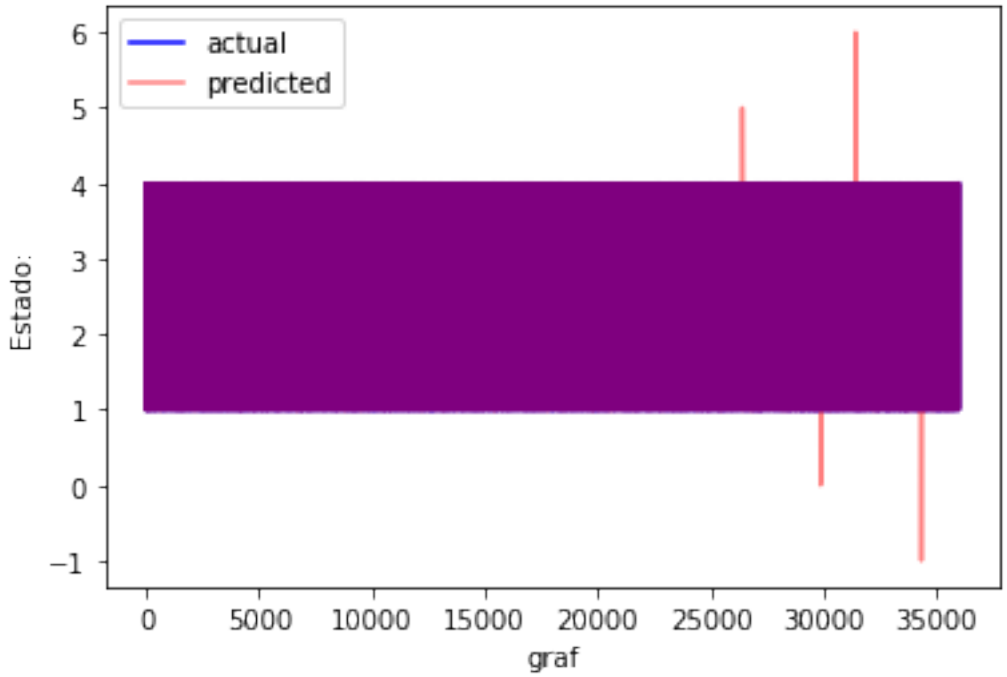
Evaluamos los resultados:

[25]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.9529768306439741
```

[26]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
```

```
plt.legend(loc="best")
plt.show()
```



[27]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[27]: 0.9722523424250007

[ ]:

Wrist_elux7_mae_Adadel_final

August 30, 2019

1 Wrist Data

Librerías:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Wrist

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
```

```
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
```

```
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT2=np.concatenate((B1,A2),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
```

```
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=25*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT3=np.concatenate((B1,A2),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
```

```
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=35*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT4=np.concatenate((B1,A2),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
```

```
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT5=np.concatenate((B1,A2),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT6=np.concatenate((B1,A2),axis=1)
```

```
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
```

```
B1 = np.asmatrix(E)
B1=B1.T
MAT7=np.concatenate((B1,A2),axis=1)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
```

```
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT8=np.concatenate((B1,A2),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
```

```
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT9=np.concatenate((B1,A2),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
```

```
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT10=np.concatenate((B1,A2),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
```

```
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT11=np.concatenate((B1,A2),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
```

```
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT12=np.concatenate((B1,A2),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
```

```
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT13=np.concatenate((B1,A2),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
```

```
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=24*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT14=np.concatenate((B1,A2),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>4), 0)
MAT14= np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
```

```
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=29*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT15=np.concatenate((B1,A2),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
 ↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Dividimos en los conjuntos de validación y entrenamiento:

[18]:
```
X = MAT[:, 1:8]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)
```

Definimos la arquitectura de la red:

[19]:
```
model = Sequential([
    Dense(128, input_shape=(7,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
```

```
    Dense(60),
    Activation('elu'),
    Dense(40),
    Activation('elu'),
    Dense(40),
    Activation('elu'),
    Dense(30),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

```
WARNING: Logging before flag parsing goes to stderr.
W0830 05:14:49.241491 18176 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:74: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

W0830 05:14:49.253420 18176 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:517: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

W0830 05:14:49.257409 18176 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:4138: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

W0830 05:14:49.350161 18176 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-packages\keras\optimizers.py:790:
The name tf.train.Optimizer is deprecated. Please use
tf.compat.v1.train.Optimizer instead.
```

Entrenamos la red:

[20]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
W0830 05:14:49.582539 18176 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:986: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.
```

```
W0830 05:14:49.587561 18176 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:973: The name tf.assign is
deprecated. Please use tf.compat.v1.assign instead.


Train on 115092 samples, validate on 28773 samples
Epoch 1/100
115092/115092 [==============================] - 23s 202us/step - loss: 0.5122 -
val_loss: 0.3439
Epoch 2/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.2748 -
val_loss: 0.2154
Epoch 3/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.2046 -
val_loss: 0.1900
Epoch 4/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.1662 -
val_loss: 0.1848
Epoch 5/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.1427 -
val_loss: 0.1321
Epoch 6/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.1279 -
val_loss: 0.1200
Epoch 7/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.1178 -
val_loss: 0.1525
Epoch 8/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.1104 -
val_loss: 0.1150
Epoch 9/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.1053 -
val_loss: 0.1538
Epoch 10/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.1007 -
val_loss: 0.0962
Epoch 11/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0966 -
val_loss: 0.1025
Epoch 12/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0937 -
val_loss: 0.0922
Epoch 13/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0917 -
val_loss: 0.1028
Epoch 14/100
```

```
115092/115092 [==============================] - 22s 192us/step - loss: 0.0882 -
val_loss: 0.0791
Epoch 15/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0873 -
val_loss: 0.0865
Epoch 16/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0846 -
val_loss: 0.0852
Epoch 17/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0819 -
val_loss: 0.0801
Epoch 18/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0823 -
val_loss: 0.0733
Epoch 19/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0803 -
val_loss: 0.0948
Epoch 20/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0785 -
val_loss: 0.0756
Epoch 21/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0778 -
val_loss: 0.0694
Epoch 22/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0749 -
val_loss: 0.0786
Epoch 23/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0730 -
val_loss: 0.0805
Epoch 24/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0722 -
val_loss: 0.0727
Epoch 25/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0708 -
val_loss: 0.0732
Epoch 26/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0703 -
val_loss: 0.0680
Epoch 27/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0692 -
val_loss: 0.0843
Epoch 28/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0682 -
val_loss: 0.0700
Epoch 29/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0675 -
val_loss: 0.0639
Epoch 30/100
```

```
115092/115092 [==============================] - 22s 192us/step - loss: 0.0672 -
val_loss: 0.0802
Epoch 31/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0671 -
val_loss: 0.0650
Epoch 32/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0655 -
val_loss: 0.0764
Epoch 33/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0662 -
val_loss: 0.0678
Epoch 34/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0662 -
val_loss: 0.0671
Epoch 35/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0652 -
val_loss: 0.0736
Epoch 36/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0639 -
val_loss: 0.0715
Epoch 37/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0641 -
val_loss: 0.0744
Epoch 38/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0630 -
val_loss: 0.0803
Epoch 39/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0628 -
val_loss: 0.0777
Epoch 40/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0628 -
val_loss: 0.0691
Epoch 41/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0618 -
val_loss: 0.0874
Epoch 42/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0610 -
val_loss: 0.0557
Epoch 43/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0605 -
val_loss: 0.0603
Epoch 44/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0613 -
val_loss: 0.0653
Epoch 45/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0599 -
val_loss: 0.0741
Epoch 46/100
```

```
115092/115092 [==============================] - 22s 193us/step - loss: 0.0604 -
val_loss: 0.0599
Epoch 47/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0590 -
val_loss: 0.0632
Epoch 48/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0600 -
val_loss: 0.0571
Epoch 49/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0587 -
val_loss: 0.0716
Epoch 50/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0601 -
val_loss: 0.0570
Epoch 51/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0590 -
val_loss: 0.0614
Epoch 52/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0587 -
val_loss: 0.0561
Epoch 53/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0589 -
val_loss: 0.0693
Epoch 54/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0577 -
val_loss: 0.0624
Epoch 55/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0573 -
val_loss: 0.0656
Epoch 56/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0577 -
val_loss: 0.0634
Epoch 57/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0582 -
val_loss: 0.0544
Epoch 58/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0585 -
val_loss: 0.0712
Epoch 59/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0571 -
val_loss: 0.0646
Epoch 60/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0576 -
val_loss: 0.0605
Epoch 61/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0579 -
val_loss: 0.0641
Epoch 62/100
```

```
115092/115092 [==============================] - 22s 189us/step - loss: 0.0572 -
val_loss: 0.0665
Epoch 63/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0574 -
val_loss: 0.0521
Epoch 64/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0574 -
val_loss: 0.0800
Epoch 65/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0565 -
val_loss: 0.0603
Epoch 66/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0564 -
val_loss: 0.0506
Epoch 67/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0552 -
val_loss: 0.0655
Epoch 68/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0564 -
val_loss: 0.0620
Epoch 69/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0561 -
val_loss: 0.0497
Epoch 70/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0563 -
val_loss: 0.0559
Epoch 71/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0555 -
val_loss: 0.0557
Epoch 72/100
115092/115092 [==============================] - 22s 187us/step - loss: 0.0570 -
val_loss: 0.0674
Epoch 73/100
115092/115092 [==============================] - 22s 187us/step - loss: 0.0565 -
val_loss: 0.0678
Epoch 74/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0572 -
val_loss: 0.0485
Epoch 75/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0563 -
val_loss: 0.0541
Epoch 76/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0556 -
val_loss: 0.0630
Epoch 77/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0551 -
val_loss: 0.0515
Epoch 78/100
```

```
115092/115092 [==============================] - 22s 188us/step - loss: 0.0537 -
val_loss: 0.0618
Epoch 79/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0552 -
val_loss: 0.0550
Epoch 80/100
115092/115092 [==============================] - 22s 187us/step - loss: 0.0559 -
val_loss: 0.0823
Epoch 81/100
115092/115092 [==============================] - 22s 187us/step - loss: 0.0549 -
val_loss: 0.0551
Epoch 82/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0557 -
val_loss: 0.0585
Epoch 83/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0553 -
val_loss: 0.0603
Epoch 84/100
115092/115092 [==============================] - 22s 187us/step - loss: 0.0538 -
val_loss: 0.0525
Epoch 85/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0542 -
val_loss: 0.0553
Epoch 86/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0538 -
val_loss: 0.0684
Epoch 87/100
115092/115092 [==============================] - 22s 187us/step - loss: 0.0543 -
val_loss: 0.0602
Epoch 88/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0545 -
val_loss: 0.0699
Epoch 89/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0536 -
val_loss: 0.0539
Epoch 90/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0535 -
val_loss: 0.0604
Epoch 91/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0539 -
val_loss: 0.0556
Epoch 92/100
115092/115092 [==============================] - 22s 187us/step - loss: 0.0521 -
val_loss: 0.0605
Epoch 93/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0539 -
val_loss: 0.0733
Epoch 94/100
```

```
115092/115092 [==============================] - 22s 188us/step - loss: 0.0544 -
val_loss: 0.0562
Epoch 95/100
115092/115092 [==============================] - 22s 187us/step - loss: 0.0527 -
val_loss: 0.0596
Epoch 96/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0543 -
val_loss: 0.0471
Epoch 97/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0530 -
val_loss: 0.0613
Epoch 98/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0531 -
val_loss: 0.0451
Epoch 99/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0525 -
val_loss: 0.0527
Epoch 100/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0524 -
val_loss: 0.0579
```

[21]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
Train on 115092 samples, validate on 28773 samples
Epoch 1/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0532 -
val_loss: 0.0545
Epoch 2/100
115092/115092 [==============================] - 22s 195us/step - loss: 0.0520 -
val_loss: 0.0481
Epoch 3/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0535 -
val_loss: 0.0554
Epoch 4/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0526 -
val_loss: 0.0483
Epoch 5/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0524 -
val_loss: 0.0850
Epoch 6/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0519 -
val_loss: 0.0532
Epoch 7/100
```

```
115092/115092 [==============================] - 22s 193us/step - loss: 0.0518 -
val_loss: 0.0584
Epoch 8/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0530 -
val_loss: 0.0617
Epoch 9/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0537 -
val_loss: 0.0578
Epoch 10/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0529 -
val_loss: 0.0478
Epoch 11/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0536 -
val_loss: 0.0550
Epoch 12/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0519 -
val_loss: 0.0536
Epoch 13/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0549 -
val_loss: 0.0505
Epoch 14/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0534 -
val_loss: 0.0582
Epoch 15/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0530 -
val_loss: 0.0547
Epoch 16/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0535 -
val_loss: 0.0683
Epoch 17/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0536 -
val_loss: 0.0536
Epoch 18/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0544 -
val_loss: 0.0621
Epoch 19/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0572 -
val_loss: 0.0762
Epoch 20/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0533 -
val_loss: 0.0754
Epoch 21/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0525 -
val_loss: 0.0673
Epoch 22/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0541 -
val_loss: 0.0479
Epoch 23/100
```

```
115092/115092 [==============================] - 22s 192us/step - loss: 0.0534 -
val_loss: 0.0596
Epoch 24/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0547 -
val_loss: 0.0544
Epoch 25/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0555 -
val_loss: 0.0598
Epoch 26/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0561 -
val_loss: 0.0525
Epoch 27/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0543 -
val_loss: 0.0626
Epoch 28/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0549 -
val_loss: 0.0595
Epoch 29/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0545 -
val_loss: 0.0535
Epoch 30/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0557 -
val_loss: 0.0547
Epoch 31/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0558 -
val_loss: 0.0535
Epoch 32/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0560 -
val_loss: 0.0496
Epoch 33/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0557 -
val_loss: 0.0606
Epoch 34/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0563 -
val_loss: 0.0480
Epoch 35/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0563 -
val_loss: 0.0556
Epoch 36/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0551 -
val_loss: 0.0608
Epoch 37/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0570 -
val_loss: 0.0544
Epoch 38/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0566 -
val_loss: 0.0592
Epoch 39/100
```

```
115092/115092 [==============================] - 22s 193us/step - loss: 0.0581 -
val_loss: 0.0585
Epoch 40/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0579 -
val_loss: 0.0672
Epoch 41/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0583 -
val_loss: 0.0603
Epoch 42/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0598 -
val_loss: 0.0579
Epoch 43/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0574 -
val_loss: 0.0502
Epoch 44/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0589 -
val_loss: 0.0623
Epoch 45/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0607 -
val_loss: 0.0509
Epoch 46/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0600 -
val_loss: 0.0751
Epoch 47/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0611 -
val_loss: 0.0525
Epoch 48/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0606 -
val_loss: 0.0675
Epoch 49/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0626 -
val_loss: 0.0588
Epoch 50/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0646 -
val_loss: 0.0640
Epoch 51/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0662 -
val_loss: 0.0703
Epoch 52/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0650 -
val_loss: 0.0725
Epoch 53/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0627 -
val_loss: 0.0709
Epoch 54/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0637 -
val_loss: 0.0587
Epoch 55/100
```

```
115092/115092 [==============================] - 22s 193us/step - loss: 0.0639 -
val_loss: 0.0523
Epoch 56/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0624 -
val_loss: 0.0635
Epoch 57/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0603 -
val_loss: 0.0542
Epoch 58/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0600 -
val_loss: 0.0534
Epoch 59/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0614 -
val_loss: 0.0624
Epoch 60/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0589 -
val_loss: 0.0615
Epoch 61/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0609 -
val_loss: 0.0756
Epoch 62/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0627 -
val_loss: 0.0837
Epoch 63/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0604 -
val_loss: 0.0702
Epoch 64/100
115092/115092 [==============================] - 22s 194us/step - loss: 0.0601 -
val_loss: 0.0566
Epoch 65/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0604 -
val_loss: 0.0750
Epoch 66/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0604 -
val_loss: 0.0726
Epoch 67/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0610 -
val_loss: 0.0601
Epoch 68/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0626 -
val_loss: 0.0865
Epoch 69/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0613 -
val_loss: 0.0710
Epoch 70/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0636 -
val_loss: 0.0581
Epoch 71/100
```

```
115092/115092 [==============================] - 22s 189us/step - loss: 0.0610 -
val_loss: 0.0733
Epoch 72/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0608 -
val_loss: 0.0683
Epoch 73/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0608 -
val_loss: 0.0612
Epoch 74/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0615 -
val_loss: 0.0547
Epoch 75/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0620 -
val_loss: 0.0683
Epoch 76/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0640 -
val_loss: 0.0535
Epoch 77/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0628 -
val_loss: 0.0631
Epoch 78/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0633 -
val_loss: 0.0658
Epoch 79/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0642 -
val_loss: 0.0629
Epoch 80/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0645 -
val_loss: 0.0692
Epoch 81/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0673 -
val_loss: 0.0617
Epoch 82/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0636 -
val_loss: 0.0592
Epoch 83/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0651 -
val_loss: 0.0693
Epoch 84/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0688 -
val_loss: 0.0576
Epoch 85/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0667 -
val_loss: 0.0672
Epoch 86/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0659 -
val_loss: 0.0589
Epoch 87/100
```

```
115092/115092 [==============================] - 22s 191us/step - loss: 0.0675 -
val_loss: 0.0562
Epoch 88/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0672 -
val_loss: 0.0515
Epoch 89/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0699 -
val_loss: 0.0720
Epoch 90/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0689 -
val_loss: 0.0605
Epoch 91/100
115092/115092 [==============================] - 22s 189us/step - loss: 0.0662 -
val_loss: 0.0653
Epoch 92/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0654 -
val_loss: 0.0574
Epoch 93/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0675 -
val_loss: 0.0791
Epoch 94/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0648 -
val_loss: 0.0865
Epoch 95/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0676 -
val_loss: 0.0643
Epoch 96/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0660 -
val_loss: 0.0679
Epoch 97/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0743 -
val_loss: 0.0790
Epoch 98/100
115092/115092 [==============================] - 22s 188us/step - loss: 0.0694 -
val_loss: 0.0648
Epoch 99/100
115092/115092 [==============================] - 22s 192us/step - loss: 0.0689 -
val_loss: 0.0656
Epoch 100/100
115092/115092 [==============================] - 22s 193us/step - loss: 0.0681 -
val_loss: 0.0645
```

[22]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[23]:
```
Y_test =np.round(Y_test, 0)
```

[24]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

```
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:3.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:3.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

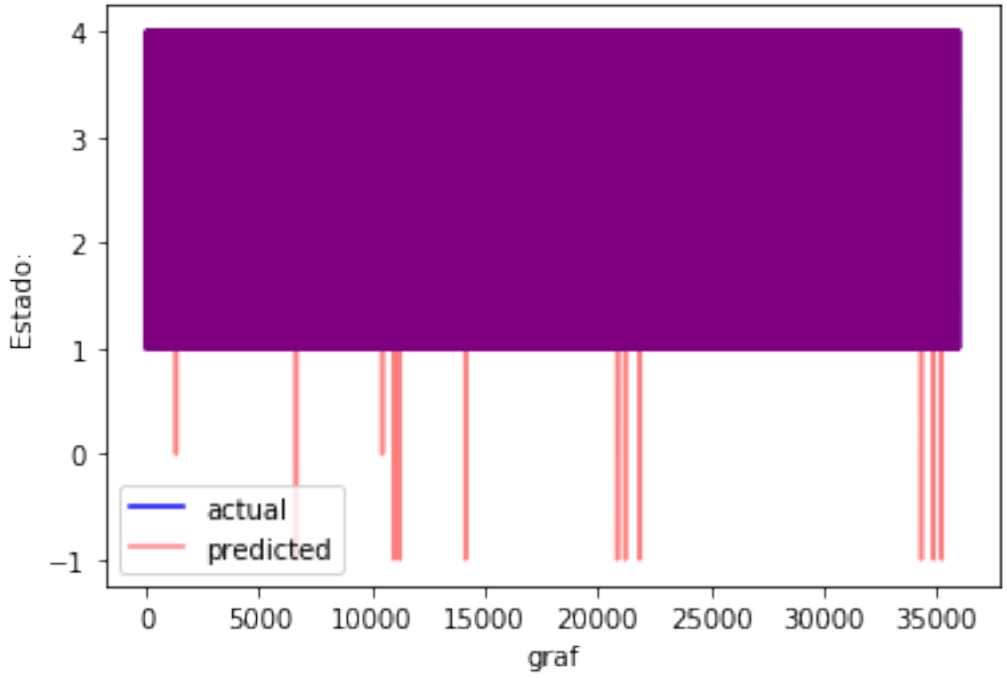
Evaluamos los resultados:

[25]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.9381804505653084
```

[26]:
```
plt.plot((Y_val),
    color="b", label="actual")
```

```
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[27]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[27]: 0.9675257875274558

[ ]:

Wrist_elux4_mae_Adadel_final

August 30, 2019

1 Wrist Data

Librerías:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Wrist

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
```

```
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
```

```
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT2=np.concatenate((B1,A2),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
```

```
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=25*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT3=np.concatenate((B1,A2),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
```

```
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=35*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT4=np.concatenate((B1,A2),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
```

```
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT5=np.concatenate((B1,A2),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT6=np.concatenate((B1,A2),axis=1)
```

```
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
```

```
B1 = np.asmatrix(E)
B1=B1.T
MAT7=np.concatenate((B1,A2),axis=1)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
```

```
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT8=np.concatenate((B1,A2),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
```

```
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT9=np.concatenate((B1,A2),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
```

```
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT10=np.concatenate((B1,A2),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
```

```
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT11=np.concatenate((B1,A2),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
```

```
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT12=np.concatenate((B1,A2),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
```

```
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT13=np.concatenate((B1,A2),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
```

```
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=24*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT14=np.concatenate((B1,A2),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>4), 0)
MAT14= np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
```

```
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=29*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT15=np.concatenate((B1,A2),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
 ↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Dividimos en los conjuntos de validación y entrenamiento:

[18]:
```
X = MAT[:, 1:10]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)
```

Definimos la arquitectura de la red:

[19]:
```
model = Sequential([
    Dense(128, input_shape=(7,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
```

```
    Dense(60),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

```
WARNING: Logging before flag parsing goes to stderr.
W0830 01:06:52.774749  7756 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:74: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

W0830 01:06:52.807661  7756 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:517: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

W0830 01:06:52.818632  7756 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:4138: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

W0830 01:06:52.884494  7756 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-packages\keras\optimizers.py:790:
The name tf.train.Optimizer is deprecated. Please use
tf.compat.v1.train.Optimizer instead.
```

Entrenamos la red:

[20]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
W0830 01:06:53.077977  7756 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:986: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.

W0830 01:06:53.082966  7756 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:973: The name tf.assign is
deprecated. Please use tf.compat.v1.assign instead.
```

```
Train on 115092 samples, validate on 28773 samples
Epoch 1/100
115092/115092 [==============================] - 21s 183us/step - loss: 0.5862 -
val_loss: 0.4021
Epoch 2/100
115092/115092 [==============================] - 18s 155us/step - loss: 0.3596 -
val_loss: 0.3465
Epoch 3/100
115092/115092 [==============================] - 17s 147us/step - loss: 0.2868 -
val_loss: 0.3225
Epoch 4/100
115092/115092 [==============================] - 17s 147us/step - loss: 0.2499 -
val_loss: 0.2759
Epoch 5/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.2254 -
val_loss: 0.2536
Epoch 6/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.2070 -
val_loss: 0.1961
Epoch 7/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.1929 -
val_loss: 0.2113
Epoch 8/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.1796 -
val_loss: 0.2095
Epoch 9/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.1695 -
val_loss: 0.1752
Epoch 10/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.1601 -
val_loss: 0.1568
Epoch 11/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.1529 -
val_loss: 0.1502
Epoch 12/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.1466 -
val_loss: 0.1469
Epoch 13/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.1413 -
val_loss: 0.1252
Epoch 14/100
115092/115092 [==============================] - 18s 153us/step - loss: 0.1367 -
val_loss: 0.1482
Epoch 15/100
115092/115092 [==============================] - 17s 150us/step - loss: 0.1338 -
val_loss: 0.1427
Epoch 16/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.1296 -
```

```
val_loss: 0.1588
Epoch 17/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.1276 -
val_loss: 0.1366
Epoch 18/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.1250 -
val_loss: 0.1260
Epoch 19/100
115092/115092 [==============================] - 17s 151us/step - loss: 0.1227 -
val_loss: 0.1468
Epoch 20/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.1205 -
val_loss: 0.1227
Epoch 21/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.1185 -
val_loss: 0.1260
Epoch 22/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.1161 -
val_loss: 0.1305
Epoch 23/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.1148 -
val_loss: 0.1211
Epoch 24/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.1139 -
val_loss: 0.1313
Epoch 25/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.1122 -
val_loss: 0.1108
Epoch 26/100
115092/115092 [==============================] - 17s 150us/step - loss: 0.1110 -
val_loss: 0.1108
Epoch 27/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.1100 -
val_loss: 0.1108
Epoch 28/100
115092/115092 [==============================] - 17s 150us/step - loss: 0.1086 -
val_loss: 0.1142
Epoch 29/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.1072 -
val_loss: 0.1465
Epoch 30/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.1064 -
val_loss: 0.1096
Epoch 31/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.1053 -
val_loss: 0.1178
Epoch 32/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.1040 -
```

```
val_loss: 0.0990
Epoch 33/100
115092/115092 [==============================] - 17s 150us/step - loss: 0.1032 -
val_loss: 0.1156
Epoch 34/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.1023 -
val_loss: 0.0990
Epoch 35/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.1015 -
val_loss: 0.0971
Epoch 36/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.1012 -
val_loss: 0.1137
Epoch 37/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0993 -
val_loss: 0.1133
Epoch 38/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0990 -
val_loss: 0.1094
Epoch 39/100
115092/115092 [==============================] - 17s 150us/step - loss: 0.0986 -
val_loss: 0.1000
Epoch 40/100
115092/115092 [==============================] - 17s 150us/step - loss: 0.0974 -
val_loss: 0.1145
Epoch 41/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0969 -
val_loss: 0.1011
Epoch 42/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0963 -
val_loss: 0.1060
Epoch 43/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0959 -
val_loss: 0.1003
Epoch 44/100
115092/115092 [==============================] - 17s 151us/step - loss: 0.0950 -
val_loss: 0.1066
Epoch 45/100
115092/115092 [==============================] - 17s 150us/step - loss: 0.0945 -
val_loss: 0.1013
Epoch 46/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0944 -
val_loss: 0.0835
Epoch 47/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0939 -
val_loss: 0.0896
Epoch 48/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0931 -
```

```
val_loss: 0.1029
Epoch 49/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0926 -
val_loss: 0.1196
Epoch 50/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0918 -
val_loss: 0.1167
Epoch 51/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0918 -
val_loss: 0.0908
Epoch 52/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0914 -
val_loss: 0.0938
Epoch 53/100
115092/115092 [==============================] - 17s 152us/step - loss: 0.0908 -
val_loss: 0.0905
Epoch 54/100
115092/115092 [==============================] - 20s 175us/step - loss: 0.0900 -
val_loss: 0.1008
Epoch 55/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0896 -
val_loss: 0.0925
Epoch 56/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0893 -
val_loss: 0.0999
Epoch 57/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0885 -
val_loss: 0.0815
Epoch 58/100
115092/115092 [==============================] - 17s 150us/step - loss: 0.0883 -
val_loss: 0.1069
Epoch 59/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0878 -
val_loss: 0.0933
Epoch 60/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0868 -
val_loss: 0.1227
Epoch 61/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0867 -
val_loss: 0.0857
Epoch 62/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0862 -
val_loss: 0.0850
Epoch 63/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0859 -
val_loss: 0.0819
Epoch 64/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0856 -
```

```
val_loss: 0.0947
Epoch 65/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0853 -
val_loss: 0.1076
Epoch 66/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0846 -
val_loss: 0.0860
Epoch 67/100
115092/115092 [==============================] - 17s 151us/step - loss: 0.0850 -
val_loss: 0.0909
Epoch 68/100
115092/115092 [==============================] - 17s 150us/step - loss: 0.0844 -
val_loss: 0.0914
Epoch 69/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0839 -
val_loss: 0.0854
Epoch 70/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0835 -
val_loss: 0.0901
Epoch 71/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0831 -
val_loss: 0.0873
Epoch 72/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0832 -
val_loss: 0.0963
Epoch 73/100
115092/115092 [==============================] - 17s 150us/step - loss: 0.0830 -
val_loss: 0.1037
Epoch 74/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0819 -
val_loss: 0.0915
Epoch 75/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0817 -
val_loss: 0.0795
Epoch 76/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0816 -
val_loss: 0.0881
Epoch 77/100
115092/115092 [==============================] - 17s 147us/step - loss: 0.0812 -
val_loss: 0.0832
Epoch 78/100
115092/115092 [==============================] - 17s 147us/step - loss: 0.0807 -
val_loss: 0.0861
Epoch 79/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0806 -
val_loss: 0.0933
Epoch 80/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0806 -
```

```
val_loss: 0.0850
Epoch 81/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0803 -
val_loss: 0.0948
Epoch 82/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0804 -
val_loss: 0.0903
Epoch 83/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0799 -
val_loss: 0.0749
Epoch 84/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0796 -
val_loss: 0.0957
Epoch 85/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0798 -
val_loss: 0.0790
Epoch 86/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0791 -
val_loss: 0.1069
Epoch 87/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0793 -
val_loss: 0.0775
Epoch 88/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0787 -
val_loss: 0.0836
Epoch 89/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0785 -
val_loss: 0.0891
Epoch 90/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0788 -
val_loss: 0.0826
Epoch 91/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0784 -
val_loss: 0.0873
Epoch 92/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0787 -
val_loss: 0.0875
Epoch 93/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0784 -
val_loss: 0.0909
Epoch 94/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0777 -
val_loss: 0.0890
Epoch 95/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0782 -
val_loss: 0.0807
Epoch 96/100
115092/115092 [==============================] - 17s 150us/step - loss: 0.0781 -
```

```
val_loss: 0.0908
Epoch 97/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0778 -
val_loss: 0.0915
Epoch 98/100
115092/115092 [==============================] - 17s 149us/step - loss: 0.0776 -
val_loss: 0.1007
Epoch 99/100
115092/115092 [==============================] - 17s 147us/step - loss: 0.0774 -
val_loss: 0.1024
Epoch 100/100
115092/115092 [==============================] - 17s 148us/step - loss: 0.0774 -
val_loss: 0.0976
```

[21]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[22]:
```
Y_test =np.round(Y_test, 0)
```

[23]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

```
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:2.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 2., predicted:2.
```

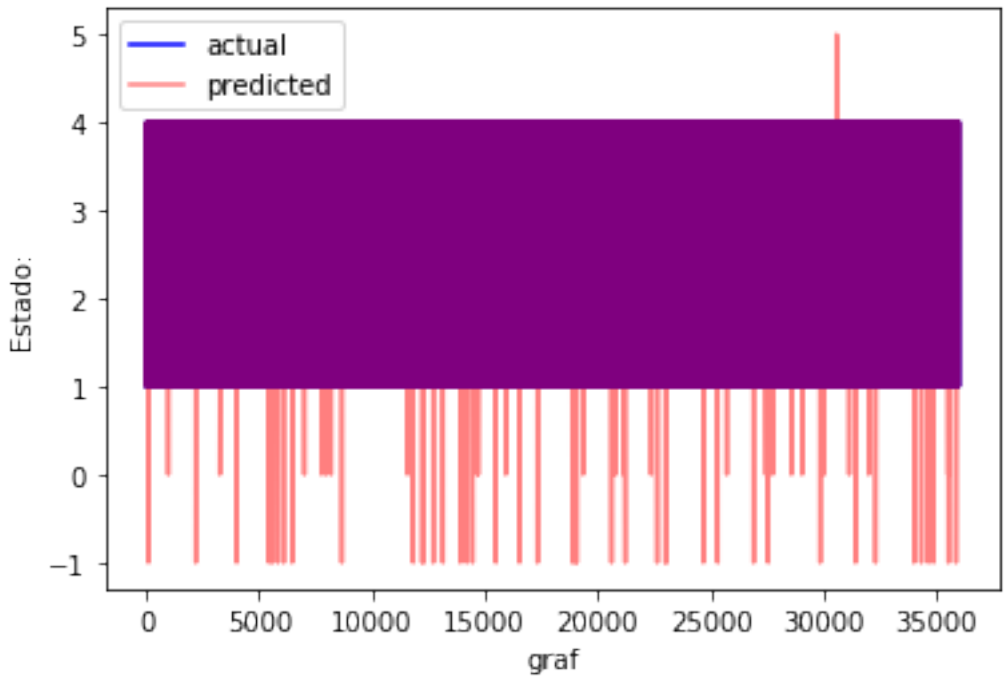
```
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

Evaluamos los resultados:

[24]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.9473311782077163
```

[25]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[26]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[26]:
```
0.9611310367837184
```

[ ]:
```

```

Wrist_elux5_mae_Adadel_final

August 30, 2019

1 Wrist Data

Librerías:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Wrist

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
```

```
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
```

```
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT2=np.concatenate((B1,A2),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
```

```
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=25*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT3=np.concatenate((B1,A2),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
```

```
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=35*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT4=np.concatenate((B1,A2),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
```

```
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT5=np.concatenate((B1,A2),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT6=np.concatenate((B1,A2),axis=1)
```

```
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
```

```
B1 = np.asmatrix(E)
B1=B1.T
MAT7=np.concatenate((B1,A2),axis=1)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
```

```
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT8=np.concatenate((B1,A2),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
```

```
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT9=np.concatenate((B1,A2),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
```

```
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT10=np.concatenate((B1,A2),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
```

```
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT11=np.concatenate((B1,A2),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
```

```
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT12=np.concatenate((B1,A2),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
```

```
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT13=np.concatenate((B1,A2),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
```

```
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=24*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT14=np.concatenate((B1,A2),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>4), 0)
MAT14= np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
```

```
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=29*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT15=np.concatenate((B1,A2),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
 ↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Dividimos en los conjuntos de validación y entrenamiento:

[18]:
```
X = MAT[:, 1:8]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)
```

Definimos la arquitectura de la red:

[19]:
```
model = Sequential([
    Dense(128, input_shape=(7,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
```

```
    Dense(48),
    Activation('elu'),
    Dense(24),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

```
WARNING: Logging before flag parsing goes to stderr.
W0830 01:41:06.452198 12492 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:74: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

W0830 01:41:06.471148 12492 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:517: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

W0830 01:41:06.475136 12492 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:4138: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

W0830 01:41:06.558912 12492 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-packages\keras\optimizers.py:790:
The name tf.train.Optimizer is deprecated. Please use
tf.compat.v1.train.Optimizer instead.
```

Entrenamos la red:

[20]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
W0830 01:41:06.767384 12492 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:986: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.

W0830 01:41:06.774366 12492 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:973: The name tf.assign is
```

```
deprecated. Please use tf.compat.v1.assign instead.


Train on 115092 samples, validate on 28773 samples
Epoch 1/100
115092/115092 [==============================] - 21s 179us/step - loss: 0.5560 -
val_loss: 0.4178
Epoch 2/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.3313 -
val_loss: 0.2706
Epoch 3/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.2469 -
val_loss: 0.2499
Epoch 4/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.2044 -
val_loss: 0.1897
Epoch 5/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.1794 -
val_loss: 0.1732
Epoch 6/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.1604 -
val_loss: 0.1548
Epoch 7/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.1475 -
val_loss: 0.2112
Epoch 8/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.1375 -
val_loss: 0.1244
Epoch 9/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.1285 -
val_loss: 0.1504
Epoch 10/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.1210 -
val_loss: 0.1193
Epoch 11/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.1160 -
val_loss: 0.1149
Epoch 12/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.1117 -
val_loss: 0.1154
Epoch 13/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.1069 -
val_loss: 0.1104
Epoch 14/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.1036 -
val_loss: 0.1029
Epoch 15/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.1002 -
```

```
val_loss: 0.1002
Epoch 16/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0973 -
val_loss: 0.0874
Epoch 17/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0938 -
val_loss: 0.0958
Epoch 18/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0918 -
val_loss: 0.0982
Epoch 19/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0904 -
val_loss: 0.0804
Epoch 20/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0886 -
val_loss: 0.0869
Epoch 21/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0865 -
val_loss: 0.0827
Epoch 22/100
115092/115092 [==============================] - 19s 162us/step - loss: 0.0856 -
val_loss: 0.0877
Epoch 23/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0840 -
val_loss: 0.0811
Epoch 24/100
115092/115092 [==============================] - 20s 170us/step - loss: 0.0821 -
val_loss: 0.1282
Epoch 25/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0813 -
val_loss: 0.0782
Epoch 26/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0799 -
val_loss: 0.0844
Epoch 27/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0790 -
val_loss: 0.0739
Epoch 28/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0786 -
val_loss: 0.0681
Epoch 29/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0771 -
val_loss: 0.0756
Epoch 30/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0757 -
val_loss: 0.0819
Epoch 31/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0752 -
```

```
val_loss: 0.0748
Epoch 32/100
115092/115092 [==============================] - 19s 166us/step - loss: 0.0751 -
val_loss: 0.0723
Epoch 33/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0737 -
val_loss: 0.0867
Epoch 34/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0732 -
val_loss: 0.0839
Epoch 35/100
115092/115092 [==============================] - 19s 166us/step - loss: 0.0729 -
val_loss: 0.0635
Epoch 36/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0716 -
val_loss: 0.0828
Epoch 37/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0714 -
val_loss: 0.0651
Epoch 38/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0705 -
val_loss: 0.0706
Epoch 39/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0707 -
val_loss: 0.0710
Epoch 40/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0702 -
val_loss: 0.0735
Epoch 41/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0697 -
val_loss: 0.0904
Epoch 42/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0691 -
val_loss: 0.0767
Epoch 43/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0688 -
val_loss: 0.0743
Epoch 44/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0686 -
val_loss: 0.0737
Epoch 45/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0677 -
val_loss: 0.0701
Epoch 46/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0674 -
val_loss: 0.0750
Epoch 47/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0670 -
```

```
val_loss: 0.0716
Epoch 48/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0664 -
val_loss: 0.0711
Epoch 49/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0668 -
val_loss: 0.0657
Epoch 50/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0665 -
val_loss: 0.0575
Epoch 51/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0658 -
val_loss: 0.0614
Epoch 52/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0657 -
val_loss: 0.0692
Epoch 53/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0652 -
val_loss: 0.0780
Epoch 54/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0653 -
val_loss: 0.0635
Epoch 55/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0649 -
val_loss: 0.0652
Epoch 56/100
115092/115092 [==============================] - 19s 162us/step - loss: 0.0650 -
val_loss: 0.0647
Epoch 57/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0639 -
val_loss: 0.0560
Epoch 58/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0643 -
val_loss: 0.0762
Epoch 59/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0640 -
val_loss: 0.0663
Epoch 60/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0642 -
val_loss: 0.0737
Epoch 61/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0637 -
val_loss: 0.0631
Epoch 62/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0635 -
val_loss: 0.0688
Epoch 63/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0639 -
```

```
val_loss: 0.0587
Epoch 64/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0636 -
val_loss: 0.0880
Epoch 65/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0633 -
val_loss: 0.0694
Epoch 66/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0626 -
val_loss: 0.0649
Epoch 67/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0614 -
val_loss: 0.0575
Epoch 68/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0616 -
val_loss: 0.0732
Epoch 69/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0609 -
val_loss: 0.0618
Epoch 70/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0607 -
val_loss: 0.0683
Epoch 71/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0597 -
val_loss: 0.0589
Epoch 72/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0603 -
val_loss: 0.0558
Epoch 73/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0600 -
val_loss: 0.0524
Epoch 74/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0595 -
val_loss: 0.0702
Epoch 75/100
115092/115092 [==============================] - 19s 162us/step - loss: 0.0592 -
val_loss: 0.0552
Epoch 76/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0590 -
val_loss: 0.0615
Epoch 77/100
115092/115092 [==============================] - 19s 161us/step - loss: 0.0586 -
val_loss: 0.0658
Epoch 78/100
115092/115092 [==============================] - 19s 161us/step - loss: 0.0591 -
val_loss: 0.0642
Epoch 79/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0585 -
```

```
val_loss: 0.0590
Epoch 80/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0587 -
val_loss: 0.0645
Epoch 81/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0591 -
val_loss: 0.0674
Epoch 82/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0582 -
val_loss: 0.0656
Epoch 83/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0579 -
val_loss: 0.0537
Epoch 84/100
115092/115092 [==============================] - 19s 162us/step - loss: 0.0579 -
val_loss: 0.0541
Epoch 85/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0574 -
val_loss: 0.0829
Epoch 86/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0579 -
val_loss: 0.0504
Epoch 87/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0570 -
val_loss: 0.0548
Epoch 88/100
115092/115092 [==============================] - 19s 162us/step - loss: 0.0573 -
val_loss: 0.0635
Epoch 89/100
115092/115092 [==============================] - 19s 161us/step - loss: 0.0566 -
val_loss: 0.0611
Epoch 90/100
115092/115092 [==============================] - 19s 161us/step - loss: 0.0567 -
val_loss: 0.0595
Epoch 91/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0571 -
val_loss: 0.0559
Epoch 92/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0564 -
val_loss: 0.0583
Epoch 93/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0570 -
val_loss: 0.0638
Epoch 94/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0561 -
val_loss: 0.0629
Epoch 95/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0567 -
```

```
val_loss: 0.0570
Epoch 96/100
115092/115092 [==============================] - 21s 182us/step - loss: 0.0563 -
val_loss: 0.0494
Epoch 97/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0565 -
val_loss: 0.0563
Epoch 98/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0558 -
val_loss: 0.0606
Epoch 99/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0555 -
val_loss: 0.0532
Epoch 100/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0558 -
val_loss: 0.0521
```

[21]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[22]:
```
Y_test =np.round(Y_test, 0)
```

[23]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

```
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:3.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
```

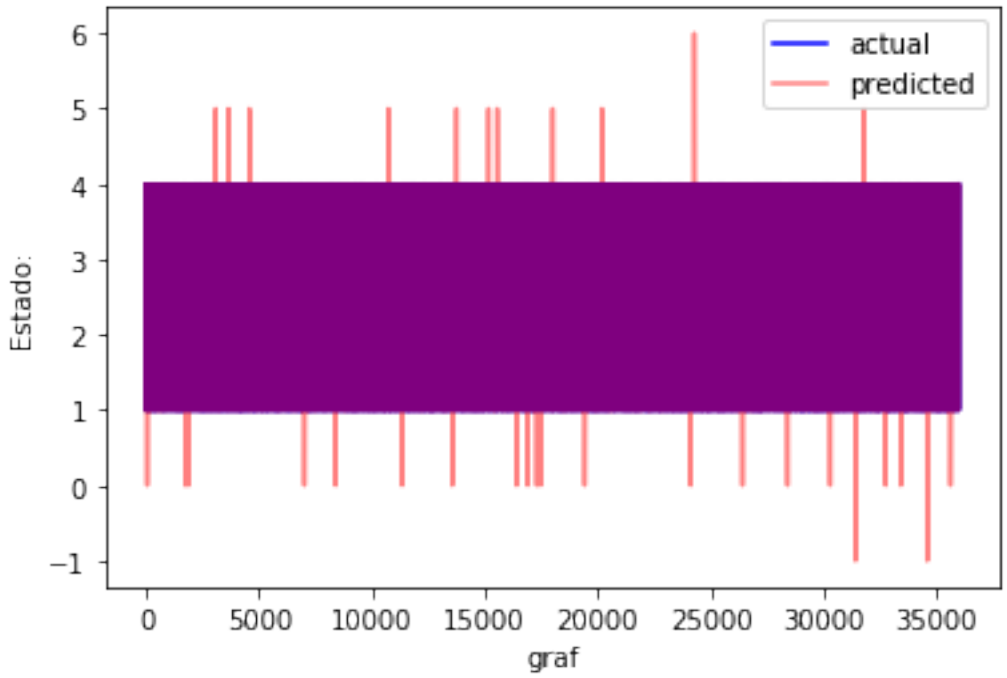
```
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

Evaluamos los resultados:

[24]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.9632204478987918
```

[25]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[26]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[26]:
```
0.9761447993994495
```

[ ]:
```

```

Wrist_elux7_mae_Adadel_final

August 30, 2019

1 Wrist Data

Librerías:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Wrist

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
```

```
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
```

```
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT2=np.concatenate((B1,A2),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
```

```
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=25*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT3=np.concatenate((B1,A2),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
```

```
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=35*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT4=np.concatenate((B1,A2),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
```

```
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT5=np.concatenate((B1,A2),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT6=np.concatenate((B1,A2),axis=1)
```

```
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
```

```
B1 = np.asmatrix(E)
B1=B1.T
MAT7=np.concatenate((B1,A2),axis=1)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
```

```
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT8=np.concatenate((B1,A2),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
```

```
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT9=np.concatenate((B1,A2),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
```

```
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT10=np.concatenate((B1,A2),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
```

```
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT11=np.concatenate((B1,A2),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
```

```
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT12=np.concatenate((B1,A2),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
```

```
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT13=np.concatenate((B1,A2),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
```

```
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=24*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT14=np.concatenate((B1,A2),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>4), 0)
MAT14= np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
```

```
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=29*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT15=np.concatenate((B1,A2),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
 ↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Dividimos en los conjuntos de validación y entrenamiento:

[18]:
```
X = MAT[:, 1:8]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)
```

Definimos la arquitectura de la red:

[19]:
```
model = Sequential([
    Dense(128, input_shape=(7,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
```

```
    Dense(60),
    Activation('elu'),
    Dense(40),
    Activation('elu'),
    Dense(40),
    Activation('elu'),
    Dense(30),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

```
WARNING: Logging before flag parsing goes to stderr.
W0830 02:17:23.968223  6340 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:74: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

W0830 02:17:23.981184  6340 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:517: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

W0830 02:17:23.984184  6340 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:4138: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

W0830 02:17:24.078895  6340 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-packages\keras\optimizers.py:790:
The name tf.train.Optimizer is deprecated. Please use
tf.compat.v1.train.Optimizer instead.
```

Entrenamos la red:

[20]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
W0830 02:17:24.318292  6340 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:986: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.
```

```
W0830 02:17:24.324240  6340 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:973: The name tf.assign is
deprecated. Please use tf.compat.v1.assign instead.


Train on 115092 samples, validate on 28773 samples
Epoch 1/100
115092/115092 [==============================] - 23s 203us/step - loss: 0.5144 -
val_loss: 0.4035
Epoch 2/100
115092/115092 [==============================] - 23s 196us/step - loss: 0.2823 -
val_loss: 0.2418
Epoch 3/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.2173 -
val_loss: 0.1953
Epoch 4/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.1869 -
val_loss: 0.1814
Epoch 5/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.1686 -
val_loss: 0.1688
Epoch 6/100
115092/115092 [==============================] - 23s 196us/step - loss: 0.1575 -
val_loss: 0.1683
Epoch 7/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.1468 -
val_loss: 0.1328
Epoch 8/100
115092/115092 [==============================] - 23s 196us/step - loss: 0.1427 -
val_loss: 0.1350
Epoch 9/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.1364 -
val_loss: 0.1455
Epoch 10/100
115092/115092 [==============================] - 23s 196us/step - loss: 0.1324 -
val_loss: 0.1454
Epoch 11/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.1285 -
val_loss: 0.1273
Epoch 12/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.1258 -
val_loss: 0.1417
Epoch 13/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.1229 -
val_loss: 0.1282
Epoch 14/100
```

```
115092/115092 [==============================] - 23s 196us/step - loss: 0.1226 -
val_loss: 0.1154
Epoch 15/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.1056 -
val_loss: 0.1073
Epoch 16/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0948 -
val_loss: 0.1141
Epoch 17/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0921 -
val_loss: 0.0848
Epoch 18/100
115092/115092 [==============================] - 23s 196us/step - loss: 0.0911 -
val_loss: 0.0927
Epoch 19/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0881 -
val_loss: 0.1075
Epoch 20/100
115092/115092 [==============================] - 23s 196us/step - loss: 0.0863 -
val_loss: 0.0938
Epoch 21/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0840 -
val_loss: 0.0953
Epoch 22/100
115092/115092 [==============================] - 23s 196us/step - loss: 0.0833 -
val_loss: 0.0850
Epoch 23/100
115092/115092 [==============================] - 23s 196us/step - loss: 0.0818 -
val_loss: 0.1343
Epoch 24/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0814 -
val_loss: 0.0835
Epoch 25/100
115092/115092 [==============================] - 23s 196us/step - loss: 0.0796 -
val_loss: 0.0820
Epoch 26/100
115092/115092 [==============================] - 23s 196us/step - loss: 0.0771 -
val_loss: 0.0750
Epoch 27/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0764 -
val_loss: 0.0780
Epoch 28/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0741 -
val_loss: 0.0680
Epoch 29/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0745 -
val_loss: 0.0795
Epoch 30/100
```

```
115092/115092 [==============================] - 23s 197us/step - loss: 0.0724 -
val_loss: 0.0751
Epoch 31/100
115092/115092 [==============================] - 23s 196us/step - loss: 0.0721 -
val_loss: 0.0768
Epoch 32/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0717 -
val_loss: 0.0634
Epoch 33/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0707 -
val_loss: 0.0780
Epoch 34/100
115092/115092 [==============================] - 23s 196us/step - loss: 0.0705 -
val_loss: 0.0776
Epoch 35/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0688 -
val_loss: 0.0810
Epoch 36/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0682 -
val_loss: 0.0772
Epoch 37/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0688 -
val_loss: 0.0897
Epoch 38/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0678 -
val_loss: 0.0644
Epoch 39/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0675 -
val_loss: 0.0636
Epoch 40/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0661 -
val_loss: 0.0653
Epoch 41/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0668 -
val_loss: 0.0645
Epoch 42/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0661 -
val_loss: 0.0787
Epoch 43/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0656 -
val_loss: 0.0733
Epoch 44/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0642 -
val_loss: 0.0654
Epoch 45/100
115092/115092 [==============================] - 23s 201us/step - loss: 0.0649 -
val_loss: 0.0593
Epoch 46/100
```

```
115092/115092 [==============================] - 23s 197us/step - loss: 0.0626 -
val_loss: 0.0760
Epoch 47/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0623 -
val_loss: 0.0654
Epoch 48/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0624 -
val_loss: 0.0688
Epoch 49/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0629 -
val_loss: 0.0658
Epoch 50/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0616 -
val_loss: 0.0669
Epoch 51/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0614 -
val_loss: 0.0697
Epoch 52/100
115092/115092 [==============================] - 23s 203us/step - loss: 0.0614 -
val_loss: 0.0535
Epoch 53/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0607 -
val_loss: 0.0578
Epoch 54/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0602 -
val_loss: 0.0674
Epoch 55/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0605 -
val_loss: 0.0615
Epoch 56/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0595 -
val_loss: 0.0685
Epoch 57/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0593 -
val_loss: 0.0560
Epoch 58/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0587 -
val_loss: 0.0689
Epoch 59/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0588 -
val_loss: 0.0681
Epoch 60/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0594 -
val_loss: 0.0607
Epoch 61/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0588 -
val_loss: 0.0688
Epoch 62/100
```

```
115092/115092 [==============================] - 23s 199us/step - loss: 0.0579 -
val_loss: 0.0540
Epoch 63/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0577 -
val_loss: 0.0540
Epoch 64/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0582 -
val_loss: 0.1376
Epoch 65/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0583 -
val_loss: 0.0680
Epoch 66/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0570 -
val_loss: 0.0755
Epoch 67/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0566 -
val_loss: 0.0639
Epoch 68/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0563 -
val_loss: 0.0585
Epoch 69/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0557 -
val_loss: 0.0691
Epoch 70/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0566 -
val_loss: 0.0715
Epoch 71/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0559 -
val_loss: 0.0617
Epoch 72/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0555 -
val_loss: 0.0523
Epoch 73/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0556 -
val_loss: 0.0834
Epoch 74/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0566 -
val_loss: 0.0519
Epoch 75/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0551 -
val_loss: 0.0591
Epoch 76/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0556 -
val_loss: 0.0571
Epoch 77/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0533 -
val_loss: 0.0579
Epoch 78/100
```

```
115092/115092 [==============================] - 23s 197us/step - loss: 0.0555 -
val_loss: 0.0559
Epoch 79/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0540 -
val_loss: 0.0587
Epoch 80/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0540 -
val_loss: 0.0550
Epoch 81/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0532 -
val_loss: 0.0598
Epoch 82/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0533 -
val_loss: 0.0654
Epoch 83/100
115092/115092 [==============================] - 23s 196us/step - loss: 0.0535 -
val_loss: 0.0527
Epoch 84/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0519 -
val_loss: 0.0453
Epoch 85/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0527 -
val_loss: 0.0621
Epoch 86/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0518 -
val_loss: 0.0567
Epoch 87/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0525 -
val_loss: 0.0447
Epoch 88/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0511 -
val_loss: 0.0560
Epoch 89/100
115092/115092 [==============================] - 23s 196us/step - loss: 0.0524 -
val_loss: 0.0545
Epoch 90/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0515 -
val_loss: 0.0630
Epoch 91/100
115092/115092 [==============================] - 23s 199us/step - loss: 0.0518 -
val_loss: 0.0635
Epoch 92/100
115092/115092 [==============================] - 23s 198us/step - loss: 0.0505 -
val_loss: 0.0579
Epoch 93/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0511 -
val_loss: 0.0539
Epoch 94/100
```

```
115092/115092 [==============================] - 22s 191us/step - loss: 0.0515 -
val_loss: 0.0457
Epoch 95/100
115092/115092 [==============================] - 23s 197us/step - loss: 0.0518 -
val_loss: 0.0516
Epoch 96/100
115092/115092 [==============================] - 21s 186us/step - loss: 0.0495 -
val_loss: 0.0511
Epoch 97/100
115092/115092 [==============================] - 21s 186us/step - loss: 0.0512 -
val_loss: 0.0580
Epoch 98/100
115092/115092 [==============================] - 22s 191us/step - loss: 0.0500 -
val_loss: 0.0704
Epoch 99/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0502 -
val_loss: 0.0618
Epoch 100/100
115092/115092 [==============================] - 22s 190us/step - loss: 0.0511 -
val_loss: 0.0520
```

[21]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[22]:
```
Y_test =np.round(Y_test, 0)
```

[23]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

```
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
```

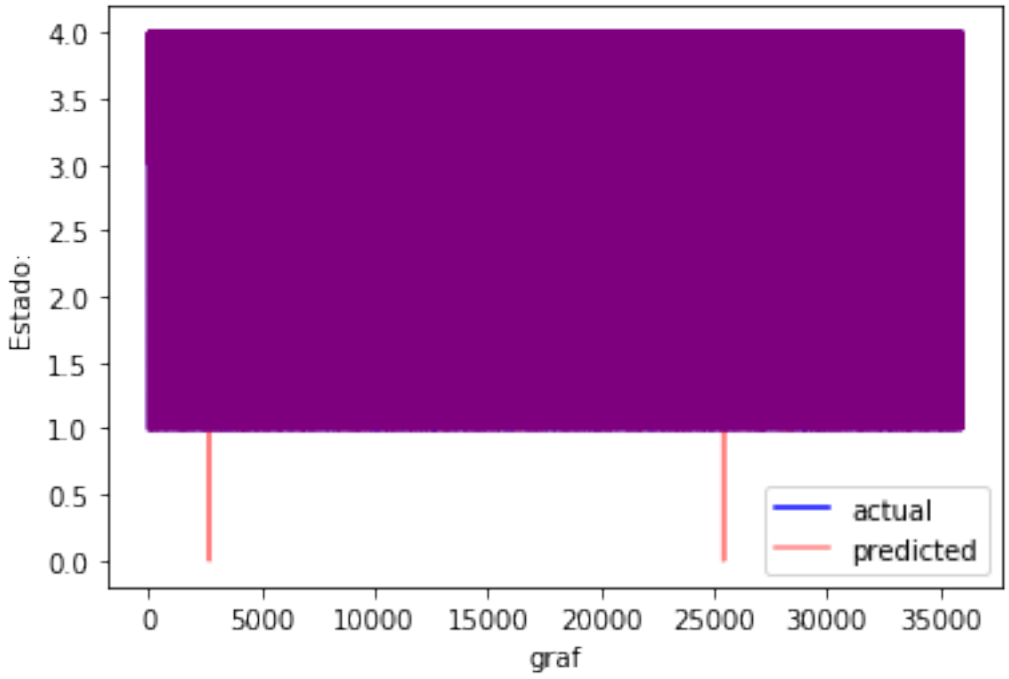
```
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

Evaluamos los resultados:

[24]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.9557391972854581
```

[25]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[26]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[26]:
```
0.9726415881224456
```

[ ]:
```

```

Wrist_seluelux5_mae_Adadel_final

September 3, 2019

1 Wrist Data

Librerías:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Wrist

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
```

```
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
```

```
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT2=np.concatenate((B1,A2),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
```

```
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=25*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT3=np.concatenate((B1,A2),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
```

```
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=35*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT4=np.concatenate((B1,A2),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
```

```
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT5=np.concatenate((B1,A2),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT6=np.concatenate((B1,A2),axis=1)
```

```
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
```

```
B1 = np.asmatrix(E)
B1=B1.T
MAT7=np.concatenate((B1,A2),axis=1)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
```

```
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT8=np.concatenate((B1,A2),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
```

```
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT9=np.concatenate((B1,A2),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
```

```
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT10=np.concatenate((B1,A2),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
```

```
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT11=np.concatenate((B1,A2),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
```

```
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT12=np.concatenate((B1,A2),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
```

```
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT13=np.concatenate((B1,A2),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
```

```
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=24*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT14=np.concatenate((B1,A2),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>4), 0)
MAT14= np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
```

```
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=29*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT15=np.concatenate((B1,A2),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
 ↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
3
```

Dividimos en los conjuntos de validación y entrenamiento:

[18]:
```
X = MAT[:, 1:8]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)
```

Definimos la arquitectura de la red:

[19]:
```
model = Sequential([
    Dense(128, input_shape=(7,)),
    Activation('elu'),
    Dense(70),
```

```
    Activation('selu'),
    Dense(48),
    Activation('elu'),
    Dense(24),
    Activation('selu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

```
WARNING: Logging before flag parsing goes to stderr.
W0903 03:28:41.603379 13956 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:74: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

W0903 03:28:41.636319 13956 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:517: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

W0903 03:28:41.647271 13956 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:4138: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

W0903 03:28:41.686223 13956 deprecation.py:323] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:3217:
add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
W0903 03:28:41.735028 13956 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-packages\keras\optimizers.py:790:
The name tf.train.Optimizer is deprecated. Please use
tf.compat.v1.train.Optimizer instead.
```

Entrenamos la red:

[20]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
W0903 03:28:42.005303 13956 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:986: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.

W0903 03:28:42.012285 13956 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:973: The name tf.assign is
deprecated. Please use tf.compat.v1.assign instead.


Train on 115092 samples, validate on 28773 samples
Epoch 1/100
115092/115092 [==============================] - 24s 206us/step - loss: 0.5585 -
val_loss: 0.3931
Epoch 2/100
115092/115092 [==============================] - 20s 176us/step - loss: 0.3358 -
val_loss: 0.2913
Epoch 3/100
115092/115092 [==============================] - 21s 178us/step - loss: 0.2545 -
val_loss: 0.2299
Epoch 4/100
115092/115092 [==============================] - 21s 181us/step - loss: 0.2128 -
val_loss: 0.1831
Epoch 5/100
115092/115092 [==============================] - 20s 176us/step - loss: 0.1832 -
val_loss: 0.2128
Epoch 6/100
115092/115092 [==============================] - 20s 176us/step - loss: 0.1635 -
val_loss: 0.1689
Epoch 7/100
115092/115092 [==============================] - 20s 177us/step - loss: 0.1488 -
val_loss: 0.1398
Epoch 8/100
115092/115092 [==============================] - 20s 177us/step - loss: 0.1387 -
val_loss: 0.1428
Epoch 9/100
115092/115092 [==============================] - 20s 176us/step - loss: 0.1312 -
val_loss: 0.1190
Epoch 10/100
115092/115092 [==============================] - 20s 177us/step - loss: 0.1249 -
val_loss: 0.1288
Epoch 11/100
115092/115092 [==============================] - 21s 179us/step - loss: 0.1186 -
val_loss: 0.1248
Epoch 12/100
115092/115092 [==============================] - 20s 176us/step - loss: 0.1133 -
val_loss: 0.1568
```

```
Epoch 13/100
115092/115092 [==============================] - 20s 177us/step - loss: 0.1096 -
val_loss: 0.1081
Epoch 14/100
115092/115092 [==============================] - 20s 177us/step - loss: 0.1061 -
val_loss: 0.1116
Epoch 15/100
115092/115092 [==============================] - 20s 178us/step - loss: 0.1030 -
val_loss: 0.0952
Epoch 16/100
115092/115092 [==============================] - 20s 177us/step - loss: 0.1000 -
val_loss: 0.1047
Epoch 17/100
115092/115092 [==============================] - 20s 177us/step - loss: 0.0974 -
val_loss: 0.0994
Epoch 18/100
115092/115092 [==============================] - 20s 178us/step - loss: 0.0953 -
val_loss: 0.0945
Epoch 19/100
115092/115092 [==============================] - 20s 177us/step - loss: 0.0927 -
val_loss: 0.0956
Epoch 20/100
115092/115092 [==============================] - 20s 177us/step - loss: 0.0907 -
val_loss: 0.0908
Epoch 21/100
115092/115092 [==============================] - 20s 177us/step - loss: 0.0885 -
val_loss: 0.0780
Epoch 22/100
115092/115092 [==============================] - 21s 178us/step - loss: 0.0871 -
val_loss: 0.1046
Epoch 23/100
115092/115092 [==============================] - 21s 180us/step - loss: 0.0855 -
val_loss: 0.1073
Epoch 24/100
115092/115092 [==============================] - 20s 177us/step - loss: 0.0846 -
val_loss: 0.0787
Epoch 25/100
115092/115092 [==============================] - 20s 170us/step - loss: 0.0829 -
val_loss: 0.0831
Epoch 26/100
115092/115092 [==============================] - 20s 170us/step - loss: 0.0820 -
val_loss: 0.0725
Epoch 27/100
115092/115092 [==============================] - 20s 172us/step - loss: 0.0808 -
val_loss: 0.0814
Epoch 28/100
115092/115092 [==============================] - 20s 172us/step - loss: 0.0799 -
val_loss: 0.0941
```

```
Epoch 29/100
115092/115092 [==============================] - 20s 172us/step - loss: 0.0783 -
val_loss: 0.0840
Epoch 30/100
115092/115092 [==============================] - 20s 171us/step - loss: 0.0775 -
val_loss: 0.0806
Epoch 31/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0755 -
val_loss: 0.0740
Epoch 32/100
115092/115092 [==============================] - 20s 170us/step - loss: 0.0751 -
val_loss: 0.0965
Epoch 33/100
115092/115092 [==============================] - 20s 171us/step - loss: 0.0749 -
val_loss: 0.0769
Epoch 34/100
115092/115092 [==============================] - 20s 176us/step - loss: 0.0739 -
val_loss: 0.0854
Epoch 35/100
115092/115092 [==============================] - 20s 170us/step - loss: 0.0734 -
val_loss: 0.0844
Epoch 36/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0721 -
val_loss: 0.0771
Epoch 37/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0718 -
val_loss: 0.0739
Epoch 38/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0712 -
val_loss: 0.0832
Epoch 39/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0706 -
val_loss: 0.0770
Epoch 40/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0704 -
val_loss: 0.0743
Epoch 41/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0697 -
val_loss: 0.0682
Epoch 42/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0689 -
val_loss: 0.0761
Epoch 43/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0685 -
val_loss: 0.0673
Epoch 44/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0686 -
val_loss: 0.0724
```

```
Epoch 45/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0676 -
val_loss: 0.0622
Epoch 46/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0682 -
val_loss: 0.0652
Epoch 47/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0678 -
val_loss: 0.0658
Epoch 48/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0675 -
val_loss: 0.0628
Epoch 49/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0669 -
val_loss: 0.0779
Epoch 50/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0670 -
val_loss: 0.0985
Epoch 51/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0668 -
val_loss: 0.0773
Epoch 52/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0662 -
val_loss: 0.0691
Epoch 53/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0658 -
val_loss: 0.0549
Epoch 54/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0658 -
val_loss: 0.0582
Epoch 55/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0653 -
val_loss: 0.0704
Epoch 56/100
115092/115092 [==============================] - 20s 172us/step - loss: 0.0648 -
val_loss: 0.0822
Epoch 57/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0649 -
val_loss: 0.0690
Epoch 58/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0637 -
val_loss: 0.0802
Epoch 59/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0643 -
val_loss: 0.0846
Epoch 60/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0635 -
val_loss: 0.0911
```

```
Epoch 61/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0633 -
val_loss: 0.0627
Epoch 62/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0625 -
val_loss: 0.0806
Epoch 63/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0623 -
val_loss: 0.0678
Epoch 64/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0618 -
val_loss: 0.0642
Epoch 65/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0622 -
val_loss: 0.0596
Epoch 66/100
115092/115092 [==============================] - 20s 173us/step - loss: 0.0608 -
val_loss: 0.0556
Epoch 67/100
115092/115092 [==============================] - 20s 172us/step - loss: 0.0609 -
val_loss: 0.0968
Epoch 68/100
115092/115092 [==============================] - 20s 173us/step - loss: 0.0607 -
val_loss: 0.0510
Epoch 69/100
115092/115092 [==============================] - 20s 170us/step - loss: 0.0602 -
val_loss: 0.0750
Epoch 70/100
115092/115092 [==============================] - 20s 175us/step - loss: 0.0597 -
val_loss: 0.0637
Epoch 71/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0593 -
val_loss: 0.1047
Epoch 72/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0593 -
val_loss: 0.0752
Epoch 73/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0594 -
val_loss: 0.0665
Epoch 74/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0587 -
val_loss: 0.0570
Epoch 75/100
115092/115092 [==============================] - 19s 166us/step - loss: 0.0585 -
val_loss: 0.0547
Epoch 76/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0588 -
val_loss: 0.0824
```

```
Epoch 77/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0585 -
val_loss: 0.0595
Epoch 78/100
115092/115092 [==============================] - 19s 166us/step - loss: 0.0585 -
val_loss: 0.0566
Epoch 79/100
115092/115092 [==============================] - 19s 166us/step - loss: 0.0583 -
val_loss: 0.0546
Epoch 80/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0577 -
val_loss: 0.0561
Epoch 81/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0578 -
val_loss: 0.0616
Epoch 82/100
115092/115092 [==============================] - 19s 166us/step - loss: 0.0575 -
val_loss: 0.0681
Epoch 83/100
115092/115092 [==============================] - 19s 166us/step - loss: 0.0576 -
val_loss: 0.0812
Epoch 84/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0577 -
val_loss: 0.0773
Epoch 85/100
115092/115092 [==============================] - 19s 166us/step - loss: 0.0581 -
val_loss: 0.0526
Epoch 86/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0566 -
val_loss: 0.0608
Epoch 87/100
115092/115092 [==============================] - 19s 166us/step - loss: 0.0569 -
val_loss: 0.0643
Epoch 88/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0577 -
val_loss: 0.0560
Epoch 89/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0572 -
val_loss: 0.0669
Epoch 90/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0570 -
val_loss: 0.0662
Epoch 91/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0570 -
val_loss: 0.0689
Epoch 92/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0563 -
val_loss: 0.0537
```

```
Epoch 93/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0566 -
val_loss: 0.0752
Epoch 94/100
115092/115092 [==============================] - 20s 171us/step - loss: 0.0576 -
val_loss: 0.0725
Epoch 95/100
115092/115092 [==============================] - 20s 170us/step - loss: 0.0573 -
val_loss: 0.0652
Epoch 96/100
115092/115092 [==============================] - 21s 181us/step - loss: 0.0563 -
val_loss: 0.0614
Epoch 97/100
115092/115092 [==============================] - 20s 178us/step - loss: 0.0572 -
val_loss: 0.0579
Epoch 98/100
115092/115092 [==============================] - 20s 170us/step - loss: 0.0567 -
val_loss: 0.0595
Epoch 99/100
115092/115092 [==============================] - 19s 166us/step - loss: 0.0567 -
val_loss: 0.0560
Epoch 100/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0564 -
val_loss: 0.0620
```

[21]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
Train on 115092 samples, validate on 28773 samples
Epoch 1/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0576 -
val_loss: 0.0536
Epoch 2/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0580 -
val_loss: 0.0571
Epoch 3/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0576 -
val_loss: 0.0580
Epoch 4/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0570 -
val_loss: 0.0694
Epoch 5/100
115092/115092 [==============================] - 20s 176us/step - loss: 0.0568 -
val_loss: 0.0517
```

```
Epoch 6/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0575 -
val_loss: 0.0556
Epoch 7/100
115092/115092 [==============================] - 20s 178us/step - loss: 0.0582 -
val_loss: 0.0745
Epoch 8/100
115092/115092 [==============================] - 20s 172us/step - loss: 0.0585 -
val_loss: 0.0694
Epoch 9/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0584 -
val_loss: 0.0840
Epoch 10/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0592 -
val_loss: 0.0656
Epoch 11/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0591 -
val_loss: 0.0564
Epoch 12/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0593 -
val_loss: 0.0526
Epoch 13/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0596 -
val_loss: 0.0633
Epoch 14/100
115092/115092 [==============================] - 20s 170us/step - loss: 0.0590 -
val_loss: 0.0770
Epoch 15/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0609 -
val_loss: 0.0773
Epoch 16/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0596 -
val_loss: 0.0585
Epoch 17/100
115092/115092 [==============================] - 20s 169us/step - loss: 0.0606 -
val_loss: 0.0545
Epoch 18/100
115092/115092 [==============================] - 19s 166us/step - loss: 0.0598 -
val_loss: 0.0803
Epoch 19/100
115092/115092 [==============================] - 19s 166us/step - loss: 0.0605 -
val_loss: 0.0716
Epoch 20/100
115092/115092 [==============================] - 19s 166us/step - loss: 0.0597 -
val_loss: 0.0577
Epoch 21/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0594 -
val_loss: 0.0714
```

```
Epoch 22/100
115092/115092 [==============================] - 19s 166us/step - loss: 0.0598 -
val_loss: 0.0571
Epoch 23/100
115092/115092 [==============================] - 20s 171us/step - loss: 0.0599 -
val_loss: 0.0770
Epoch 24/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0596 -
val_loss: 0.0605
Epoch 25/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0596 -
val_loss: 0.0611
Epoch 26/100
115092/115092 [==============================] - 20s 170us/step - loss: 0.0597 -
val_loss: 0.0854
Epoch 27/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0597 -
val_loss: 0.0715
Epoch 28/100
115092/115092 [==============================] - 19s 166us/step - loss: 0.0586 -
val_loss: 0.0634
Epoch 29/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0577 -
val_loss: 0.0508
Epoch 30/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0568 -
val_loss: 0.0734
Epoch 31/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0570 -
val_loss: 0.0531
Epoch 32/100
115092/115092 [==============================] - 19s 166us/step - loss: 0.0569 -
val_loss: 0.0577
Epoch 33/100
115092/115092 [==============================] - 19s 166us/step - loss: 0.0567 -
val_loss: 0.0583
Epoch 34/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0561 -
val_loss: 0.0630
Epoch 35/100
115092/115092 [==============================] - 20s 173us/step - loss: 0.0558 -
val_loss: 0.0518
Epoch 36/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0561 -
val_loss: 0.0729
Epoch 37/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0563 -
val_loss: 0.0564
```

```
Epoch 38/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0550 -
val_loss: 0.0648
Epoch 39/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0561 -
val_loss: 0.0534
Epoch 40/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0555 -
val_loss: 0.0655
Epoch 41/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0551 -
val_loss: 0.0541
Epoch 42/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0558 -
val_loss: 0.0538
Epoch 43/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0565 -
val_loss: 0.0546
Epoch 44/100
115092/115092 [==============================] - 19s 166us/step - loss: 0.0558 -
val_loss: 0.0565
Epoch 45/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0561 -
val_loss: 0.0593
Epoch 46/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0565 -
val_loss: 0.0510
Epoch 47/100
115092/115092 [==============================] - 19s 166us/step - loss: 0.0560 -
val_loss: 0.0534
Epoch 48/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0558 -
val_loss: 0.0512
Epoch 49/100
115092/115092 [==============================] - 20s 170us/step - loss: 0.0553 -
val_loss: 0.0652
Epoch 50/100
115092/115092 [==============================] - 19s 166us/step - loss: 0.0552 -
val_loss: 0.0693
Epoch 51/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0546 -
val_loss: 0.0643
Epoch 52/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0560 -
val_loss: 0.0505
Epoch 53/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0550 -
val_loss: 0.0620
```

```
Epoch 54/100
115092/115092 [==============================] - 20s 173us/step - loss: 0.0561 -
val_loss: 0.0579
Epoch 55/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0560 -
val_loss: 0.0567
Epoch 56/100
115092/115092 [==============================] - 20s 172us/step - loss: 0.0566 -
val_loss: 0.0616
Epoch 57/100
115092/115092 [==============================] - 20s 170us/step - loss: 0.0566 -
val_loss: 0.0540
Epoch 58/100
115092/115092 [==============================] - 20s 172us/step - loss: 0.0572 -
val_loss: 0.0736
Epoch 59/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0568 -
val_loss: 0.0618
Epoch 60/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0566 -
val_loss: 0.0618
Epoch 61/100
115092/115092 [==============================] - 19s 165us/step - loss: 0.0569 -
val_loss: 0.0540
Epoch 62/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0574 -
val_loss: 0.0660
Epoch 63/100
115092/115092 [==============================] - 20s 170us/step - loss: 0.0554 -
val_loss: 0.0735
Epoch 64/100
115092/115092 [==============================] - 20s 172us/step - loss: 0.0584 -
val_loss: 0.0520
Epoch 65/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0568 -
val_loss: 0.0588
Epoch 66/100
115092/115092 [==============================] - 20s 173us/step - loss: 0.0564 -
val_loss: 0.0562
Epoch 67/100
115092/115092 [==============================] - 19s 166us/step - loss: 0.0571 -
val_loss: 0.0644
Epoch 68/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0570 -
val_loss: 0.0654
Epoch 69/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0560 -
val_loss: 0.0840
```

```
Epoch 70/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.0555 -
val_loss: 0.0614
Epoch 71/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.0566 -
val_loss: 0.0678
Epoch 72/100
115092/115092 [==============================] - 20s 172us/step - loss: 0.0563 -
val_loss: 0.0644
Epoch 73/100
115092/115092 [==============================] - 20s 170us/step - loss: 0.0579 -
val_loss: 0.0670
Epoch 74/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0563 -
val_loss: 0.0671
Epoch 75/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0567 -
val_loss: 0.0639
Epoch 76/100
115092/115092 [==============================] - 19s 167us/step - loss: 0.0562 -
val_loss: 0.0562
Epoch 77/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0569 -
val_loss: 0.0661
Epoch 78/100
115092/115092 [==============================] - 19s 166us/step - loss: 0.0571 -
val_loss: 0.0631
Epoch 79/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0568 -
val_loss: 0.0490
Epoch 80/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0567 -
val_loss: 0.0542
Epoch 81/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0564 -
val_loss: 0.0586
Epoch 82/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0564 -
val_loss: 0.0623
Epoch 83/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0563 -
val_loss: 0.0889
Epoch 84/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0559 -
val_loss: 0.0608
Epoch 85/100
115092/115092 [==============================] - 20s 170us/step - loss: 0.0565 -
val_loss: 0.0787
```

```
Epoch 86/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0569 -
val_loss: 0.0522
Epoch 87/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0560 -
val_loss: 0.0676
Epoch 88/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0566 -
val_loss: 0.0757
Epoch 89/100
115092/115092 [==============================] - 20s 170us/step - loss: 0.0566 -
val_loss: 0.0536
Epoch 90/100
115092/115092 [==============================] - 20s 171us/step - loss: 0.0569 -
val_loss: 0.0655
Epoch 91/100
115092/115092 [==============================] - 20s 175us/step - loss: 0.0557 -
val_loss: 0.0641
Epoch 92/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0561 -
val_loss: 0.0596
Epoch 93/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0560 -
val_loss: 0.0563
Epoch 94/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0568 -
val_loss: 0.0588
Epoch 95/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0568 -
val_loss: 0.0620
Epoch 96/100
115092/115092 [==============================] - 19s 168us/step - loss: 0.0567 -
val_loss: 0.0696
Epoch 97/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0575 -
val_loss: 0.0672
Epoch 98/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0578 -
val_loss: 0.0558
Epoch 99/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0585 -
val_loss: 0.0671
Epoch 100/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.0574 -
val_loss: 0.0526
```

[22]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[23]:
```
Y_test =np.round(Y_test, 0)
```

[24]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

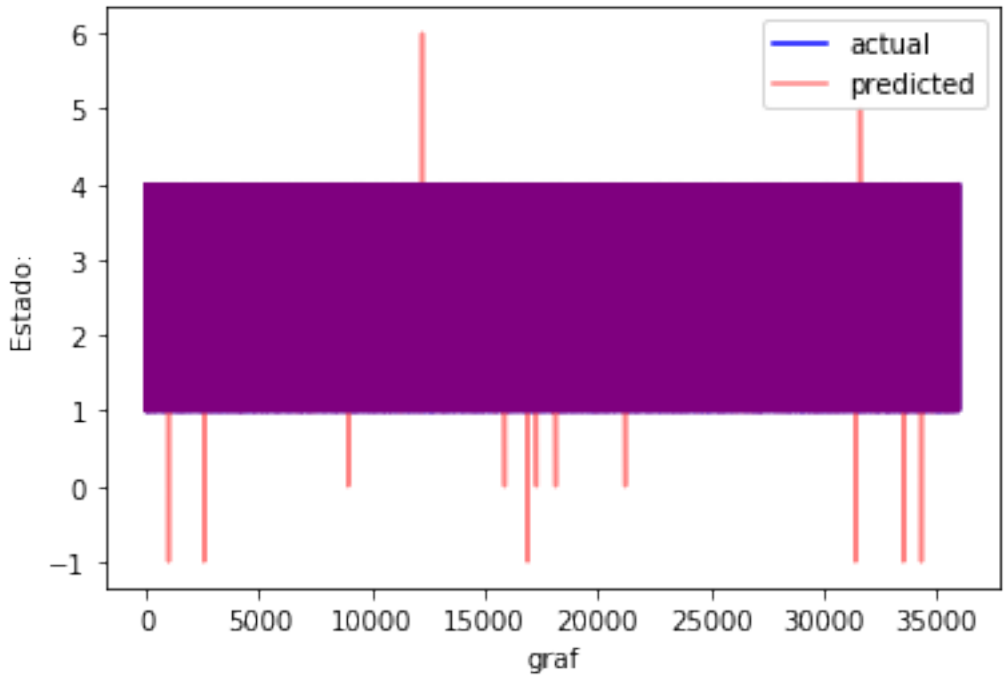
```
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

Evaluamos los resultados:

[25]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.9600521866248108
```

[26]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[27]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[27]: 0.977840798509745

[ ]:

Wrist_seluelux5_mae_Adadel_final

August 31, 2019

1 Wrist Data

Librerías:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Wrist

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
```

```
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>=4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
```

```
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT2=np.concatenate((B1,A2),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>=4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
```

```
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=25*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT3=np.concatenate((B1,A2),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>=4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
```

```
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=35*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT4=np.concatenate((B1,A2),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>=4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
```

```
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT5=np.concatenate((B1,A2),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>=4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT6=np.concatenate((B1,A2),axis=1)
```

```
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>=4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
```

```
B1 = np.asmatrix(E)
B1=B1.T
MAT7=np.concatenate((B1,A2),axis=1)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>=4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
```

```
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT8=np.concatenate((B1,A2),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>=4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
```

```
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT9=np.concatenate((B1,A2),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>=4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
```

```
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT10=np.concatenate((B1,A2),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>=4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
```

```
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT11=np.concatenate((B1,A2),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>=4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
```

```
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT12=np.concatenate((B1,A2),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>=4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
```

```
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT13=np.concatenate((B1,A2),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>=4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
```

```
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=24*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT14=np.concatenate((B1,A2),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>=4), 0)
MAT14= np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
```

```
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=29*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT15=np.concatenate((B1,A2),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>=4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
 ↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Dividimos en los conjuntos de validación y entrenamiento:

[18]:
```
X = MAT[:, 1:8]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)
```

Definimos la arquitectura de la red:

[19]:
```
model = Sequential([
    Dense(128, input_shape=(7,)),
    Activation('elu'),
    Dense(70),
    Activation('selu'),
```

```
    Dense(48),
    Activation('elu'),
    Dense(24),
    Activation('selu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

```
WARNING: Logging before flag parsing goes to stderr.
W0829 22:04:57.830875 31448 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:74: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

W0829 22:04:57.864757 31448 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:517: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

W0829 22:04:57.877796 31448 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:4138: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

W0829 22:04:57.919682 31448 deprecation.py:323] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:3217:
add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
W0829 22:04:57.966590 31448 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-packages\keras\optimizers.py:790:
The name tf.train.Optimizer is deprecated. Please use
tf.compat.v1.train.Optimizer instead.
```

Entrenamos la red:

[20]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
W0829 22:04:58.267327 31448 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
```

```
packages\keras\backend\tensorflow_backend.py:986: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.

W0829 22:04:58.274302 31448 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:973: The name tf.assign is
deprecated. Please use tf.compat.v1.assign instead.


Train on 84868 samples, validate on 21218 samples
Epoch 1/100
84868/84868 [==============================] - 19s 226us/step - loss: 0.4211 -
val_loss: 0.2801
Epoch 2/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.2503 -
val_loss: 0.2200
Epoch 3/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.1915 -
val_loss: 0.1721
Epoch 4/100
84868/84868 [==============================] - 15s 181us/step - loss: 0.1607 -
val_loss: 0.1665
Epoch 5/100
84868/84868 [==============================] - 15s 182us/step - loss: 0.1422 -
val_loss: 0.1556
Epoch 6/100
84868/84868 [==============================] - 16s 183us/step - loss: 0.1306 -
val_loss: 0.1213
Epoch 7/100
84868/84868 [==============================] - 15s 183us/step - loss: 0.1214 -
val_loss: 0.1268
Epoch 8/100
84868/84868 [==============================] - 15s 182us/step - loss: 0.1145 -
val_loss: 0.1465
Epoch 9/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.1098 -
val_loss: 0.1213
Epoch 10/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.1053 -
val_loss: 0.0991
Epoch 11/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.1011 -
val_loss: 0.0942
Epoch 12/100
84868/84868 [==============================] - 16s 183us/step - loss: 0.0975 -
val_loss: 0.0972
Epoch 13/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0949 -
```

```
val_loss: 0.0974
Epoch 14/100
84868/84868 [==============================] - 15s 182us/step - loss: 0.0906 -
val_loss: 0.0905
Epoch 15/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0871 -
val_loss: 0.0932
Epoch 16/100
84868/84868 [==============================] - 15s 181us/step - loss: 0.0838 -
val_loss: 0.0904
Epoch 17/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0821 -
val_loss: 0.0878
Epoch 18/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0802 -
val_loss: 0.0864
Epoch 19/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0788 -
val_loss: 0.0744
Epoch 20/100
84868/84868 [==============================] - 15s 181us/step - loss: 0.0773 -
val_loss: 0.0749
Epoch 21/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0763 -
val_loss: 0.0766
Epoch 22/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0755 -
val_loss: 0.0909
Epoch 23/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0738 -
val_loss: 0.0965
Epoch 24/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0735 -
val_loss: 0.0708
Epoch 25/100
84868/84868 [==============================] - 16s 194us/step - loss: 0.0725 -
val_loss: 0.0931
Epoch 26/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0721 -
val_loss: 0.0745
Epoch 27/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0712 -
val_loss: 0.0804
Epoch 28/100
84868/84868 [==============================] - 15s 181us/step - loss: 0.0704 -
val_loss: 0.0848
Epoch 29/100
84868/84868 [==============================] - 17s 201us/step - loss: 0.0701 -
```

```
val_loss: 0.0759
Epoch 30/100
84868/84868 [==============================] - 17s 202us/step - loss: 0.0688 -
val_loss: 0.0822
Epoch 31/100
84868/84868 [==============================] - 17s 197us/step - loss: 0.0683 -
val_loss: 0.0838
Epoch 32/100
84868/84868 [==============================] - 15s 181us/step - loss: 0.0683 -
val_loss: 0.0711
Epoch 33/100
84868/84868 [==============================] - 15s 181us/step - loss: 0.0673 -
val_loss: 0.0656
Epoch 34/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0662 -
val_loss: 0.0696
Epoch 35/100
84868/84868 [==============================] - 15s 182us/step - loss: 0.0656 -
val_loss: 0.0721
Epoch 36/100
84868/84868 [==============================] - 15s 181us/step - loss: 0.0656 -
val_loss: 0.0727
Epoch 37/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0653 -
val_loss: 0.0670
Epoch 38/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0648 -
val_loss: 0.0641
Epoch 39/100
84868/84868 [==============================] - 15s 181us/step - loss: 0.0643 -
val_loss: 0.0733
Epoch 40/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0650 -
val_loss: 0.0751
Epoch 41/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0634 -
val_loss: 0.0679
Epoch 42/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0642 -
val_loss: 0.0637
Epoch 43/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0629 -
val_loss: 0.0569
Epoch 44/100
84868/84868 [==============================] - 15s 178us/step - loss: 0.0628 -
val_loss: 0.0741
Epoch 45/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0614 -
```

```
val_loss: 0.0679
Epoch 46/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0603 -
val_loss: 0.0712
Epoch 47/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0599 -
val_loss: 0.0719
Epoch 48/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0601 -
val_loss: 0.0641
Epoch 49/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0599 -
val_loss: 0.0622
Epoch 50/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0600 -
val_loss: 0.0609
Epoch 51/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0597 -
val_loss: 0.0662
Epoch 52/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0589 -
val_loss: 0.0688
Epoch 53/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0586 -
val_loss: 0.0713
Epoch 54/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0584 -
val_loss: 0.0667
Epoch 55/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0586 -
val_loss: 0.0684
Epoch 56/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0585 -
val_loss: 0.0613
Epoch 57/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0585 -
val_loss: 0.0592
Epoch 58/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0583 -
val_loss: 0.0610
Epoch 59/100
84868/84868 [==============================] - 15s 181us/step - loss: 0.0580 -
val_loss: 0.0726
Epoch 60/100
84868/84868 [==============================] - 15s 181us/step - loss: 0.0579 -
val_loss: 0.0688
Epoch 61/100
84868/84868 [==============================] - 15s 182us/step - loss: 0.0573 -
```

```
val_loss: 0.0678
Epoch 62/100
84868/84868 [==============================] - 15s 181us/step - loss: 0.0575 -
val_loss: 0.0666
Epoch 63/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0574 -
val_loss: 0.0581
Epoch 64/100
84868/84868 [==============================] - 15s 181us/step - loss: 0.0572 -
val_loss: 0.0635
Epoch 65/100
84868/84868 [==============================] - 15s 178us/step - loss: 0.0566 -
val_loss: 0.0629
Epoch 66/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0553 -
val_loss: 0.0671
Epoch 67/100
84868/84868 [==============================] - 15s 181us/step - loss: 0.0555 -
val_loss: 0.0630
Epoch 68/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0555 -
val_loss: 0.0679
Epoch 69/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0550 -
val_loss: 0.0587
Epoch 70/100
84868/84868 [==============================] - 15s 181us/step - loss: 0.0548 -
val_loss: 0.0623
Epoch 71/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0542 -
val_loss: 0.0580
Epoch 72/100
84868/84868 [==============================] - 15s 181us/step - loss: 0.0538 -
val_loss: 0.0576
Epoch 73/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0542 -
val_loss: 0.0624
Epoch 74/100
84868/84868 [==============================] - 16s 183us/step - loss: 0.0552 -
val_loss: 0.0587
Epoch 75/100
84868/84868 [==============================] - 16s 185us/step - loss: 0.0545 -
val_loss: 0.0665
Epoch 76/100
84868/84868 [==============================] - 15s 181us/step - loss: 0.0537 -
val_loss: 0.0596
Epoch 77/100
84868/84868 [==============================] - 15s 181us/step - loss: 0.0534 -
```

```
val_loss: 0.0645
Epoch 78/100
84868/84868 [==============================] - 15s 182us/step - loss: 0.0531 -
val_loss: 0.0656
Epoch 79/100
84868/84868 [==============================] - 15s 182us/step - loss: 0.0536 -
val_loss: 0.0608
Epoch 80/100
84868/84868 [==============================] - 15s 182us/step - loss: 0.0531 -
val_loss: 0.0562
Epoch 81/100
84868/84868 [==============================] - 15s 182us/step - loss: 0.0521 -
val_loss: 0.0528
Epoch 82/100
84868/84868 [==============================] - 15s 182us/step - loss: 0.0519 -
val_loss: 0.0565
Epoch 83/100
84868/84868 [==============================] - 15s 181us/step - loss: 0.0521 -
val_loss: 0.0623
Epoch 84/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0517 -
val_loss: 0.0563
Epoch 85/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0515 -
val_loss: 0.0529
Epoch 86/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0512 -
val_loss: 0.0569
Epoch 87/100
84868/84868 [==============================] - 15s 178us/step - loss: 0.0517 -
val_loss: 0.0501
Epoch 88/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0514 -
val_loss: 0.0513
Epoch 89/100
84868/84868 [==============================] - 15s 178us/step - loss: 0.0515 -
val_loss: 0.0541
Epoch 90/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0521 -
val_loss: 0.0521
Epoch 91/100
84868/84868 [==============================] - 15s 181us/step - loss: 0.0500 -
val_loss: 0.0545
Epoch 92/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0509 -
val_loss: 0.0550
Epoch 93/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0503 -
```

```
val_loss: 0.0641
Epoch 94/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0502 -
val_loss: 0.0533
Epoch 95/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0504 -
val_loss: 0.0571
Epoch 96/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0504 -
val_loss: 0.0568
Epoch 97/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0499 -
val_loss: 0.0535
Epoch 98/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0497 -
val_loss: 0.0546
Epoch 99/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0493 -
val_loss: 0.0554
Epoch 100/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0505 -
val_loss: 0.0557
```

[21]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
Train on 84868 samples, validate on 21218 samples
Epoch 1/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0502 -
val_loss: 0.0488
Epoch 2/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.0501 -
val_loss: 0.0592
Epoch 3/100
84868/84868 [==============================] - 15s 182us/step - loss: 0.0505 -
val_loss: 0.0683
Epoch 4/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0502 -
val_loss: 0.0499
Epoch 5/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0499 -
val_loss: 0.0507
Epoch 6/100
84868/84868 [==============================] - 15s 181us/step - loss: 0.0498 -
```

```
val_loss: 0.0511
Epoch 7/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0497 -
val_loss: 0.0567
Epoch 8/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0498 -
val_loss: 0.0636
Epoch 9/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0499 -
val_loss: 0.0536
Epoch 10/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0497 -
val_loss: 0.0548
Epoch 11/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0494 -
val_loss: 0.0592
Epoch 12/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0494 -
val_loss: 0.0524
Epoch 13/100
84868/84868 [==============================] - 15s 178us/step - loss: 0.0495 -
val_loss: 0.0558
Epoch 14/100
84868/84868 [==============================] - 15s 178us/step - loss: 0.0490 -
val_loss: 0.0609
Epoch 15/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0498 -
val_loss: 0.0530
Epoch 16/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0494 -
val_loss: 0.0537
Epoch 17/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0499 -
val_loss: 0.0532
Epoch 18/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0492 -
val_loss: 0.0723
Epoch 19/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0490 -
val_loss: 0.0436
Epoch 20/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0491 -
val_loss: 0.0481
Epoch 21/100
84868/84868 [==============================] - 15s 182us/step - loss: 0.0494 -
val_loss: 0.0627
Epoch 22/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0498 -
```

```
val_loss: 0.0519
Epoch 23/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0492 -
val_loss: 0.0558
Epoch 24/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0493 -
val_loss: 0.0677
Epoch 25/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0493 -
val_loss: 0.0616
Epoch 26/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0487 -
val_loss: 0.0505
Epoch 27/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0488 -
val_loss: 0.0456
Epoch 28/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0480 -
val_loss: 0.0520
Epoch 29/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0494 -
val_loss: 0.0493
Epoch 30/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0491 -
val_loss: 0.0537
Epoch 31/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0483 -
val_loss: 0.0490
Epoch 32/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0484 -
val_loss: 0.0590
Epoch 33/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0487 -
val_loss: 0.0478
Epoch 34/100
84868/84868 [==============================] - 15s 178us/step - loss: 0.0481 -
val_loss: 0.0560
Epoch 35/100
84868/84868 [==============================] - 15s 178us/step - loss: 0.0491 -
val_loss: 0.0517
Epoch 36/100
84868/84868 [==============================] - 15s 178us/step - loss: 0.0486 -
val_loss: 0.0596
Epoch 37/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0493 -
val_loss: 0.0563
Epoch 38/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0514 -
```

```
val_loss: 0.0549
Epoch 39/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0495 -
val_loss: 0.0500
Epoch 40/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0498 -
val_loss: 0.0663
Epoch 41/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0496 -
val_loss: 0.0492
Epoch 42/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0503 -
val_loss: 0.0482
Epoch 43/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0498 -
val_loss: 0.0587
Epoch 44/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0501 -
val_loss: 0.0526
Epoch 45/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0501 -
val_loss: 0.0501
Epoch 46/100
84868/84868 [==============================] - 15s 178us/step - loss: 0.0495 -
val_loss: 0.0530
Epoch 47/100
84868/84868 [==============================] - 15s 181us/step - loss: 0.0490 -
val_loss: 0.0547
Epoch 48/100
84868/84868 [==============================] - 15s 180us/step - loss: 0.0506 -
val_loss: 0.0506
Epoch 49/100
84868/84868 [==============================] - 16s 194us/step - loss: 0.0505 -
val_loss: 0.0635
Epoch 50/100
84868/84868 [==============================] - 17s 202us/step - loss: 0.0493 -
val_loss: 0.0573
Epoch 51/100
84868/84868 [==============================] - 17s 204us/step - loss: 0.0494 -
val_loss: 0.0599
Epoch 52/100
84868/84868 [==============================] - 17s 199us/step - loss: 0.0487 -
val_loss: 0.0494
Epoch 53/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0498 -
val_loss: 0.0555
Epoch 54/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.0497 -
```

```
val_loss: 0.0590
Epoch 55/100
84868/84868 [==============================] - 18s 213us/step - loss: 0.0500 -
val_loss: 0.0700
Epoch 56/100
84868/84868 [==============================] - 17s 200us/step - loss: 0.0490 -
val_loss: 0.0461
Epoch 57/100
84868/84868 [==============================] - 17s 197us/step - loss: 0.0484 -
val_loss: 0.0514
Epoch 58/100
84868/84868 [==============================] - 17s 195us/step - loss: 0.0495 -
val_loss: 0.0478
Epoch 59/100
84868/84868 [==============================] - 17s 194us/step - loss: 0.0480 -
val_loss: 0.0623
Epoch 60/100
84868/84868 [==============================] - 16s 194us/step - loss: 0.0487 -
val_loss: 0.0507
Epoch 61/100
84868/84868 [==============================] - 17s 195us/step - loss: 0.0496 -
val_loss: 0.0473
Epoch 62/100
84868/84868 [==============================] - 16s 194us/step - loss: 0.0499 -
val_loss: 0.0506
Epoch 63/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.0486 -
val_loss: 0.0775
Epoch 64/100
84868/84868 [==============================] - 17s 194us/step - loss: 0.0492 -
val_loss: 0.0496
Epoch 65/100
84868/84868 [==============================] - 17s 196us/step - loss: 0.0500 -
val_loss: 0.0784
Epoch 66/100
84868/84868 [==============================] - 16s 194us/step - loss: 0.0499 -
val_loss: 0.0471
Epoch 67/100
84868/84868 [==============================] - 16s 194us/step - loss: 0.0507 -
val_loss: 0.0640
Epoch 68/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.0504 -
val_loss: 0.0477
Epoch 69/100
84868/84868 [==============================] - 16s 194us/step - loss: 0.0489 -
val_loss: 0.0475
Epoch 70/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.0497 -
```

```
val_loss: 0.0464
Epoch 71/100
84868/84868 [==============================] - 16s 194us/step - loss: 0.0492 -
val_loss: 0.0540
Epoch 72/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.0492 -
val_loss: 0.0508
Epoch 73/100
84868/84868 [==============================] - 17s 204us/step - loss: 0.0484 -
val_loss: 0.0615
Epoch 74/100
84868/84868 [==============================] - 17s 197us/step - loss: 0.0484 -
val_loss: 0.0530
Epoch 75/100
84868/84868 [==============================] - 18s 207us/step - loss: 0.0489 -
val_loss: 0.0501
Epoch 76/100
84868/84868 [==============================] - 18s 217us/step - loss: 0.0487 -
val_loss: 0.0450
Epoch 77/100
84868/84868 [==============================] - 17s 205us/step - loss: 0.0485 -
val_loss: 0.0718
Epoch 78/100
84868/84868 [==============================] - 16s 194us/step - loss: 0.0483 -
val_loss: 0.0485
Epoch 79/100
84868/84868 [==============================] - 16s 186us/step - loss: 0.0486 -
val_loss: 0.0527
Epoch 80/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0484 -
val_loss: 0.0411
Epoch 81/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0480 -
val_loss: 0.0442
Epoch 82/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0485 -
val_loss: 0.0562
Epoch 83/100
84868/84868 [==============================] - 15s 178us/step - loss: 0.0489 -
val_loss: 0.0515
Epoch 84/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0484 -
val_loss: 0.0468
Epoch 85/100
84868/84868 [==============================] - 15s 178us/step - loss: 0.0495 -
val_loss: 0.0463
Epoch 86/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0479 -
```

```
val_loss: 0.0468
Epoch 87/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0473 -
val_loss: 0.0567
Epoch 88/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0478 -
val_loss: 0.0476
Epoch 89/100
84868/84868 [==============================] - 15s 178us/step - loss: 0.0486 -
val_loss: 0.0487
Epoch 90/100
84868/84868 [==============================] - 15s 178us/step - loss: 0.0488 -
val_loss: 0.0542
Epoch 91/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0470 -
val_loss: 0.0468
Epoch 92/100
84868/84868 [==============================] - 15s 177us/step - loss: 0.0482 -
val_loss: 0.0465
Epoch 93/100
84868/84868 [==============================] - 15s 176us/step - loss: 0.0480 -
val_loss: 0.0530
Epoch 94/100
84868/84868 [==============================] - 15s 177us/step - loss: 0.0478 -
val_loss: 0.0469
Epoch 95/100
84868/84868 [==============================] - 15s 177us/step - loss: 0.0494 -
val_loss: 0.0491
Epoch 96/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0484 -
val_loss: 0.0552
Epoch 97/100
84868/84868 [==============================] - 15s 179us/step - loss: 0.0487 -
val_loss: 0.0633
Epoch 98/100
84868/84868 [==============================] - 15s 177us/step - loss: 0.0484 -
val_loss: 0.0442
Epoch 99/100
84868/84868 [==============================] - 15s 177us/step - loss: 0.0468 -
val_loss: 0.0426
Epoch 100/100
84868/84868 [==============================] - 15s 178us/step - loss: 0.0481 -
val_loss: 0.0552
```

[22]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[23]:
```
Y_test =np.round(Y_test, 0)
```

[24]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

```
Estado: 3., predicted:3.
Estado: 1., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

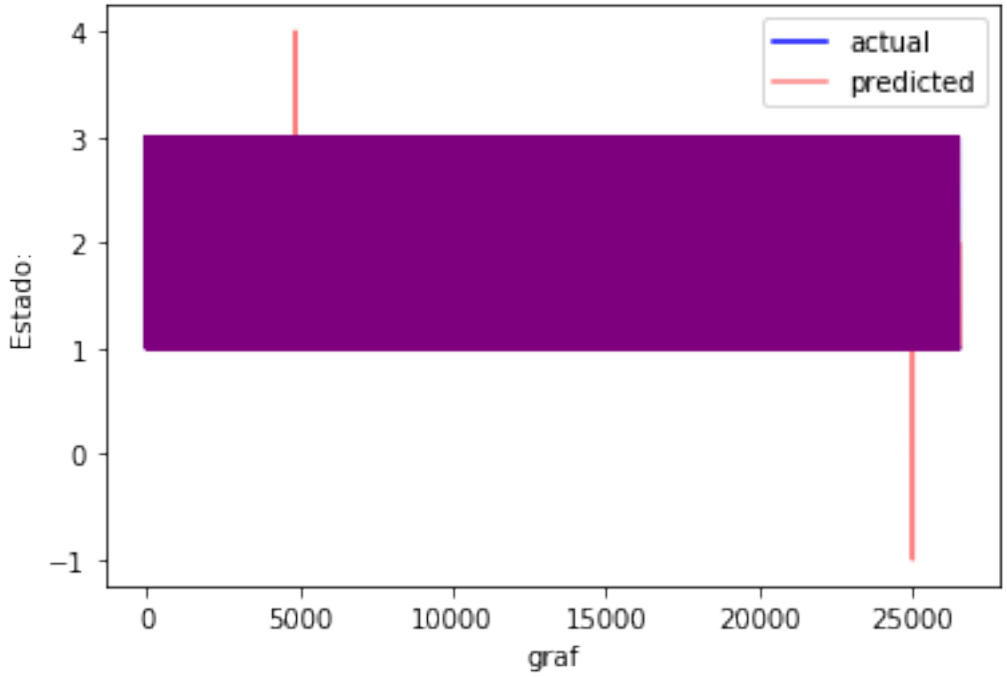
Evaluamos los resultados:

[25]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.8923511571157836
```

[26]:
```
plt.plot((Y_val),
    color="b", label="actual")
```

```
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[27]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[27]: 0.9690822713219214

[ ]:

TFM_wrist_elux4_mae_Adadel_final

August 21, 2019

1 Wrist Data

Librerías:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Wrist

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
```

```
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>=4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
```

```
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT2=np.concatenate((B1,A2),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>=4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
```

```
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=25*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT3=np.concatenate((B1,A2),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>=4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
```

```
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=35*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT4=np.concatenate((B1,A2),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>=4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
```

```
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT5=np.concatenate((B1,A2),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>=4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT6=np.concatenate((B1,A2),axis=1)
```

```
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>=4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
```

```
B1 = np.asmatrix(E)
B1=B1.T
MAT7=np.concatenate((B1,A2),axis=1)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>=4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
```

```
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT8=np.concatenate((B1,A2),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>=4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
```

```
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT9=np.concatenate((B1,A2),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>=4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
```

```
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT10=np.concatenate((B1,A2),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>=4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
```

```
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT11=np.concatenate((B1,A2),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>=4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
```

```
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT12=np.concatenate((B1,A2),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>=4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
```

```
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT13=np.concatenate((B1,A2),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>=4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
```

```
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=24*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT14=np.concatenate((B1,A2),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>=4), 0)
MAT14= np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
```

```
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=29*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT15=np.concatenate((B1,A2),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>=4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
 ↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Dividimos en los conjuntos de validación y entrenamiento:

[18]:
```
X = MAT[:, 1:10]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)
```

Definimos la arquitectura de la red:

[23]:
```
model = Sequential([
    Dense(128, input_shape=(7,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
```

```
    Dense(60),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

Entrenamos la red:

[24]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
Train on 84868 samples, validate on 21218 samples
Epoch 1/100
84868/84868 [==============================] - 15s 181us/step - loss: 0.4074 -
val_loss: 0.2950
Epoch 2/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.2448 -
val_loss: 0.2329
Epoch 3/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.1999 -
val_loss: 0.1898
Epoch 4/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.1763 -
val_loss: 0.1624
Epoch 5/100
84868/84868 [==============================] - 12s 147us/step - loss: 0.1620 -
val_loss: 0.1461
Epoch 6/100
84868/84868 [==============================] - 12s 143us/step - loss: 0.1517 -
val_loss: 0.1501
Epoch 7/100
84868/84868 [==============================] - 12s 147us/step - loss: 0.1429 -
val_loss: 0.1437
Epoch 8/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.1363 -
val_loss: 0.1312
Epoch 9/100
84868/84868 [==============================] - 12s 142us/step - loss: 0.1305 -
val_loss: 0.1508
Epoch 10/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.1254 -
val_loss: 0.1165
Epoch 11/100
84868/84868 [==============================] - 13s 148us/step - loss: 0.1213 -
```

```
val_loss: 0.1399
Epoch 12/100
84868/84868 [==============================] - 12s 144us/step - loss: 0.1183 -
val_loss: 0.1313
Epoch 13/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.1157 -
val_loss: 0.1373
Epoch 14/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.1128 -
val_loss: 0.1326
Epoch 15/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.1108 -
val_loss: 0.1060
Epoch 16/100
84868/84868 [==============================] - 12s 136us/step - loss: 0.1084 -
val_loss: 0.1185
Epoch 17/100
84868/84868 [==============================] - 11s 133us/step - loss: 0.1071 -
val_loss: 0.1065
Epoch 18/100
84868/84868 [==============================] - 11s 133us/step - loss: 0.1045 -
val_loss: 0.1149
Epoch 19/100
84868/84868 [==============================] - 11s 134us/step - loss: 0.1033 -
val_loss: 0.1060
Epoch 20/100
84868/84868 [==============================] - 11s 133us/step - loss: 0.1021 -
val_loss: 0.1173
Epoch 21/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.1002 -
val_loss: 0.1101
Epoch 22/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.0985 -
val_loss: 0.0917
Epoch 23/100
84868/84868 [==============================] - 11s 133us/step - loss: 0.0972 -
val_loss: 0.1184
Epoch 24/100
84868/84868 [==============================] - 11s 133us/step - loss: 0.0960 -
val_loss: 0.1014
Epoch 25/100
84868/84868 [==============================] - 11s 133us/step - loss: 0.0955 -
val_loss: 0.0972
Epoch 26/100
84868/84868 [==============================] - 11s 133us/step - loss: 0.0950 -
val_loss: 0.0986
Epoch 27/100
84868/84868 [==============================] - 11s 133us/step - loss: 0.0932 -
```

```
val_loss: 0.1078
Epoch 28/100
84868/84868 [==============================] - 11s 133us/step - loss: 0.0924 -
val_loss: 0.0936
Epoch 29/100
84868/84868 [==============================] - 11s 134us/step - loss: 0.0918 -
val_loss: 0.1078
Epoch 30/100
84868/84868 [==============================] - 11s 135us/step - loss: 0.0908 -
val_loss: 0.1083
Epoch 31/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.0901 -
val_loss: 0.0989
Epoch 32/100
84868/84868 [==============================] - 11s 133us/step - loss: 0.0894 -
val_loss: 0.1030
Epoch 33/100
84868/84868 [==============================] - 11s 133us/step - loss: 0.0888 -
val_loss: 0.0890
Epoch 34/100
84868/84868 [==============================] - 11s 133us/step - loss: 0.0882 -
val_loss: 0.0956
Epoch 35/100
84868/84868 [==============================] - 11s 134us/step - loss: 0.0876 -
val_loss: 0.1056
Epoch 36/100
84868/84868 [==============================] - 11s 133us/step - loss: 0.0871 -
val_loss: 0.0870
Epoch 37/100
84868/84868 [==============================] - 11s 133us/step - loss: 0.0860 -
val_loss: 0.0998
Epoch 38/100
84868/84868 [==============================] - 12s 144us/step - loss: 0.0854 -
val_loss: 0.0930
Epoch 39/100
84868/84868 [==============================] - 12s 144us/step - loss: 0.0850 -
val_loss: 0.0924
Epoch 40/100
84868/84868 [==============================] - 12s 143us/step - loss: 0.0846 -
val_loss: 0.0913
Epoch 41/100
84868/84868 [==============================] - 12s 145us/step - loss: 0.0832 -
val_loss: 0.0913
Epoch 42/100
84868/84868 [==============================] - 12s 144us/step - loss: 0.0833 -
val_loss: 0.0985
Epoch 43/100
84868/84868 [==============================] - 12s 144us/step - loss: 0.0832 -
```

```
val_loss: 0.0940
Epoch 44/100
84868/84868 [==============================] - 12s 146us/step - loss: 0.0832 -
val_loss: 0.1046
Epoch 45/100
84868/84868 [==============================] - 12s 146us/step - loss: 0.0826 -
val_loss: 0.1055
Epoch 46/100
84868/84868 [==============================] - 12s 145us/step - loss: 0.0814 -
val_loss: 0.0785
Epoch 47/100
84868/84868 [==============================] - 12s 146us/step - loss: 0.0814 -
val_loss: 0.0913
Epoch 48/100
84868/84868 [==============================] - 12s 144us/step - loss: 0.0811 -
val_loss: 0.0901
Epoch 49/100
84868/84868 [==============================] - 12s 141us/step - loss: 0.0804 -
val_loss: 0.0793
Epoch 50/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0805 -
val_loss: 0.0810
Epoch 51/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0799 -
val_loss: 0.0876
Epoch 52/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0797 -
val_loss: 0.0823
Epoch 53/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0792 -
val_loss: 0.0772
Epoch 54/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0783 -
val_loss: 0.0869
Epoch 55/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0785 -
val_loss: 0.0767
Epoch 56/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0779 -
val_loss: 0.1017
Epoch 57/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0773 -
val_loss: 0.0977
Epoch 58/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0768 -
val_loss: 0.0703
Epoch 59/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0767 -
```

```
val_loss: 0.0856
Epoch 60/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0760 -
val_loss: 0.0753
Epoch 61/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0761 -
val_loss: 0.0738
Epoch 62/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0759 -
val_loss: 0.0731
Epoch 63/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0756 -
val_loss: 0.0803
Epoch 64/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0755 -
val_loss: 0.0641
Epoch 65/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0752 -
val_loss: 0.0976
Epoch 66/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0747 -
val_loss: 0.0869
Epoch 67/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0743 -
val_loss: 0.0701
Epoch 68/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0743 -
val_loss: 0.0778
Epoch 69/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0740 -
val_loss: 0.0822
Epoch 70/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0736 -
val_loss: 0.0916
Epoch 71/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0733 -
val_loss: 0.0763
Epoch 72/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0732 -
val_loss: 0.0869
Epoch 73/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0728 -
val_loss: 0.0954
Epoch 74/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0731 -
val_loss: 0.0844
Epoch 75/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0725 -
```

```
val_loss: 0.0758
Epoch 76/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0727 -
val_loss: 0.0819
Epoch 77/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0727 -
val_loss: 0.0761
Epoch 78/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0722 -
val_loss: 0.0779
Epoch 79/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0726 -
val_loss: 0.0786
Epoch 80/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0718 -
val_loss: 0.0675
Epoch 81/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0716 -
val_loss: 0.0756
Epoch 82/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0709 -
val_loss: 0.0887
Epoch 83/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0708 -
val_loss: 0.0744
Epoch 84/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0711 -
val_loss: 0.0810
Epoch 85/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0707 -
val_loss: 0.0723
Epoch 86/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0703 -
val_loss: 0.0865
Epoch 87/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0707 -
val_loss: 0.0790
Epoch 88/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0705 -
val_loss: 0.0675
Epoch 89/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0696 -
val_loss: 0.0781
Epoch 90/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0697 -
val_loss: 0.0758
Epoch 91/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0695 -
```

```
val_loss: 0.0680
Epoch 92/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0689 -
val_loss: 0.0848
Epoch 93/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0689 -
val_loss: 0.0977
Epoch 94/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0685 -
val_loss: 0.0837
Epoch 95/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0684 -
val_loss: 0.0720
Epoch 96/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0679 -
val_loss: 0.0784
Epoch 97/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0682 -
val_loss: 0.0692
Epoch 98/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0679 -
val_loss: 0.0655
Epoch 99/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0673 -
val_loss: 0.0601
Epoch 100/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0674 -
val_loss: 0.0835
```

[25]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
Train on 84868 samples, validate on 21218 samples
Epoch 1/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0667 -
val_loss: 0.0720
Epoch 2/100
84868/84868 [==============================] - 12s 136us/step - loss: 0.0666 -
val_loss: 0.0674
Epoch 3/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0669 -
val_loss: 0.0683
Epoch 4/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0667 -
```

```
val_loss: 0.0816
Epoch 5/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0666 -
val_loss: 0.0710
Epoch 6/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0663 -
val_loss: 0.0751
Epoch 7/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0663 -
val_loss: 0.0910
Epoch 8/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0661 -
val_loss: 0.0644
Epoch 9/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0661 -
val_loss: 0.0563
Epoch 10/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0656 -
val_loss: 0.0770
Epoch 11/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0655 -
val_loss: 0.0713
Epoch 12/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0655 -
val_loss: 0.0982
Epoch 13/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0658 -
val_loss: 0.0819
Epoch 14/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0650 -
val_loss: 0.0831
Epoch 15/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0652 -
val_loss: 0.0703
Epoch 16/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0647 -
val_loss: 0.0884
Epoch 17/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0645 -
val_loss: 0.0734
Epoch 18/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0642 -
val_loss: 0.0636
Epoch 19/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0648 -
val_loss: 0.0676
Epoch 20/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0645 -
```

```
val_loss: 0.0941
Epoch 21/100
84868/84868 [==============================] - 12s 136us/step - loss: 0.0654 -
val_loss: 0.0711
Epoch 22/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0640 -
val_loss: 0.0617
Epoch 23/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0645 -
val_loss: 0.0808
Epoch 24/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0642 -
val_loss: 0.0615
Epoch 25/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0645 -
val_loss: 0.0681
Epoch 26/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0639 -
val_loss: 0.0616
Epoch 27/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0637 -
val_loss: 0.0655
Epoch 28/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0639 -
val_loss: 0.0803
Epoch 29/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0634 -
val_loss: 0.0718
Epoch 30/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0641 -
val_loss: 0.0615
Epoch 31/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0629 -
val_loss: 0.0672
Epoch 32/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0638 -
val_loss: 0.0912
Epoch 33/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0636 -
val_loss: 0.0624
Epoch 34/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0629 -
val_loss: 0.0561
Epoch 35/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0632 -
val_loss: 0.0592
Epoch 36/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0628 -
```

```
val_loss: 0.0666
Epoch 37/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0629 -
val_loss: 0.0568
Epoch 38/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0633 -
val_loss: 0.0717
Epoch 39/100
84868/84868 [==============================] - 12s 136us/step - loss: 0.0627 -
val_loss: 0.0880
Epoch 40/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0629 -
val_loss: 0.0540
Epoch 41/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0628 -
val_loss: 0.0690
Epoch 42/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0623 -
val_loss: 0.0712
Epoch 43/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0623 -
val_loss: 0.0856
Epoch 44/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0623 -
val_loss: 0.0730
Epoch 45/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0628 -
val_loss: 0.0728
Epoch 46/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0634 -
val_loss: 0.0622
Epoch 47/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0630 -
val_loss: 0.0675
Epoch 48/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0629 -
val_loss: 0.0565
Epoch 49/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0627 -
val_loss: 0.0712
Epoch 50/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0624 -
val_loss: 0.0752
Epoch 51/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0622 -
val_loss: 0.0807
Epoch 52/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0625 -
```

```
val_loss: 0.0761
Epoch 53/100
84868/84868 [==============================] - 12s 141us/step - loss: 0.0615 -
val_loss: 0.0793
Epoch 54/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0620 -
val_loss: 0.0807
Epoch 55/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0625 -
val_loss: 0.0812
Epoch 56/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0615 -
val_loss: 0.0665
Epoch 57/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0620 -
val_loss: 0.0648
Epoch 58/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0623 -
val_loss: 0.0626
Epoch 59/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0620 -
val_loss: 0.0592
Epoch 60/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0612 -
val_loss: 0.0638
Epoch 61/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0621 -
val_loss: 0.0728
Epoch 62/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0618 -
val_loss: 0.0643
Epoch 63/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0620 -
val_loss: 0.0892
Epoch 64/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0620 -
val_loss: 0.0616
Epoch 65/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0617 -
val_loss: 0.0740
Epoch 66/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0611 -
val_loss: 0.0542
Epoch 67/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0614 -
val_loss: 0.0751
Epoch 68/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0610 -
```

```
val_loss: 0.0686
Epoch 69/100
84868/84868 [==============================] - 12s 136us/step - loss: 0.0607 -
val_loss: 0.0693
Epoch 70/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0612 -
val_loss: 0.0598
Epoch 71/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0611 -
val_loss: 0.0802
Epoch 72/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0611 -
val_loss: 0.0623
Epoch 73/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0610 -
val_loss: 0.0722
Epoch 74/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0606 -
val_loss: 0.0645
Epoch 75/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0610 -
val_loss: 0.0763
Epoch 76/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0608 -
val_loss: 0.0729
Epoch 77/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0610 -
val_loss: 0.0721
Epoch 78/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0606 -
val_loss: 0.0599
Epoch 79/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0607 -
val_loss: 0.0644
Epoch 80/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0606 -
val_loss: 0.0737
Epoch 81/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0605 -
val_loss: 0.0751
Epoch 82/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0604 -
val_loss: 0.0637
Epoch 83/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0599 -
val_loss: 0.0663
Epoch 84/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0605 -
```

```
val_loss: 0.0753
Epoch 85/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0597 -
val_loss: 0.0600
Epoch 86/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0599 -
val_loss: 0.0609
Epoch 87/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0599 -
val_loss: 0.0572
Epoch 88/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0596 -
val_loss: 0.0711
Epoch 89/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0601 -
val_loss: 0.0541
Epoch 90/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0597 -
val_loss: 0.0615
Epoch 91/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0598 -
val_loss: 0.0572
Epoch 92/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0595 -
val_loss: 0.0574
Epoch 93/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0598 -
val_loss: 0.0531
Epoch 94/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0588 -
val_loss: 0.0511
Epoch 95/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0597 -
val_loss: 0.0628
Epoch 96/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0598 -
val_loss: 0.0670
Epoch 97/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0592 -
val_loss: 0.0818
Epoch 98/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0596 -
val_loss: 0.0786
Epoch 99/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0590 -
val_loss: 0.0634
Epoch 100/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0592 -
```

```
val_loss: 0.0581
```

[26]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
↪validation_split=0.2)
```

```
Train on 84868 samples, validate on 21218 samples
Epoch 1/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0593 -
val_loss: 0.0749
Epoch 2/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0590 -
val_loss: 0.0555
Epoch 3/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0592 -
val_loss: 0.0622
Epoch 4/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0590 -
val_loss: 0.0620
Epoch 5/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0586 -
val_loss: 0.0559
Epoch 6/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0594 -
val_loss: 0.0579
Epoch 7/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0587 -
val_loss: 0.0558
Epoch 8/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0593 -
val_loss: 0.0708
Epoch 9/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0586 -
val_loss: 0.0659
Epoch 10/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0587 -
val_loss: 0.0872
Epoch 11/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0592 -
val_loss: 0.0523
Epoch 12/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0583 -
val_loss: 0.0834
Epoch 13/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0581 -
```

```
val_loss: 0.0604
Epoch 14/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0588 -
val_loss: 0.0552
Epoch 15/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0586 -
val_loss: 0.0658
Epoch 16/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0576 -
val_loss: 0.0582
Epoch 17/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0585 -
val_loss: 0.0760
Epoch 18/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0586 -
val_loss: 0.0640
Epoch 19/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0579 -
val_loss: 0.0701
Epoch 20/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0578 -
val_loss: 0.0589
Epoch 21/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0582 -
val_loss: 0.0596
Epoch 22/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0579 -
val_loss: 0.0768
Epoch 23/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0575 -
val_loss: 0.0744
Epoch 24/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0576 -
val_loss: 0.0686
Epoch 25/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0579 -
val_loss: 0.0539
Epoch 26/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0572 -
val_loss: 0.0657
Epoch 27/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0577 -
val_loss: 0.0633
Epoch 28/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0574 -
val_loss: 0.0649
Epoch 29/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0573 -
```

```
val_loss: 0.0502
Epoch 30/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0570 -
val_loss: 0.0574
Epoch 31/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0567 -
val_loss: 0.0651
Epoch 32/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0571 -
val_loss: 0.0560
Epoch 33/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0571 -
val_loss: 0.0568
Epoch 34/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0564 -
val_loss: 0.0515
Epoch 35/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0566 -
val_loss: 0.0617
Epoch 36/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0568 -
val_loss: 0.0657
Epoch 37/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0569 -
val_loss: 0.0513
Epoch 38/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0565 -
val_loss: 0.0658
Epoch 39/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0566 -
val_loss: 0.0495
Epoch 40/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0564 -
val_loss: 0.0627
Epoch 41/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0565 -
val_loss: 0.0712
Epoch 42/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0572 -
val_loss: 0.0575
Epoch 43/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0566 -
val_loss: 0.0524
Epoch 44/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0564 -
val_loss: 0.0817
Epoch 45/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0559 -
```

```
val_loss: 0.0548
Epoch 46/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0562 -
val_loss: 0.0530
Epoch 47/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0563 -
val_loss: 0.0857
Epoch 48/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0568 -
val_loss: 0.0693
Epoch 49/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0559 -
val_loss: 0.0750
Epoch 50/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0558 -
val_loss: 0.0532
Epoch 51/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0560 -
val_loss: 0.0585
Epoch 52/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0557 -
val_loss: 0.0618
Epoch 53/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0564 -
val_loss: 0.0661
Epoch 54/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0558 -
val_loss: 0.0654
Epoch 55/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0561 -
val_loss: 0.0526
Epoch 56/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0558 -
val_loss: 0.0623
Epoch 57/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0556 -
val_loss: 0.0712
Epoch 58/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0507 -
val_loss: 0.0584
Epoch 59/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0481 -
val_loss: 0.0647
Epoch 60/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0482 -
val_loss: 0.0578
Epoch 61/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0476 -
```

```
val_loss: 0.0467
Epoch 62/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0469 -
val_loss: 0.0469
Epoch 63/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0476 -
val_loss: 0.0552
Epoch 64/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0475 -
val_loss: 0.0391
Epoch 65/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0470 -
val_loss: 0.0517
Epoch 66/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0473 -
val_loss: 0.0608
Epoch 67/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0472 -
val_loss: 0.0437
Epoch 68/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0464 -
val_loss: 0.0570
Epoch 69/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0467 -
val_loss: 0.0598
Epoch 70/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0464 -
val_loss: 0.0519
Epoch 71/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0466 -
val_loss: 0.0480
Epoch 72/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0463 -
val_loss: 0.0531
Epoch 73/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0468 -
val_loss: 0.0625
Epoch 74/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0469 -
val_loss: 0.0705
Epoch 75/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0466 -
val_loss: 0.0667
Epoch 76/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0461 -
val_loss: 0.0455
Epoch 77/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0472 -
```

```
val_loss: 0.0456
Epoch 78/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0468 -
val_loss: 0.0632
Epoch 79/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0470 -
val_loss: 0.0641
Epoch 80/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0464 -
val_loss: 0.0479
Epoch 81/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0465 -
val_loss: 0.0578
Epoch 82/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0466 -
val_loss: 0.0499
Epoch 83/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0466 -
val_loss: 0.0509
Epoch 84/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0472 -
val_loss: 0.0546
Epoch 85/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0467 -
val_loss: 0.0466
Epoch 86/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0470 -
val_loss: 0.0550
Epoch 87/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0464 -
val_loss: 0.0529
Epoch 88/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0468 -
val_loss: 0.0524
Epoch 89/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0466 -
val_loss: 0.0526
Epoch 90/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0467 -
val_loss: 0.0569
Epoch 91/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0467 -
val_loss: 0.0465
Epoch 92/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0468 -
val_loss: 0.0658
Epoch 93/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0464 -
```

```
val_loss: 0.0462
Epoch 94/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0467 -
val_loss: 0.0573
Epoch 95/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0470 -
val_loss: 0.0475
Epoch 96/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0465 -
val_loss: 0.0647
Epoch 97/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0467 -
val_loss: 0.0551
Epoch 98/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0469 -
val_loss: 0.0539
Epoch 99/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0471 -
val_loss: 0.0506
Epoch 100/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0468 -
val_loss: 0.0484
```

[27]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[28]:
```
Y_test =np.round(Y_test, 0)
```

[29]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

```
Estado: 3., predicted:3.
Estado: 1., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

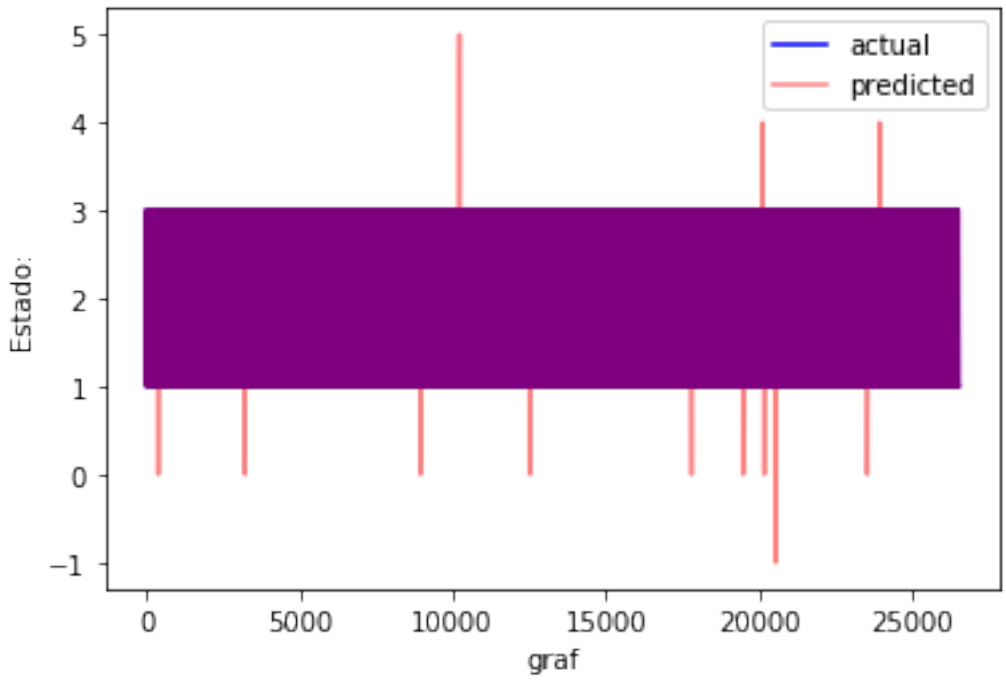
```
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

Evaluamos los resultados:

[30]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.9481733666596786
```

[31]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[32]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[32]: 0.9816755900761632

[ ]:

Wrist_elux5_mae_Adadel_final

August 26, 2019

1 Wrist Data

Librerías:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Wrist

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
```

```
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>=4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
```

```
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT2=np.concatenate((B1,A2),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>=4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
```

```
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=25*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT3=np.concatenate((B1,A2),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>=4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
```

```
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=35*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT4=np.concatenate((B1,A2),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>=4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
```

```
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT5=np.concatenate((B1,A2),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>=4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT6=np.concatenate((B1,A2),axis=1)
```

```
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>=4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
```

```
B1 = np.asmatrix(E)
B1=B1.T
MAT7=np.concatenate((B1,A2),axis=1)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>=4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
```

```
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT8=np.concatenate((B1,A2),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>=4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
```

```
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT9=np.concatenate((B1,A2),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>=4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
```

```
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT10=np.concatenate((B1,A2),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>=4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
```

```
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT11=np.concatenate((B1,A2),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>=4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
```

```
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT12=np.concatenate((B1,A2),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>=4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
```

```
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT13=np.concatenate((B1,A2),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>=4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
```

```
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=24*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT14=np.concatenate((B1,A2),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>=4), 0)
MAT14= np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
```

```
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=29*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT15=np.concatenate((B1,A2),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>=4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
 ↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Dividimos en los conjuntos de validación y entrenamiento:

[37]:
```
X = MAT[:, 1:8]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)
```

Definimos la arquitectura de la red:

[38]:
```
model = Sequential([
    Dense(128, input_shape=(7,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
```

```
    Dense(48),
    Activation('elu'),
    Dense(24),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

Entrenamos la red:

[39]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
Train on 84868 samples, validate on 21218 samples
Epoch 1/100
84868/84868 [==============================] - 14s 162us/step - loss: 0.3536 -
val_loss: 0.2678
Epoch 2/100
84868/84868 [==============================] - 14s 160us/step - loss: 0.2102 -
val_loss: 0.1813
Epoch 3/100
84868/84868 [==============================] - 15s 172us/step - loss: 0.1675 -
val_loss: 0.1591
Epoch 4/100
84868/84868 [==============================] - 14s 170us/step - loss: 0.1488 -
val_loss: 0.1369
Epoch 5/100
84868/84868 [==============================] - 14s 163us/step - loss: 0.1358 -
val_loss: 0.1355
Epoch 6/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.1271 -
val_loss: 0.1336
Epoch 7/100
84868/84868 [==============================] - 14s 163us/step - loss: 0.1197 -
val_loss: 0.1297
Epoch 8/100
84868/84868 [==============================] - 15s 174us/step - loss: 0.1117 -
val_loss: 0.1246
Epoch 9/100
84868/84868 [==============================] - 13s 159us/step - loss: 0.1079 -
val_loss: 0.1012
Epoch 10/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.1039 -
val_loss: 0.1194
```

```
Epoch 11/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.1014 -
val_loss: 0.0978
Epoch 12/100
84868/84868 [==============================] - 13s 159us/step - loss: 0.0994 -
val_loss: 0.1030
Epoch 13/100
84868/84868 [==============================] - 14s 165us/step - loss: 0.0971 -
val_loss: 0.1096
Epoch 14/100
84868/84868 [==============================] - 14s 162us/step - loss: 0.0948 -
val_loss: 0.0897
Epoch 15/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0931 -
val_loss: 0.0955
Epoch 16/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0909 -
val_loss: 0.0865
Epoch 17/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0893 -
val_loss: 0.0842
Epoch 18/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0883 -
val_loss: 0.0861
Epoch 19/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0868 -
val_loss: 0.0916
Epoch 20/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0861 -
val_loss: 0.0841
Epoch 21/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0852 -
val_loss: 0.0932
Epoch 22/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0844 -
val_loss: 0.0769
Epoch 23/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0834 -
val_loss: 0.0940
Epoch 24/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0829 -
val_loss: 0.0863
Epoch 25/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0816 -
val_loss: 0.0809
Epoch 26/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0807 -
val_loss: 0.0809
```

```
Epoch 27/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0798 -
val_loss: 0.0827
Epoch 28/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0791 -
val_loss: 0.0842
Epoch 29/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0790 -
val_loss: 0.0882
Epoch 30/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0774 -
val_loss: 0.0812
Epoch 31/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0769 -
val_loss: 0.0757
Epoch 32/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0758 -
val_loss: 0.0790
Epoch 33/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0754 -
val_loss: 0.0799
Epoch 34/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0749 -
val_loss: 0.0711
Epoch 35/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0743 -
val_loss: 0.0778
Epoch 36/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0739 -
val_loss: 0.0823
Epoch 37/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0733 -
val_loss: 0.0903
Epoch 38/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0730 -
val_loss: 0.0672
Epoch 39/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0719 -
val_loss: 0.0663
Epoch 40/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0717 -
val_loss: 0.0938
Epoch 41/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0713 -
val_loss: 0.0696
Epoch 42/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0703 -
val_loss: 0.0959
```

```
Epoch 43/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0702 -
val_loss: 0.0830
Epoch 44/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0699 -
val_loss: 0.0812
Epoch 45/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0702 -
val_loss: 0.0681
Epoch 46/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0688 -
val_loss: 0.0782
Epoch 47/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0692 -
val_loss: 0.0734
Epoch 48/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0693 -
val_loss: 0.0796
Epoch 49/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0677 -
val_loss: 0.0663
Epoch 50/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0626 -
val_loss: 0.0528
Epoch 51/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0603 -
val_loss: 0.0772
Epoch 52/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0603 -
val_loss: 0.0712
Epoch 53/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0601 -
val_loss: 0.0639
Epoch 54/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0588 -
val_loss: 0.0544
Epoch 55/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0586 -
val_loss: 0.0750
Epoch 56/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0575 -
val_loss: 0.0637
Epoch 57/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0580 -
val_loss: 0.0577
Epoch 58/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0582 -
val_loss: 0.0641
```

```
Epoch 59/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0570 -
val_loss: 0.0609
Epoch 60/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0574 -
val_loss: 0.0634
Epoch 61/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0577 -
val_loss: 0.0647
Epoch 62/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0564 -
val_loss: 0.0679
Epoch 63/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0563 -
val_loss: 0.0763
Epoch 64/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0560 -
val_loss: 0.0457
Epoch 65/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0556 -
val_loss: 0.0545
Epoch 66/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0543 -
val_loss: 0.0453
Epoch 67/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0552 -
val_loss: 0.0638
Epoch 68/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0539 -
val_loss: 0.0608
Epoch 69/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0543 -
val_loss: 0.0723
Epoch 70/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0536 -
val_loss: 0.0679
Epoch 71/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0535 -
val_loss: 0.0711
Epoch 72/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0529 -
val_loss: 0.0667
Epoch 73/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0534 -
val_loss: 0.0514
Epoch 74/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0520 -
val_loss: 0.0532
```

```
Epoch 75/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0526 -
val_loss: 0.0589
Epoch 76/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0519 -
val_loss: 0.0546
Epoch 77/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0518 -
val_loss: 0.0535
Epoch 78/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0520 -
val_loss: 0.0426
Epoch 79/100
84868/84868 [==============================] - 14s 162us/step - loss: 0.0524 -
val_loss: 0.0741
Epoch 80/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0516 -
val_loss: 0.0588
Epoch 81/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0519 -
val_loss: 0.0534
Epoch 82/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0519 -
val_loss: 0.0526
Epoch 83/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0507 -
val_loss: 0.0670
Epoch 84/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0513 -
val_loss: 0.0547
Epoch 85/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0503 -
val_loss: 0.0582
Epoch 86/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0499 -
val_loss: 0.0511
Epoch 87/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0506 -
val_loss: 0.0488
Epoch 88/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0497 -
val_loss: 0.0446
Epoch 89/100
84868/84868 [==============================] - 14s 161us/step - loss: 0.0499 -
val_loss: 0.0523
Epoch 90/100
84868/84868 [==============================] - 14s 165us/step - loss: 0.0491 -
val_loss: 0.0535
```

```
Epoch 91/100
84868/84868 [==============================] - 14s 165us/step - loss: 0.0498 -
val_loss: 0.0462
Epoch 92/100
84868/84868 [==============================] - 14s 164us/step - loss: 0.0499 -
val_loss: 0.0530
Epoch 93/100
84868/84868 [==============================] - 14s 166us/step - loss: 0.0494 -
val_loss: 0.0559
Epoch 94/100
84868/84868 [==============================] - 14s 165us/step - loss: 0.0489 -
val_loss: 0.0626
Epoch 95/100
84868/84868 [==============================] - 14s 165us/step - loss: 0.0494 -
val_loss: 0.0564
Epoch 96/100
84868/84868 [==============================] - 14s 167us/step - loss: 0.0486 -
val_loss: 0.0440
Epoch 97/100
84868/84868 [==============================] - 14s 165us/step - loss: 0.0486 -
val_loss: 0.0614
Epoch 98/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0485 -
val_loss: 0.0548
Epoch 99/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0478 -
val_loss: 0.0836
Epoch 100/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0480 -
val_loss: 0.0543
```

[40]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
Train on 84868 samples, validate on 21218 samples
Epoch 1/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0473 -
val_loss: 0.0456
Epoch 2/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0474 -
val_loss: 0.0583
Epoch 3/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0479 -
val_loss: 0.0558
```

```
Epoch 4/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0474 -
val_loss: 0.0600
Epoch 5/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0475 -
val_loss: 0.0395
Epoch 6/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0466 -
val_loss: 0.0453
Epoch 7/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0473 -
val_loss: 0.0539
Epoch 8/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0466 -
val_loss: 0.0328
Epoch 9/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0473 -
val_loss: 0.0503
Epoch 10/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0468 -
val_loss: 0.0411
Epoch 11/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0472 -
val_loss: 0.0617
Epoch 12/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0473 -
val_loss: 0.0471
Epoch 13/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0465 -
val_loss: 0.0571
Epoch 14/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0466 -
val_loss: 0.0407
Epoch 15/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0477 -
val_loss: 0.0500
Epoch 16/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0469 -
val_loss: 0.0600
Epoch 17/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0464 -
val_loss: 0.0677
Epoch 18/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0466 -
val_loss: 0.0418
Epoch 19/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0469 -
val_loss: 0.0751
```

```
Epoch 20/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0461 -
val_loss: 0.0667
Epoch 21/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0463 -
val_loss: 0.0565
Epoch 22/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0462 -
val_loss: 0.0620
Epoch 23/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0457 -
val_loss: 0.0667
Epoch 24/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0462 -
val_loss: 0.0468
Epoch 25/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0470 -
val_loss: 0.0398
Epoch 26/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0464 -
val_loss: 0.0562
Epoch 27/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0467 -
val_loss: 0.0565
Epoch 28/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0458 -
val_loss: 0.0689
Epoch 29/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0459 -
val_loss: 0.0488
Epoch 30/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0454 -
val_loss: 0.0552
Epoch 31/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0454 -
val_loss: 0.0674
Epoch 32/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0466 -
val_loss: 0.0476
Epoch 33/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0459 -
val_loss: 0.0492
Epoch 34/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0470 -
val_loss: 0.0612
Epoch 35/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0467 -
val_loss: 0.0575
```

```
Epoch 36/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0464 -
val_loss: 0.0670
Epoch 37/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0463 -
val_loss: 0.0546
Epoch 38/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0467 -
val_loss: 0.0772
Epoch 39/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0462 -
val_loss: 0.0430
Epoch 40/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0463 -
val_loss: 0.0663
Epoch 41/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0458 -
val_loss: 0.0496
Epoch 42/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0460 -
val_loss: 0.0465
Epoch 43/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0451 -
val_loss: 0.0659
Epoch 44/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0464 -
val_loss: 0.0492
Epoch 45/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0462 -
val_loss: 0.0615
Epoch 46/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0460 -
val_loss: 0.0604
Epoch 47/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0465 -
val_loss: 0.0433
Epoch 48/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0457 -
val_loss: 0.0654
Epoch 49/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0459 -
val_loss: 0.0459
Epoch 50/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0466 -
val_loss: 0.0578
Epoch 51/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0465 -
val_loss: 0.0544
```

```
Epoch 52/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0466 -
val_loss: 0.0444
Epoch 53/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0461 -
val_loss: 0.0463
Epoch 54/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0463 -
val_loss: 0.0438
Epoch 55/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0457 -
val_loss: 0.0676
Epoch 56/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0468 -
val_loss: 0.0457
Epoch 57/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0464 -
val_loss: 0.0671
Epoch 58/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0464 -
val_loss: 0.0426
Epoch 59/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0463 -
val_loss: 0.0672
Epoch 60/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0472 -
val_loss: 0.0423
Epoch 61/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0467 -
val_loss: 0.0561
Epoch 62/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0466 -
val_loss: 0.0569
Epoch 63/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0464 -
val_loss: 0.0451
Epoch 64/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0464 -
val_loss: 0.0603
Epoch 65/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0470 -
val_loss: 0.0518
Epoch 66/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0466 -
val_loss: 0.0440
Epoch 67/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0469 -
val_loss: 0.0458
```

```
Epoch 68/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0470 -
val_loss: 0.0459
Epoch 69/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0467 -
val_loss: 0.0482
Epoch 70/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0471 -
val_loss: 0.0454
Epoch 71/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0475 -
val_loss: 0.0573
Epoch 72/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0466 -
val_loss: 0.0423
Epoch 73/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0468 -
val_loss: 0.0469
Epoch 74/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0467 -
val_loss: 0.0595
Epoch 75/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0469 -
val_loss: 0.0400
Epoch 76/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0471 -
val_loss: 0.0489
Epoch 77/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0463 -
val_loss: 0.0657
Epoch 78/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0465 -
val_loss: 0.0737
Epoch 79/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0457 -
val_loss: 0.0490
Epoch 80/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0466 -
val_loss: 0.0409
Epoch 81/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0471 -
val_loss: 0.0770
Epoch 82/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0461 -
val_loss: 0.0457
Epoch 83/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0467 -
val_loss: 0.0541
```

```
Epoch 84/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0466 -
val_loss: 0.0431
Epoch 85/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0471 -
val_loss: 0.0557
Epoch 86/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0455 -
val_loss: 0.0569
Epoch 87/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0458 -
val_loss: 0.0549
Epoch 88/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0459 -
val_loss: 0.0556
Epoch 89/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0455 -
val_loss: 0.0665
Epoch 90/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0453 -
val_loss: 0.0570
Epoch 91/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0448 -
val_loss: 0.0477
Epoch 92/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0452 -
val_loss: 0.0613
Epoch 93/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0455 -
val_loss: 0.0568
Epoch 94/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0459 -
val_loss: 0.0626
Epoch 95/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0461 -
val_loss: 0.0410
Epoch 96/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0444 -
val_loss: 0.0490
Epoch 97/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0439 -
val_loss: 0.0484
Epoch 98/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0451 -
val_loss: 0.0614
Epoch 99/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0454 -
val_loss: 0.0376
```

```
Epoch 100/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0442 -
val_loss: 0.0439
```

[41]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
Train on 84868 samples, validate on 21218 samples
Epoch 1/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0445 -
val_loss: 0.0369
Epoch 2/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0442 -
val_loss: 0.0542
Epoch 3/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0449 -
val_loss: 0.0541
Epoch 4/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0443 -
val_loss: 0.0369
Epoch 5/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0448 -
val_loss: 0.0429
Epoch 6/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0449 -
val_loss: 0.0443
Epoch 7/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0448 -
val_loss: 0.0547
Epoch 8/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0442 -
val_loss: 0.0502
Epoch 9/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0448 -
val_loss: 0.0526
Epoch 10/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0444 -
val_loss: 0.0563
Epoch 11/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0450 -
val_loss: 0.0435
Epoch 12/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0445 -
val_loss: 0.0483
```

```
Epoch 13/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0453 -
val_loss: 0.0474
Epoch 14/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0444 -
val_loss: 0.0466
Epoch 15/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0446 -
val_loss: 0.0482
Epoch 16/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0445 -
val_loss: 0.0376
Epoch 17/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0437 -
val_loss: 0.0489
Epoch 18/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0433 -
val_loss: 0.0577
Epoch 19/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0438 -
val_loss: 0.0484
Epoch 20/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0436 -
val_loss: 0.0621
Epoch 21/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0436 -
val_loss: 0.0416
Epoch 22/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0441 -
val_loss: 0.0419
Epoch 23/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0435 -
val_loss: 0.0470
Epoch 24/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0439 -
val_loss: 0.0462
Epoch 25/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0444 -
val_loss: 0.0455
Epoch 26/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0437 -
val_loss: 0.0570
Epoch 27/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0440 -
val_loss: 0.0551
Epoch 28/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0438 -
val_loss: 0.0581
```

```
Epoch 29/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0447 -
val_loss: 0.0393
Epoch 30/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0439 -
val_loss: 0.0575
Epoch 31/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0443 -
val_loss: 0.0374
Epoch 32/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0437 -
val_loss: 0.0628
Epoch 33/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0483 -
val_loss: 0.0500
Epoch 34/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0461 -
val_loss: 0.0454
Epoch 35/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0463 -
val_loss: 0.0513
Epoch 36/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0465 -
val_loss: 0.0393
Epoch 37/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0446 -
val_loss: 0.0419
Epoch 38/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0461 -
val_loss: 0.0475
Epoch 39/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0446 -
val_loss: 0.0550
Epoch 40/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0445 -
val_loss: 0.0480
Epoch 41/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0456 -
val_loss: 0.0504
Epoch 42/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0451 -
val_loss: 0.0534
Epoch 43/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0448 -
val_loss: 0.0642
Epoch 44/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0440 -
val_loss: 0.0404
```

```
Epoch 45/100
84868/84868 [==============================] - 14s 159us/step - loss: 0.0456 -
val_loss: 0.0450
Epoch 46/100
84868/84868 [==============================] - 14s 160us/step - loss: 0.0455 -
val_loss: 0.0393
Epoch 47/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0444 -
val_loss: 0.0419
Epoch 48/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0450 -
val_loss: 0.0444
Epoch 49/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0456 -
val_loss: 0.0593
Epoch 50/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0453 -
val_loss: 0.0472
Epoch 51/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0452 -
val_loss: 0.0400
Epoch 52/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0447 -
val_loss: 0.0507
Epoch 53/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0457 -
val_loss: 0.0665
Epoch 54/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0445 -
val_loss: 0.0341
Epoch 55/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0450 -
val_loss: 0.0496
Epoch 56/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0467 -
val_loss: 0.0479
Epoch 57/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0454 -
val_loss: 0.0400
Epoch 58/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0465 -
val_loss: 0.0573
Epoch 59/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0463 -
val_loss: 0.0405
Epoch 60/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0462 -
val_loss: 0.0387
```

```
Epoch 61/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0452 -
val_loss: 0.0475
Epoch 62/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0462 -
val_loss: 0.0582
Epoch 63/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0459 -
val_loss: 0.0505
Epoch 64/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0460 -
val_loss: 0.0548
Epoch 65/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0470 -
val_loss: 0.0677
Epoch 66/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0464 -
val_loss: 0.0473
Epoch 67/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0464 -
val_loss: 0.0517
Epoch 68/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0463 -
val_loss: 0.0561
Epoch 69/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0462 -
val_loss: 0.0514
Epoch 70/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0461 -
val_loss: 0.0672
Epoch 71/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0466 -
val_loss: 0.0405
Epoch 72/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0463 -
val_loss: 0.0533
Epoch 73/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0469 -
val_loss: 0.0521
Epoch 74/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0473 -
val_loss: 0.0715
Epoch 75/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0469 -
val_loss: 0.0627
Epoch 76/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0486 -
val_loss: 0.0574
```

```
Epoch 77/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0494 -
val_loss: 0.0563
Epoch 78/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0500 -
val_loss: 0.0508
Epoch 79/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0489 -
val_loss: 0.0578
Epoch 80/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0482 -
val_loss: 0.0632
Epoch 81/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0484 -
val_loss: 0.0569
Epoch 82/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0494 -
val_loss: 0.0627
Epoch 83/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0493 -
val_loss: 0.0573
Epoch 84/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0500 -
val_loss: 0.0498
Epoch 85/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0492 -
val_loss: 0.0516
Epoch 86/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0476 -
val_loss: 0.0542
Epoch 87/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0488 -
val_loss: 0.0637
Epoch 88/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0506 -
val_loss: 0.0456
Epoch 89/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0494 -
val_loss: 0.0636
Epoch 90/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0506 -
val_loss: 0.0519
Epoch 91/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0494 -
val_loss: 0.0718
Epoch 92/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0496 -
val_loss: 0.0555
```

```
Epoch 93/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0496 -
val_loss: 0.0626
Epoch 94/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0518 -
val_loss: 0.0531
Epoch 95/100
84868/84868 [==============================] - 14s 160us/step - loss: 0.0511 -
val_loss: 0.0679
Epoch 96/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0514 -
val_loss: 0.0667
Epoch 97/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0498 -
val_loss: 0.0589
Epoch 98/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0506 -
val_loss: 0.0530
Epoch 99/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0491 -
val_loss: 0.0597
Epoch 100/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0515 -
val_loss: 0.0516
```

[42]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[43]:
```
Y_test =np.round(Y_test, 0)
```

[44]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

```
Estado: 3., predicted:3.
Estado: 1., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
```

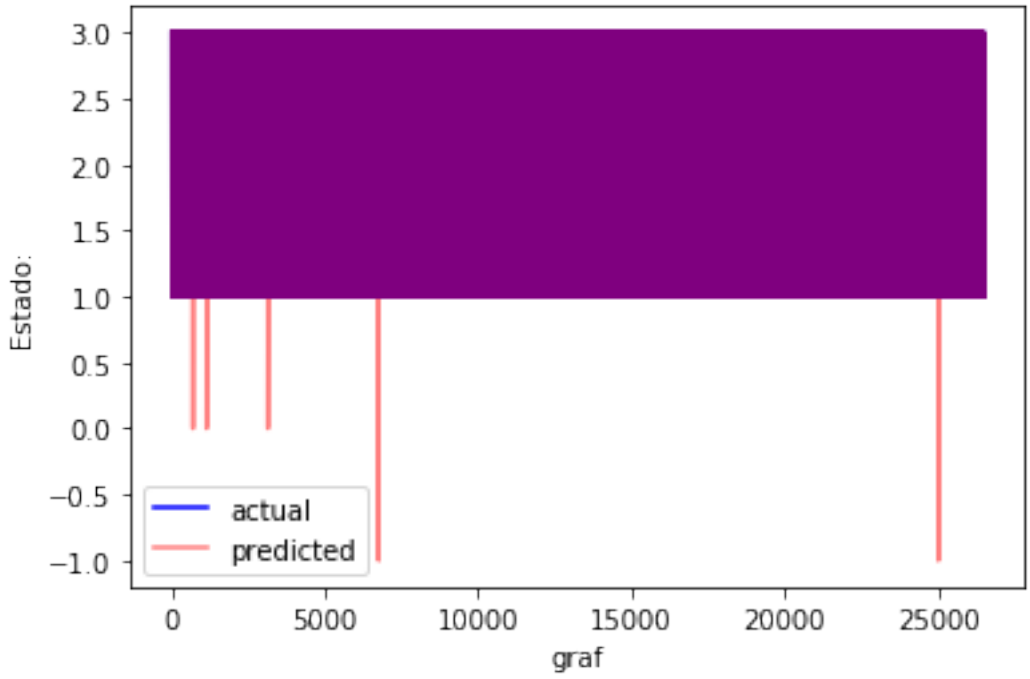
```
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:3.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

Evaluamos los resultados:

[45]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.9343474100437075
```

[46]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[47]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[47]: 0.9774903853404721

[ ]:

Wrist_elux7_mae_Adadel_final

August 26, 2019

1 Wrist Data

Librerías:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Wrist

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
```

```
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>=4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
```

```
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT2=np.concatenate((B1,A2),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>=4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
```

```
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=25*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT3=np.concatenate((B1,A2),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>=4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
```

```
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=35*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT4=np.concatenate((B1,A2),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>=4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
```

```
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT5=np.concatenate((B1,A2),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>=4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT6=np.concatenate((B1,A2),axis=1)
```

```
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>=4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
```

```
B1 = np.asmatrix(E)
B1=B1.T
MAT7=np.concatenate((B1,A2),axis=1)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>=4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
```

```
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT8=np.concatenate((B1,A2),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>=4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
```

```
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT9=np.concatenate((B1,A2),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>=4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
```

```
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT10=np.concatenate((B1,A2),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>=4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
```

```
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT11=np.concatenate((B1,A2),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>=4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
```

```
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT12=np.concatenate((B1,A2),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>=4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
```

```
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT13=np.concatenate((B1,A2),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>=4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
```

```
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=24*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT14=np.concatenate((B1,A2),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>=4), 0)
MAT14= np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
```

```
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=29*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT15=np.concatenate((B1,A2),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>=4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
 ↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Dividimos en los conjuntos de validación y entrenamiento:

[18]:
```
X = MAT[:, 1:8]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)
```

Definimos la arquitectura de la red:

[19]:
```
model = Sequential([
    Dense(128, input_shape=(7,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
```

```
    Dense(60),
    Activation('elu'),
    Dense(40),
    Activation('elu'),
    Dense(40),
    Activation('elu'),
    Dense(30),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

```
WARNING: Logging before flag parsing goes to stderr.
W0826 19:48:56.430401 10972 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:74: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

W0826 19:48:56.463053 10972 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:517: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

W0826 19:48:56.473266 10972 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:4138: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

W0826 19:48:56.567031 10972 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-packages\keras\optimizers.py:790:
The name tf.train.Optimizer is deprecated. Please use
tf.compat.v1.train.Optimizer instead.
```

Entrenamos la red:

[20]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
W0826 19:48:56.823332 10972 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:986: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.
```

```
W0826 19:48:56.829314 10972 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:973: The name tf.assign is
deprecated. Please use tf.compat.v1.assign instead.


Train on 84868 samples, validate on 21218 samples
Epoch 1/100
84868/84868 [==============================] - 21s 248us/step - loss: 0.3493 -
val_loss: 0.2701
Epoch 2/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.1877 -
val_loss: 0.1668
Epoch 3/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.1453 -
val_loss: 0.1336
Epoch 4/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.1276 -
val_loss: 0.1546
Epoch 5/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.1161 -
val_loss: 0.1220
Epoch 6/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.1104 -
val_loss: 0.0999
Epoch 7/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.1058 -
val_loss: 0.1076
Epoch 8/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.1019 -
val_loss: 0.0884
Epoch 9/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0975 -
val_loss: 0.0951
Epoch 10/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0949 -
val_loss: 0.0978
Epoch 11/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0920 -
val_loss: 0.1036
Epoch 12/100
84868/84868 [==============================] - 16s 186us/step - loss: 0.0918 -
val_loss: 0.0928
Epoch 13/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0858 -
val_loss: 0.0812
Epoch 14/100
```

```
84868/84868 [==============================] - 16s 188us/step - loss: 0.0823 -
val_loss: 0.0872
Epoch 15/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0799 -
val_loss: 0.0814
Epoch 16/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0799 -
val_loss: 0.0770
Epoch 17/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0778 -
val_loss: 0.0906
Epoch 18/100
84868/84868 [==============================] - 16s 185us/step - loss: 0.0753 -
val_loss: 0.0834
Epoch 19/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0743 -
val_loss: 0.0724
Epoch 20/100
84868/84868 [==============================] - 16s 186us/step - loss: 0.0733 -
val_loss: 0.0802
Epoch 21/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0722 -
val_loss: 0.0688
Epoch 22/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0712 -
val_loss: 0.0704
Epoch 23/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0713 -
val_loss: 0.0821
Epoch 24/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0696 -
val_loss: 0.0695
Epoch 25/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0680 -
val_loss: 0.0793
Epoch 26/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0666 -
val_loss: 0.0677
Epoch 27/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0667 -
val_loss: 0.0853
Epoch 28/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0669 -
val_loss: 0.0751
Epoch 29/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0664 -
val_loss: 0.0665
Epoch 30/100
```

```
84868/84868 [==============================] - 16s 187us/step - loss: 0.0663 -
val_loss: 0.0637
Epoch 31/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0658 -
val_loss: 0.0613
Epoch 32/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0643 -
val_loss: 0.0677
Epoch 33/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0649 -
val_loss: 0.0755
Epoch 34/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0649 -
val_loss: 0.0612
Epoch 35/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0634 -
val_loss: 0.0635
Epoch 36/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0638 -
val_loss: 0.0811
Epoch 37/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0623 -
val_loss: 0.0704
Epoch 38/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0629 -
val_loss: 0.0682
Epoch 39/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0618 -
val_loss: 0.0745
Epoch 40/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0625 -
val_loss: 0.0677
Epoch 41/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0610 -
val_loss: 0.0691
Epoch 42/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0610 -
val_loss: 0.0651
Epoch 43/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0614 -
val_loss: 0.0964
Epoch 44/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0599 -
val_loss: 0.0649
Epoch 45/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0604 -
val_loss: 0.0790
Epoch 46/100
```

```
84868/84868 [==============================] - 16s 191us/step - loss: 0.0604 -
val_loss: 0.0678
Epoch 47/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0605 -
val_loss: 0.0585
Epoch 48/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0595 -
val_loss: 0.0589
Epoch 49/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0602 -
val_loss: 0.0635
Epoch 50/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0609 -
val_loss: 0.0758
Epoch 51/100
84868/84868 [==============================] - 17s 203us/step - loss: 0.0603 -
val_loss: 0.0802
Epoch 52/100
84868/84868 [==============================] - 17s 203us/step - loss: 0.0594 -
val_loss: 0.0675
Epoch 53/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.0598 -
val_loss: 0.0692
Epoch 54/100
84868/84868 [==============================] - 17s 197us/step - loss: 0.0585 -
val_loss: 0.0683
Epoch 55/100
84868/84868 [==============================] - 16s 186us/step - loss: 0.0601 -
val_loss: 0.0662
Epoch 56/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0587 -
val_loss: 0.0642
Epoch 57/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0587 -
val_loss: 0.0596
Epoch 58/100
84868/84868 [==============================] - 16s 186us/step - loss: 0.0578 -
val_loss: 0.0799
Epoch 59/100
84868/84868 [==============================] - 16s 186us/step - loss: 0.0590 -
val_loss: 0.0656
Epoch 60/100
84868/84868 [==============================] - 16s 186us/step - loss: 0.0576 -
val_loss: 0.0565
Epoch 61/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0573 -
val_loss: 0.0615
Epoch 62/100
```

```
84868/84868 [==============================] - 16s 193us/step - loss: 0.0572 -
val_loss: 0.0698
Epoch 63/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0556 -
val_loss: 0.0646
Epoch 64/100
84868/84868 [==============================] - 17s 195us/step - loss: 0.0564 -
val_loss: 0.0570
Epoch 65/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0554 -
val_loss: 0.0635
Epoch 66/100
84868/84868 [==============================] - 16s 185us/step - loss: 0.0552 -
val_loss: 0.0555
Epoch 67/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.0548 -
val_loss: 0.0597
Epoch 68/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0557 -
val_loss: 0.0539
Epoch 69/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0548 -
val_loss: 0.0569
Epoch 70/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0548 -
val_loss: 0.0580
Epoch 71/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0547 -
val_loss: 0.0581
Epoch 72/100
84868/84868 [==============================] - 16s 186us/step - loss: 0.0545 -
val_loss: 0.0538
Epoch 73/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0549 -
val_loss: 0.0567
Epoch 74/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.0555 -
val_loss: 0.0587
Epoch 75/100
84868/84868 [==============================] - 17s 195us/step - loss: 0.0556 -
val_loss: 0.0619
Epoch 76/100
84868/84868 [==============================] - 17s 197us/step - loss: 0.0549 -
val_loss: 0.0577
Epoch 77/100
84868/84868 [==============================] - 17s 204us/step - loss: 0.0557 -
val_loss: 0.0661
Epoch 78/100
```

```
84868/84868 [==============================] - 18s 207us/step - loss: 0.0556 -
val_loss: 0.0631
Epoch 79/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0557 -
val_loss: 0.0700
Epoch 80/100
84868/84868 [==============================] - 16s 186us/step - loss: 0.0556 -
val_loss: 0.0499
Epoch 81/100
84868/84868 [==============================] - 17s 201us/step - loss: 0.0556 -
val_loss: 0.0588
Epoch 82/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0544 -
val_loss: 0.0587
Epoch 83/100
84868/84868 [==============================] - 18s 209us/step - loss: 0.0555 -
val_loss: 0.0571
Epoch 84/100
84868/84868 [==============================] - 17s 195us/step - loss: 0.0547 -
val_loss: 0.0601
Epoch 85/100
84868/84868 [==============================] - 17s 200us/step - loss: 0.0554 -
val_loss: 0.0606
Epoch 86/100
84868/84868 [==============================] - 17s 202us/step - loss: 0.0544 -
val_loss: 0.0558
Epoch 87/100
84868/84868 [==============================] - 18s 210us/step - loss: 0.0531 -
val_loss: 0.0542
Epoch 88/100
84868/84868 [==============================] - 17s 200us/step - loss: 0.0542 -
val_loss: 0.0702
Epoch 89/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0546 -
val_loss: 0.0577
Epoch 90/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0546 -
val_loss: 0.0603
Epoch 91/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0551 -
val_loss: 0.0596
Epoch 92/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0528 -
val_loss: 0.0600
Epoch 93/100
84868/84868 [==============================] - 16s 185us/step - loss: 0.0533 -
val_loss: 0.0535
Epoch 94/100
```

```
84868/84868 [==============================] - 16s 186us/step - loss: 0.0539 -
val_loss: 0.0659
Epoch 95/100
84868/84868 [==============================] - 16s 194us/step - loss: 0.0562 -
val_loss: 0.0620
Epoch 96/100
84868/84868 [==============================] - 19s 219us/step - loss: 0.0554 -
val_loss: 0.0633
Epoch 97/100
84868/84868 [==============================] - 18s 216us/step - loss: 0.0533 -
val_loss: 0.0643
Epoch 98/100
84868/84868 [==============================] - 18s 217us/step - loss: 0.0551 -
val_loss: 0.0549
Epoch 99/100
84868/84868 [==============================] - 18s 210us/step - loss: 0.0534 -
val_loss: 0.0554
Epoch 100/100
84868/84868 [==============================] - 18s 210us/step - loss: 0.0539 -
val_loss: 0.0548
```

[21]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
Train on 84868 samples, validate on 21218 samples
Epoch 1/100
84868/84868 [==============================] - 17s 200us/step - loss: 0.0551 -
val_loss: 0.0609
Epoch 2/100
84868/84868 [==============================] - 17s 200us/step - loss: 0.0539 -
val_loss: 0.0571
Epoch 3/100
84868/84868 [==============================] - 17s 198us/step - loss: 0.0535 -
val_loss: 0.0556
Epoch 4/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0534 -
val_loss: 0.0613
Epoch 5/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0538 -
val_loss: 0.0649
Epoch 6/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0525 -
val_loss: 0.0636
Epoch 7/100
```

```
84868/84868 [==============================] - 16s 190us/step - loss: 0.0528 -
val_loss: 0.0568
Epoch 8/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0524 -
val_loss: 0.0502
Epoch 9/100
84868/84868 [==============================] - 16s 186us/step - loss: 0.0523 -
val_loss: 0.0515
Epoch 10/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0553 -
val_loss: 0.0611
Epoch 11/100
84868/84868 [==============================] - 17s 195us/step - loss: 0.0524 -
val_loss: 0.0499
Epoch 12/100
84868/84868 [==============================] - 18s 209us/step - loss: 0.0534 -
val_loss: 0.0572
Epoch 13/100
84868/84868 [==============================] - 17s 198us/step - loss: 0.0516 -
val_loss: 0.0574
Epoch 14/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0530 -
val_loss: 0.0538
Epoch 15/100
84868/84868 [==============================] - 16s 186us/step - loss: 0.0525 -
val_loss: 0.0553
Epoch 16/100
84868/84868 [==============================] - 16s 183us/step - loss: 0.0533 -
val_loss: 0.0511
Epoch 17/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0524 -
val_loss: 0.0921
Epoch 18/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0518 -
val_loss: 0.0649
Epoch 19/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0524 -
val_loss: 0.0606
Epoch 20/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0532 -
val_loss: 0.0526
Epoch 21/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0531 -
val_loss: 0.0642
Epoch 22/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0516 -
val_loss: 0.0573
Epoch 23/100
```

```
84868/84868 [==============================] - 16s 188us/step - loss: 0.0517 -
val_loss: 0.0466
Epoch 24/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0512 -
val_loss: 0.0640
Epoch 25/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0513 -
val_loss: 0.0544
Epoch 26/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0506 -
val_loss: 0.0606
Epoch 27/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0506 -
val_loss: 0.0682
Epoch 28/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0504 -
val_loss: 0.0690
Epoch 29/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0514 -
val_loss: 0.0467
Epoch 30/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0517 -
val_loss: 0.0769
Epoch 31/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0507 -
val_loss: 0.0700
Epoch 32/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0517 -
val_loss: 0.0518
Epoch 33/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0537 -
val_loss: 0.0530
Epoch 34/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0505 -
val_loss: 0.0627
Epoch 35/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0522 -
val_loss: 0.0580
Epoch 36/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0502 -
val_loss: 0.0482
Epoch 37/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0511 -
val_loss: 0.0645
Epoch 38/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0506 -
val_loss: 0.0497
Epoch 39/100
```

```
84868/84868 [==============================] - 16s 188us/step - loss: 0.0511 -
val_loss: 0.0526
Epoch 40/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0506 -
val_loss: 0.0531
Epoch 41/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0500 -
val_loss: 0.0546
Epoch 42/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0512 -
val_loss: 0.0645
Epoch 43/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0508 -
val_loss: 0.0563
Epoch 44/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0508 -
val_loss: 0.0481
Epoch 45/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0510 -
val_loss: 0.0630
Epoch 46/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0489 -
val_loss: 0.0481
Epoch 47/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0502 -
val_loss: 0.0635
Epoch 48/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0519 -
val_loss: 0.0515
Epoch 49/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0509 -
val_loss: 0.0569
Epoch 50/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0515 -
val_loss: 0.0517
Epoch 51/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0495 -
val_loss: 0.0542
Epoch 52/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0508 -
val_loss: 0.0481
Epoch 53/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0508 -
val_loss: 0.0505
Epoch 54/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0504 -
val_loss: 0.0599
Epoch 55/100
```

```
84868/84868 [==============================] - 16s 188us/step - loss: 0.0522 -
val_loss: 0.0643
Epoch 56/100
84868/84868 [==============================] - 16s 186us/step - loss: 0.0512 -
val_loss: 0.0591
Epoch 57/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0510 -
val_loss: 0.0605
Epoch 58/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0520 -
val_loss: 0.0530
Epoch 59/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0509 -
val_loss: 0.0641
Epoch 60/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0497 -
val_loss: 0.0553
Epoch 61/100
84868/84868 [==============================] - 16s 186us/step - loss: 0.0522 -
val_loss: 0.0566
Epoch 62/100
84868/84868 [==============================] - 16s 186us/step - loss: 0.0504 -
val_loss: 0.0517
Epoch 63/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0503 -
val_loss: 0.0629
Epoch 64/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0522 -
val_loss: 0.0583
Epoch 65/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0506 -
val_loss: 0.0576
Epoch 66/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0516 -
val_loss: 0.0586
Epoch 67/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0511 -
val_loss: 0.0540
Epoch 68/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0514 -
val_loss: 0.0483
Epoch 69/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0526 -
val_loss: 0.0586
Epoch 70/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0515 -
val_loss: 0.0524
Epoch 71/100
```

```
84868/84868 [==============================] - 16s 188us/step - loss: 0.0508 -
val_loss: 0.0438
Epoch 72/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0518 -
val_loss: 0.0537
Epoch 73/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0520 -
val_loss: 0.0533
Epoch 74/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0515 -
val_loss: 0.0472
Epoch 75/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0526 -
val_loss: 0.0727
Epoch 76/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0528 -
val_loss: 0.0470
Epoch 77/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0539 -
val_loss: 0.0530
Epoch 78/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0531 -
val_loss: 0.0461
Epoch 79/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0520 -
val_loss: 0.0556
Epoch 80/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0514 -
val_loss: 0.0580
Epoch 81/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0521 -
val_loss: 0.0538
Epoch 82/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0537 -
val_loss: 0.0558
Epoch 83/100
84868/84868 [==============================] - 17s 197us/step - loss: 0.0529 -
val_loss: 0.0522
Epoch 84/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0516 -
val_loss: 0.0497
Epoch 85/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0534 -
val_loss: 0.0603
Epoch 86/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0529 -
val_loss: 0.0633
Epoch 87/100
```

```
84868/84868 [==============================] - 16s 188us/step - loss: 0.0538 -
val_loss: 0.0516
Epoch 88/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0527 -
val_loss: 0.0692
Epoch 89/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0519 -
val_loss: 0.0564
Epoch 90/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0540 -
val_loss: 0.0525
Epoch 91/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0521 -
val_loss: 0.0582
Epoch 92/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0527 -
val_loss: 0.0528
Epoch 93/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0523 -
val_loss: 0.0624
Epoch 94/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0520 -
val_loss: 0.0635
Epoch 95/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0519 -
val_loss: 0.0589
Epoch 96/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0552 -
val_loss: 0.0482
Epoch 97/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0524 -
val_loss: 0.0576
Epoch 98/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0529 -
val_loss: 0.0543
Epoch 99/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0576 -
val_loss: 0.0592
Epoch 100/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0538 -
val_loss: 0.0545
```

[22]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
Train on 84868 samples, validate on 21218 samples
Epoch 1/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0543 -
val_loss: 0.0598
Epoch 2/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0541 -
val_loss: 0.0671
Epoch 3/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0564 -
val_loss: 0.0758
Epoch 4/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0543 -
val_loss: 0.0635
Epoch 5/100
84868/84868 [==============================] - 16s 186us/step - loss: 0.0537 -
val_loss: 0.0620
Epoch 6/100
84868/84868 [==============================] - 16s 186us/step - loss: 0.0556 -
val_loss: 0.0566
Epoch 7/100
84868/84868 [==============================] - 16s 186us/step - loss: 0.0554 -
val_loss: 0.0589
Epoch 8/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0556 -
val_loss: 0.0566
Epoch 9/100
84868/84868 [==============================] - 16s 186us/step - loss: 0.0575 -
val_loss: 0.0818
Epoch 10/100
84868/84868 [==============================] - 16s 186us/step - loss: 0.0549 -
val_loss: 0.0564
Epoch 11/100
84868/84868 [==============================] - 16s 186us/step - loss: 0.0588 -
val_loss: 0.0568
Epoch 12/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0555 -
val_loss: 0.0621
Epoch 13/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0570 -
val_loss: 0.0517
Epoch 14/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0539 -
val_loss: 0.0623
Epoch 15/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0546 -
val_loss: 0.0592
Epoch 16/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0544 -
```

```
val_loss: 0.0551
Epoch 17/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0548 -
val_loss: 0.0576
Epoch 18/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0542 -
val_loss: 0.0551
Epoch 19/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0578 -
val_loss: 0.0578
Epoch 20/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0564 -
val_loss: 0.0671
Epoch 21/100
84868/84868 [==============================] - 16s 183us/step - loss: 0.0547 -
val_loss: 0.0517
Epoch 22/100
84868/84868 [==============================] - 16s 183us/step - loss: 0.0608 -
val_loss: 0.0827
Epoch 23/100
84868/84868 [==============================] - 16s 184us/step - loss: 0.0571 -
val_loss: 0.0547
Epoch 24/100
84868/84868 [==============================] - 16s 183us/step - loss: 0.0562 -
val_loss: 0.0527
Epoch 25/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0571 -
val_loss: 0.0550
Epoch 26/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0565 -
val_loss: 0.0549
Epoch 27/100
84868/84868 [==============================] - 16s 183us/step - loss: 0.0561 -
val_loss: 0.0591
Epoch 28/100
84868/84868 [==============================] - 15s 182us/step - loss: 0.0565 -
val_loss: 0.0645
Epoch 29/100
84868/84868 [==============================] - 16s 183us/step - loss: 0.0597 -
val_loss: 0.0582
Epoch 30/100
84868/84868 [==============================] - 15s 182us/step - loss: 0.0575 -
val_loss: 0.0633
Epoch 31/100
84868/84868 [==============================] - 15s 181us/step - loss: 0.0560 -
val_loss: 0.0526
Epoch 32/100
84868/84868 [==============================] - 15s 182us/step - loss: 0.0631 -
```

```
val_loss: 0.0575
Epoch 33/100
84868/84868 [==============================] - 15s 182us/step - loss: 0.0622 -
val_loss: 0.0549
Epoch 34/100
84868/84868 [==============================] - 15s 182us/step - loss: 0.0596 -
val_loss: 0.0623
Epoch 35/100
84868/84868 [==============================] - 16s 186us/step - loss: 0.0572 -
val_loss: 0.0623
Epoch 36/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0569 -
val_loss: 0.0620
Epoch 37/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0626 -
val_loss: 0.0740
Epoch 38/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0594 -
val_loss: 0.0642
Epoch 39/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0616 -
val_loss: 0.0532
Epoch 40/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0617 -
val_loss: 0.0818
Epoch 41/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0627 -
val_loss: 0.0582
Epoch 42/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0583 -
val_loss: 0.0580
Epoch 43/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0592 -
val_loss: 0.0616
Epoch 44/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0623 -
val_loss: 0.0560
Epoch 45/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0626 -
val_loss: 0.0790
Epoch 46/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0648 -
val_loss: 0.0715
Epoch 47/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0624 -
val_loss: 0.0493
Epoch 48/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0627 -
```

```
val_loss: 0.0548
Epoch 49/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0619 -
val_loss: 0.0689
Epoch 50/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0663 -
val_loss: 0.0599
Epoch 51/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0578 -
val_loss: 0.0653
Epoch 52/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0598 -
val_loss: 0.0570
Epoch 53/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0604 -
val_loss: 0.0547
Epoch 54/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0577 -
val_loss: 0.0677
Epoch 55/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0621 -
val_loss: 0.0656
Epoch 56/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0604 -
val_loss: 0.0552
Epoch 57/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0576 -
val_loss: 0.0606
Epoch 58/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0595 -
val_loss: 0.0653
Epoch 59/100
84868/84868 [==============================] - 17s 195us/step - loss: 0.0590 -
val_loss: 0.0680
Epoch 60/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0584 -
val_loss: 0.0540
Epoch 61/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0581 -
val_loss: 0.0636
Epoch 62/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0616 -
val_loss: 0.0482
Epoch 63/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0595 -
val_loss: 0.0543
Epoch 64/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0578 -
```

```
val_loss: 0.0588
Epoch 65/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0588 -
val_loss: 0.0727
Epoch 66/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0581 -
val_loss: 0.0703
Epoch 67/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0571 -
val_loss: 0.0583
Epoch 68/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0585 -
val_loss: 0.0516
Epoch 69/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0607 -
val_loss: 0.0543
Epoch 70/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0588 -
val_loss: 0.0624
Epoch 71/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0578 -
val_loss: 0.0717
Epoch 72/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0560 -
val_loss: 0.0578
Epoch 73/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0585 -
val_loss: 0.0763
Epoch 74/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0581 -
val_loss: 0.0582
Epoch 75/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0557 -
val_loss: 0.0590
Epoch 76/100
84868/84868 [==============================] - 17s 196us/step - loss: 0.0566 -
val_loss: 0.0650
Epoch 77/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0593 -
val_loss: 0.0555
Epoch 78/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0583 -
val_loss: 0.0575
Epoch 79/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0566 -
val_loss: 0.0618
Epoch 80/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0567 -
```

```
val_loss: 0.0506
Epoch 81/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0571 -
val_loss: 0.0533
Epoch 82/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0577 -
val_loss: 0.0521
Epoch 83/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0550 -
val_loss: 0.0542
Epoch 84/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0547 -
val_loss: 0.0536
Epoch 85/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0569 -
val_loss: 0.0744
Epoch 86/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0553 -
val_loss: 0.0562
Epoch 87/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0537 -
val_loss: 0.0608
Epoch 88/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0596 -
val_loss: 0.0517
Epoch 89/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0548 -
val_loss: 0.0690
Epoch 90/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0565 -
val_loss: 0.0575
Epoch 91/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0566 -
val_loss: 0.0620
Epoch 92/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0576 -
val_loss: 0.0505
Epoch 93/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0569 -
val_loss: 0.0517
Epoch 94/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0573 -
val_loss: 0.0632
Epoch 95/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0578 -
val_loss: 0.0584
Epoch 96/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0572 -
```

```
val_loss: 0.0609
Epoch 97/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0549 -
val_loss: 0.0502
Epoch 98/100
84868/84868 [==============================] - 17s 201us/step - loss: 0.0554 -
val_loss: 0.0562
Epoch 99/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.0568 -
val_loss: 0.0840
Epoch 100/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0573 -
val_loss: 0.0621
```

[23]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[24]:
```
Y_test =np.round(Y_test, 0)
```

[25]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

```
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
```

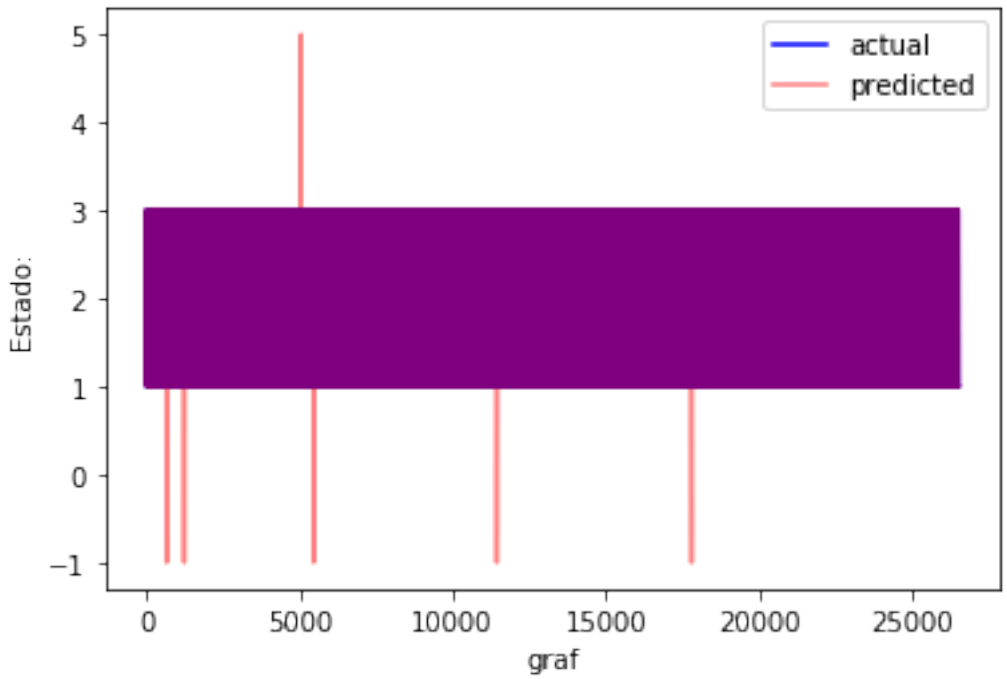
```
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

Evaluamos los resultados:

[26]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.8573412608945405
```

[27]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[28]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[28]:
```
0.9578086117185732
```

[ ]:
```

```

Wrist_elux4_mae_Adadel_final

August 27, 2019

1 Wrist Data

Librerías:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Wrist

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
```

```
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>=4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
```

```
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT2=np.concatenate((B1,A2),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>=4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
```

```
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=25*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT3=np.concatenate((B1,A2),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>=4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
```

```
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=35*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT4=np.concatenate((B1,A2),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>=4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
```

```
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT5=np.concatenate((B1,A2),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>=4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT6=np.concatenate((B1,A2),axis=1)
```

```
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>=4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
```

```
B1 = np.asmatrix(E)
B1=B1.T
MAT7=np.concatenate((B1,A2),axis=1)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>=4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
```

```
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT8=np.concatenate((B1,A2),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>=4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
```

```
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT9=np.concatenate((B1,A2),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>=4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
```

```
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT10=np.concatenate((B1,A2),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>=4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
```

```
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT11=np.concatenate((B1,A2),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>=4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
```

```
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT12=np.concatenate((B1,A2),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>=4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
```

```
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT13=np.concatenate((B1,A2),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>=4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
```

```
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=24*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT14=np.concatenate((B1,A2),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>=4), 0)
MAT14= np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
```

```
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=29*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT15=np.concatenate((B1,A2),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>=4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
 ↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Dividimos en los conjuntos de validación y entrenamiento:

[18]:
```
X = MAT[:, 1:10]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)
```

Definimos la arquitectura de la red:

[19]:
```
model = Sequential([
    Dense(128, input_shape=(7,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
```

```
    Dense(60),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

```
WARNING: Logging before flag parsing goes to stderr.
W0827 17:13:51.872752 16308 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:74: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

W0827 17:13:51.907613 16308 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:517: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

W0827 17:13:51.918725 16308 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:4138: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

W0827 17:13:51.979421 16308 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-packages\keras\optimizers.py:790:
The name tf.train.Optimizer is deprecated. Please use
tf.compat.v1.train.Optimizer instead.
```

Entrenamos la red:

[20]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
W0827 17:13:52.172904 16308 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:986: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.

W0827 17:13:52.178901 16308 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:973: The name tf.assign is
deprecated. Please use tf.compat.v1.assign instead.
```

```
Train on 84868 samples, validate on 21218 samples
Epoch 1/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.3930 -
val_loss: 0.2707
Epoch 2/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.2372 -
val_loss: 0.2054
Epoch 3/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.1939 -
val_loss: 0.1913
Epoch 4/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.1695 -
val_loss: 0.2000
Epoch 5/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.1551 -
val_loss: 0.1491
Epoch 6/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.1447 -
val_loss: 0.1327
Epoch 7/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.1364 -
val_loss: 0.1520
Epoch 8/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.1309 -
val_loss: 0.1303
Epoch 9/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.1254 -
val_loss: 0.1247
Epoch 10/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.1212 -
val_loss: 0.1369
Epoch 11/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.1176 -
val_loss: 0.1087
Epoch 12/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.1145 -
val_loss: 0.1130
Epoch 13/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.1110 -
val_loss: 0.1023
Epoch 14/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.1089 -
val_loss: 0.1204
Epoch 15/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.1064 -
val_loss: 0.1058
Epoch 16/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.1038 -
```

```
val_loss: 0.1162
Epoch 17/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.1020 -
val_loss: 0.1071
Epoch 18/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.1006 -
val_loss: 0.1133
Epoch 19/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0986 -
val_loss: 0.1220
Epoch 20/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0974 -
val_loss: 0.1014
Epoch 21/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0961 -
val_loss: 0.1050
Epoch 22/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0952 -
val_loss: 0.1058
Epoch 23/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0940 -
val_loss: 0.0992
Epoch 24/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0931 -
val_loss: 0.1051
Epoch 25/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0923 -
val_loss: 0.0874
Epoch 26/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0916 -
val_loss: 0.0869
Epoch 27/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0904 -
val_loss: 0.0927
Epoch 28/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0897 -
val_loss: 0.0909
Epoch 29/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0892 -
val_loss: 0.0896
Epoch 30/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0881 -
val_loss: 0.0899
Epoch 31/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0874 -
val_loss: 0.1076
Epoch 32/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0867 -
```

```
val_loss: 0.0878
Epoch 33/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0866 -
val_loss: 0.0895
Epoch 34/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0859 -
val_loss: 0.0993
Epoch 35/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0857 -
val_loss: 0.0855
Epoch 36/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0842 -
val_loss: 0.0852
Epoch 37/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0838 -
val_loss: 0.0944
Epoch 38/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0836 -
val_loss: 0.1087
Epoch 39/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0834 -
val_loss: 0.0881
Epoch 40/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0828 -
val_loss: 0.0870
Epoch 41/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0819 -
val_loss: 0.0822
Epoch 42/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0821 -
val_loss: 0.0820
Epoch 43/100
84868/84868 [==============================] - 12s 136us/step - loss: 0.0822 -
val_loss: 0.0795
Epoch 44/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0816 -
val_loss: 0.0799
Epoch 45/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0808 -
val_loss: 0.0788
Epoch 46/100
84868/84868 [==============================] - 12s 136us/step - loss: 0.0814 -
val_loss: 0.0907
Epoch 47/100
84868/84868 [==============================] - 12s 136us/step - loss: 0.0806 -
val_loss: 0.0954
Epoch 48/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0802 -
```

```
val_loss: 0.1004
Epoch 49/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0798 -
val_loss: 0.1134
Epoch 50/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0794 -
val_loss: 0.1081
Epoch 51/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0797 -
val_loss: 0.0875
Epoch 52/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0790 -
val_loss: 0.0781
Epoch 53/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0786 -
val_loss: 0.0924
Epoch 54/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0784 -
val_loss: 0.1053
Epoch 55/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0783 -
val_loss: 0.0946
Epoch 56/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0778 -
val_loss: 0.0929
Epoch 57/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0776 -
val_loss: 0.0716
Epoch 58/100
84868/84868 [==============================] - 12s 136us/step - loss: 0.0776 -
val_loss: 0.0712
Epoch 59/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0776 -
val_loss: 0.0895
Epoch 60/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0772 -
val_loss: 0.0934
Epoch 61/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0773 -
val_loss: 0.0812
Epoch 62/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0765 -
val_loss: 0.0880
Epoch 63/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0768 -
val_loss: 0.0733
Epoch 64/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0766 -
```

```
val_loss: 0.0861
Epoch 65/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0764 -
val_loss: 0.0863
Epoch 66/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0757 -
val_loss: 0.1002
Epoch 67/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0763 -
val_loss: 0.0874
Epoch 68/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0756 -
val_loss: 0.0794
Epoch 69/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0748 -
val_loss: 0.0814
Epoch 70/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0743 -
val_loss: 0.0778
Epoch 71/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0746 -
val_loss: 0.0916
Epoch 72/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0740 -
val_loss: 0.0866
Epoch 73/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0740 -
val_loss: 0.0787
Epoch 74/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0737 -
val_loss: 0.0827
Epoch 75/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0738 -
val_loss: 0.0861
Epoch 76/100
84868/84868 [==============================] - 12s 136us/step - loss: 0.0738 -
val_loss: 0.0728
Epoch 77/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0727 -
val_loss: 0.0746
Epoch 78/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0732 -
val_loss: 0.0704
Epoch 79/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0729 -
val_loss: 0.0725
Epoch 80/100
84868/84868 [==============================] - 12s 136us/step - loss: 0.0730 -
```

```
val_loss: 0.0827
Epoch 81/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0728 -
val_loss: 0.0910
Epoch 82/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0732 -
val_loss: 0.1024
Epoch 83/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0722 -
val_loss: 0.0725
Epoch 84/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0730 -
val_loss: 0.0810
Epoch 85/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0723 -
val_loss: 0.0703
Epoch 86/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0715 -
val_loss: 0.0706
Epoch 87/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0713 -
val_loss: 0.0744
Epoch 88/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0722 -
val_loss: 0.0711
Epoch 89/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0710 -
val_loss: 0.0892
Epoch 90/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0713 -
val_loss: 0.0826
Epoch 91/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0711 -
val_loss: 0.0772
Epoch 92/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0711 -
val_loss: 0.0903
Epoch 93/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0709 -
val_loss: 0.0690
Epoch 94/100
84868/84868 [==============================] - 12s 136us/step - loss: 0.0704 -
val_loss: 0.0792
Epoch 95/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0700 -
val_loss: 0.0649
Epoch 96/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0700 -
```

```
val_loss: 0.0639
Epoch 97/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0693 -
val_loss: 0.0697
Epoch 98/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0699 -
val_loss: 0.0900
Epoch 99/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0689 -
val_loss: 0.0661
Epoch 100/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0684 -
val_loss: 0.0611
```

[21]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
Train on 84868 samples, validate on 21218 samples
Epoch 1/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0692 -
val_loss: 0.0807
Epoch 2/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0686 -
val_loss: 0.0892
Epoch 3/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0682 -
val_loss: 0.0716
Epoch 4/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0682 -
val_loss: 0.0779
Epoch 5/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0676 -
val_loss: 0.0639
Epoch 6/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0678 -
val_loss: 0.0830
Epoch 7/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0672 -
val_loss: 0.0815
Epoch 8/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0672 -
val_loss: 0.0724
Epoch 9/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0665 -
```

```
val_loss: 0.0739
Epoch 10/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0666 -
val_loss: 0.0884
Epoch 11/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0668 -
val_loss: 0.0846
Epoch 12/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0655 -
val_loss: 0.0717
Epoch 13/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0660 -
val_loss: 0.0650
Epoch 14/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0653 -
val_loss: 0.0727
Epoch 15/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0654 -
val_loss: 0.0661
Epoch 16/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0653 -
val_loss: 0.0627
Epoch 17/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0635 -
val_loss: 0.0732
Epoch 18/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0612 -
val_loss: 0.0754
Epoch 19/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0607 -
val_loss: 0.0765
Epoch 20/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0600 -
val_loss: 0.0573
Epoch 21/100
84868/84868 [==============================] - 12s 136us/step - loss: 0.0609 -
val_loss: 0.0705
Epoch 22/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0596 -
val_loss: 0.0648
Epoch 23/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0587 -
val_loss: 0.0719
Epoch 24/100
84868/84868 [==============================] - 12s 136us/step - loss: 0.0585 -
val_loss: 0.0682
Epoch 25/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0581 -
```

```
val_loss: 0.0720
Epoch 26/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0579 -
val_loss: 0.0757
Epoch 27/100
84868/84868 [==============================] - 12s 136us/step - loss: 0.0576 -
val_loss: 0.0643
Epoch 28/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0579 -
val_loss: 0.0613
Epoch 29/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0567 -
val_loss: 0.0667
Epoch 30/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0573 -
val_loss: 0.0611
Epoch 31/100
84868/84868 [==============================] - 12s 136us/step - loss: 0.0567 -
val_loss: 0.0800
Epoch 32/100
84868/84868 [==============================] - 12s 136us/step - loss: 0.0566 -
val_loss: 0.0697
Epoch 33/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0562 -
val_loss: 0.0761
Epoch 34/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0558 -
val_loss: 0.0732
Epoch 35/100
84868/84868 [==============================] - 12s 136us/step - loss: 0.0557 -
val_loss: 0.0582
Epoch 36/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0552 -
val_loss: 0.0613
Epoch 37/100
84868/84868 [==============================] - 12s 136us/step - loss: 0.0557 -
val_loss: 0.0581
Epoch 38/100
84868/84868 [==============================] - 12s 136us/step - loss: 0.0554 -
val_loss: 0.0635
Epoch 39/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0551 -
val_loss: 0.0612
Epoch 40/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0549 -
val_loss: 0.0636
Epoch 41/100
84868/84868 [==============================] - 12s 146us/step - loss: 0.0548 -
```

```
val_loss: 0.0837
Epoch 42/100
84868/84868 [==============================] - 12s 147us/step - loss: 0.0552 -
val_loss: 0.0546
Epoch 43/100
84868/84868 [==============================] - 13s 147us/step - loss: 0.0551 -
val_loss: 0.0826
Epoch 44/100
84868/84868 [==============================] - 13s 149us/step - loss: 0.0548 -
val_loss: 0.0614
Epoch 45/100
84868/84868 [==============================] - 12s 143us/step - loss: 0.0551 -
val_loss: 0.0664
Epoch 46/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0545 -
val_loss: 0.0739
Epoch 47/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0540 -
val_loss: 0.0699
Epoch 48/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0541 -
val_loss: 0.0699
Epoch 49/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0537 -
val_loss: 0.0735
Epoch 50/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0543 -
val_loss: 0.0627
Epoch 51/100
84868/84868 [==============================] - 12s 136us/step - loss: 0.0536 -
val_loss: 0.0537
Epoch 52/100
84868/84868 [==============================] - 12s 136us/step - loss: 0.0539 -
val_loss: 0.0620
Epoch 53/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0538 -
val_loss: 0.0514
Epoch 54/100
84868/84868 [==============================] - 12s 136us/step - loss: 0.0532 -
val_loss: 0.0579
Epoch 55/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0535 -
val_loss: 0.0635
Epoch 56/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0538 -
val_loss: 0.0772
Epoch 57/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0533 -
```

```
val_loss: 0.0610
Epoch 58/100
84868/84868 [==============================] - 12s 136us/step - loss: 0.0530 -
val_loss: 0.0937
Epoch 59/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0533 -
val_loss: 0.0683
Epoch 60/100
84868/84868 [==============================] - 12s 146us/step - loss: 0.0532 -
val_loss: 0.0480
Epoch 61/100
84868/84868 [==============================] - 14s 160us/step - loss: 0.0526 -
val_loss: 0.0528
Epoch 62/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0525 -
val_loss: 0.0567
Epoch 63/100
84868/84868 [==============================] - 11s 134us/step - loss: 0.0527 -
val_loss: 0.0552
Epoch 64/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.0520 -
val_loss: 0.0670
Epoch 65/100
84868/84868 [==============================] - 11s 131us/step - loss: 0.0527 -
val_loss: 0.0804
Epoch 66/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.0519 -
val_loss: 0.0561
Epoch 67/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.0518 -
val_loss: 0.0602
Epoch 68/100
84868/84868 [==============================] - 11s 131us/step - loss: 0.0521 -
val_loss: 0.0633
Epoch 69/100
84868/84868 [==============================] - 11s 131us/step - loss: 0.0517 -
val_loss: 0.0525
Epoch 70/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.0510 -
val_loss: 0.0534
Epoch 71/100
84868/84868 [==============================] - 11s 133us/step - loss: 0.0515 -
val_loss: 0.0582
Epoch 72/100
84868/84868 [==============================] - 11s 131us/step - loss: 0.0507 -
val_loss: 0.0597
Epoch 73/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.0506 -
```

```
val_loss: 0.0600
Epoch 74/100
84868/84868 [==============================] - 11s 131us/step - loss: 0.0513 -
val_loss: 0.0637
Epoch 75/100
84868/84868 [==============================] - 11s 131us/step - loss: 0.0511 -
val_loss: 0.0550
Epoch 76/100
84868/84868 [==============================] - 11s 131us/step - loss: 0.0513 -
val_loss: 0.0629
Epoch 77/100
84868/84868 [==============================] - 11s 133us/step - loss: 0.0508 -
val_loss: 0.0618
Epoch 78/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.0505 -
val_loss: 0.0530
Epoch 79/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.0505 -
val_loss: 0.0649
Epoch 80/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.0503 -
val_loss: 0.0515
Epoch 81/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.0509 -
val_loss: 0.0507
Epoch 82/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.0500 -
val_loss: 0.0571
Epoch 83/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.0500 -
val_loss: 0.0734
Epoch 84/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.0502 -
val_loss: 0.0518
Epoch 85/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.0494 -
val_loss: 0.0580
Epoch 86/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.0503 -
val_loss: 0.0429
Epoch 87/100
84868/84868 [==============================] - 11s 133us/step - loss: 0.0497 -
val_loss: 0.0743
Epoch 88/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.0498 -
val_loss: 0.0591
Epoch 89/100
84868/84868 [==============================] - 11s 133us/step - loss: 0.0504 -
```

```
val_loss: 0.0509
Epoch 90/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.0501 -
val_loss: 0.0550
Epoch 91/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.0500 -
val_loss: 0.0535
Epoch 92/100
84868/84868 [==============================] - 11s 131us/step - loss: 0.0505 -
val_loss: 0.0555
Epoch 93/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.0498 -
val_loss: 0.0598
Epoch 94/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.0500 -
val_loss: 0.0443
Epoch 95/100
84868/84868 [==============================] - 11s 131us/step - loss: 0.0501 -
val_loss: 0.0498
Epoch 96/100
84868/84868 [==============================] - 11s 131us/step - loss: 0.0495 -
val_loss: 0.0580
Epoch 97/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.0502 -
val_loss: 0.0490
Epoch 98/100
84868/84868 [==============================] - 11s 131us/step - loss: 0.0492 -
val_loss: 0.0718
Epoch 99/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.0493 -
val_loss: 0.0584
Epoch 100/100
84868/84868 [==============================] - 11s 132us/step - loss: 0.0496 -
val_loss: 0.0576
```

[22]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[23]:
```
Y_test =np.round(Y_test, 0)
```

[24]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

```
Estado: 3., predicted:3.
```

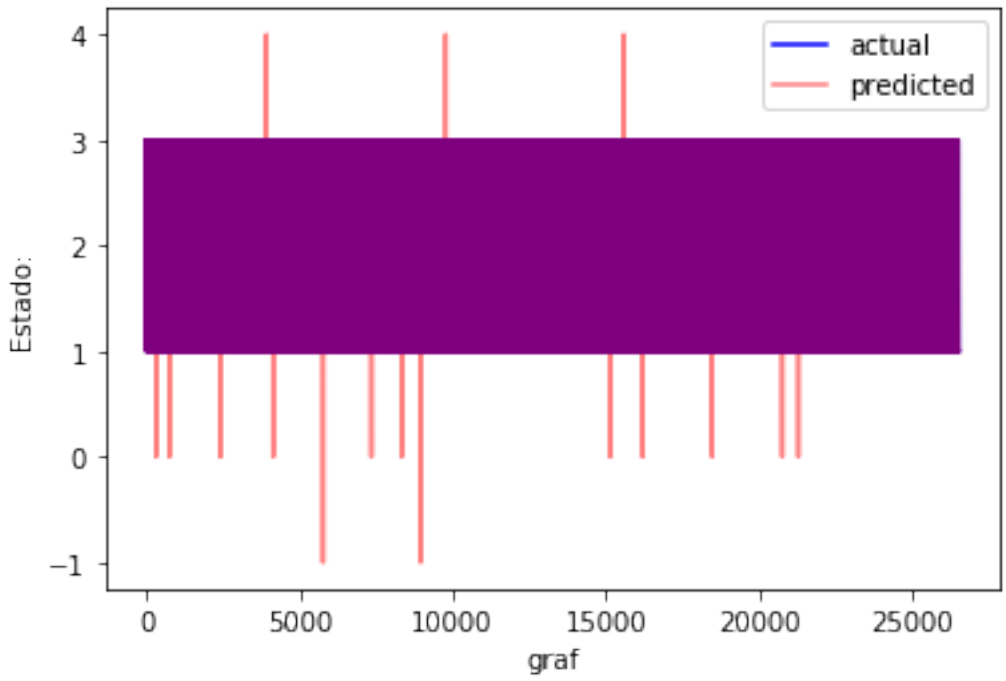
```
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

Evaluamos los resultados:

[25]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.9449951782739201
```

[26]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[27]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[27]: 0.9811100218686374

[ ]:

Wrist_elux5_mae_Adadel_final

August 27, 2019

1 Wrist Data

Librerías:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Wrist

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
```

```
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>=4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
```

```
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT2=np.concatenate((B1,A2),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>=4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
```

```
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=25*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT3=np.concatenate((B1,A2),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>=4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
```

```
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=35*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT4=np.concatenate((B1,A2),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>=4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
```

```
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT5=np.concatenate((B1,A2),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>=4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT6=np.concatenate((B1,A2),axis=1)
```

```
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>=4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
```

```
B1 = np.asmatrix(E)
B1=B1.T
MAT7=np.concatenate((B1,A2),axis=1)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>=4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
```

```
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT8=np.concatenate((B1,A2),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>=4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
```

```
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT9=np.concatenate((B1,A2),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>=4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
```

```
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT10=np.concatenate((B1,A2),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>=4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
```

```
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT11=np.concatenate((B1,A2),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>=4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
```

```
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT12=np.concatenate((B1,A2),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>=4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
```

```
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT13=np.concatenate((B1,A2),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>=4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
```

```
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=24*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT14=np.concatenate((B1,A2),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>=4), 0)
MAT14= np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
```

```
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=29*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT15=np.concatenate((B1,A2),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>=4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
 ↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Dividimos en los conjuntos de validación y entrenamiento:

[18]:
```
X = MAT[:, 1:8]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)
```

Definimos la arquitectura de la red:

[19]:
```
model = Sequential([
    Dense(128, input_shape=(7,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
```

```
    Dense(48),
    Activation('elu'),
    Dense(24),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

```
WARNING: Logging before flag parsing goes to stderr.
W0827 17:58:21.155245 18896 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:74: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

W0827 17:58:21.167284 18896 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:517: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

W0827 17:58:21.171230 18896 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:4138: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

W0827 17:58:21.235178 18896 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-packages\keras\optimizers.py:790:
The name tf.train.Optimizer is deprecated. Please use
tf.compat.v1.train.Optimizer instead.
```

Entrenamos la red:

[20]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
W0827 17:58:21.421569 18896 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:986: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.

W0827 17:58:21.426519 18896 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:973: The name tf.assign is
```

```
deprecated. Please use tf.compat.v1.assign instead.


Train on 84868 samples, validate on 21218 samples
Epoch 1/100
84868/84868 [==============================] - 14s 168us/step - loss: 0.3635 -
val_loss: 0.2587
Epoch 2/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.2186 -
val_loss: 0.1946
Epoch 3/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.1677 -
val_loss: 0.1745
Epoch 4/100
84868/84868 [==============================] - 14s 160us/step - loss: 0.1433 -
val_loss: 0.1260
Epoch 5/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.1290 -
val_loss: 0.1332
Epoch 6/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.1189 -
val_loss: 0.1256
Epoch 7/100
84868/84868 [==============================] - 14s 160us/step - loss: 0.1120 -
val_loss: 0.1143
Epoch 8/100
84868/84868 [==============================] - 14s 161us/step - loss: 0.1073 -
val_loss: 0.1023
Epoch 9/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.1036 -
val_loss: 0.1002
Epoch 10/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0996 -
val_loss: 0.1003
Epoch 11/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0966 -
val_loss: 0.0997
Epoch 12/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0939 -
val_loss: 0.0886
Epoch 13/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0922 -
val_loss: 0.1055
Epoch 14/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0898 -
val_loss: 0.1087
Epoch 15/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0876 -
```

```
val_loss: 0.0902
Epoch 16/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0855 -
val_loss: 0.0782
Epoch 17/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0835 -
val_loss: 0.0817
Epoch 18/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0827 -
val_loss: 0.0946
Epoch 19/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0800 -
val_loss: 0.0773
Epoch 20/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0771 -
val_loss: 0.0736
Epoch 21/100
84868/84868 [==============================] - 14s 160us/step - loss: 0.0765 -
val_loss: 0.0659
Epoch 22/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0744 -
val_loss: 0.0796
Epoch 23/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0725 -
val_loss: 0.0792
Epoch 24/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0721 -
val_loss: 0.0812
Epoch 25/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0714 -
val_loss: 0.0808
Epoch 26/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0700 -
val_loss: 0.0646
Epoch 27/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0690 -
val_loss: 0.0779
Epoch 28/100
84868/84868 [==============================] - 13s 159us/step - loss: 0.0684 -
val_loss: 0.0831
Epoch 29/100
84868/84868 [==============================] - 13s 159us/step - loss: 0.0672 -
val_loss: 0.0751
Epoch 30/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0667 -
val_loss: 0.0697
Epoch 31/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0654 -
```

```
val_loss: 0.0661
Epoch 32/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0649 -
val_loss: 0.0639
Epoch 33/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0644 -
val_loss: 0.0642
Epoch 34/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0643 -
val_loss: 0.0702
Epoch 35/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0633 -
val_loss: 0.0709
Epoch 36/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0628 -
val_loss: 0.0588
Epoch 37/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0628 -
val_loss: 0.0676
Epoch 38/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0623 -
val_loss: 0.0728
Epoch 39/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0615 -
val_loss: 0.0593
Epoch 40/100
84868/84868 [==============================] - 13s 159us/step - loss: 0.0611 -
val_loss: 0.0724
Epoch 41/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0606 -
val_loss: 0.0611
Epoch 42/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0605 -
val_loss: 0.0625
Epoch 43/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0601 -
val_loss: 0.0649
Epoch 44/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0594 -
val_loss: 0.0656
Epoch 45/100
84868/84868 [==============================] - 13s 159us/step - loss: 0.0591 -
val_loss: 0.0631
Epoch 46/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0590 -
val_loss: 0.0813
Epoch 47/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0582 -
```

```
val_loss: 0.0578
Epoch 48/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0587 -
val_loss: 0.0585
Epoch 49/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0576 -
val_loss: 0.0640
Epoch 50/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0577 -
val_loss: 0.0507
Epoch 51/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0566 -
val_loss: 0.0613
Epoch 52/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0563 -
val_loss: 0.0635
Epoch 53/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0560 -
val_loss: 0.0559
Epoch 54/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0558 -
val_loss: 0.0642
Epoch 55/100
84868/84868 [==============================] - 13s 159us/step - loss: 0.0558 -
val_loss: 0.0741
Epoch 56/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0559 -
val_loss: 0.0534
Epoch 57/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0546 -
val_loss: 0.0576
Epoch 58/100
84868/84868 [==============================] - 14s 159us/step - loss: 0.0546 -
val_loss: 0.0497
Epoch 59/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0538 -
val_loss: 0.0745
Epoch 60/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0531 -
val_loss: 0.0637
Epoch 61/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0535 -
val_loss: 0.0595
Epoch 62/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0536 -
val_loss: 0.0652
Epoch 63/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0529 -
```

```
val_loss: 0.0640
Epoch 64/100
84868/84868 [==============================] - 14s 161us/step - loss: 0.0519 -
val_loss: 0.0726
Epoch 65/100
84868/84868 [==============================] - 14s 165us/step - loss: 0.0522 -
val_loss: 0.0652
Epoch 66/100
84868/84868 [==============================] - 14s 161us/step - loss: 0.0513 -
val_loss: 0.0529
Epoch 67/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0524 -
val_loss: 0.0535
Epoch 68/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0522 -
val_loss: 0.0643
Epoch 69/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0510 -
val_loss: 0.0537
Epoch 70/100
84868/84868 [==============================] - 13s 153us/step - loss: 0.0514 -
val_loss: 0.0612
Epoch 71/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0505 -
val_loss: 0.0484
Epoch 72/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0501 -
val_loss: 0.0681
Epoch 73/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0509 -
val_loss: 0.0615
Epoch 74/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0500 -
val_loss: 0.0517
Epoch 75/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0504 -
val_loss: 0.0584
Epoch 76/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0495 -
val_loss: 0.0595
Epoch 77/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0494 -
val_loss: 0.0542
Epoch 78/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0497 -
val_loss: 0.0615
Epoch 79/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0488 -
```

```
val_loss: 0.0580
Epoch 80/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0492 -
val_loss: 0.0493
Epoch 81/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0496 -
val_loss: 0.0534
Epoch 82/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0490 -
val_loss: 0.0656
Epoch 83/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0491 -
val_loss: 0.0583
Epoch 84/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0483 -
val_loss: 0.0631
Epoch 85/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0481 -
val_loss: 0.0550
Epoch 86/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0484 -
val_loss: 0.0651
Epoch 87/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0490 -
val_loss: 0.0697
Epoch 88/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0477 -
val_loss: 0.0573
Epoch 89/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0484 -
val_loss: 0.0461
Epoch 90/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0484 -
val_loss: 0.0481
Epoch 91/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0477 -
val_loss: 0.0652
Epoch 92/100
84868/84868 [==============================] - 13s 153us/step - loss: 0.0481 -
val_loss: 0.0509
Epoch 93/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0473 -
val_loss: 0.0579
Epoch 94/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0476 -
val_loss: 0.0501
Epoch 95/100
84868/84868 [==============================] - 14s 160us/step - loss: 0.0473 -
```

```
val_loss: 0.0496
Epoch 96/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0468 -
val_loss: 0.0506
Epoch 97/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0473 -
val_loss: 0.0486
Epoch 98/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0463 -
val_loss: 0.0556
Epoch 99/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0464 -
val_loss: 0.0588
Epoch 100/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0466 -
val_loss: 0.0538
```

[21]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
Train on 84868 samples, validate on 21218 samples
Epoch 1/100
84868/84868 [==============================] - 14s 161us/step - loss: 0.0465 -
val_loss: 0.0531
Epoch 2/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0451 -
val_loss: 0.0405
Epoch 3/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0447 -
val_loss: 0.0584
Epoch 4/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0480 -
val_loss: 0.0530
Epoch 5/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0452 -
val_loss: 0.0437
Epoch 6/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0448 -
val_loss: 0.0480
Epoch 7/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0449 -
val_loss: 0.0460
Epoch 8/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0451 -
```

```
val_loss: 0.0420
Epoch 9/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0451 -
val_loss: 0.0488
Epoch 10/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0448 -
val_loss: 0.0462
Epoch 11/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0445 -
val_loss: 0.0484
Epoch 12/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0451 -
val_loss: 0.0563
Epoch 13/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0447 -
val_loss: 0.0486
Epoch 14/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0443 -
val_loss: 0.0519
Epoch 15/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0433 -
val_loss: 0.0535
Epoch 16/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0434 -
val_loss: 0.0488
Epoch 17/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0431 -
val_loss: 0.0523
Epoch 18/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0428 -
val_loss: 0.0592
Epoch 19/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0430 -
val_loss: 0.0629
Epoch 20/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0429 -
val_loss: 0.0526
Epoch 21/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0425 -
val_loss: 0.0444
Epoch 22/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0423 -
val_loss: 0.0373
Epoch 23/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0429 -
val_loss: 0.0497
Epoch 24/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0430 -
```

```
val_loss: 0.0640
Epoch 25/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0428 -
val_loss: 0.0574
Epoch 26/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0428 -
val_loss: 0.0520
Epoch 27/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0431 -
val_loss: 0.0358
Epoch 28/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0427 -
val_loss: 0.0450
Epoch 29/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0422 -
val_loss: 0.0389
Epoch 30/100
84868/84868 [==============================] - 13s 153us/step - loss: 0.0431 -
val_loss: 0.0413
Epoch 31/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0425 -
val_loss: 0.0488
Epoch 32/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0427 -
val_loss: 0.0399
Epoch 33/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0422 -
val_loss: 0.0580
Epoch 34/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0434 -
val_loss: 0.0483
Epoch 35/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0425 -
val_loss: 0.0593
Epoch 36/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0429 -
val_loss: 0.0506
Epoch 37/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0429 -
val_loss: 0.0532
Epoch 38/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0422 -
val_loss: 0.0454
Epoch 39/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0420 -
val_loss: 0.0576
Epoch 40/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0423 -
```

```
val_loss: 0.0531
Epoch 41/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0422 -
val_loss: 0.0428
Epoch 42/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0420 -
val_loss: 0.0450
Epoch 43/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0417 -
val_loss: 0.0450
Epoch 44/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0412 -
val_loss: 0.0531
Epoch 45/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0410 -
val_loss: 0.0422
Epoch 46/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0413 -
val_loss: 0.0421
Epoch 47/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0409 -
val_loss: 0.0395
Epoch 48/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0412 -
val_loss: 0.0469
Epoch 49/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0417 -
val_loss: 0.0509
Epoch 50/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0410 -
val_loss: 0.0579
Epoch 51/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0413 -
val_loss: 0.0534
Epoch 52/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0413 -
val_loss: 0.0472
Epoch 53/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0410 -
val_loss: 0.0538
Epoch 54/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0409 -
val_loss: 0.0362
Epoch 55/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0406 -
val_loss: 0.0515
Epoch 56/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0410 -
```

```
val_loss: 0.0511
Epoch 57/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0401 -
val_loss: 0.0383
Epoch 58/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0407 -
val_loss: 0.0659
Epoch 59/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0400 -
val_loss: 0.0470
Epoch 60/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0412 -
val_loss: 0.0383
Epoch 61/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0416 -
val_loss: 0.0649
Epoch 62/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0406 -
val_loss: 0.0484
Epoch 63/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0420 -
val_loss: 0.0377
Epoch 64/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0405 -
val_loss: 0.0429
Epoch 65/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0406 -
val_loss: 0.0511
Epoch 66/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0403 -
val_loss: 0.0479
Epoch 67/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0404 -
val_loss: 0.0444
Epoch 68/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0403 -
val_loss: 0.0411
Epoch 69/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0399 -
val_loss: 0.0475
Epoch 70/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0403 -
val_loss: 0.0440
Epoch 71/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0406 -
val_loss: 0.0473
Epoch 72/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0399 -
```

```
val_loss: 0.0428
Epoch 73/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0399 -
val_loss: 0.0507
Epoch 74/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0393 -
val_loss: 0.0471
Epoch 75/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0392 -
val_loss: 0.0472
Epoch 76/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0398 -
val_loss: 0.0513
Epoch 77/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0396 -
val_loss: 0.0388
Epoch 78/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0392 -
val_loss: 0.0487
Epoch 79/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0409 -
val_loss: 0.0430
Epoch 80/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0391 -
val_loss: 0.0558
Epoch 81/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0396 -
val_loss: 0.0401
Epoch 82/100
84868/84868 [==============================] - 13s 153us/step - loss: 0.0396 -
val_loss: 0.0388
Epoch 83/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0393 -
val_loss: 0.0493
Epoch 84/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0387 -
val_loss: 0.0530
Epoch 85/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0389 -
val_loss: 0.0558
Epoch 86/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0401 -
val_loss: 0.0432
Epoch 87/100
84868/84868 [==============================] - 13s 153us/step - loss: 0.0400 -
val_loss: 0.0536
Epoch 88/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0394 -
```

```
val_loss: 0.0429
Epoch 89/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0393 -
val_loss: 0.0500
Epoch 90/100
84868/84868 [==============================] - 13s 153us/step - loss: 0.0397 -
val_loss: 0.0555
Epoch 91/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0391 -
val_loss: 0.0446
Epoch 92/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0400 -
val_loss: 0.0553
Epoch 93/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0407 -
val_loss: 0.0454
Epoch 94/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0397 -
val_loss: 0.0411
Epoch 95/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0397 -
val_loss: 0.0513
Epoch 96/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0400 -
val_loss: 0.0428
Epoch 97/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0403 -
val_loss: 0.0534
Epoch 98/100
84868/84868 [==============================] - 13s 153us/step - loss: 0.0395 -
val_loss: 0.0462
Epoch 99/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0394 -
val_loss: 0.0473
Epoch 100/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0404 -
val_loss: 0.0491
```

[22]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[23]:
```
Y_test =np.round(Y_test, 0)
```

[24]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
```

```
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
↪array2string(prediction))
```

```
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

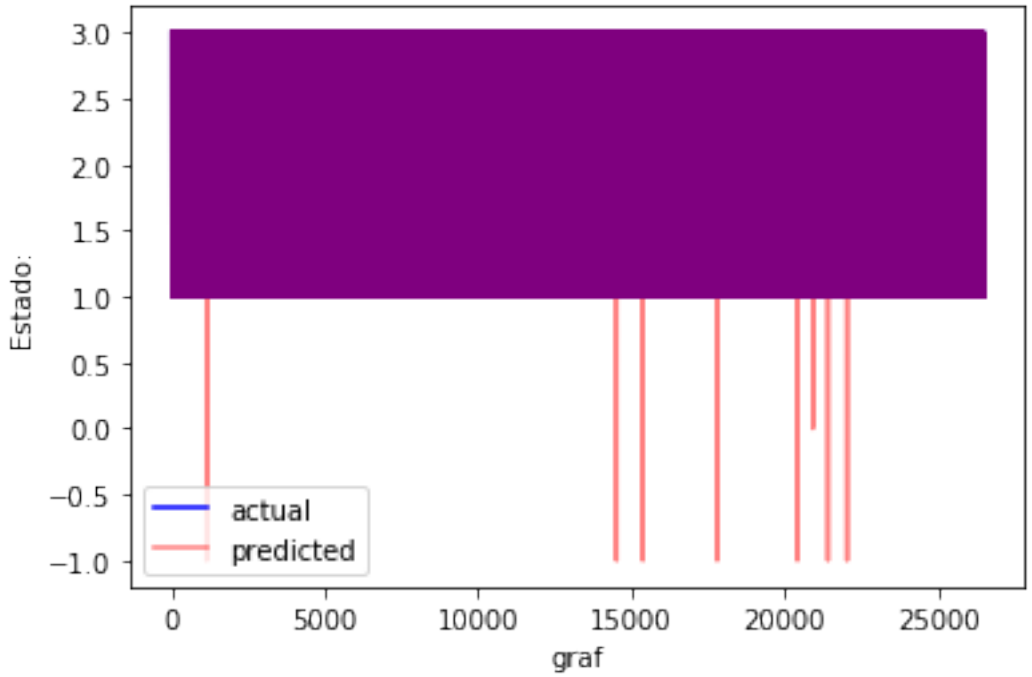
Evaluamos los resultados:

[25]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.9373327182053833
```

[26]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
```

```
plt.legend(loc="best")
plt.show()
```



[27]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[27]:
```
0.9814493627931529
```

[ ]:
```

```

Wrist_elux7_mae_Adadel_final

August 27, 2019

1 Wrist Data

Librerías:

[3]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Wrist

Cargamos los datos:

[4]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
```

```
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>=4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
```

```
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT2=np.concatenate((B1,A2),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>=4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
```

```
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=25*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT3=np.concatenate((B1,A2),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>=4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
```

```
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=35*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT4=np.concatenate((B1,A2),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>=4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
```

```
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT5=np.concatenate((B1,A2),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>=4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT6=np.concatenate((B1,A2),axis=1)
```

```
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>=4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
```

```
B1 = np.asmatrix(E)
B1=B1.T
MAT7=np.concatenate((B1,A2),axis=1)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>=4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
```

```
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT8=np.concatenate((B1,A2),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>=4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
```

```
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT9=np.concatenate((B1,A2),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>=4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
```

```
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT10=np.concatenate((B1,A2),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>=4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
```

```
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT11=np.concatenate((B1,A2),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>=4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
```

```
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT12=np.concatenate((B1,A2),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>=4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
```

```
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT13=np.concatenate((B1,A2),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>=4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[17]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
```

```
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=24*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT14=np.concatenate((B1,A2),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>=4), 0)
MAT14= np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[18]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
```

```
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=29*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT15=np.concatenate((B1,A2),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>=4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[19]:
```
MAT=np.
 ↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Dividimos en los conjuntos de validación y entrenamiento:

[20]:
```
X = MAT[:, 1:8]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)
```

Definimos la arquitectura de la red:

[21]:
```
model = Sequential([
    Dense(128, input_shape=(7,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
```

```
    Dense(60),
    Activation('elu'),
    Dense(40),
    Activation('elu'),
    Dense(40),
    Activation('elu'),
    Dense(30),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

```
WARNING: Logging before flag parsing goes to stderr.
W0827 18:46:22.944923 19300 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:74: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

W0827 18:46:22.959837 19300 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:517: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

W0827 18:46:22.963827 19300 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:4138: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

W0827 18:46:23.061565 19300 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-packages\keras\optimizers.py:790:
The name tf.train.Optimizer is deprecated. Please use
tf.compat.v1.train.Optimizer instead.
```

Entrenamos la red:

[22]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
W0827 18:46:23.283006 19300 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:986: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.
```

```
W0827 18:46:23.288008 19300 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:973: The name tf.assign is
deprecated. Please use tf.compat.v1.assign instead.


Train on 84868 samples, validate on 21218 samples
Epoch 1/100
84868/84868 [==============================] - 17s 205us/step - loss: 0.3384 -
val_loss: 0.2349
Epoch 2/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.1988 -
val_loss: 0.1797
Epoch 3/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.1505 -
val_loss: 0.1360
Epoch 4/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.1323 -
val_loss: 0.1396
Epoch 5/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.1198 -
val_loss: 0.1167
Epoch 6/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.1122 -
val_loss: 0.1205
Epoch 7/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.1045 -
val_loss: 0.1122
Epoch 8/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0988 -
val_loss: 0.1034
Epoch 9/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0939 -
val_loss: 0.1069
Epoch 10/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0897 -
val_loss: 0.0963
Epoch 11/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0862 -
val_loss: 0.0863
Epoch 12/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0850 -
val_loss: 0.0950
Epoch 13/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0816 -
val_loss: 0.0891
Epoch 14/100
```

```
84868/84868 [==============================] - 16s 192us/step - loss: 0.0804 -
val_loss: 0.0757
Epoch 15/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.0787 -
val_loss: 0.0725
Epoch 16/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0756 -
val_loss: 0.0716
Epoch 17/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0736 -
val_loss: 0.0804
Epoch 18/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0724 -
val_loss: 0.0741
Epoch 19/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0714 -
val_loss: 0.0753
Epoch 20/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0703 -
val_loss: 0.0797
Epoch 21/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0704 -
val_loss: 0.0712
Epoch 22/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0693 -
val_loss: 0.0669
Epoch 23/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0683 -
val_loss: 0.0696
Epoch 24/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0668 -
val_loss: 0.0768
Epoch 25/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0680 -
val_loss: 0.0655
Epoch 26/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0656 -
val_loss: 0.0638
Epoch 27/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0661 -
val_loss: 0.0636
Epoch 28/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0652 -
val_loss: 0.0724
Epoch 29/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0653 -
val_loss: 0.0600
Epoch 30/100
```

```
84868/84868 [==============================] - 16s 188us/step - loss: 0.0636 -
val_loss: 0.0605
Epoch 31/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0640 -
val_loss: 0.0679
Epoch 32/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0625 -
val_loss: 0.0728
Epoch 33/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0633 -
val_loss: 0.0610
Epoch 34/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0612 -
val_loss: 0.0616
Epoch 35/100
84868/84868 [==============================] - 16s 194us/step - loss: 0.0605 -
val_loss: 0.0593
Epoch 36/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0608 -
val_loss: 0.0629
Epoch 37/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0605 -
val_loss: 0.0590
Epoch 38/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0599 -
val_loss: 0.0643
Epoch 39/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0592 -
val_loss: 0.0615
Epoch 40/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0602 -
val_loss: 0.0659
Epoch 41/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0597 -
val_loss: 0.0685
Epoch 42/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0586 -
val_loss: 0.0599
Epoch 43/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0582 -
val_loss: 0.0658
Epoch 44/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0595 -
val_loss: 0.0710
Epoch 45/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0590 -
val_loss: 0.0645
Epoch 46/100
```

```
84868/84868 [==============================] - 16s 190us/step - loss: 0.0580 -
val_loss: 0.0640
Epoch 47/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0589 -
val_loss: 0.0655
Epoch 48/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0582 -
val_loss: 0.0615
Epoch 49/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0585 -
val_loss: 0.0686
Epoch 50/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0580 -
val_loss: 0.0631
Epoch 51/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0572 -
val_loss: 0.0620
Epoch 52/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0582 -
val_loss: 0.0568
Epoch 53/100
84868/84868 [==============================] - 16s 194us/step - loss: 0.0573 -
val_loss: 0.0615
Epoch 54/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0574 -
val_loss: 0.0561
Epoch 55/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0576 -
val_loss: 0.0634
Epoch 56/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0565 -
val_loss: 0.0649
Epoch 57/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0558 -
val_loss: 0.0662
Epoch 58/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0550 -
val_loss: 0.0590
Epoch 59/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0549 -
val_loss: 0.0649
Epoch 60/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0568 -
val_loss: 0.0536
Epoch 61/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0561 -
val_loss: 0.0613
Epoch 62/100
```

```
84868/84868 [==============================] - 16s 191us/step - loss: 0.0566 -
val_loss: 0.0565
Epoch 63/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0561 -
val_loss: 0.0595
Epoch 64/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0562 -
val_loss: 0.0576
Epoch 65/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0553 -
val_loss: 0.0563
Epoch 66/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0561 -
val_loss: 0.0552
Epoch 67/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0554 -
val_loss: 0.0508
Epoch 68/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0557 -
val_loss: 0.0627
Epoch 69/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0549 -
val_loss: 0.0542
Epoch 70/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0563 -
val_loss: 0.0679
Epoch 71/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0556 -
val_loss: 0.0614
Epoch 72/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0563 -
val_loss: 0.0521
Epoch 73/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0556 -
val_loss: 0.1056
Epoch 74/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0577 -
val_loss: 0.0635
Epoch 75/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0563 -
val_loss: 0.0671
Epoch 76/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0545 -
val_loss: 0.0580
Epoch 77/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0538 -
val_loss: 0.0541
Epoch 78/100
```

```
84868/84868 [==============================] - 16s 191us/step - loss: 0.0512 -
val_loss: 0.0527
Epoch 79/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0512 -
val_loss: 0.0436
Epoch 80/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0510 -
val_loss: 0.0500
Epoch 81/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0516 -
val_loss: 0.0541
Epoch 82/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0489 -
val_loss: 0.0673
Epoch 83/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0503 -
val_loss: 0.0507
Epoch 84/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0500 -
val_loss: 0.0636
Epoch 85/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0495 -
val_loss: 0.0545
Epoch 86/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0499 -
val_loss: 0.0616
Epoch 87/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0510 -
val_loss: 0.0528
Epoch 88/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0493 -
val_loss: 0.0500
Epoch 89/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0484 -
val_loss: 0.0491
Epoch 90/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0493 -
val_loss: 0.0526
Epoch 91/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0499 -
val_loss: 0.0692
Epoch 92/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0492 -
val_loss: 0.0523
Epoch 93/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0492 -
val_loss: 0.0488
Epoch 94/100
```

```
84868/84868 [==============================] - 16s 191us/step - loss: 0.0483 -
val_loss: 0.0483
Epoch 95/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0475 -
val_loss: 0.0427
Epoch 96/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0478 -
val_loss: 0.0596
Epoch 97/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0479 -
val_loss: 0.0505
Epoch 98/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0482 -
val_loss: 0.0582
Epoch 99/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0478 -
val_loss: 0.0441
Epoch 100/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0485 -
val_loss: 0.0578
```

[23]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
Train on 84868 samples, validate on 21218 samples
Epoch 1/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0481 -
val_loss: 0.0599
Epoch 2/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0493 -
val_loss: 0.0586
Epoch 3/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0477 -
val_loss: 0.0480
Epoch 4/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0474 -
val_loss: 0.0558
Epoch 5/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0475 -
val_loss: 0.0562
Epoch 6/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0483 -
val_loss: 0.0565
Epoch 7/100
```

```
84868/84868 [==============================] - 16s 188us/step - loss: 0.0481 -
val_loss: 0.0536
Epoch 8/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0476 -
val_loss: 0.0625
Epoch 9/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0491 -
val_loss: 0.0678
Epoch 10/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0484 -
val_loss: 0.0596
Epoch 11/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0496 -
val_loss: 0.0644
Epoch 12/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0488 -
val_loss: 0.0550
Epoch 13/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0472 -
val_loss: 0.0484
Epoch 14/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0494 -
val_loss: 0.0581
Epoch 15/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0492 -
val_loss: 0.0510
Epoch 16/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0473 -
val_loss: 0.0448
Epoch 17/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0483 -
val_loss: 0.0489
Epoch 18/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0497 -
val_loss: 0.0528
Epoch 19/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0483 -
val_loss: 0.0541
Epoch 20/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0477 -
val_loss: 0.0540
Epoch 21/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0483 -
val_loss: 0.0632
Epoch 22/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0481 -
val_loss: 0.0409
Epoch 23/100
```

```
84868/84868 [==============================] - 16s 191us/step - loss: 0.0497 -
val_loss: 0.0503
Epoch 24/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0505 -
val_loss: 0.0578
Epoch 25/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0497 -
val_loss: 0.0548
Epoch 26/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0499 -
val_loss: 0.0572
Epoch 27/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0493 -
val_loss: 0.0439
Epoch 28/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0524 -
val_loss: 0.0554
Epoch 29/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0506 -
val_loss: 0.0534
Epoch 30/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0491 -
val_loss: 0.0531
Epoch 31/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0510 -
val_loss: 0.0715
Epoch 32/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0497 -
val_loss: 0.0530
Epoch 33/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0508 -
val_loss: 0.0473
Epoch 34/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0487 -
val_loss: 0.0479
Epoch 35/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0490 -
val_loss: 0.0577
Epoch 36/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0513 -
val_loss: 0.0463
Epoch 37/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0537 -
val_loss: 0.0933
Epoch 38/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0511 -
val_loss: 0.0534
Epoch 39/100
```

```
84868/84868 [==============================] - 16s 191us/step - loss: 0.0522 -
val_loss: 0.0546
Epoch 40/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0520 -
val_loss: 0.0700
Epoch 41/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0526 -
val_loss: 0.0526
Epoch 42/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0548 -
val_loss: 0.0583
Epoch 43/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0537 -
val_loss: 0.0524
Epoch 44/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0520 -
val_loss: 0.0489
Epoch 45/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0507 -
val_loss: 0.0568
Epoch 46/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0520 -
val_loss: 0.0437
Epoch 47/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0524 -
val_loss: 0.0591
Epoch 48/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0524 -
val_loss: 0.0601
Epoch 49/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0508 -
val_loss: 0.0518
Epoch 50/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0523 -
val_loss: 0.0510
Epoch 51/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0533 -
val_loss: 0.0543
Epoch 52/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0498 -
val_loss: 0.0472
Epoch 53/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0522 -
val_loss: 0.0433
Epoch 54/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0536 -
val_loss: 0.0693
Epoch 55/100
```

```
84868/84868 [==============================] - 16s 190us/step - loss: 0.0528 -
val_loss: 0.0492
Epoch 56/100
84868/84868 [==============================] - 24s 287us/step - loss: 0.0519 -
val_loss: 0.0596
Epoch 57/100
84868/84868 [==============================] - 34s 402us/step - loss: 0.0517 -
val_loss: 0.0600
Epoch 58/100
84868/84868 [==============================] - 30s 358us/step - loss: 0.0535 -
val_loss: 0.0621
Epoch 59/100
84868/84868 [==============================] - 42s 496us/step - loss: 0.0546 -
val_loss: 0.0620
Epoch 60/100
84868/84868 [==============================] - 40s 474us/step - loss: 0.0512 -
val_loss: 0.0466
Epoch 61/100
84868/84868 [==============================] - 29s 343us/step - loss: 0.0508 -
val_loss: 0.0571
Epoch 62/100
84868/84868 [==============================] - 44s 518us/step - loss: 0.0529 -
val_loss: 0.0527
Epoch 63/100
84868/84868 [==============================] - 32s 379us/step - loss: 0.0501 -
val_loss: 0.0608
Epoch 64/100
84868/84868 [==============================] - 31s 363us/step - loss: 0.0508 -
val_loss: 0.0563
Epoch 65/100
84868/84868 [==============================] - 31s 369us/step - loss: 0.0503 -
val_loss: 0.0625
Epoch 66/100
84868/84868 [==============================] - 34s 403us/step - loss: 0.0502 -
val_loss: 0.0538
Epoch 67/100
84868/84868 [==============================] - 39s 460us/step - loss: 0.0514 -
val_loss: 0.0634
Epoch 68/100
84868/84868 [==============================] - 30s 356us/step - loss: 0.0510 -
val_loss: 0.0536
Epoch 69/100
84868/84868 [==============================] - 39s 465us/step - loss: 0.0505 -
val_loss: 0.0755
Epoch 70/100
84868/84868 [==============================] - 32s 377us/step - loss: 0.0471 -
val_loss: 0.0560
Epoch 71/100
```

```
84868/84868 [==============================] - 33s 386us/step - loss: 0.0476 -
val_loss: 0.0507
Epoch 72/100
84868/84868 [==============================] - 17s 202us/step - loss: 0.0448 -
val_loss: 0.0472
Epoch 73/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0463 -
val_loss: 0.0570
Epoch 74/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0442 -
val_loss: 0.0505
Epoch 75/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0461 -
val_loss: 0.0713
Epoch 76/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0457 -
val_loss: 0.0408
Epoch 77/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0444 -
val_loss: 0.0476
Epoch 78/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0441 -
val_loss: 0.0461
Epoch 79/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0467 -
val_loss: 0.0632
Epoch 80/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0455 -
val_loss: 0.0375
Epoch 81/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0433 -
val_loss: 0.0460
Epoch 82/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0439 -
val_loss: 0.0594
Epoch 83/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0460 -
val_loss: 0.0531
Epoch 84/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0453 -
val_loss: 0.0467
Epoch 85/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0444 -
val_loss: 0.0442
Epoch 86/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0452 -
val_loss: 0.0477
Epoch 87/100
```

```
84868/84868 [==============================] - 16s 190us/step - loss: 0.0452 -
val_loss: 0.0546
Epoch 88/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0453 -
val_loss: 0.0446
Epoch 89/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0435 -
val_loss: 0.0430
Epoch 90/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0441 -
val_loss: 0.0406
Epoch 91/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0429 -
val_loss: 0.0508
Epoch 92/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0439 -
val_loss: 0.0425
Epoch 93/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0440 -
val_loss: 0.0435
Epoch 94/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0454 -
val_loss: 0.0429
Epoch 95/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0447 -
val_loss: 0.0835
Epoch 96/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0464 -
val_loss: 0.0504
Epoch 97/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0442 -
val_loss: 0.0468
Epoch 98/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0432 -
val_loss: 0.0455
Epoch 99/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0438 -
val_loss: 0.0624
Epoch 100/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0418 -
val_loss: 0.0512
```

[24]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[25]:
```
Y_test =np.round(Y_test, 0)
```

[26]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

```
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

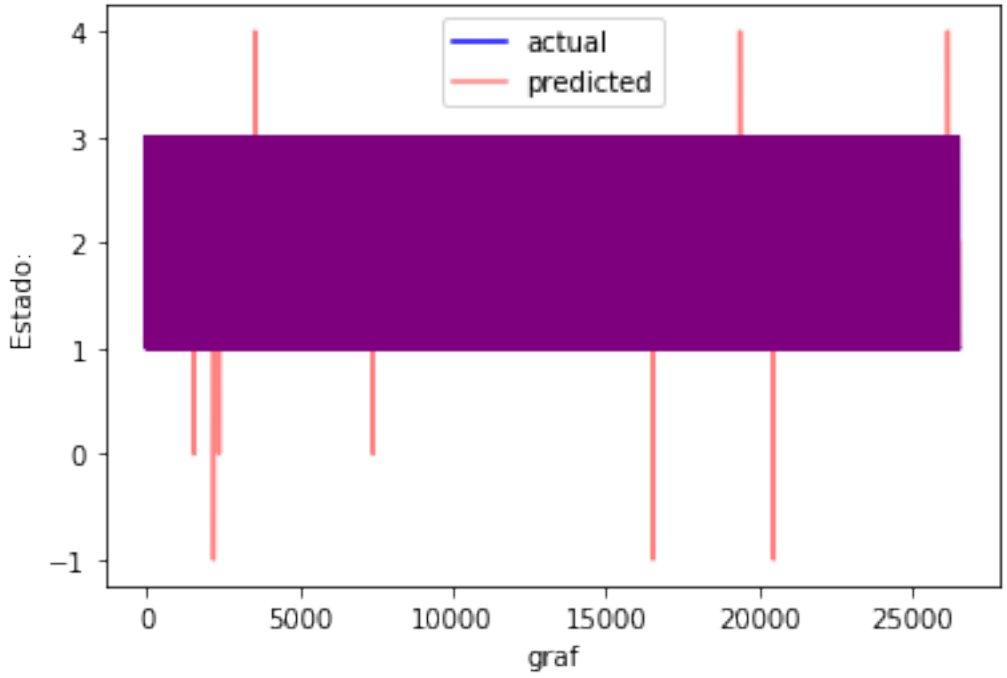
Evaluamos los resultados:

[27]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.9039227793796867
```

[28]:
```
plt.plot((Y_val),
    color="b", label="actual")
```

```
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[29]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[29]: 0.9731543624161074

[ ]:

Wrist_elux4_mae_Adadel_final

August 28, 2019

1 Wrist Data

Librerías:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Wrist

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
```

```
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>=4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
```

```
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT2=np.concatenate((B1,A2),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>=4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
```

```
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=25*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT3=np.concatenate((B1,A2),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>=4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
```

```
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=35*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT4=np.concatenate((B1,A2),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>=4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
```

```
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT5=np.concatenate((B1,A2),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>=4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT6=np.concatenate((B1,A2),axis=1)
```

```
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>=4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
```

```
B1 = np.asmatrix(E)
B1=B1.T
MAT7=np.concatenate((B1,A2),axis=1)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>=4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
```

```
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT8=np.concatenate((B1,A2),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>=4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
```

```
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT9=np.concatenate((B1,A2),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>=4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
```

```
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT10=np.concatenate((B1,A2),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>=4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
```

```
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT11=np.concatenate((B1,A2),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>=4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
```

```
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT12=np.concatenate((B1,A2),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>=4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
```

```
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT13=np.concatenate((B1,A2),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>=4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
```

```
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=24*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT14=np.concatenate((B1,A2),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>=4), 0)
MAT14= np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
```

```
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=29*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT15=np.concatenate((B1,A2),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>=4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
 ↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Dividimos en los conjuntos de validación y entrenamiento:

[18]:
```
X = MAT[:, 1:10]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)
```

Definimos la arquitectura de la red:

[19]:
```
model = Sequential([
    Dense(128, input_shape=(7,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
```

```
    Dense(60),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

```
WARNING: Logging before flag parsing goes to stderr.
W0828 16:31:07.587659 16372 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:74: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

W0828 16:31:07.622563 16372 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:517: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

W0828 16:31:07.634500 16372 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:4138: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

W0828 16:31:07.699326 16372 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-packages\keras\optimizers.py:790:
The name tf.train.Optimizer is deprecated. Please use
tf.compat.v1.train.Optimizer instead.
```

Entrenamos la red:

[20]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
W0828 16:31:07.902820 16372 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:986: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.

W0828 16:31:07.908801 16372 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:973: The name tf.assign is
deprecated. Please use tf.compat.v1.assign instead.
```

```
Train on 84868 samples, validate on 21218 samples
Epoch 1/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.4390 -
val_loss: 0.3262
Epoch 2/100
84868/84868 [==============================] - 12s 142us/step - loss: 0.2624 -
val_loss: 0.2508
Epoch 3/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.2222 -
val_loss: 0.2135
Epoch 4/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.2000 -
val_loss: 0.1893
Epoch 5/100
84868/84868 [==============================] - 12s 141us/step - loss: 0.1845 -
val_loss: 0.1879
Epoch 6/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.1719 -
val_loss: 0.1836
Epoch 7/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.1588 -
val_loss: 0.1509
Epoch 8/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.1419 -
val_loss: 0.1455
Epoch 9/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.1325 -
val_loss: 0.1495
Epoch 10/100
84868/84868 [==============================] - 12s 143us/step - loss: 0.1269 -
val_loss: 0.1265
Epoch 11/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.1224 -
val_loss: 0.1430
Epoch 12/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.1172 -
val_loss: 0.1209
Epoch 13/100
84868/84868 [==============================] - 12s 141us/step - loss: 0.1143 -
val_loss: 0.1333
Epoch 14/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.1112 -
val_loss: 0.1083
Epoch 15/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.1079 -
val_loss: 0.1116
Epoch 16/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.1058 -
```

```
val_loss: 0.1085
Epoch 17/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.1040 -
val_loss: 0.0986
Epoch 18/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.1023 -
val_loss: 0.1022
Epoch 19/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.1009 -
val_loss: 0.1193
Epoch 20/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0990 -
val_loss: 0.1074
Epoch 21/100
84868/84868 [==============================] - 12s 141us/step - loss: 0.0971 -
val_loss: 0.1192
Epoch 22/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0960 -
val_loss: 0.0973
Epoch 23/100
84868/84868 [==============================] - 12s 142us/step - loss: 0.0951 -
val_loss: 0.1113
Epoch 24/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0931 -
val_loss: 0.1159
Epoch 25/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0921 -
val_loss: 0.0888
Epoch 26/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0910 -
val_loss: 0.1148
Epoch 27/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0905 -
val_loss: 0.0897
Epoch 28/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0893 -
val_loss: 0.1111
Epoch 29/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0885 -
val_loss: 0.0898
Epoch 30/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0883 -
val_loss: 0.1090
Epoch 31/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0872 -
val_loss: 0.0922
Epoch 32/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0868 -
```

```
val_loss: 0.1000
Epoch 33/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0858 -
val_loss: 0.1038
Epoch 34/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0852 -
val_loss: 0.0820
Epoch 35/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0851 -
val_loss: 0.0808
Epoch 36/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0840 -
val_loss: 0.0962
Epoch 37/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0836 -
val_loss: 0.0781
Epoch 38/100
84868/84868 [==============================] - 12s 142us/step - loss: 0.0834 -
val_loss: 0.0941
Epoch 39/100
84868/84868 [==============================] - 12s 141us/step - loss: 0.0833 -
val_loss: 0.0906
Epoch 40/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0822 -
val_loss: 0.0842
Epoch 41/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0813 -
val_loss: 0.0873
Epoch 42/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0812 -
val_loss: 0.0855
Epoch 43/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0805 -
val_loss: 0.0809
Epoch 44/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0803 -
val_loss: 0.0915
Epoch 45/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0801 -
val_loss: 0.0798
Epoch 46/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0795 -
val_loss: 0.0854
Epoch 47/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0793 -
val_loss: 0.0939
Epoch 48/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0787 -
```

```
val_loss: 0.0887
Epoch 49/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0783 -
val_loss: 0.0777
Epoch 50/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0783 -
val_loss: 0.0854
Epoch 51/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0777 -
val_loss: 0.0967
Epoch 52/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0772 -
val_loss: 0.0861
Epoch 53/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0769 -
val_loss: 0.1039
Epoch 54/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0769 -
val_loss: 0.0904
Epoch 55/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0773 -
val_loss: 0.0856
Epoch 56/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0770 -
val_loss: 0.0883
Epoch 57/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0761 -
val_loss: 0.1035
Epoch 58/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0761 -
val_loss: 0.0813
Epoch 59/100
84868/84868 [==============================] - 12s 143us/step - loss: 0.0762 -
val_loss: 0.0983
Epoch 60/100
84868/84868 [==============================] - 12s 146us/step - loss: 0.0761 -
val_loss: 0.0851
Epoch 61/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0757 -
val_loss: 0.0937
Epoch 62/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0750 -
val_loss: 0.0761
Epoch 63/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0758 -
val_loss: 0.0802
Epoch 64/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0749 -
```

```
val_loss: 0.0980
Epoch 65/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0743 -
val_loss: 0.0867
Epoch 66/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0743 -
val_loss: 0.0802
Epoch 67/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0738 -
val_loss: 0.0786
Epoch 68/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0739 -
val_loss: 0.1063
Epoch 69/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0736 -
val_loss: 0.0824
Epoch 70/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0735 -
val_loss: 0.0764
Epoch 71/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0735 -
val_loss: 0.0843
Epoch 72/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0735 -
val_loss: 0.0696
Epoch 73/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0732 -
val_loss: 0.0900
Epoch 74/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0731 -
val_loss: 0.0829
Epoch 75/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0727 -
val_loss: 0.0951
Epoch 76/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0727 -
val_loss: 0.0766
Epoch 77/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0726 -
val_loss: 0.0829
Epoch 78/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0726 -
val_loss: 0.0616
Epoch 79/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0717 -
val_loss: 0.0963
Epoch 80/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0710 -
```

```
val_loss: 0.0774
Epoch 81/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0716 -
val_loss: 0.0634
Epoch 82/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0714 -
val_loss: 0.0741
Epoch 83/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0708 -
val_loss: 0.0842
Epoch 84/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0705 -
val_loss: 0.0723
Epoch 85/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0700 -
val_loss: 0.0869
Epoch 86/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0704 -
val_loss: 0.0961
Epoch 87/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0702 -
val_loss: 0.0714
Epoch 88/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0699 -
val_loss: 0.0868
Epoch 89/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0702 -
val_loss: 0.0852
Epoch 90/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0696 -
val_loss: 0.0660
Epoch 91/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0692 -
val_loss: 0.0859
Epoch 92/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0694 -
val_loss: 0.0837
Epoch 93/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0693 -
val_loss: 0.0835
Epoch 94/100
84868/84868 [==============================] - 12s 137us/step - loss: 0.0691 -
val_loss: 0.0707
Epoch 95/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0689 -
val_loss: 0.0701
Epoch 96/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0689 -
```

```
val_loss: 0.0705
Epoch 97/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0683 -
val_loss: 0.0765
Epoch 98/100
84868/84868 [==============================] - 12s 140us/step - loss: 0.0685 -
val_loss: 0.0791
Epoch 99/100
84868/84868 [==============================] - 12s 139us/step - loss: 0.0682 -
val_loss: 0.0643
Epoch 100/100
84868/84868 [==============================] - 12s 138us/step - loss: 0.0678 -
val_loss: 0.0686
```

[21]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[22]:
```
Y_test =np.round(Y_test, 0)
```

[23]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

```
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
```

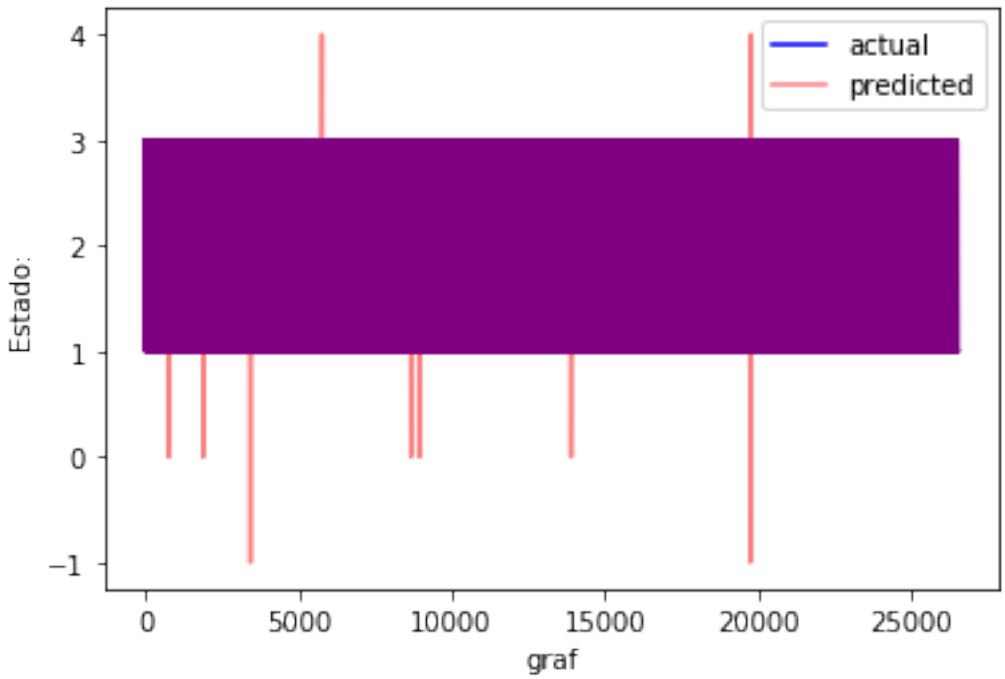
```
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

Evaluamos los resultados:

[24]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.8910521232193426
```

[25]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[26]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[26]:
```
0.965538043888093
```

[ ]:
```

```

Wrist_elux5_mae_Adadel_final

August 28, 2019

1 Wrist Data

Librerías:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Wrist

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
```

```
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>=4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
```

```
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT2=np.concatenate((B1,A2),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>=4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
```

```
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=25*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT3=np.concatenate((B1,A2),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>=4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
```

```
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=35*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT4=np.concatenate((B1,A2),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>=4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
```

```
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT5=np.concatenate((B1,A2),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>=4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT6=np.concatenate((B1,A2),axis=1)
```

```
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>=4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
```

```
B1 = np.asmatrix(E)
B1=B1.T
MAT7=np.concatenate((B1,A2),axis=1)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>=4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
```

```
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT8=np.concatenate((B1,A2),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>=4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
```

```
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT9=np.concatenate((B1,A2),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>=4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
```

```
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT10=np.concatenate((B1,A2),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>=4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
```

```
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT11=np.concatenate((B1,A2),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>=4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
```

```
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT12=np.concatenate((B1,A2),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>=4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
```

```
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT13=np.concatenate((B1,A2),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>=4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
```

```
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=24*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT14=np.concatenate((B1,A2),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>=4), 0)
MAT14= np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
```

```
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=29*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT15=np.concatenate((B1,A2),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>=4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
 ↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Dividimos en los conjuntos de validación y entrenamiento:

[18]:
```
X = MAT[:, 1:8]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)
```

Definimos la arquitectura de la red:

[19]:
```
model = Sequential([
    Dense(128, input_shape=(7,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
```

```
    Dense(48),
    Activation('elu'),
    Dense(24),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

```
WARNING: Logging before flag parsing goes to stderr.
W0828 18:41:15.603651 23756 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:74: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

W0828 18:41:15.617621 23756 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:517: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

W0828 18:41:15.621641 23756 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:4138: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

W0828 18:41:15.691451 23756 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-packages\keras\optimizers.py:790:
The name tf.train.Optimizer is deprecated. Please use
tf.compat.v1.train.Optimizer instead.
```

Entrenamos la red:

[20]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
W0828 18:41:15.885896 23756 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:986: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.

W0828 18:41:15.890924 23756 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:973: The name tf.assign is
```

```
deprecated. Please use tf.compat.v1.assign instead.


Train on 84868 samples, validate on 21218 samples
Epoch 1/100
84868/84868 [==============================] - 15s 171us/step - loss: 0.3657 -
val_loss: 0.2790
Epoch 2/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.2181 -
val_loss: 0.1970
Epoch 3/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.1727 -
val_loss: 0.1643
Epoch 4/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.1439 -
val_loss: 0.1497
Epoch 5/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.1286 -
val_loss: 0.1286
Epoch 6/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.1194 -
val_loss: 0.1196
Epoch 7/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.1114 -
val_loss: 0.1178
Epoch 8/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.1059 -
val_loss: 0.1143
Epoch 9/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.1001 -
val_loss: 0.1088
Epoch 10/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0959 -
val_loss: 0.0914
Epoch 11/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0925 -
val_loss: 0.0882
Epoch 12/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0897 -
val_loss: 0.0932
Epoch 13/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0873 -
val_loss: 0.0862
Epoch 14/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0852 -
val_loss: 0.0839
Epoch 15/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0828 -
```

```
val_loss: 0.0839
Epoch 16/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0821 -
val_loss: 0.0860
Epoch 17/100
84868/84868 [==============================] - 14s 167us/step - loss: 0.0798 -
val_loss: 0.0924
Epoch 18/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0783 -
val_loss: 0.0865
Epoch 19/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0768 -
val_loss: 0.0800
Epoch 20/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0760 -
val_loss: 0.0775
Epoch 21/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0743 -
val_loss: 0.0773
Epoch 22/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0731 -
val_loss: 0.0760
Epoch 23/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0714 -
val_loss: 0.0854
Epoch 24/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0704 -
val_loss: 0.0772
Epoch 25/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0690 -
val_loss: 0.0762
Epoch 26/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0686 -
val_loss: 0.0735
Epoch 27/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0678 -
val_loss: 0.0752
Epoch 28/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0667 -
val_loss: 0.0688
Epoch 29/100
84868/84868 [==============================] - 13s 153us/step - loss: 0.0658 -
val_loss: 0.0779
Epoch 30/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0650 -
val_loss: 0.0677
Epoch 31/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0646 -
```

```
val_loss: 0.0725
Epoch 32/100
84868/84868 [==============================] - 13s 153us/step - loss: 0.0639 -
val_loss: 0.0658
Epoch 33/100
84868/84868 [==============================] - 13s 152us/step - loss: 0.0635 -
val_loss: 0.0760
Epoch 34/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0627 -
val_loss: 0.0721
Epoch 35/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0624 -
val_loss: 0.0688
Epoch 36/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0614 -
val_loss: 0.0606
Epoch 37/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0611 -
val_loss: 0.0850
Epoch 38/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0618 -
val_loss: 0.0620
Epoch 39/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0606 -
val_loss: 0.0825
Epoch 40/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0598 -
val_loss: 0.0715
Epoch 41/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0600 -
val_loss: 0.0701
Epoch 42/100
84868/84868 [==============================] - 13s 153us/step - loss: 0.0598 -
val_loss: 0.0581
Epoch 43/100
84868/84868 [==============================] - 13s 158us/step - loss: 0.0588 -
val_loss: 0.0551
Epoch 44/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0584 -
val_loss: 0.0698
Epoch 45/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0590 -
val_loss: 0.0605
Epoch 46/100
84868/84868 [==============================] - 13s 153us/step - loss: 0.0584 -
val_loss: 0.0601
Epoch 47/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0578 -
```

```
val_loss: 0.0573
Epoch 48/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0575 -
val_loss: 0.0547
Epoch 49/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0568 -
val_loss: 0.0641
Epoch 50/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0561 -
val_loss: 0.0696
Epoch 51/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0560 -
val_loss: 0.0733
Epoch 52/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0563 -
val_loss: 0.0633
Epoch 53/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0560 -
val_loss: 0.0542
Epoch 54/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0553 -
val_loss: 0.0636
Epoch 55/100
84868/84868 [==============================] - 13s 153us/step - loss: 0.0562 -
val_loss: 0.0652
Epoch 56/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0558 -
val_loss: 0.0634
Epoch 57/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0547 -
val_loss: 0.0625
Epoch 58/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0548 -
val_loss: 0.0544
Epoch 59/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0552 -
val_loss: 0.0532
Epoch 60/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0550 -
val_loss: 0.0614
Epoch 61/100
84868/84868 [==============================] - 13s 151us/step - loss: 0.0544 -
val_loss: 0.0514
Epoch 62/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0535 -
val_loss: 0.0516
Epoch 63/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0543 -
```

```
val_loss: 0.0664
Epoch 64/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0527 -
val_loss: 0.0695
Epoch 65/100
84868/84868 [==============================] - 13s 153us/step - loss: 0.0530 -
val_loss: 0.0668
Epoch 66/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0522 -
val_loss: 0.0583
Epoch 67/100
84868/84868 [==============================] - 13s 153us/step - loss: 0.0518 -
val_loss: 0.0621
Epoch 68/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0517 -
val_loss: 0.0658
Epoch 69/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0520 -
val_loss: 0.0508
Epoch 70/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0513 -
val_loss: 0.0638
Epoch 71/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0517 -
val_loss: 0.0570
Epoch 72/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0508 -
val_loss: 0.0639
Epoch 73/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0513 -
val_loss: 0.0600
Epoch 74/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0504 -
val_loss: 0.0532
Epoch 75/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0506 -
val_loss: 0.0498
Epoch 76/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0516 -
val_loss: 0.0564
Epoch 77/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0504 -
val_loss: 0.0607
Epoch 78/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0506 -
val_loss: 0.0506
Epoch 79/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0509 -
```

```
val_loss: 0.0577
Epoch 80/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0504 -
val_loss: 0.0545
Epoch 81/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0513 -
val_loss: 0.0556
Epoch 82/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0510 -
val_loss: 0.0516
Epoch 83/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0504 -
val_loss: 0.0513
Epoch 84/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0506 -
val_loss: 0.0602
Epoch 85/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0509 -
val_loss: 0.0646
Epoch 86/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0500 -
val_loss: 0.0453
Epoch 87/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0508 -
val_loss: 0.0506
Epoch 88/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0504 -
val_loss: 0.0555
Epoch 89/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0498 -
val_loss: 0.0502
Epoch 90/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0497 -
val_loss: 0.0637
Epoch 91/100
84868/84868 [==============================] - 13s 157us/step - loss: 0.0498 -
val_loss: 0.0577
Epoch 92/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0502 -
val_loss: 0.0552
Epoch 93/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0491 -
val_loss: 0.0572
Epoch 94/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0500 -
val_loss: 0.0462
Epoch 95/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0497 -
```

```
val_loss: 0.0641
Epoch 96/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0494 -
val_loss: 0.0451
Epoch 97/100
84868/84868 [==============================] - 13s 153us/step - loss: 0.0502 -
val_loss: 0.0499
Epoch 98/100
84868/84868 [==============================] - 13s 154us/step - loss: 0.0490 -
val_loss: 0.0498
Epoch 99/100
84868/84868 [==============================] - 13s 156us/step - loss: 0.0496 -
val_loss: 0.0556
Epoch 100/100
84868/84868 [==============================] - 13s 155us/step - loss: 0.0488 -
val_loss: 0.0546
```

[21]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[22]:
```
Y_test =np.round(Y_test, 0)
```

[23]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

```
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
```

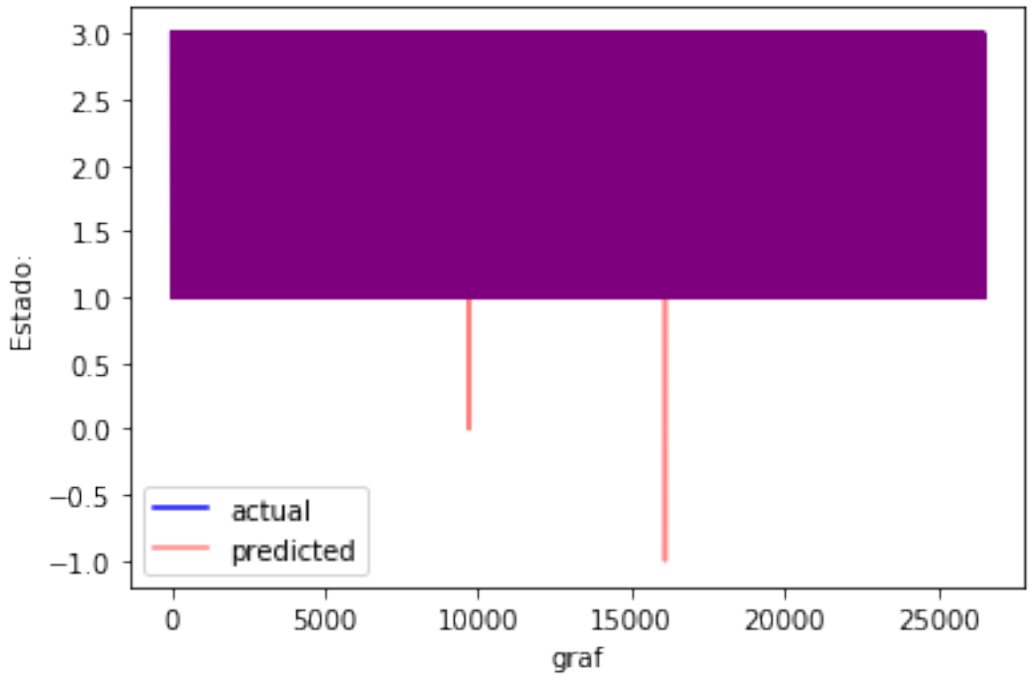
```
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

Evaluamos los resultados:

[24]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.8935274261160827
```

[25]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[26]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[26]:
```
0.9730412487746022
```

[ ]:
```

```

Wrist_elux7_mae_Adadel_final

August 28, 2019

1 Wrist Data

Librerías:

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
Using TensorFlow backend.
```

1.1 Wrist

Cargamos los datos:

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
```

```
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>=4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
```

```
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT2=np.concatenate((B1,A2),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>=4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
```

```
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=25*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT3=np.concatenate((B1,A2),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>=4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
```

```
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=35*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT4=np.concatenate((B1,A2),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>=4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
```

```
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT5=np.concatenate((B1,A2),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>=4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT6=np.concatenate((B1,A2),axis=1)
```

```
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>=4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
```

```
B1 = np.asmatrix(E)
B1=B1.T
MAT7=np.concatenate((B1,A2),axis=1)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>=4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
```

```
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT8=np.concatenate((B1,A2),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>=4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
```

```
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT9=np.concatenate((B1,A2),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>=4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
```

```
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT10=np.concatenate((B1,A2),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>=4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
```

```
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT11=np.concatenate((B1,A2),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>=4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
```

```
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT12=np.concatenate((B1,A2),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>=4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
```

```
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT13=np.concatenate((B1,A2),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>=4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
```

```
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=24*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT14=np.concatenate((B1,A2),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>=4), 0)
MAT14= np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
```

```
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=29*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT15=np.concatenate((B1,A2),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>=4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

Unimos los datos:

[17]:
```
MAT=np.
 ↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

Dividimos en los conjuntos de validación y entrenamiento:

[18]:
```
X = MAT[:, 1:8]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)
```

Definimos la arquitectura de la red:

[19]:
```
model = Sequential([
    Dense(128, input_shape=(7,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
```

```
    Dense(60),
    Activation('elu'),
    Dense(40),
    Activation('elu'),
    Dense(40),
    Activation('elu'),
    Dense(30),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

```
WARNING: Logging before flag parsing goes to stderr.
W0828 19:17:10.691833  3912 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:74: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

W0828 19:17:10.709784  3912 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:517: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

W0828 19:17:10.712776  3912 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:4138: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

W0828 19:17:10.799576  3912 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-packages\keras\optimizers.py:790:
The name tf.train.Optimizer is deprecated. Please use
tf.compat.v1.train.Optimizer instead.
```

Entrenamos la red:

[20]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_train, Y_train, batch_size=BATCH_SIZE, epochs=NUM_EPOCHS,␣
 ↪validation_split=0.2)
```

```
W0828 19:17:11.028969  3912 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:986: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.
```

```
W0828 19:17:11.033917  3912 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:973: The name tf.assign is
deprecated. Please use tf.compat.v1.assign instead.


Train on 84868 samples, validate on 21218 samples
Epoch 1/100
84868/84868 [==============================] - 17s 202us/step - loss: 0.3514 -
val_loss: 0.2646
Epoch 2/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.1969 -
val_loss: 0.1673
Epoch 3/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.1511 -
val_loss: 0.1494
Epoch 4/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.1276 -
val_loss: 0.1175
Epoch 5/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.1134 -
val_loss: 0.1056
Epoch 6/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.1030 -
val_loss: 0.0972
Epoch 7/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0959 -
val_loss: 0.0963
Epoch 8/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0917 -
val_loss: 0.0900
Epoch 9/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0896 -
val_loss: 0.0792
Epoch 10/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0875 -
val_loss: 0.0826
Epoch 11/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0860 -
val_loss: 0.0895
Epoch 12/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0839 -
val_loss: 0.1022
Epoch 13/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0799 -
val_loss: 0.0764
Epoch 14/100
```

```
84868/84868 [==============================] - 16s 189us/step - loss: 0.0799 -
val_loss: 0.0705
Epoch 15/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0784 -
val_loss: 0.0696
Epoch 16/100
84868/84868 [==============================] - 17s 195us/step - loss: 0.0754 -
val_loss: 0.0814
Epoch 17/100
84868/84868 [==============================] - 16s 186us/step - loss: 0.0743 -
val_loss: 0.0927
Epoch 18/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0726 -
val_loss: 0.0675
Epoch 19/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0714 -
val_loss: 0.0684
Epoch 20/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0704 -
val_loss: 0.0641
Epoch 21/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0689 -
val_loss: 0.0605
Epoch 22/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0675 -
val_loss: 0.0719
Epoch 23/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0639 -
val_loss: 0.0773
Epoch 24/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0652 -
val_loss: 0.0973
Epoch 25/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0657 -
val_loss: 0.0676
Epoch 26/100
84868/84868 [==============================] - 16s 189us/step - loss: 0.0624 -
val_loss: 0.0691
Epoch 27/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0589 -
val_loss: 0.0702
Epoch 28/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0580 -
val_loss: 0.0647
Epoch 29/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0577 -
val_loss: 0.0701
Epoch 30/100
```

```
84868/84868 [==============================] - 16s 188us/step - loss: 0.0572 -
val_loss: 0.0565
Epoch 31/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0562 -
val_loss: 0.0594
Epoch 32/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0563 -
val_loss: 0.0503
Epoch 33/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0546 -
val_loss: 0.0581
Epoch 34/100
84868/84868 [==============================] - 16s 188us/step - loss: 0.0542 -
val_loss: 0.0632
Epoch 35/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0545 -
val_loss: 0.0648
Epoch 36/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0550 -
val_loss: 0.0591
Epoch 37/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0543 -
val_loss: 0.0585
Epoch 38/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0518 -
val_loss: 0.0594
Epoch 39/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.0513 -
val_loss: 0.0531
Epoch 40/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0529 -
val_loss: 0.0517
Epoch 41/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0519 -
val_loss: 0.0525
Epoch 42/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0502 -
val_loss: 0.0557
Epoch 43/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0506 -
val_loss: 0.0543
Epoch 44/100
84868/84868 [==============================] - 16s 194us/step - loss: 0.0500 -
val_loss: 0.0473
Epoch 45/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0498 -
val_loss: 0.0440
Epoch 46/100
```

```
84868/84868 [==============================] - 16s 190us/step - loss: 0.0501 -
val_loss: 0.0541
Epoch 47/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.0493 -
val_loss: 0.0513
Epoch 48/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.0490 -
val_loss: 0.0527
Epoch 49/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.0492 -
val_loss: 0.0500
Epoch 50/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0492 -
val_loss: 0.0569
Epoch 51/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0488 -
val_loss: 0.0600
Epoch 52/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.0497 -
val_loss: 0.0756
Epoch 53/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0477 -
val_loss: 0.0521
Epoch 54/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0481 -
val_loss: 0.0457
Epoch 55/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0484 -
val_loss: 0.0615
Epoch 56/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.0480 -
val_loss: 0.0545
Epoch 57/100
84868/84868 [==============================] - 17s 195us/step - loss: 0.0489 -
val_loss: 0.0488
Epoch 58/100
84868/84868 [==============================] - 17s 202us/step - loss: 0.0486 -
val_loss: 0.0475
Epoch 59/100
84868/84868 [==============================] - 17s 199us/step - loss: 0.0476 -
val_loss: 0.0443
Epoch 60/100
84868/84868 [==============================] - 17s 200us/step - loss: 0.0475 -
val_loss: 0.0503
Epoch 61/100
84868/84868 [==============================] - 17s 196us/step - loss: 0.0457 -
val_loss: 0.0479
Epoch 62/100
```

```
84868/84868 [==============================] - 16s 187us/step - loss: 0.0464 -
val_loss: 0.0490
Epoch 63/100
84868/84868 [==============================] - 16s 187us/step - loss: 0.0462 -
val_loss: 0.0535
Epoch 64/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0459 -
val_loss: 0.0511
Epoch 65/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0452 -
val_loss: 0.0795
Epoch 66/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.0465 -
val_loss: 0.0547
Epoch 67/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0468 -
val_loss: 0.0476
Epoch 68/100
84868/84868 [==============================] - 16s 194us/step - loss: 0.0470 -
val_loss: 0.0523
Epoch 69/100
84868/84868 [==============================] - 17s 196us/step - loss: 0.0459 -
val_loss: 0.0473
Epoch 70/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0456 -
val_loss: 0.0479
Epoch 71/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0456 -
val_loss: 0.0440
Epoch 72/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0457 -
val_loss: 0.0505
Epoch 73/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0464 -
val_loss: 0.0516
Epoch 74/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0466 -
val_loss: 0.0475
Epoch 75/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0451 -
val_loss: 0.0494
Epoch 76/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0465 -
val_loss: 0.0486
Epoch 77/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0459 -
val_loss: 0.0523
Epoch 78/100
```

```
84868/84868 [==============================] - 16s 192us/step - loss: 0.0450 -
val_loss: 0.0487
Epoch 79/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0446 -
val_loss: 0.0466
Epoch 80/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0436 -
val_loss: 0.0527
Epoch 81/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0446 -
val_loss: 0.0504
Epoch 82/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0442 -
val_loss: 0.0445
Epoch 83/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0449 -
val_loss: 0.0580
Epoch 84/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0438 -
val_loss: 0.0434
Epoch 85/100
84868/84868 [==============================] - 16s 191us/step - loss: 0.0443 -
val_loss: 0.0633
Epoch 86/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0455 -
val_loss: 0.0488
Epoch 87/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0434 -
val_loss: 0.0540
Epoch 88/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0421 -
val_loss: 0.0458
Epoch 89/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.0438 -
val_loss: 0.0366
Epoch 90/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0433 -
val_loss: 0.0522
Epoch 91/100
84868/84868 [==============================] - 16s 190us/step - loss: 0.0427 -
val_loss: 0.0536
Epoch 92/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0427 -
val_loss: 0.0529
Epoch 93/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.0445 -
val_loss: 0.0529
Epoch 94/100
```

```
84868/84868 [==============================] - 16s 193us/step - loss: 0.0441 -
val_loss: 0.0501
Epoch 95/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.0430 -
val_loss: 0.0436
Epoch 96/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.0432 -
val_loss: 0.0464
Epoch 97/100
84868/84868 [==============================] - 17s 195us/step - loss: 0.0429 -
val_loss: 0.0498
Epoch 98/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.0436 -
val_loss: 0.0437
Epoch 99/100
84868/84868 [==============================] - 16s 192us/step - loss: 0.0424 -
val_loss: 0.0457
Epoch 100/100
84868/84868 [==============================] - 16s 193us/step - loss: 0.0435 -
val_loss: 0.0582
```

[21]:
```
Y_test = model.predict(X_val).flatten()
```

Redondeamos los resultados puesto que deben de ser enteros:

[22]:
```
Y_test =np.round(Y_test, 0)
```

[23]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    prediction =  Y_test[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(prediction))
```

```
Estado: 3., predicted:3.
Estado: 1., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
```

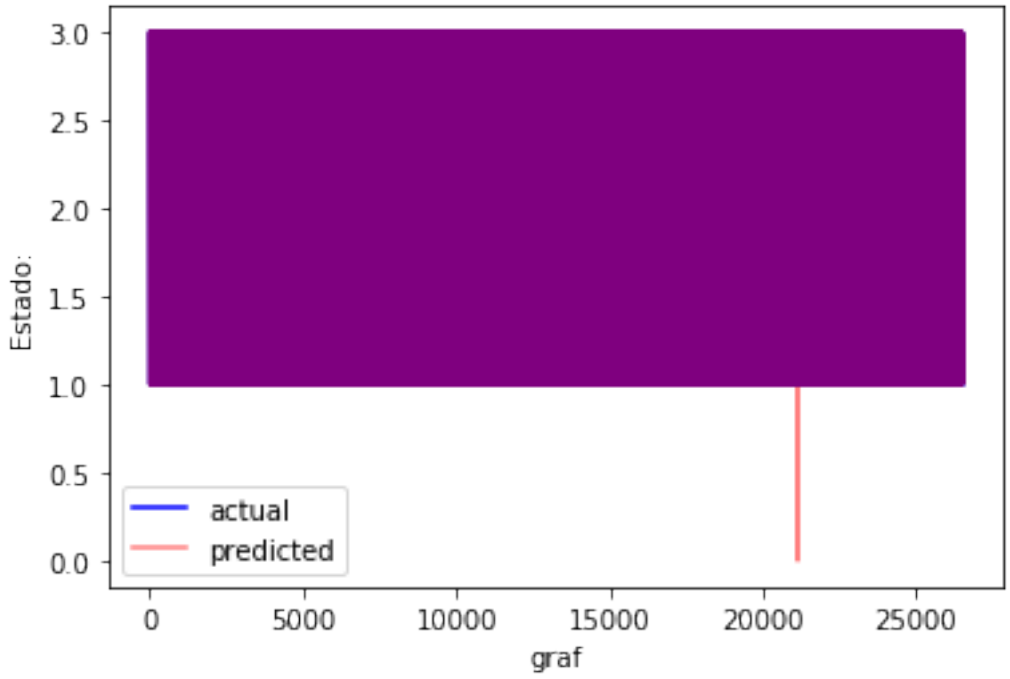
```
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

Evaluamos los resultados:

[24]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test, Y_val)
print (r2)
```

```
0.9113167266916418
```

[25]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_test),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[26]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_test
accur=np.sum(Accu)
accur/len(Y_val)
```

[26]:
```
0.9678003167181962
```

[ ]:
```

```

TFM_wrist_data_ENSEMBLE-final

August 23, 2019

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import sys
import os
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
```

```
Using TensorFlow backend.
```

0.1 Ensemble

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
```

```
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
```

```
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT2=np.concatenate((B1,A2),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
```

```
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=25*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT3=np.concatenate((B1,A2),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
```

```
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=35*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT4=np.concatenate((B1,A2),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
```

```
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT5=np.concatenate((B1,A2),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
```

```
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT6=np.concatenate((B1,A2),axis=1)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT7=np.concatenate((B1,A2),axis=1)
```

```
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
```

```
B1 = np.asmatrix(E)
B1=B1.T
MAT8=np.concatenate((B1,A2),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
```

```
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT9=np.concatenate((B1,A2),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
```

```
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT10=np.concatenate((B1,A2),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
```

```
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT11=np.concatenate((B1,A2),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
```

```
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT12=np.concatenate((B1,A2),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
```

```
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT13=np.concatenate((B1,A2),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
```

```
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=24*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT14=np.concatenate((B1,A2),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>4), 0)
MAT14= np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
```

```
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=29*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT15=np.concatenate((B1,A2),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

[17]:
```
MAT=np.
 ↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

[18]:
```
X = MAT[:, 1:8]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)
```

[19]:
```
X_trainA= np.delete(X_train,3 , 1)
X_trainB= np.delete(X_train,4 , 1)
X_valA= np.delete(X_val,3, 1)
X_valB= np.delete(X_val,4 , 1)
```

[24]:
```
model = Sequential([
    Dense(128, input_shape=(6,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
    Dense(48),
    Activation('elu'),
```

```
    Dense(24),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

[25]:
```
NUM_EPOCHS =100
BATCH_SIZE = 15

history = model.fit(X_trainA, Y_train, batch_size=BATCH_SIZE,␣
 ↪epochs=NUM_EPOCHS, validation_split=0.2)
```

```
Train on 115092 samples, validate on 28773 samples
Epoch 1/100
115092/115092 [==============================] - 25s 220us/step - loss: 0.5107 -
val_loss: 0.4044
Epoch 2/100
115092/115092 [==============================] - 25s 214us/step - loss: 0.2917 -
val_loss: 0.2468
Epoch 3/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.2212 -
val_loss: 0.2127
Epoch 4/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.1850 -
val_loss: 0.1663
Epoch 5/100
115092/115092 [==============================] - 24s 212us/step - loss: 0.1632 -
val_loss: 0.1613
Epoch 6/100
115092/115092 [==============================] - 24s 212us/step - loss: 0.1479 -
val_loss: 0.1500
Epoch 7/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.1359 -
val_loss: 0.1363
Epoch 8/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.1273 -
val_loss: 0.1370
Epoch 9/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.1204 -
val_loss: 0.1197
Epoch 10/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.1142 -
val_loss: 0.1229
Epoch 11/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.1101 -
val_loss: 0.1110
```

```
Epoch 12/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.1065 -
val_loss: 0.1056
Epoch 13/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.1033 -
val_loss: 0.1306
Epoch 14/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0999 -
val_loss: 0.1107
Epoch 15/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0965 -
val_loss: 0.0925
Epoch 16/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0950 -
val_loss: 0.1023
Epoch 17/100
115092/115092 [==============================] - 24s 212us/step - loss: 0.0928 -
val_loss: 0.0929
Epoch 18/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0911 -
val_loss: 0.0914
Epoch 19/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0894 -
val_loss: 0.1122
Epoch 20/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0884 -
val_loss: 0.0868
Epoch 21/100
115092/115092 [==============================] - 24s 212us/step - loss: 0.0869 -
val_loss: 0.0875
Epoch 22/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0847 -
val_loss: 0.0852
Epoch 23/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0832 -
val_loss: 0.0786
Epoch 24/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0830 -
val_loss: 0.0895
Epoch 25/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0817 -
val_loss: 0.0824
Epoch 26/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0798 -
val_loss: 0.0709
Epoch 27/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0792 -
val_loss: 0.0848
```

```
Epoch 28/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0781 -
val_loss: 0.0832
Epoch 29/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0771 -
val_loss: 0.0791
Epoch 30/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0760 -
val_loss: 0.0745
Epoch 31/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0748 -
val_loss: 0.0842
Epoch 32/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0737 -
val_loss: 0.0707
Epoch 33/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0733 -
val_loss: 0.0689
Epoch 34/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0726 -
val_loss: 0.0901
Epoch 35/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0722 -
val_loss: 0.0745
Epoch 36/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0705 -
val_loss: 0.0700
Epoch 37/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0703 -
val_loss: 0.0631
Epoch 38/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0700 -
val_loss: 0.0605
Epoch 39/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0697 -
val_loss: 0.0840
Epoch 40/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0695 -
val_loss: 0.0674
Epoch 41/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0695 -
val_loss: 0.0679
Epoch 42/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0686 -
val_loss: 0.0627
Epoch 43/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0676 -
val_loss: 0.0644
```

```
Epoch 44/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0667 -
val_loss: 0.0839
Epoch 45/100
115092/115092 [==============================] - 24s 212us/step - loss: 0.0665 -
val_loss: 0.0685
Epoch 46/100
115092/115092 [==============================] - 24s 212us/step - loss: 0.0650 -
val_loss: 0.0744
Epoch 47/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0644 -
val_loss: 0.0612
Epoch 48/100
115092/115092 [==============================] - 25s 217us/step - loss: 0.0645 -
val_loss: 0.0954
Epoch 49/100
115092/115092 [==============================] - 24s 212us/step - loss: 0.0643 -
val_loss: 0.0644
Epoch 50/100
115092/115092 [==============================] - 24s 212us/step - loss: 0.0647 -
val_loss: 0.0511
Epoch 51/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0640 -
val_loss: 0.0740
Epoch 52/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0635 -
val_loss: 0.0709
Epoch 53/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0628 -
val_loss: 0.0652
Epoch 54/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0624 -
val_loss: 0.0603
Epoch 55/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0621 -
val_loss: 0.0614
Epoch 56/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0618 -
val_loss: 0.0635
Epoch 57/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0620 -
val_loss: 0.0631
Epoch 58/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0621 -
val_loss: 0.0620
Epoch 59/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0619 -
val_loss: 0.0770
```

```
Epoch 60/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0624 -
val_loss: 0.0670
Epoch 61/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0614 -
val_loss: 0.1005
Epoch 62/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0629 -
val_loss: 0.0597
Epoch 63/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0626 -
val_loss: 0.0655
Epoch 64/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0637 -
val_loss: 0.0592
Epoch 65/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0625 -
val_loss: 0.0639
Epoch 66/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0627 -
val_loss: 0.0570
Epoch 67/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0636 -
val_loss: 0.0631
Epoch 68/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0628 -
val_loss: 0.0654
Epoch 69/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0645 -
val_loss: 0.0598
Epoch 70/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0651 -
val_loss: 0.0674
Epoch 71/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0654 -
val_loss: 0.0630
Epoch 72/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0653 -
val_loss: 0.0617
Epoch 73/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0652 -
val_loss: 0.0560
Epoch 74/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0654 -
val_loss: 0.0580
Epoch 75/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0658 -
val_loss: 0.1430
```

```
Epoch 76/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0655 -
val_loss: 0.0602
Epoch 77/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0664 -
val_loss: 0.0771
Epoch 78/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0679 -
val_loss: 0.0799
Epoch 79/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0694 -
val_loss: 0.0640
Epoch 80/100
115092/115092 [==============================] - 24s 212us/step - loss: 0.0671 -
val_loss: 0.0661
Epoch 81/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0674 -
val_loss: 0.0845
Epoch 82/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0666 -
val_loss: 0.0653
Epoch 83/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0676 -
val_loss: 0.0821
Epoch 84/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0673 -
val_loss: 0.0690
Epoch 85/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0684 -
val_loss: 0.0666
Epoch 86/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0677 -
val_loss: 0.0785
Epoch 87/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0675 -
val_loss: 0.0698
Epoch 88/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0673 -
val_loss: 0.0775
Epoch 89/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0685 -
val_loss: 0.0626
Epoch 90/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0687 -
val_loss: 0.0646
Epoch 91/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0692 -
val_loss: 0.0596
```

```
Epoch 92/100
115092/115092 [==============================] - 24s 212us/step - loss: 0.0686 -
val_loss: 0.0674
Epoch 93/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0670 -
val_loss: 0.0672
Epoch 94/100
115092/115092 [==============================] - 25s 213us/step - loss: 0.0662 -
val_loss: 0.0620
Epoch 95/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0665 -
val_loss: 0.0627
Epoch 96/100
115092/115092 [==============================] - 24s 212us/step - loss: 0.0649 -
val_loss: 0.1057
Epoch 97/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0669 -
val_loss: 0.0692
Epoch 98/100
115092/115092 [==============================] - 24s 211us/step - loss: 0.0669 -
val_loss: 0.0746
Epoch 99/100
115092/115092 [==============================] - 24s 210us/step - loss: 0.0658 -
val_loss: 0.0740
Epoch 100/100
115092/115092 [==============================] - 24s 212us/step - loss: 0.0650 -
val_loss: 0.0608
```

[26]:
```
Y_testA = model.predict(X_valA).flatten()
```

[27]:
```
Y_testA =np.round(Y_testA, 0)
```

[28]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    predictionA =  Y_testA[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(predictionA))
```

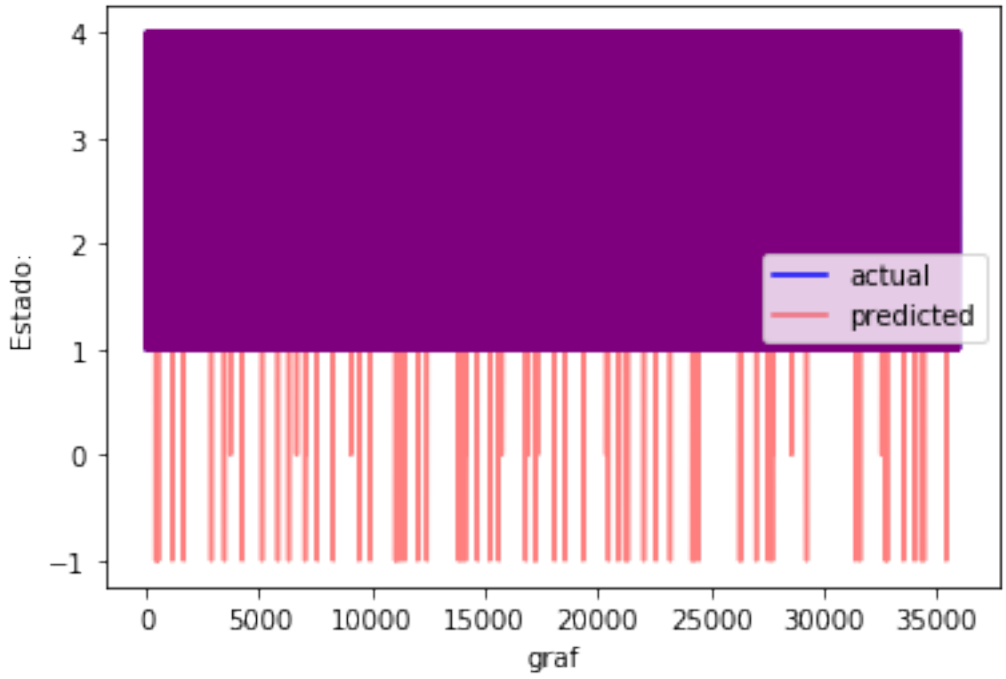
```
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:3.
```

```
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:2.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

[29]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_testA, Y_val)
print (r2)
```

```
0.9455439396680531
```

[30]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_testA),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[31]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_testA
accur=np.sum(Accu)
accur/len(Y_val)
```

[31]:
```
0.9715572608224206
```

[32]:
```
model = Sequential([
    Dense(128, input_shape=(6,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
    Dense(48),
    Activation('elu'),
    Dense(24),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

[33]:
```
NUM_EPOCHS =100
BATCH_SIZE = 20

history = model.fit(X_trainB, Y_train, batch_size=BATCH_SIZE,␣
 ↪epochs=NUM_EPOCHS, validation_split=0.2)
```

```
Train on 115092 samples, validate on 28773 samples
Epoch 1/100
115092/115092 [==============================] - 19s 162us/step - loss: 0.6112 -
val_loss: 0.4906
Epoch 2/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.3958 -
val_loss: 0.3515
Epoch 3/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.3198 -
val_loss: 0.2893
Epoch 4/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.2816 -
val_loss: 0.2707
Epoch 5/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.2546 -
val_loss: 0.2396
Epoch 6/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.2246 -
val_loss: 0.2046
Epoch 7/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.2022 -
val_loss: 0.2077
Epoch 8/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1886 -
val_loss: 0.2034
Epoch 9/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1794 -
val_loss: 0.1701
Epoch 10/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.1715 -
val_loss: 0.1950
Epoch 11/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1636 -
val_loss: 0.1571
Epoch 12/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1586 -
val_loss: 0.1504
Epoch 13/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1547 -
val_loss: 0.1522
Epoch 14/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1501 -
val_loss: 0.1574
Epoch 15/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1474 -
val_loss: 0.1407
Epoch 16/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1443 -
```

```
val_loss: 0.1471
Epoch 17/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1422 -
val_loss: 0.1458
Epoch 18/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1404 -
val_loss: 0.1503
Epoch 19/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1389 -
val_loss: 0.1495
Epoch 20/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1370 -
val_loss: 0.1496
Epoch 21/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.1358 -
val_loss: 0.1370
Epoch 22/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1348 -
val_loss: 0.1290
Epoch 23/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1339 -
val_loss: 0.1368
Epoch 24/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1324 -
val_loss: 0.1331
Epoch 25/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1315 -
val_loss: 0.1372
Epoch 26/100
115092/115092 [==============================] - 19s 161us/step - loss: 0.1313 -
val_loss: 0.1279
Epoch 27/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1305 -
val_loss: 0.1325
Epoch 28/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1290 -
val_loss: 0.1294
Epoch 29/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1287 -
val_loss: 0.1622
Epoch 30/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1269 -
val_loss: 0.1525
Epoch 31/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1262 -
val_loss: 0.1315
Epoch 32/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1260 -
```

```
val_loss: 0.1321
Epoch 33/100
115092/115092 [==============================] - 18s 160us/step - loss: 0.1245 -
val_loss: 0.1277
Epoch 34/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1245 -
val_loss: 0.1249
Epoch 35/100
115092/115092 [==============================] - 18s 160us/step - loss: 0.1234 -
val_loss: 0.1254
Epoch 36/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1221 -
val_loss: 0.1168
Epoch 37/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1207 -
val_loss: 0.1330
Epoch 38/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1208 -
val_loss: 0.1148
Epoch 39/100
115092/115092 [==============================] - 18s 160us/step - loss: 0.1204 -
val_loss: 0.1181
Epoch 40/100
115092/115092 [==============================] - 18s 160us/step - loss: 0.1202 -
val_loss: 0.1238
Epoch 41/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1193 -
val_loss: 0.1340
Epoch 42/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1186 -
val_loss: 0.1160
Epoch 43/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1183 -
val_loss: 0.1274
Epoch 44/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1177 -
val_loss: 0.1163
Epoch 45/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1178 -
val_loss: 0.1248
Epoch 46/100
115092/115092 [==============================] - 18s 160us/step - loss: 0.1171 -
val_loss: 0.1183
Epoch 47/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1163 -
val_loss: 0.1113
Epoch 48/100
115092/115092 [==============================] - 18s 160us/step - loss: 0.1163 -
```

```
val_loss: 0.1228
Epoch 49/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1165 -
val_loss: 0.1167
Epoch 50/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1153 -
val_loss: 0.1135
Epoch 51/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1160 -
val_loss: 0.1195
Epoch 52/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1155 -
val_loss: 0.1205
Epoch 53/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1145 -
val_loss: 0.1411
Epoch 54/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1136 -
val_loss: 0.1123
Epoch 55/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1137 -
val_loss: 0.1254
Epoch 56/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1146 -
val_loss: 0.1128
Epoch 57/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1130 -
val_loss: 0.1189
Epoch 58/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1129 -
val_loss: 0.1286
Epoch 59/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1130 -
val_loss: 0.1164
Epoch 60/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1128 -
val_loss: 0.1244
Epoch 61/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1123 -
val_loss: 0.1102
Epoch 62/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1120 -
val_loss: 0.1175
Epoch 63/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1117 -
val_loss: 0.1084
Epoch 64/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1103 -
```

```
val_loss: 0.1182
Epoch 65/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.1105 -
val_loss: 0.1169
Epoch 66/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1093 -
val_loss: 0.1100
Epoch 67/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.1087 -
val_loss: 0.1110
Epoch 68/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.1087 -
val_loss: 0.1075
Epoch 69/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.1081 -
val_loss: 0.1050
Epoch 70/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1074 -
val_loss: 0.1151
Epoch 71/100
115092/115092 [==============================] - 18s 160us/step - loss: 0.1078 -
val_loss: 0.1182
Epoch 72/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1068 -
val_loss: 0.1131
Epoch 73/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1063 -
val_loss: 0.1046
Epoch 74/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.1056 -
val_loss: 0.1116
Epoch 75/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.1053 -
val_loss: 0.1073
Epoch 76/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.1056 -
val_loss: 0.1120
Epoch 77/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.1056 -
val_loss: 0.1048
Epoch 78/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.1051 -
val_loss: 0.0977
Epoch 79/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1042 -
val_loss: 0.1168
Epoch 80/100
115092/115092 [==============================] - 19s 169us/step - loss: 0.1048 -
```

```
val_loss: 0.1058
Epoch 81/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.1037 -
val_loss: 0.0987
Epoch 82/100
115092/115092 [==============================] - 19s 162us/step - loss: 0.1044 -
val_loss: 0.0978
Epoch 83/100
115092/115092 [==============================] - 19s 163us/step - loss: 0.1032 -
val_loss: 0.1057
Epoch 84/100
115092/115092 [==============================] - 19s 161us/step - loss: 0.1028 -
val_loss: 0.1032
Epoch 85/100
115092/115092 [==============================] - 19s 161us/step - loss: 0.1025 -
val_loss: 0.1204
Epoch 86/100
115092/115092 [==============================] - 19s 164us/step - loss: 0.1019 -
val_loss: 0.0985
Epoch 87/100
115092/115092 [==============================] - 18s 160us/step - loss: 0.1010 -
val_loss: 0.1022
Epoch 88/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1007 -
val_loss: 0.0965
Epoch 89/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.1012 -
val_loss: 0.0964
Epoch 90/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.1008 -
val_loss: 0.1176
Epoch 91/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.1002 -
val_loss: 0.1072
Epoch 92/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.1007 -
val_loss: 0.1081
Epoch 93/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0998 -
val_loss: 0.1093
Epoch 94/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0994 -
val_loss: 0.1171
Epoch 95/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0996 -
val_loss: 0.0996
Epoch 96/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0995 -
```

```
val_loss: 0.0955
Epoch 97/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0985 -
val_loss: 0.0968
Epoch 98/100
115092/115092 [==============================] - 18s 159us/step - loss: 0.0982 -
val_loss: 0.1178
Epoch 99/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0979 -
val_loss: 0.1069
Epoch 100/100
115092/115092 [==============================] - 18s 158us/step - loss: 0.0970 -
val_loss: 0.0968
```

[34]:
```
Y_testB = model.predict(X_valB).flatten()
```

[35]:
```
Y_testB =np.round(Y_testB, 0)
```

[36]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    labelB =  YA[i]
    predictionB =  Y_testB[i]
    print("Estado: "+ np.array2string(labelB) + ", predicted:" + np.
 ↪array2string(predictionB))
```

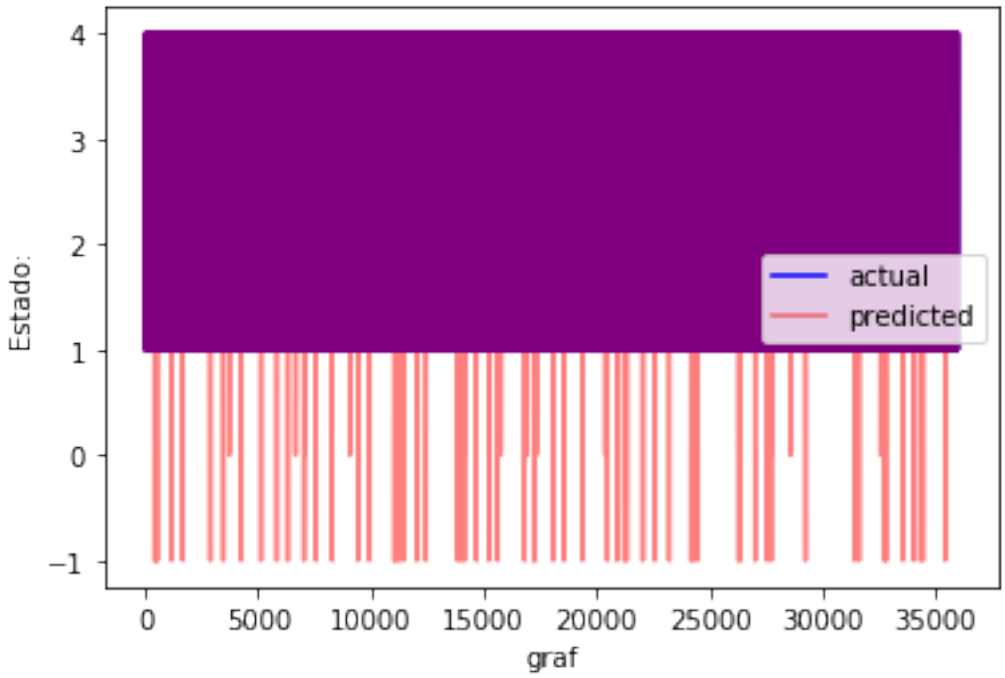
```
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
```

```
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

[37]:
```
r2 = r2_score(Y_testB, Y_val)
print (r2)
```

```
0.9236670228821766
```

[38]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_testA),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[39]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_testB
accur=np.sum(Accu)
```

```
accur/len(Y_val)
```

[39]: 0.9430311118525315

Unimos los datos quedandonos con el minimo media y maximo para ver si con alguna de las opciones podemos mejorar los resultados

[40]:
```
YtA = np.asmatrix(Y_testA)
YtB = np.asmatrix(Y_testB)
ense1=np.zeros(len(Y_testA))
ense2=np.zeros(len(Y_testA))
ense3=np.zeros(len(Y_testA))
Yt=np.concatenate((YtA.T,YtB.T),axis=1)
for i in range(len(Yt)):
    ense1[i]=np.min(Yt[i,:])
    ense2[i]=np.max(Yt[i,:])
    ense3[i]=np.mean(Yt[i,:])

ense3 =np.round(ense3, 0)
```

[45]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==ense1
accur=np.sum(Accu)
accur/len(Y_val)
```

[45]: 0.9559874329246254

[41]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==ense2
accur=np.sum(Accu)
accur/len(Y_val)
```

[41]: 0.9586009397503267

[46]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==ense3
accur=np.sum(Accu)
accur/len(Y_val)
```

[46]: 0.9556537937553868

[42]:
```
r2 = r2_score(ense1, Y_val)
print (r2)
```

```
0.9305066644684276
```

[43]:
```
r2 = r2_score(ense2, Y_val)
print (r2)
```

```
0.9386869691515516
```

[44]:
```
r2 = r2_score(ense3, Y_val)
print (r2)
```

```
0.9486139227380893
```

[ ]:
```

```

TFM_wrist_data_ENSEMBLE2-final

September 4, 2019

[1]:
```
from keras.layers import Input
from keras.layers.core import Dense, Activation
from keras.models import Sequential, Model
from keras.layers import Activation
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pickle
import numpy as np
import sys
import os
import pandas
from pandas import set_option
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
```

```
Using TensorFlow backend.
```

0.1 Ensemble

[2]:
```
data = pickle.load(open('S2.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
```

```
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT1=np.concatenate((B1,A2),axis=1)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]>4), 0)
MAT1 = np.delete(MAT1, np.where(MAT1[:,0]<=0), 0)
```

[3]:
```
data = pickle.load(open('S3.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
```

```
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT2=np.concatenate((B1,A2),axis=1)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]>4), 0)
MAT2 = np.delete(MAT2, np.where(MAT2[:,0]<=0), 0)
```

[4]:
```
data = pickle.load(open('S4.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
```

```
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=25*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT3=np.concatenate((B1,A2),axis=1)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]>4), 0)
MAT3 = np.delete(MAT3, np.where(MAT3[:,0]<=0), 0)
```

[5]:
```
data = pickle.load(open('S5.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
```

```
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=35*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT4=np.concatenate((B1,A2),axis=1)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]>4), 0)
MAT4 = np.delete(MAT4, np.where(MAT4[:,0]<=0), 0)
```

[6]:
```
data = pickle.load(open('S6.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
```

```
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT5=np.concatenate((B1,A2),axis=1)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]>4), 0)
MAT5 = np.delete(MAT5, np.where(MAT5[:,0]<=0), 0)
```

[7]:
```
data = pickle.load(open('S7.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
```

```
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT6=np.concatenate((B1,A2),axis=1)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]>4), 0)
MAT6 = np.delete(MAT6, np.where(MAT6[:,0]<=0), 0)
```

[8]:
```
data = pickle.load(open('S8.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT7=np.concatenate((B1,A2),axis=1)
```

```
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]>4), 0)
MAT7 = np.delete(MAT7, np.where(MAT7[:,0]<=0), 0)
```

[9]:
```
data = pickle.load(open('S9.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
```

```
B1 = np.asmatrix(E)
B1=B1.T
MAT8=np.concatenate((B1,A2),axis=1)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]>4), 0)
MAT8 = np.delete(MAT8, np.where(MAT8[:,0]<=0), 0)
```

[10]:
```
data = pickle.load(open('S10.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
```

```
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT9=np.concatenate((B1,A2),axis=1)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]>4), 0)
MAT9 = np.delete(MAT9, np.where(MAT9[:,0]<=0), 0)
```

[11]:
```
data = pickle.load(open('S11.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
```

```
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=26*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT10=np.concatenate((B1,A2),axis=1)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]>4), 0)
MAT10 = np.delete(MAT10, np.where(MAT10[:,0]<=0), 0)
```

[12]:
```
data = pickle.load(open('S13.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
```

```
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT11=np.concatenate((B1,A2),axis=1)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]>4), 0)
MAT11 = np.delete(MAT11, np.where(MAT11[:,0]<=0), 0)
```

[13]:
```
data = pickle.load(open('S14.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
```

```
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=27*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT12=np.concatenate((B1,A2),axis=1)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]>4), 0)
MAT12 = np.delete(MAT12, np.where(MAT12[:,0]<=0), 0)
```

[14]:
```
data = pickle.load(open('S15.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
```

```
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=28*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT13=np.concatenate((B1,A2),axis=1)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]>4), 0)
MAT13 = np.delete(MAT13, np.where(MAT13[:,0]<=0), 0)
```

[15]:
```
data = pickle.load(open('S16.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
```

```
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=24*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT14=np.concatenate((B1,A2),axis=1)
MAT14 = np.delete(MAT14, np.where(MAT14[:,0]>4), 0)
MAT14= np.delete(MAT14, np.where(MAT14[:,0]<=0), 0)
```

[16]:
```
data = pickle.load(open('S17.pkl','rb'),encoding='latin1')
a=data['signal']['wrist']['ACC']
b=data['signal']['wrist']['BVP']
c=data['signal']['wrist']['EDA']
d=data['signal']['wrist']['TEMP']
A2l=data['label']
mind=min(len(a),len(b),len(c),len(d),len(A2l))
Cal1=len(a)/mind
Cal2=len(b)/mind
Cal3=len(c)/mind
Cal4=len(d)/mind
Cal5=len(A2l)/mind
l1=int(len(a)/Cal1)
A=[]
for i in range(l1):
    k=i*Cal1
    at=a[int(k)]
    A.append(at)
l2=int(len(b)/Cal2)
B=[]
for i in range(l2):
    k=i*Cal2
    at=b[int(k)]
```

```
    B.append(at)
l3=int(len(c)/Cal3)
C=[]
for i in range(l3):
    k=i*Cal3
    at=c[int(k)]
    C.append(at)
l4=int(len(d)/Cal4)
D=[]
for i in range(l4):
    k=i*Cal4
    at=d[int(k)]
    D.append(at)
l5=int(len(A2l)/Cal5)
E=[]
for i in range(l5):
    k=i*Cal5
    at=A2l[int(k)]
    E.append(at)
A2a=29*(np.ones((mind,1)))
A2=np.concatenate((A,B,C,D,A2a),axis=1)
B1 = np.asmatrix(E)
B1=B1.T
MAT15=np.concatenate((B1,A2),axis=1)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]>4), 0)
MAT15 = np.delete(MAT15, np.where(MAT15[:,0]<=0), 0)
```

[17]:
```
MAT=np.
 ↪concatenate((MAT1,MAT2,MAT3,MAT4,MAT5,MAT6,MAT7,MAT8,MAT9,MAT10,MAT11,MAT12,MAT13,MAT14,MAT
```

[18]:
```
X = MAT[:, 1:8]
Y = MAT[:, 0]
validation_size = 0.2
seed = 7
X_train, X_val, Y_train, Y_val = train_test_split(X, Y,␣
 ↪test_size=validation_size, random_state=seed)
```

[19]:
```
X_trainA= np.delete(X_train,3 , 1)
X_trainB= np.delete(X_train,4 , 1)
X_valA= np.delete(X_val,3, 1)
X_valB= np.delete(X_val,4 , 1)
```

[20]:
```
model = Sequential([
    Dense(128, input_shape=(6,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
    Dense(60),
    Activation('elu'),
```

```
    Dense(1),
    Activation('elu'),
])

model.compile(optimizer='Adadelta',loss='mae')
```

```
WARNING: Logging before flag parsing goes to stderr.
W0903 22:06:30.329254 19528 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:74: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

W0903 22:06:30.364161 19528 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:517: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

W0903 22:06:30.376129 19528 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:4138: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

W0903 22:06:30.440954 19528 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-packages\keras\optimizers.py:790:
The name tf.train.Optimizer is deprecated. Please use
tf.compat.v1.train.Optimizer instead.
```

[21]:
```
NUM_EPOCHS =200
BATCH_SIZE = 20

history = model.fit(X_trainA, Y_train, batch_size=BATCH_SIZE,␣
 ↪epochs=NUM_EPOCHS, validation_split=0.2)
```

```
W0903 22:06:30.655380 19528 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:986: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.

W0903 22:06:30.661365 19528 deprecation_wrapper.py:119] From
C:\Users\jorge\Anaconda3\envs\TFMGPU\lib\site-
packages\keras\backend\tensorflow_backend.py:973: The name tf.assign is
deprecated. Please use tf.compat.v1.assign instead.


Train on 115092 samples, validate on 28773 samples
Epoch 1/200
```

```
115092/115092 [==============================] - 20s 175us/step - loss: 0.5623 -
val_loss: 0.3846
Epoch 2/200
115092/115092 [==============================] - 17s 144us/step - loss: 0.3386 -
val_loss: 0.3603
Epoch 3/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.2668 -
val_loss: 0.2803
Epoch 4/200
115092/115092 [==============================] - 17s 144us/step - loss: 0.2250 -
val_loss: 0.2247
Epoch 5/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.2010 -
val_loss: 0.2149
Epoch 6/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.1836 -
val_loss: 0.2007
Epoch 7/200
115092/115092 [==============================] - 16s 138us/step - loss: 0.1706 -
val_loss: 0.1680
Epoch 8/200
115092/115092 [==============================] - 16s 138us/step - loss: 0.1591 -
val_loss: 0.1638
Epoch 9/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.1493 -
val_loss: 0.1500
Epoch 10/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.1411 -
val_loss: 0.1274
Epoch 11/200
115092/115092 [==============================] - 17s 143us/step - loss: 0.1340 -
val_loss: 0.1309
Epoch 12/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.1283 -
val_loss: 0.1240
Epoch 13/200
115092/115092 [==============================] - 16s 143us/step - loss: 0.1240 -
val_loss: 0.1212
Epoch 14/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.1201 -
val_loss: 0.1258
Epoch 15/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.1169 -
val_loss: 0.1391
Epoch 16/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.1144 -
val_loss: 0.1317
Epoch 17/200
```

```
115092/115092 [==============================] - 16s 139us/step - loss: 0.1124 -
val_loss: 0.1361
Epoch 18/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.1097 -
val_loss: 0.1193
Epoch 19/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.1073 -
val_loss: 0.1104
Epoch 20/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.1054 -
val_loss: 0.1036
Epoch 21/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.1031 -
val_loss: 0.1000
Epoch 22/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.1013 -
val_loss: 0.1135
Epoch 23/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.1000 -
val_loss: 0.0997
Epoch 24/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0983 -
val_loss: 0.1090
Epoch 25/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0970 -
val_loss: 0.1112
Epoch 26/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0955 -
val_loss: 0.0986
Epoch 27/200
115092/115092 [==============================] - 17s 144us/step - loss: 0.0941 -
val_loss: 0.0981
Epoch 28/200
115092/115092 [==============================] - 17s 144us/step - loss: 0.0928 -
val_loss: 0.1028
Epoch 29/200
115092/115092 [==============================] - 17s 144us/step - loss: 0.0917 -
val_loss: 0.0955
Epoch 30/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0907 -
val_loss: 0.1047
Epoch 31/200
115092/115092 [==============================] - 16s 143us/step - loss: 0.0898 -
val_loss: 0.1090
Epoch 32/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.0886 -
val_loss: 0.0887
Epoch 33/200
```

```
115092/115092 [==============================] - 16s 141us/step - loss: 0.0879 -
val_loss: 0.0874
Epoch 34/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0869 -
val_loss: 0.0893
Epoch 35/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0861 -
val_loss: 0.0974
Epoch 36/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0857 -
val_loss: 0.0893
Epoch 37/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0853 -
val_loss: 0.0955
Epoch 38/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0846 -
val_loss: 0.0880
Epoch 39/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0837 -
val_loss: 0.0817
Epoch 40/200
115092/115092 [==============================] - 16s 138us/step - loss: 0.0828 -
val_loss: 0.0861
Epoch 41/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0825 -
val_loss: 0.0968
Epoch 42/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0821 -
val_loss: 0.0841
Epoch 43/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0817 -
val_loss: 0.0798
Epoch 44/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0809 -
val_loss: 0.0813
Epoch 45/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0802 -
val_loss: 0.0903
Epoch 46/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.0805 -
val_loss: 0.0861
Epoch 47/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.0797 -
val_loss: 0.0906
Epoch 48/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.0790 -
val_loss: 0.0830
Epoch 49/200
```

```
115092/115092 [==============================] - 17s 145us/step - loss: 0.0785 -
val_loss: 0.0816
Epoch 50/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.0783 -
val_loss: 0.0914
Epoch 51/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0780 -
val_loss: 0.0825
Epoch 52/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0776 -
val_loss: 0.0947
Epoch 53/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0770 -
val_loss: 0.0802
Epoch 54/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0767 -
val_loss: 0.0829
Epoch 55/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0764 -
val_loss: 0.0797
Epoch 56/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0760 -
val_loss: 0.0835
Epoch 57/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0756 -
val_loss: 0.0751
Epoch 58/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0753 -
val_loss: 0.0909
Epoch 59/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0749 -
val_loss: 0.0964
Epoch 60/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0748 -
val_loss: 0.0913
Epoch 61/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0742 -
val_loss: 0.0961
Epoch 62/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0744 -
val_loss: 0.0790
Epoch 63/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0735 -
val_loss: 0.0816
Epoch 64/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0734 -
val_loss: 0.0790
Epoch 65/200
```

```
115092/115092 [==============================] - 16s 140us/step - loss: 0.0736 -
val_loss: 0.0787
Epoch 66/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0731 -
val_loss: 0.0732
Epoch 67/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0726 -
val_loss: 0.0994
Epoch 68/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0725 -
val_loss: 0.0855
Epoch 69/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.0719 -
val_loss: 0.0763
Epoch 70/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.0719 -
val_loss: 0.0722
Epoch 71/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0714 -
val_loss: 0.0850
Epoch 72/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0719 -
val_loss: 0.0735
Epoch 73/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.0707 -
val_loss: 0.0790
Epoch 74/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0713 -
val_loss: 0.0791
Epoch 75/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.0707 -
val_loss: 0.0722
Epoch 76/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0707 -
val_loss: 0.0885
Epoch 77/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0704 -
val_loss: 0.0775
Epoch 78/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0699 -
val_loss: 0.0755
Epoch 79/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0699 -
val_loss: 0.0666
Epoch 80/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0696 -
val_loss: 0.0656
Epoch 81/200
```

```
115092/115092 [==============================] - 16s 139us/step - loss: 0.0692 -
val_loss: 0.0802
Epoch 82/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0691 -
val_loss: 0.0826
Epoch 83/200
115092/115092 [==============================] - 17s 144us/step - loss: 0.0689 -
val_loss: 0.0813
Epoch 84/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.0685 -
val_loss: 0.0690
Epoch 85/200
115092/115092 [==============================] - 17s 148us/step - loss: 0.0683 -
val_loss: 0.0701
Epoch 86/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.0683 -
val_loss: 0.0690
Epoch 87/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0676 -
val_loss: 0.0681
Epoch 88/200
115092/115092 [==============================] - 16s 138us/step - loss: 0.0674 -
val_loss: 0.0724
Epoch 89/200
115092/115092 [==============================] - 16s 137us/step - loss: 0.0677 -
val_loss: 0.0902
Epoch 90/200
115092/115092 [==============================] - 16s 137us/step - loss: 0.0674 -
val_loss: 0.0902
Epoch 91/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0671 -
val_loss: 0.0711
Epoch 92/200
115092/115092 [==============================] - 16s 137us/step - loss: 0.0669 -
val_loss: 0.0611
Epoch 93/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0668 -
val_loss: 0.0734
Epoch 94/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0665 -
val_loss: 0.0742
Epoch 95/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0663 -
val_loss: 0.0723
Epoch 96/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0662 -
val_loss: 0.0654
Epoch 97/200
```

```
115092/115092 [==============================] - 16s 141us/step - loss: 0.0658 -
val_loss: 0.0657
Epoch 98/200
115092/115092 [==============================] - 16s 138us/step - loss: 0.0659 -
val_loss: 0.0758
Epoch 99/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0655 -
val_loss: 0.0678
Epoch 100/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.0655 -
val_loss: 0.0749
Epoch 101/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0658 -
val_loss: 0.0771
Epoch 102/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0652 -
val_loss: 0.0921
Epoch 103/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0652 -
val_loss: 0.0775
Epoch 104/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0651 -
val_loss: 0.0655
Epoch 105/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0650 -
val_loss: 0.0794
Epoch 106/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0653 -
val_loss: 0.0665
Epoch 107/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0649 -
val_loss: 0.0685
Epoch 108/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0650 -
val_loss: 0.0804
Epoch 109/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0649 -
val_loss: 0.0735
Epoch 110/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0643 -
val_loss: 0.0726
Epoch 111/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0639 -
val_loss: 0.0745
Epoch 112/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0642 -
val_loss: 0.0598
Epoch 113/200
```

```
115092/115092 [==============================] - 17s 145us/step - loss: 0.0642 -
val_loss: 0.0724
Epoch 114/200
115092/115092 [==============================] - 16s 138us/step - loss: 0.0642 -
val_loss: 0.0621
Epoch 115/200
115092/115092 [==============================] - 16s 137us/step - loss: 0.0640 -
val_loss: 0.0751
Epoch 116/200
115092/115092 [==============================] - 16s 136us/step - loss: 0.0636 -
val_loss: 0.0732
Epoch 117/200
115092/115092 [==============================] - 16s 137us/step - loss: 0.0634 -
val_loss: 0.0782
Epoch 118/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0634 -
val_loss: 0.0577
Epoch 119/200
115092/115092 [==============================] - 16s 136us/step - loss: 0.0631 -
val_loss: 0.0787
Epoch 120/200
115092/115092 [==============================] - 16s 136us/step - loss: 0.0636 -
val_loss: 0.0907
Epoch 121/200
115092/115092 [==============================] - 16s 136us/step - loss: 0.0628 -
val_loss: 0.0714
Epoch 122/200
115092/115092 [==============================] - 16s 136us/step - loss: 0.0631 -
val_loss: 0.0578
Epoch 123/200
115092/115092 [==============================] - 16s 136us/step - loss: 0.0624 -
val_loss: 0.0754
Epoch 124/200
115092/115092 [==============================] - 16s 135us/step - loss: 0.0629 -
val_loss: 0.0648
Epoch 125/200
115092/115092 [==============================] - 16s 136us/step - loss: 0.0626 -
val_loss: 0.0718
Epoch 126/200
115092/115092 [==============================] - 16s 136us/step - loss: 0.0626 -
val_loss: 0.0697
Epoch 127/200
115092/115092 [==============================] - 16s 135us/step - loss: 0.0627 -
val_loss: 0.0677
Epoch 128/200
115092/115092 [==============================] - 16s 135us/step - loss: 0.0625 -
val_loss: 0.0628
Epoch 129/200
```

```
115092/115092 [==============================] - 16s 136us/step - loss: 0.0622 -
val_loss: 0.0719
Epoch 130/200
115092/115092 [==============================] - 16s 135us/step - loss: 0.0621 -
val_loss: 0.0597
Epoch 131/200
115092/115092 [==============================] - 16s 136us/step - loss: 0.0620 -
val_loss: 0.0634
Epoch 132/200
115092/115092 [==============================] - 16s 136us/step - loss: 0.0618 -
val_loss: 0.0632
Epoch 133/200
115092/115092 [==============================] - 16s 136us/step - loss: 0.0618 -
val_loss: 0.0620
Epoch 134/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0616 -
val_loss: 0.0786
Epoch 135/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.0615 -
val_loss: 0.0713
Epoch 136/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0615 -
val_loss: 0.0629
Epoch 137/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.0609 -
val_loss: 0.0527
Epoch 138/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.0616 -
val_loss: 0.0633
Epoch 139/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0618 -
val_loss: 0.0739
Epoch 140/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.0613 -
val_loss: 0.0644
Epoch 141/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.0611 -
val_loss: 0.0728
Epoch 142/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.0614 -
val_loss: 0.0855
Epoch 143/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.0608 -
val_loss: 0.0645
Epoch 144/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0611 -
val_loss: 0.0693
Epoch 145/200
```

```
115092/115092 [==============================] - 17s 145us/step - loss: 0.0609 -
val_loss: 0.0850
Epoch 146/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.0610 -
val_loss: 0.0709
Epoch 147/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0606 -
val_loss: 0.0625
Epoch 148/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0607 -
val_loss: 0.0661
Epoch 149/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0602 -
val_loss: 0.0803
Epoch 150/200
115092/115092 [==============================] - 17s 144us/step - loss: 0.0603 -
val_loss: 0.0702
Epoch 151/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0602 -
val_loss: 0.0625
Epoch 152/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0601 -
val_loss: 0.0675
Epoch 153/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.0603 -
val_loss: 0.0695
Epoch 154/200
115092/115092 [==============================] - 17s 144us/step - loss: 0.0600 -
val_loss: 0.0669
Epoch 155/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.0601 -
val_loss: 0.0644
Epoch 156/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0600 -
val_loss: 0.0754
Epoch 157/200
115092/115092 [==============================] - 16s 143us/step - loss: 0.0598 -
val_loss: 0.0768
Epoch 158/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.0595 -
val_loss: 0.0649
Epoch 159/200
115092/115092 [==============================] - 17s 144us/step - loss: 0.0599 -
val_loss: 0.0737
Epoch 160/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.0600 -
val_loss: 0.0528
Epoch 161/200
```

```
115092/115092 [==============================] - 16s 140us/step - loss: 0.0596 -
val_loss: 0.0688
Epoch 162/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.0598 -
val_loss: 0.0584
Epoch 163/200
115092/115092 [==============================] - 16s 138us/step - loss: 0.0595 -
val_loss: 0.0903
Epoch 164/200
115092/115092 [==============================] - 16s 138us/step - loss: 0.0595 -
val_loss: 0.0657
Epoch 165/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0595 -
val_loss: 0.0596
Epoch 166/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0592 -
val_loss: 0.0615
Epoch 167/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0592 -
val_loss: 0.0605
Epoch 168/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0591 -
val_loss: 0.0703
Epoch 169/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0597 -
val_loss: 0.0676
Epoch 170/200
115092/115092 [==============================] - 16s 143us/step - loss: 0.0591 -
val_loss: 0.0707
Epoch 171/200
115092/115092 [==============================] - 16s 138us/step - loss: 0.0590 -
val_loss: 0.0654
Epoch 172/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0589 -
val_loss: 0.0729
Epoch 173/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0588 -
val_loss: 0.0621
Epoch 174/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0592 -
val_loss: 0.0622
Epoch 175/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0591 -
val_loss: 0.0646
Epoch 176/200
115092/115092 [==============================] - 17s 148us/step - loss: 0.0589 -
val_loss: 0.0722
Epoch 177/200
```

```
115092/115092 [==============================] - 17s 145us/step - loss: 0.0590 -
val_loss: 0.0635
Epoch 178/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0588 -
val_loss: 0.0639
Epoch 179/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0583 -
val_loss: 0.0682
Epoch 180/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.0584 -
val_loss: 0.0749
Epoch 181/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.0583 -
val_loss: 0.0621
Epoch 182/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.0585 -
val_loss: 0.0579
Epoch 183/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0582 -
val_loss: 0.0713
Epoch 184/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0585 -
val_loss: 0.0687
Epoch 185/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0584 -
val_loss: 0.0713
Epoch 186/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0588 -
val_loss: 0.0675
Epoch 187/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0588 -
val_loss: 0.0595
Epoch 188/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0582 -
val_loss: 0.0747
Epoch 189/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0579 -
val_loss: 0.0604
Epoch 190/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.0583 -
val_loss: 0.0606
Epoch 191/200
115092/115092 [==============================] - 16s 138us/step - loss: 0.0581 -
val_loss: 0.0571
Epoch 192/200
115092/115092 [==============================] - 16s 138us/step - loss: 0.0585 -
val_loss: 0.0756
Epoch 193/200
```

```
115092/115092 [==============================] - 16s 138us/step - loss: 0.0580 -
val_loss: 0.0600
Epoch 194/200
115092/115092 [==============================] - 16s 138us/step - loss: 0.0581 -
val_loss: 0.0661
Epoch 195/200
115092/115092 [==============================] - 16s 138us/step - loss: 0.0580 -
val_loss: 0.0547
Epoch 196/200
115092/115092 [==============================] - 16s 138us/step - loss: 0.0582 -
val_loss: 0.0648
Epoch 197/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0581 -
val_loss: 0.0634
Epoch 198/200
115092/115092 [==============================] - 16s 138us/step - loss: 0.0578 -
val_loss: 0.0550
Epoch 199/200
115092/115092 [==============================] - 16s 138us/step - loss: 0.0579 -
val_loss: 0.0635
Epoch 200/200
115092/115092 [==============================] - 16s 139us/step - loss: 0.0582 -
val_loss: 0.0692
```

[22]:
```
Y_testA = model.predict(X_valA).flatten()
```

[23]:
```
Y_testA =np.round(Y_testA, 0)
```

[24]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    label =  YA[i]
    predictionA =  Y_testA[i]
    print("Estado: "+ np.array2string(label) + ", predicted:" + np.
 ↪array2string(predictionA))
```

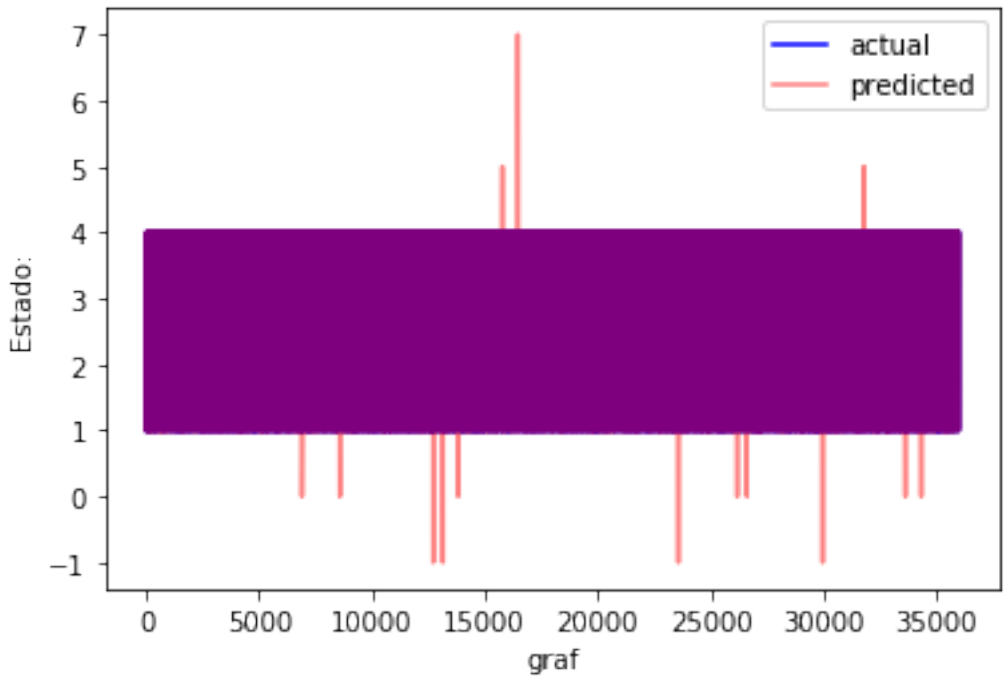
```
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

```
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

[25]:
```
from sklearn.metrics import r2_score
r2 = r2_score(Y_testA, Y_val)
print (r2)
```

```
0.9623221863076288
```

[26]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_testA),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[27]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_testA
accur=np.sum(Accu)
accur/len(Y_val)
```

[27]:
```
0.9718352934634527
```

[28]:
```
model = Sequential([
    Dense(128, input_shape=(6,)),
    Activation('elu'),
    Dense(70),
    Activation('elu'),
    Dense(60),
    Activation('elu'),
    Dense(1),
    Activation('elu'),
])


model.compile(optimizer='Adadelta',loss='mae')
```

[29]:
```
NUM_EPOCHS =200
BATCH_SIZE = 20

history = model.fit(X_trainB, Y_train, batch_size=BATCH_SIZE,␣
 ↪epochs=NUM_EPOCHS, validation_split=0.2)
```

```
Train on 115092 samples, validate on 28773 samples
Epoch 1/200
115092/115092 [==============================] - 17s 149us/step - loss: 0.6444 -
val_loss: 0.4960
Epoch 2/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.4209 -
val_loss: 0.4074
Epoch 3/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.3398 -
val_loss: 0.3533
Epoch 4/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.2947 -
val_loss: 0.3078
Epoch 5/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.2652 -
val_loss: 0.2337
Epoch 6/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.2450 -
val_loss: 0.2362
Epoch 7/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.2311 -
val_loss: 0.2506
Epoch 8/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.2195 -
val_loss: 0.2046
Epoch 9/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.2107 -
val_loss: 0.2227
Epoch 10/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.2043 -
val_loss: 0.2069
Epoch 11/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1981 -
val_loss: 0.2119
Epoch 12/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1929 -
val_loss: 0.1847
Epoch 13/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1881 -
val_loss: 0.1772
Epoch 14/200
115092/115092 [==============================] - 17s 148us/step - loss: 0.1843 -
val_loss: 0.1850
Epoch 15/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1802 -
val_loss: 0.1848
Epoch 16/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1766 -
```

```
val_loss: 0.1691
Epoch 17/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1731 -
val_loss: 0.1793
Epoch 18/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1704 -
val_loss: 0.1650
Epoch 19/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1685 -
val_loss: 0.1750
Epoch 20/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.1657 -
val_loss: 0.2017
Epoch 21/200
115092/115092 [==============================] - 18s 153us/step - loss: 0.1641 -
val_loss: 0.1774
Epoch 22/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.1622 -
val_loss: 0.1709
Epoch 23/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1598 -
val_loss: 0.1615
Epoch 24/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.1578 -
val_loss: 0.1844
Epoch 25/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1559 -
val_loss: 0.1554
Epoch 26/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1544 -
val_loss: 0.1484
Epoch 27/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1527 -
val_loss: 0.1693
Epoch 28/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1514 -
val_loss: 0.1678
Epoch 29/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.1499 -
val_loss: 0.1496
Epoch 30/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1486 -
val_loss: 0.1473
Epoch 31/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1478 -
val_loss: 0.1547
Epoch 32/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1456 -
```

```
val_loss: 0.1678
Epoch 33/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.1445 -
val_loss: 0.1546
Epoch 34/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1432 -
val_loss: 0.1553
Epoch 35/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1432 -
val_loss: 0.1860
Epoch 36/200
115092/115092 [==============================] - 17s 148us/step - loss: 0.1413 -
val_loss: 0.1444
Epoch 37/200
115092/115092 [==============================] - 17s 148us/step - loss: 0.1409 -
val_loss: 0.1547
Epoch 38/200
115092/115092 [==============================] - 17s 148us/step - loss: 0.1395 -
val_loss: 0.1414
Epoch 39/200
115092/115092 [==============================] - 17s 149us/step - loss: 0.1386 -
val_loss: 0.1397
Epoch 40/200
115092/115092 [==============================] - 17s 150us/step - loss: 0.1374 -
val_loss: 0.1514
Epoch 41/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1366 -
val_loss: 0.1453
Epoch 42/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1352 -
val_loss: 0.1370
Epoch 43/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.1352 -
val_loss: 0.1332
Epoch 44/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1339 -
val_loss: 0.1560
Epoch 45/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1330 -
val_loss: 0.1462
Epoch 46/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1327 -
val_loss: 0.1369
Epoch 47/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1333 -
val_loss: 0.1353
Epoch 48/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.1312 -
```

```
val_loss: 0.1438
Epoch 49/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.1311 -
val_loss: 0.1373
Epoch 50/200
115092/115092 [==============================] - 17s 149us/step - loss: 0.1308 -
val_loss: 0.1400
Epoch 51/200
115092/115092 [==============================] - 17s 149us/step - loss: 0.1303 -
val_loss: 0.1248
Epoch 52/200
115092/115092 [==============================] - 17s 148us/step - loss: 0.1296 -
val_loss: 0.1410
Epoch 53/200
115092/115092 [==============================] - 17s 149us/step - loss: 0.1286 -
val_loss: 0.1485
Epoch 54/200
115092/115092 [==============================] - 17s 149us/step - loss: 0.1284 -
val_loss: 0.1577
Epoch 55/200
115092/115092 [==============================] - 18s 153us/step - loss: 0.1275 -
val_loss: 0.1312
Epoch 56/200
115092/115092 [==============================] - 17s 149us/step - loss: 0.1272 -
val_loss: 0.1275
Epoch 57/200
115092/115092 [==============================] - 17s 149us/step - loss: 0.1259 -
val_loss: 0.1431
Epoch 58/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.1261 -
val_loss: 0.1315
Epoch 59/200
115092/115092 [==============================] - 17s 149us/step - loss: 0.1244 -
val_loss: 0.1290
Epoch 60/200
115092/115092 [==============================] - 17s 148us/step - loss: 0.1240 -
val_loss: 0.1200
Epoch 61/200
115092/115092 [==============================] - 17s 149us/step - loss: 0.1244 -
val_loss: 0.1366
Epoch 62/200
115092/115092 [==============================] - 17s 149us/step - loss: 0.1241 -
val_loss: 0.1476
Epoch 63/200
115092/115092 [==============================] - 17s 151us/step - loss: 0.1224 -
val_loss: 0.1337
Epoch 64/200
115092/115092 [==============================] - 17s 150us/step - loss: 0.1224 -
```

```
val_loss: 0.1350
Epoch 65/200
115092/115092 [==============================] - 17s 149us/step - loss: 0.1218 -
val_loss: 0.1338
Epoch 66/200
115092/115092 [==============================] - 17s 150us/step - loss: 0.1217 -
val_loss: 0.1361
Epoch 67/200
115092/115092 [==============================] - 17s 149us/step - loss: 0.1213 -
val_loss: 0.1269
Epoch 68/200
115092/115092 [==============================] - 17s 149us/step - loss: 0.1210 -
val_loss: 0.1278
Epoch 69/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.1202 -
val_loss: 0.1323
Epoch 70/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.1199 -
val_loss: 0.1355
Epoch 71/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1189 -
val_loss: 0.1184
Epoch 72/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1189 -
val_loss: 0.1222
Epoch 73/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1184 -
val_loss: 0.1208
Epoch 74/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1181 -
val_loss: 0.1359
Epoch 75/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1180 -
val_loss: 0.1308
Epoch 76/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1174 -
val_loss: 0.1293
Epoch 77/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1163 -
val_loss: 0.1172
Epoch 78/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1173 -
val_loss: 0.1223
Epoch 79/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1168 -
val_loss: 0.1184
Epoch 80/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1164 -
```

```
val_loss: 0.1106
Epoch 81/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1167 -
val_loss: 0.1480
Epoch 82/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1157 -
val_loss: 0.1256
Epoch 83/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1155 -
val_loss: 0.1180
Epoch 84/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1150 -
val_loss: 0.1245
Epoch 85/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1149 -
val_loss: 0.1282
Epoch 86/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1150 -
val_loss: 0.1138
Epoch 87/200
115092/115092 [==============================] - 17s 148us/step - loss: 0.1144 -
val_loss: 0.1100
Epoch 88/200
115092/115092 [==============================] - 17s 149us/step - loss: 0.1140 -
val_loss: 0.1084
Epoch 89/200
115092/115092 [==============================] - 17s 148us/step - loss: 0.1136 -
val_loss: 0.1219
Epoch 90/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1136 -
val_loss: 0.1238
Epoch 91/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1133 -
val_loss: 0.1328
Epoch 92/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1126 -
val_loss: 0.1134
Epoch 93/200
115092/115092 [==============================] - 17s 149us/step - loss: 0.1128 -
val_loss: 0.1277
Epoch 94/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1125 -
val_loss: 0.1223
Epoch 95/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1127 -
val_loss: 0.1170
Epoch 96/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1124 -
```

```
val_loss: 0.1332
Epoch 97/200
115092/115092 [==============================] - 17s 149us/step - loss: 0.1119 -
val_loss: 0.1346
Epoch 98/200
115092/115092 [==============================] - 18s 154us/step - loss: 0.1123 -
val_loss: 0.1160
Epoch 99/200
115092/115092 [==============================] - 17s 150us/step - loss: 0.1119 -
val_loss: 0.1123
Epoch 100/200
115092/115092 [==============================] - 17s 148us/step - loss: 0.1118 -
val_loss: 0.1169
Epoch 101/200
115092/115092 [==============================] - 17s 144us/step - loss: 0.1113 -
val_loss: 0.1151
Epoch 102/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1109 -
val_loss: 0.1426
Epoch 103/200
115092/115092 [==============================] - 17s 144us/step - loss: 0.1111 -
val_loss: 0.1010
Epoch 104/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1108 -
val_loss: 0.1463
Epoch 105/200
115092/115092 [==============================] - 16s 143us/step - loss: 0.1105 -
val_loss: 0.1223
Epoch 106/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.1106 -
val_loss: 0.1102
Epoch 107/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.1108 -
val_loss: 0.1317
Epoch 108/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.1103 -
val_loss: 0.1114
Epoch 109/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1100 -
val_loss: 0.1230
Epoch 110/200
115092/115092 [==============================] - 16s 143us/step - loss: 0.1092 -
val_loss: 0.1144
Epoch 111/200
115092/115092 [==============================] - 16s 143us/step - loss: 0.1097 -
val_loss: 0.1059
Epoch 112/200
115092/115092 [==============================] - 16s 143us/step - loss: 0.1098 -
```

```
val_loss: 0.1163
Epoch 113/200
115092/115092 [==============================] - 17s 143us/step - loss: 0.1098 -
val_loss: 0.1138
Epoch 114/200
115092/115092 [==============================] - 17s 143us/step - loss: 0.1096 -
val_loss: 0.1347
Epoch 115/200
115092/115092 [==============================] - 17s 148us/step - loss: 0.1092 -
val_loss: 0.1270
Epoch 116/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.1093 -
val_loss: 0.1092
Epoch 117/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.1087 -
val_loss: 0.1200
Epoch 118/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1086 -
val_loss: 0.1133
Epoch 119/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.1087 -
val_loss: 0.1038
Epoch 120/200
115092/115092 [==============================] - 17s 144us/step - loss: 0.1084 -
val_loss: 0.1117
Epoch 121/200
115092/115092 [==============================] - 17s 144us/step - loss: 0.1089 -
val_loss: 0.1108
Epoch 122/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.1084 -
val_loss: 0.1067
Epoch 123/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.1079 -
val_loss: 0.1163
Epoch 124/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.1084 -
val_loss: 0.1300
Epoch 125/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.1082 -
val_loss: 0.1308
Epoch 126/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.1080 -
val_loss: 0.1068
Epoch 127/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.1075 -
val_loss: 0.1108
Epoch 128/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1073 -
```

```
val_loss: 0.1269
Epoch 129/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1071 -
val_loss: 0.1223
Epoch 130/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1073 -
val_loss: 0.1094
Epoch 131/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1073 -
val_loss: 0.1216
Epoch 132/200
115092/115092 [==============================] - 16s 143us/step - loss: 0.1067 -
val_loss: 0.1176
Epoch 133/200
115092/115092 [==============================] - 17s 144us/step - loss: 0.1065 -
val_loss: 0.0967
Epoch 134/200
115092/115092 [==============================] - 16s 143us/step - loss: 0.1064 -
val_loss: 0.1095
Epoch 135/200
115092/115092 [==============================] - 17s 144us/step - loss: 0.1060 -
val_loss: 0.1152
Epoch 136/200
115092/115092 [==============================] - 17s 144us/step - loss: 0.1064 -
val_loss: 0.1155
Epoch 137/200
115092/115092 [==============================] - 16s 140us/step - loss: 0.1065 -
val_loss: 0.1291
Epoch 138/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.1058 -
val_loss: 0.1020
Epoch 139/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.1054 -
val_loss: 0.1261
Epoch 140/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.1052 -
val_loss: 0.1136
Epoch 141/200
115092/115092 [==============================] - 16s 143us/step - loss: 0.1057 -
val_loss: 0.1058
Epoch 142/200
115092/115092 [==============================] - 17s 143us/step - loss: 0.1053 -
val_loss: 0.1028
Epoch 143/200
115092/115092 [==============================] - 17s 144us/step - loss: 0.1051 -
val_loss: 0.1033
Epoch 144/200
115092/115092 [==============================] - 17s 144us/step - loss: 0.1051 -
```

```
val_loss: 0.1090
Epoch 145/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1053 -
val_loss: 0.1050
Epoch 146/200
115092/115092 [==============================] - 16s 141us/step - loss: 0.1047 -
val_loss: 0.1041
Epoch 147/200
115092/115092 [==============================] - 18s 160us/step - loss: 0.1047 -
val_loss: 0.1113
Epoch 148/200
115092/115092 [==============================] - 19s 164us/step - loss: 0.1052 -
val_loss: 0.1038
Epoch 149/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1047 -
val_loss: 0.1003
Epoch 150/200
115092/115092 [==============================] - 17s 152us/step - loss: 0.1046 -
val_loss: 0.1121
Epoch 151/200
115092/115092 [==============================] - 17s 150us/step - loss: 0.1039 -
val_loss: 0.1151
Epoch 152/200
115092/115092 [==============================] - 17s 149us/step - loss: 0.1044 -
val_loss: 0.1154
Epoch 153/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1038 -
val_loss: 0.1178
Epoch 154/200
115092/115092 [==============================] - 17s 148us/step - loss: 0.1045 -
val_loss: 0.1108
Epoch 155/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1045 -
val_loss: 0.1021
Epoch 156/200
115092/115092 [==============================] - 18s 152us/step - loss: 0.1043 -
val_loss: 0.1185
Epoch 157/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.1040 -
val_loss: 0.1153
Epoch 158/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1031 -
val_loss: 0.1024
Epoch 159/200
115092/115092 [==============================] - 17s 151us/step - loss: 0.1038 -
val_loss: 0.1036
Epoch 160/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1038 -
```

```
val_loss: 0.1239
Epoch 161/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1036 -
val_loss: 0.0982
Epoch 162/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1035 -
val_loss: 0.1105
Epoch 163/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1037 -
val_loss: 0.1134
Epoch 164/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.1037 -
val_loss: 0.1120
Epoch 165/200
115092/115092 [==============================] - 17s 149us/step - loss: 0.1030 -
val_loss: 0.1108
Epoch 166/200
115092/115092 [==============================] - 16s 143us/step - loss: 0.1032 -
val_loss: 0.1073
Epoch 167/200
115092/115092 [==============================] - 16s 143us/step - loss: 0.1033 -
val_loss: 0.0950
Epoch 168/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.1026 -
val_loss: 0.1012
Epoch 169/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1031 -
val_loss: 0.1053
Epoch 170/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.1025 -
val_loss: 0.1211
Epoch 171/200
115092/115092 [==============================] - 17s 151us/step - loss: 0.1026 -
val_loss: 0.1054
Epoch 172/200
115092/115092 [==============================] - 17s 143us/step - loss: 0.1030 -
val_loss: 0.1028
Epoch 173/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.1025 -
val_loss: 0.1181
Epoch 174/200
115092/115092 [==============================] - 17s 144us/step - loss: 0.1024 -
val_loss: 0.1123
Epoch 175/200
115092/115092 [==============================] - 16s 143us/step - loss: 0.1025 -
val_loss: 0.1001
Epoch 176/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1017 -
```

```
val_loss: 0.1080
Epoch 177/200
115092/115092 [==============================] - 16s 143us/step - loss: 0.1023 -
val_loss: 0.1189
Epoch 178/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1023 -
val_loss: 0.1135
Epoch 179/200
115092/115092 [==============================] - 16s 143us/step - loss: 0.1016 -
val_loss: 0.0995
Epoch 180/200
115092/115092 [==============================] - 16s 142us/step - loss: 0.1020 -
val_loss: 0.1069
Epoch 181/200
115092/115092 [==============================] - 16s 143us/step - loss: 0.1022 -
val_loss: 0.1106
Epoch 182/200
115092/115092 [==============================] - 17s 144us/step - loss: 0.1014 -
val_loss: 0.0946
Epoch 183/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1013 -
val_loss: 0.1164
Epoch 184/200
115092/115092 [==============================] - 17s 144us/step - loss: 0.1020 -
val_loss: 0.0955
Epoch 185/200
115092/115092 [==============================] - 17s 144us/step - loss: 0.1019 -
val_loss: 0.1013
Epoch 186/200
115092/115092 [==============================] - 17s 144us/step - loss: 0.1019 -
val_loss: 0.1122
Epoch 187/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1013 -
val_loss: 0.1167
Epoch 188/200
115092/115092 [==============================] - 16s 143us/step - loss: 0.1013 -
val_loss: 0.1096
Epoch 189/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1010 -
val_loss: 0.0990
Epoch 190/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1013 -
val_loss: 0.1040
Epoch 191/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1014 -
val_loss: 0.1204
Epoch 192/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1006 -
```

```
val_loss: 0.1054
Epoch 193/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1007 -
val_loss: 0.1065
Epoch 194/200
115092/115092 [==============================] - 17s 145us/step - loss: 0.1007 -
val_loss: 0.1027
Epoch 195/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.1012 -
val_loss: 0.0985
Epoch 196/200
115092/115092 [==============================] - 17s 148us/step - loss: 0.1006 -
val_loss: 0.0982
Epoch 197/200
115092/115092 [==============================] - 17s 146us/step - loss: 0.1003 -
val_loss: 0.1237
Epoch 198/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.1009 -
val_loss: 0.1182
Epoch 199/200
115092/115092 [==============================] - 17s 147us/step - loss: 0.1007 -
val_loss: 0.1021
Epoch 200/200
115092/115092 [==============================] - 17s 143us/step - loss: 0.1004 -
val_loss: 0.1063
```

[30]:
```
Y_testB = model.predict(X_valB).flatten()
```

[31]:
```
Y_testB =np.round(Y_testB, 0)
```

[32]:
```
for i in range(30):
    YA= np.squeeze(np.asarray(Y_val))
    labelB =  YA[i]
    predictionB =  Y_testB[i]
    print("Estado: "+ np.array2string(labelB) + ", predicted:" + np.
 ↪array2string(predictionB))
```

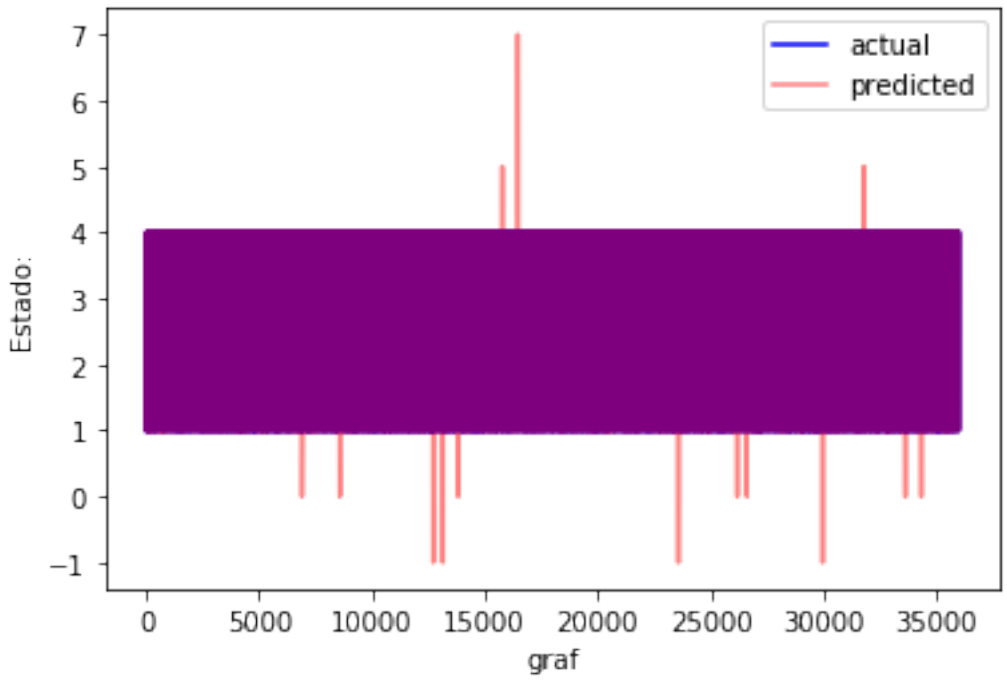
```
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 4., predicted:4.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
```

```
Estado: 1., predicted:1.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 3., predicted:3.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 2., predicted:2.
Estado: 4., predicted:4.
Estado: 2., predicted:2.
Estado: 1., predicted:1.
Estado: 3., predicted:3.
Estado: 1., predicted:1.
Estado: 1., predicted:1.
```

[33]:
```
r2 = r2_score(Y_testB, Y_val)
print (r2)
```

```
0.9340475520483761
```

[34]:
```
plt.plot((Y_val),
    color="b", label="actual")
plt.plot((Y_testA),
    color="r", alpha=0.5, label="predicted")
plt.xlabel("graf")
plt.ylabel("Estado:")
plt.legend(loc="best")
plt.show()
```



[35]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==Y_testB
accur=np.sum(Accu)
accur/len(Y_val)
```

[35]:
```
0.9520393694219701
```

Unimos los datos quedandonos con el minimo media y maximo para ver si con alguna de las opciones podemos mejorar los resultados

[36]:
```
YtA = np.asmatrix(Y_testA)
YtB = np.asmatrix(Y_testB)
ense1=np.zeros(len(Y_testA))
ense2=np.zeros(len(Y_testA))
ense3=np.zeros(len(Y_testA))
Yt=np.concatenate((YtA.T,YtB.T),axis=1)
for i in range(len(Yt)):
    ense1[i]=np.min(Yt[i,:])
    ense2[i]=np.max(Yt[i,:])
    ense3[i]=np.mean(Yt[i,:])

ense3 =np.round(ense3, 0)
```

[37]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==ense1
accur=np.sum(Accu)
accur/len(Y_val)
```

[37]:
```
0.9547362860399811
```

[38]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==ense2
accur=np.sum(Accu)
accur/len(Y_val)
```

[38]:
```
0.9691383768454417
```

[39]:
```
Y_val2 = np.array(Y_val.T)[0]
Accu=Y_val2==ense3
accur=np.sum(Accu)
accur/len(Y_val)
```

[39]:
```
0.964995690494064
```

[40]:
```
r2 = r2_score(ense1, Y_val)
print (r2)
```

```
0.9414155730076385
```

[41]:
```
r2 = r2_score(ense2, Y_val)
print (r2)
```

```
0.9546724540831304
```

[42]:
```
r2 = r2_score(ense3, Y_val)
print (r2)
```

```
0.960546544725396
```

[ ]:

[ ]:

[ ]: