Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Aplicación de técnicas de Machine Learning a un problema práctico de reposición de inventario.

Application of Machine Learning techniques to a practical problem of stock replenishment.

Daniel Rodríguez Suárez

La Laguna, 6 de julio de 2020

D. **José Luis Roda García**, con N.I.F. 43.356.123-L Profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. **Carlos J. Pérez González**, con N.I.F. 45.452.719-G Profesor Ayte. Doctor adscrito al Departamento de matemáticas, estadística e investigación operativa de la Universidad de La Laguna, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"Aplicación de técnicas de Machine Learning a un problema práctico de reposición de stock."

ha sido realizada bajo su dirección por D. **Daniel Rodríguez Suárez**,

con N.I.F. 51.165.316-E.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 6 de julio de 2020

Agradecimientos

A los profesores: José Luis Roda por poner en marcha todo este proyecto y sin el cual todo esto habría sido imposible y a Carlos J. Pérez por sumarse al mismo y aportar una gran cantidad de ideas y de conocimiento además de por haber estado siempre disponibles y haberme demostrado que todavía existe motivación en la enseñanza universitaria.

A la gente de TITSA: En especial a Ginés León, por abrirme las puertas al departamento de Data Science y Big Data de donde se ha obtenido todo el material de este proyecto y también a Alejandro Sánchez-Romo por su disponibilidad a la hora de proporcionar dicho material.

A mi gran amigo Miguel Jiménez el cual me ha descubierto gran parte de este mundo y, sin el cual este proyecto probablemente no existiría y además ha aportado ideas al mismo.

Y, por último, pero no menos importante a mi familia y a mis amigos por estar siempre ahí sobre todo en estos tiempos difíciles que nos está tocando vivir.

Licencia

Resumen

En este trabajo se explora el uso de nuevas tecnologías basadas en Inteligencia Artificial y Machine Learning, para un problema práctico de predicción de inventario en la compañía Transportes Interurbanos de Tenerife S.A (TITSA). Primeramente, se explorará el estado actual del problema y las soluciones propuestas para el mismo o para problemas similares. A continuación, se comparará la eficacia de los métodos no paramétricos basados en Inteligencia Artificial desarrollados mediante redes neuronales LSTM frente a métodos paramétricos tradicionales como el ARIMA. Además, se mostrará el proceso completo, como es el tratamiento de los datos para poder ser procesado por los algoritmos no paramétricos correctamente, las herramientas utilizadas como pueden ser las librerías de Python para el manejo de datos y las específicas para la creación de la Inteligencia Artificial y, también el uso del PowerBI para la representación y examen por parte del desarrollador, de los resultados alcanzados.

Palabras clave: Inteligencia Artificial, Big Data, Predicción, Inventario, ARIMA, LSTM, Python.

Abstract

This paper explores the use of new technologies based on Artificial Intelligence and Machine Learning, for a practical problem of inventory prediction in the company Transportes Interurbanos de Tenerife S.A (TITSA). Firstly, the current state of the problem and the proposed solutions for the same or similar problems will be explored. Next, the effectiveness of non-parametric methods based on Artificial Intelligence developed through LSTM neural networks will be compared with traditional parametric methods such as ARIMA. Furthermore, the complete process will be shown, such as the treatment of the data to be able to be processed by the non-parametric algorithms correctly, the tools used such as the Python libraries for the handling of data and the specific ones for the creation of the Artificial Intelligence and, also the use of PowerBI for the representation and examination by the developer of the results achieved.

Keywords: Artificial Intelligence, Big Data, Prediction, Inventory, ARIMA, LSTM, Python.

Índice

Índice de figuras

Índice de tablas

Capítulo 1
Antecedentes

1.1 Introducción

En los últimos años, los campos en las tecnologías relacionadas con el Big Data e Inteligencia Artificial han experimentado un crecimiento exponencial gracias a la aplicación práctica de las mismas.

Cada vez más las empresas de todo el mundo manejan más información, la cual antes era imposible de analizar y obtener algo valioso de ella. Sin embargo, hoy en día se ve cada más el potencial de esa información que antaño se consideraba inútil o inabordable.

En el caso de este Trabajo de Fin de Grado se tiene un problema que consiste en la optimización mediante técnicas de análisis estadístico y Machine Learning del almacenamiento de stock de las piezas de recambio en uno o varios de los almacenes de las guaguas de la compañía TITSA [1].

Si bien en el anteproyecto se había evaluado el uso de fuentes de datos abiertos (Open Data) diversas como meteorología, poblacionales, turismo, etc, se optó por el estudio sobre este conjunto de datos proporcionado por TITSA ya que es un problema práctico y al que se le pueden aplicar múltiples técnicas de análisis estadístico.

El objetivo era conseguir aprovechar todas las características posibles que ofrecían los datos de la compañía para los resultados de los modelos, pero tal y como se verá en este trabajo, el conjunto de datos de TITSA es inmenso y fue imposible incorporar características adicionales.

Eso se deja para trabajos futuros ya que TITSA ha demostrado bastante interés en este asunto debido a que hay un gran potencial en este tipo de tecnologías.

Una vez obtenidos los datos el problema consiste en la predicción con la mayor exactitud y el menor margen de error posible de la demanda de piezas a corto plazo según ciertos aspectos de los que dependan las guaguas en las que estén (fechas, rutas, temperatura, duración, tiempo, número de personas, modelo, marca, etc). Tratándose así de un problema de Big Data ya que uno de los objetivos será intentar utilizar esa información para lograr una predicción más exacta a partir de todos esos parámetros.

Al obtenerse estos resultados se puede generar una predicción del material necesario que se va a necesitar en un futuro gracias a lo aprendido debido a la demanda histórica. De esta forma, se evitan insuficiencias en el stock por falta de suministros y un ahorro en los costes al pedirse lo justo y necesario siempre dentro de los márgenes de seguridad.

Entre las dificultades presentes en este problema se pueden mencionar:

1. La gran variedad en las guaguas, las cuales llevan piezas distintas según marca y modelo.
2. Para cada tipo de guagua en sí, existe un gran número de piezas distintas. Esto sumado a lo anterior hace que el número de variables dependientes sea mucho mayor que el de variables independientes.
3. Hay piezas intercambiables entre algunos modelos y marcas las cuales se deberían tener en cuenta.

4. El histórico de los datos puede ser completamente inútil para algunos casos en concreto en los que la información del histórico no aporte absolutamente nada remarcable ni ningún antecedente estadístico. Por ejemplo, con la compra de guaguas eléctricas, híbridas, etc
5. Por último, hay piezas mucho más caras y difíciles de conseguir que otras. Dichas piezas tienen prioridad a la hora del encargo y, por lo tanto, del análisis.

A partir de los datos de un dataset (conjunto de datos) histórico donde los mismos están ordenados de forma cronológica y, además dicho orden es único e inalterable, se aplicarán modelos de series temporales, por lo que es que es importante sentar algunas bases de las mismas.

Por ejemplo, en la siguiente figura se muestra una serie temporal real que expone en toneladas la producción mensual de chocolate en Australia desde junio 1958 hasta diciembre 1990. Esta serie servirá para ver en qué se descomponen las mismas y qué componentes son más importantes.

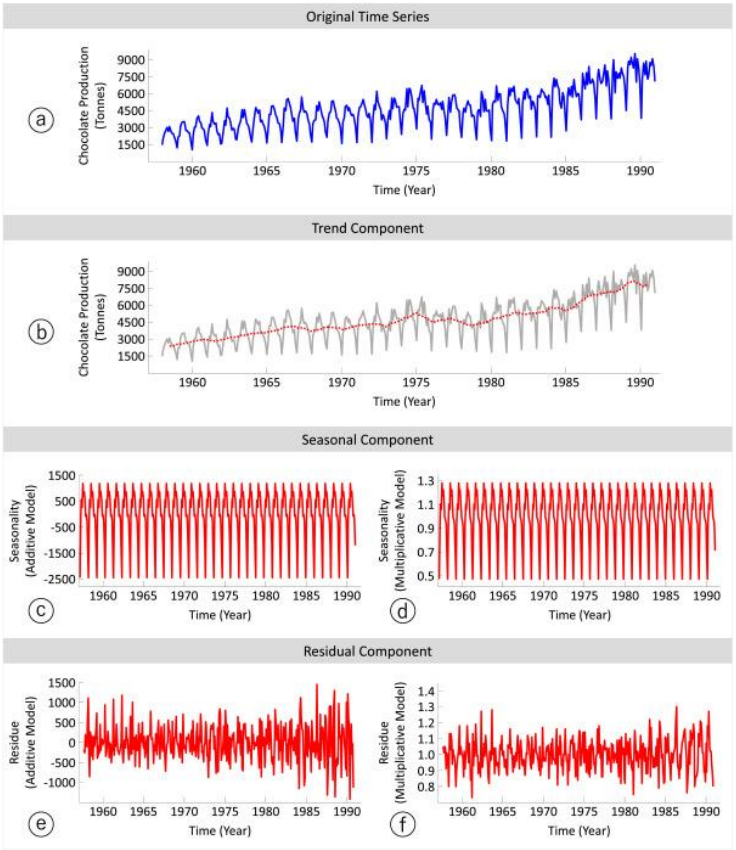


Figura 1: Componentes de la serie temporal en la producción de chocolate [2] De arriba hacia abajo se puede observar (a) la serie temporal original, (b) la tendencia, (c) y (d) la estacionalidad, (e) y (f) el ruido o componente residual.

Las series se pueden descomponer de forma aditiva, p.e. $Z_T = T_t + S_t + R_t$, o multiplicativa, p.e. $Z_t = T_t * S_t * R_t$

Tal y como se muestra en la figura 1:

- La tendencia (T) es un aumento o disminución a largo plazo de los datos que puede asumir una gran variedad de patrones (lineal, exponencial, polinomial). Se pueden encontrar series en tiempo real con una tendencia creciente en fenómenos relacionados con el desarrollo demográfico, el cambio gradual de los hábitos de consumo y la demanda de tecnologías en los sectores sociales.
- La estacionalidad (S) es la aparición de patrones cíclicos de variación que se repiten, a intervalos de tiempo relativamente constantes, junto con el componente de tendencia.

Ejemplos de patrones estacionales son el aumento en las ventas de aires acondicionados en verano y ropa abrigada en invierno.
El residuo (R) son las fluctuaciones a corto plazo que no son sistemáticas ni predecibles.

En este trabajo se intentará crear el mejor o los mejores modelos para intentar predecir el número de piezas de cara a un futuro a corto plazo.

1.2 Estado del arte

Durante más de medio siglo, los modelos estadísticos autorregresivos (AR) y modelos de medias móviles (MA) han influido en los campos de procesamiento y análisis de series temporales. Esto es debido a su gran adaptabilidad y facilidad a la hora de la aplicación sobre las mismas. [3]

Sin embargo, en los últimos veinte años, gracias a la expansión de las técnicas aplicables sobre el campo del data mining (minería de datos) y big data (conjunto masivo de datos), existe un interés creciente en la adaptación de los métodos de aprendizaje automático, especialmente aquellos que son de aprendizaje no supervisado y que no siguen las reglas de los modelos estadísticos más estandarizados. Se habla en este caso, de modelos no paramétricos capaces de procesar un gran volumen de datos y que cada vez son más fáciles de implementar gracias a los distintos lenguajes de programación y librerías adjuntas orientadas a ello, haciendo la tarea de la optimización del modelo lo más complicado siendo ya, la propia instalación algo banal que debe hacerse antes de la verdadera complicación matemática que supone. Aquí por supuesto hablamos de redes neuronales artificiales (ANN). [4].

Los investigadores de la comunidad estadística y de aprendizaje automático han contribuido a varios aspectos del proceso de predicción, como la asistencia en la selección del modelo más prometedor [5]. En muchos casos, el modelado no se realiza con una sola técnica, sino que se realiza con la hibridación de varias de las mismas [6]. Tal y como se puede ver en la siguiente Figura, las redes neuronales han conseguido muchísima atención frente a los tradicionales.

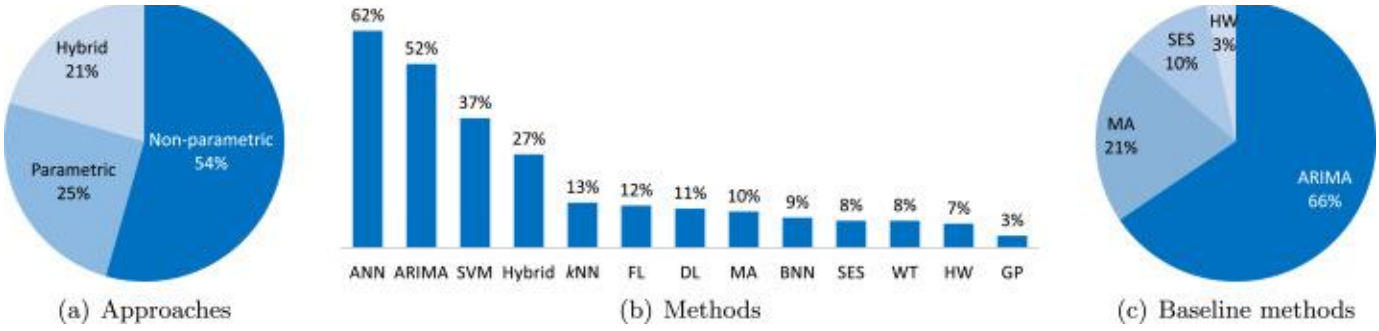


Figura 2: Sumario con los métodos más usados. Los acrónimos son: Artificial Neural Networks (ANN), modelos ARIMA – Autoregressive Integrated Moving Average (ARIMA) or Seasonal ARIMA (SARIMA) –, Support Vector Machines (SVM), k-Nearest Neighbors (kNN), Fuzzy Logic (FL), Deep Learning (DL), Bayesian Neural Networks (BNN), Simple Exponential Smoothing (SES), Wavelet Transform (WT), Holt-Winters (HW) models, and Gaussian Process (GP). [2]

1.2.1 Predicción en las series temporales

La predicción en las series temporales cubre 6 pasos, tal y como se ilustra en la figura 3:

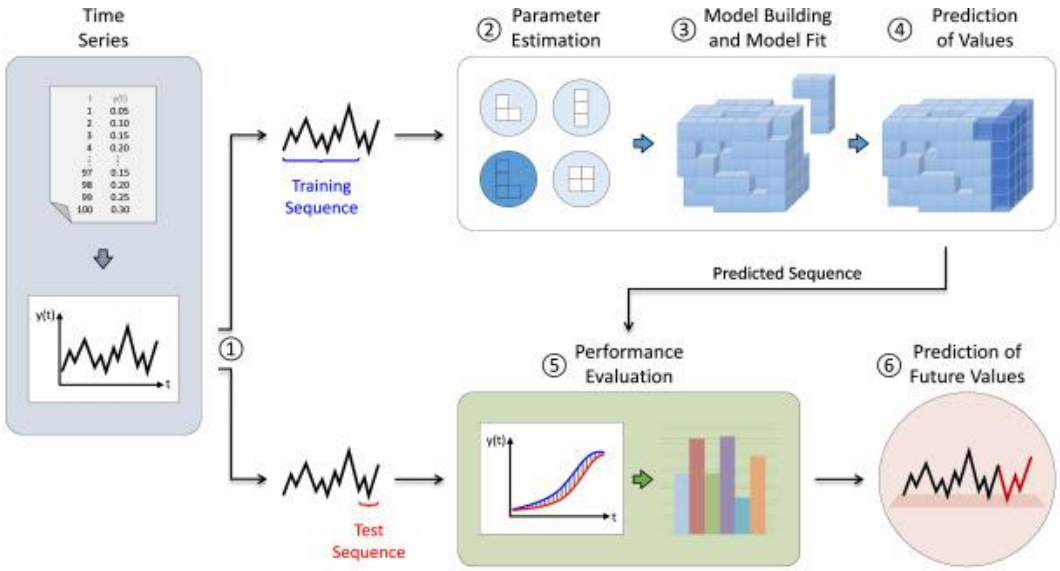


Figura 3: Proceso de predicción en las series temporales [2]

1. Se divide la serie temporal en dos secuencias: una antes de la predicción, que está destinada al entrenamiento del modelo *Training Sequence* (Secuencia de entrenamiento) y otro después de ese período *Test Sequence* (Secuencia de prueba o validación), que se utiliza para evaluar la calidad del modelo ajustado. La secuencia de entrenamiento debe ser, por definición, mayor que la de validación ya que gracias a ella el modelo podrá entrenarse.
2. Se elige la estructura del modelo en función de las características de los datos y de la propia serie temporal (estacionalidad, tendencia, etc.). El algoritmo recibe como entrada la secuencia de entrenamiento, que se subdivide en subsecuencias (muestras) para entrenamiento y validación, y un conjunto de parámetros predefinidos. En cada iteración, el algoritmo busca los valores de los parámetros que minimicen el error predictivo del modelo llegando así al valor óptimo posible (aquí se presentan varios problemas ya que según el algoritmo puede caer en óptimos locales o globales, pero normalmente eso es cuestión de no elegir el método correcto para el caso correcto). Para evitar problemas de sobreajuste (overfitting) y validación por memorización, es decir el modelo no aprende a interpretar, sino que simplemente está memorizando la estructura de los datos, durante el entrenamiento, se usa el conjunto de validación.
3. Se construye el modelo con los valores de los parámetros acomodados previamente y se ajusta a los datos de la secuencia de entrenamiento. Este modelo después se extrapola para las muestras de evaluación. Si los parámetros elegidos anteriormente son malos, se incrementará el error de predicción por lo que habrá que cambiar dichos parámetros.
4. Se elige la estrategia para predecir los valores de una serie temporal de varios períodos por delante. Dichos períodos por delante se conocen como horizontes de predicción h y, al ser varios por delante debe cumplirse $h > 1$.La estrategia que se utiliza es la de realizar diferentes pasos recursivos , donde la predicción de $h > 1$ se realiza h veces sucesivas considerando un modelo predictivo con h = 1. Cuando se realizan las diferentes iteraciones de los horizontes y se extrapola el modelo, se pueden utilizar las predicciones obtenidas en los distintos horizontes, aunque, hay que tener en cuenta eso sí, que son extrapolaciones y no son datos reales. En la figura 4 se puede ver un ejemplo sobre una red neuronal simple.

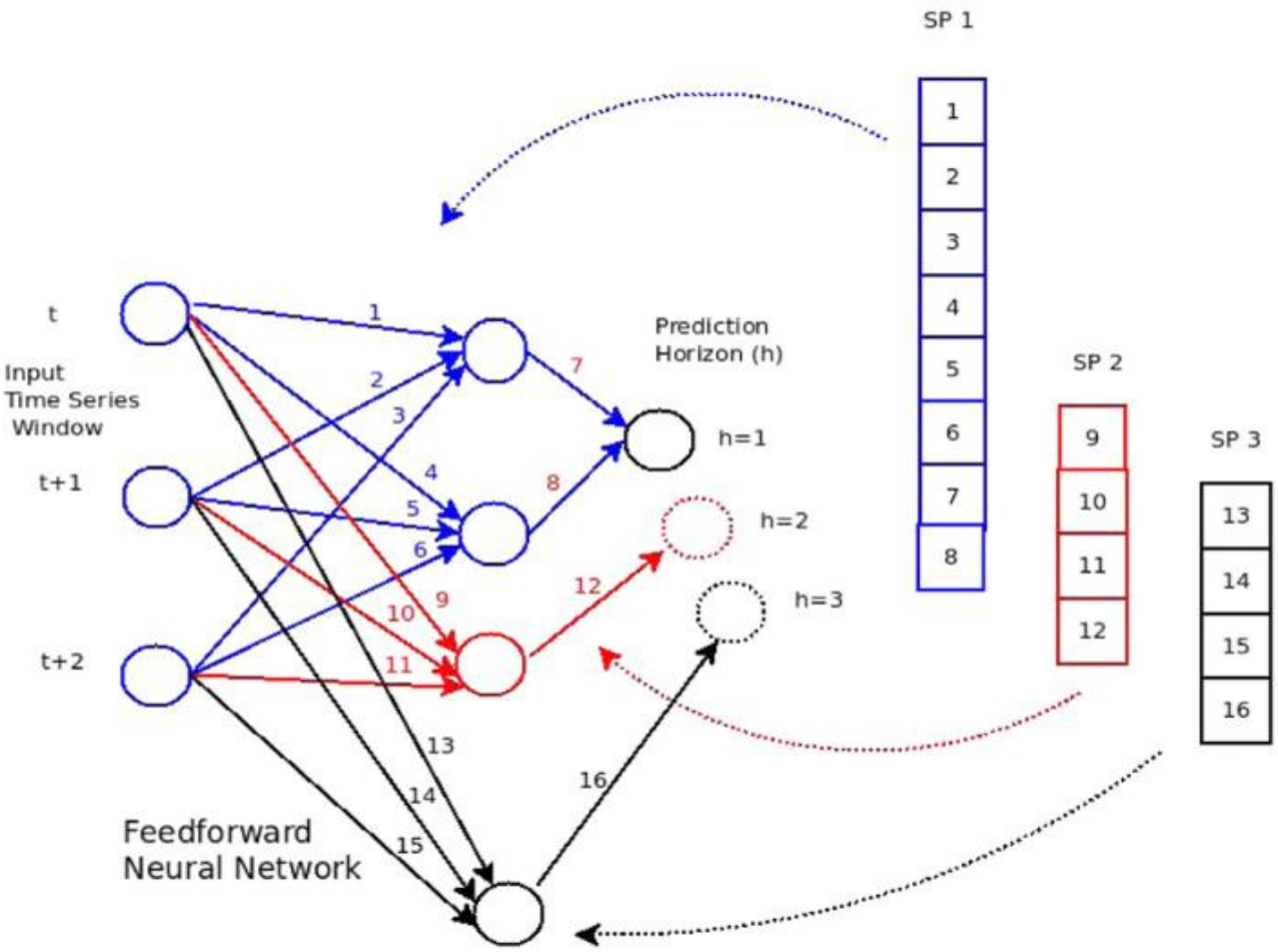


Figura 4: Transferencia de conocimiento en una red neuronal o ventana deslizante para diferentes horizontes de predicción (h). La Tarea 2 utiliza el conocimiento de la Tarea 1. El mismo concepto se utiliza para la Tarea 3 que utiliza el conocimiento de las tareas anteriores creándose así un procedimiento recursivo. [7]

5. Se comparan los valores pronosticados con la secuencia de validación (Test Sequence) para medir la precisión del modelo. Normalmente se suele usar el modelo que mejor se ajuste a este test set siguiendo las distintas medidas que existen para ello (MSE (Mean Squared Error), Curva ROC, F1 Score, etc).

6. Por último, se hacen predicciones para períodos futuros de la serie temporal. Al estar trabajando con datos reales es importante nunca perder de vista el error asociado, debido a que el modelo puede ser excelente durante un periodo de tiempo inicial corto, para la predicción de dichos valores hay diferentes técnicas, pero en este caso se usará la técnica expuesta recientemente, es decir, la transferencia recursiva de los valores en los horizontes de predicción predichos.

1.2.2 Métodos de predicción paramétricos

*"*Un modelo paramétrico es un modelo de aprendizaje que resume los datos con un conjunto de parámetros de tamaño fijo (independiente del número de ejemplos de entrenamiento). No importa la cantidad de datos que arroje a un modelo paramétrico, no cambiará de opinión sobre cuántos parámetros necesita." (Intelligence: A Modern Approach, 1994, p737*)*

Este tipo de algoritmos siempre implican ajustarse a la función objetivo.

Regresión lineal:

$$b_0 + b_1 * x_1 + b_2 * x_2 = 0$$

Donde $b_0 b_1 b_2$ son los coeficientes de la línea que controlan la intersección y la pendiente, y $x_1 x_2$ son dos variables de entrada.

A menudo, la forma funcional asumida es una combinación lineal de las variables de entrada y, como tal, los algoritmos de aprendizaje automático paramétricos también se denominan "algoritmos de aprendizaje automático lineal".

El problema es que la función subyacente real desconocida puede no ser una función lineal como una línea. Podría ser casi una línea y requerir una pequeña transformación de los datos de entrada para funcionar correctamente (función logística). O simplemente la función no se ajusta a una línea por lo que el modelo no se ajusta.

Algunos ejemplos más de algoritmos paramétricos de aprendizaje automático incluyen:

- Regresión logística
- Naive Bayes
- AR, MA, ARMA, ARIMA, SARIMA
- ARCH, GARCH

Las ventajas de los métodos de predicción paramétricos son:

- Más simples: estos métodos son más fáciles de entender e interpretar resultados.
- Velocidad: los modelos paramétricos son muy rápidos en lo que a rendimiento se refiere.
- Menos datos: no requieren tantos datos de entrenamiento y pueden funcionar bien incluso si el ajuste a los datos no es perfecto.

Desventajas

- Restringido: al tomar una forma para la función, estos métodos están altamente restringidos a la forma especificada.
- Complejidad limitada: los métodos son más adecuados para problemas más simples.
- Mal ajuste: en la práctica, es poco probable que los métodos coincidan con los datos reales de ajuste, en este caso series temporales.

Para las series temporales se pueden dividir los modelos en dos grupos de acuerdo a su complejidad matemática [8]:

- Suavizamiento exponencial.
- Los modelos SARIMA, ARCH y GARCH.

1.2.3 Modelos de series

Los *Autoregressive models* (AR) son modelos muy naturales y conceptos muy poderosos a la vez que simples, debido a que son extremadamente buenos a la hora de aprender solo con los datos que más correlación le proporcionan, identificando así mejor los patrones que afectan a la variable que se debe predecir y evitando el overfitting (sobreajuste) [10]. En un AR se pronostica la variable de interés usando una combinación lineal de los valores pasados. La fórmula es:

$$y_t = c + \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \cdots + \varphi_p y_{t-p} + \varepsilon_t$$

siendo y_t la variable de estudio (p.e., num. De piezas), φ_i los coeficientes del modelo de orden p AR de orden p (i=1,..,p) y ε_t es el término de error. Este tipo de modelos se usan cuanto la serie es estacionaria.

Por otro lado, se tienen los *moving-average models* (MA), los cuales tienen en cuenta los errores en las mediciones pasadas en vez de las proyecciones directas. La fórmula.

$$y_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

siendo μ la media de la serie, θ_1 los parámetros del modelo MA(θ) con j=1,..,q y ε_t los términos de error.

Week	Sales	3 MA	Error	|Error|	Error^2	|%Error|
1	39					
2	44					
3	40					
4	45	41	4	4	16	8.89%
5	38	43	-5	5	25	13.16%
6	43	41	2	2	4	4.65%
7	39	42	-3	3	9	7.69%
8		40	Total	14	54	34.39%
				MAD 3.5	MSE 13.5	MAPE 8.60%

Figura 5: Ejemplo de MA(3), proyectando la semana 4,5,6,7 y 8, junto con los cálculos del MAD, MSE Y MAPE, como se puede observar el MA consiste en tomar los elementos anteriores y proyectar la media sobre el elemento a predecir, al ser un 3MA se cogen los 3 elementos anteriores y se hace la media sobre ellos. En la semana 4 el primer 3MA realizado consiste en (40+44+39) /3 = 41

Los modelos de *suavizado exponencial* descomponen las series temporales en componentes cuyos valores se suavizan mediante pesos que decaen exponencialmente con el tiempo. Al final, una estructura aditiva o multiplicativa recompone los componentes suavizados para predecir valores futuros *[10]*, siendo su ecuación:

$$F_{t+1} = \alpha A_t + (1-\alpha)F_t$$

Week	Sales	Forecast	Error	Error^2
1	39	39.00		
2	44	39.00	5.00	25.00
3	40	40.00	0.00	0.00
4	45	40.00	5.00	25.00
5	38	41.00	-3.00	9.00
6	43	40.40	2.60	6.76
7	39	40.92	-1.92	3.69

Figura 6: Ejemplo de suavizado exponencial. A partir de las Sales (At) y las Forecast (Ft) y con un $\alpha = 0.2$ se puede hacer una previsión de la semana 8 la cual daría 40.54 ($F8 = 0.2(39) + (0.8)(40.92)$, también están calculados los errores y los errores cuadráticos.[44]

El problema de este tipo de modelo es que no van bien cuando hay algún tipo de tendencia en los datos, lo que es un inconveniente. En dichas situaciones se procede a realizar un doble suavizamiento exponencial, el cual es la aplicación recursiva del filtro 2 veces. Aunque se ha demostrado que en muchos casos el uso de este tipo de modelos es altamente ineficaz.

Para medir el error se usa el Median Absolute Deviation (MAD), Mean Square Error (MSE), Mean Absolute Percent error (MAPE) para obtener una medida del error.

Los modelos ARIMA son aquellos que implican tres procedimientos estadísticos: autorregresión (AR), integración(I) y *Moving Average* (MA). La autorregresión expresa la correlación entre observaciones, es decir, cuánto influyen los valores actuales en los siguientes. El procedimiento de integración indica el número de diferencias requeridas para garantizar la estacionalidad de la serie. Por último, la parte MA comprende factores desconocidos que no pueden explicarse por los valores pasados de series temporales. [2]

Los modelos autorregresivos de media móvil (ARIMA) son una combinación de los modelos AR y MA. Para ello se descompone la función en el modelo AR y el MA dándole el alcance deseado.

$$y'_t = c + \varphi_1 y'_{t-1} + \cdots + \varphi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

Donde p es el orden del AR y q es el orden del MA e $y'_t=(1-B)^d y_t$ una diferencia de orden d.

Una diferenciación de orden 1 sería:

$$y'_t = y_t - y_{t-1}$$

Una diferenciación de orden 2 sería:

$$y''_t = y_t - y_{t-1}$$

$$y'_t = y''_t - y''_{t-1} = (y_t - y_{t-1}) - (y_{t-1} - y_{t-2})$$

Un ARIMA de orden (p,0,0) es un AR, mientras que si es de orden (0,0, q) es un MA.

Por último, se también es usado en este tipo de problemas el modelo autorregresivo con heterocedasticidad condicional (ARCH), el cual, a diferencia de los ya vistos anteriormente, se ajusta a los errores utilizando un AR (Figura 10) que puedan surgir en la creación de un modelo, encontrando en ellos un patrón, (si se comete el error en un tiempo determinado es muy probable que se haya cometido el tiempo inmediatamente anterior también).

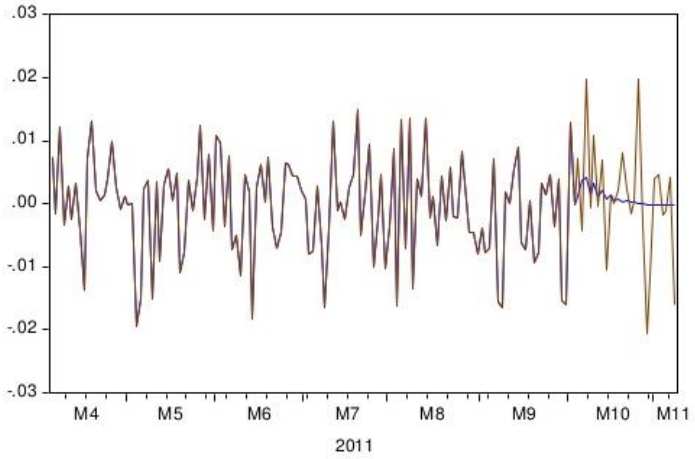


Figura 7: Representación de los residuos presentados en un modelo hipotético sobre una serie temporal, la representación es simplemente la resta entre el número real y lo estimado, se puede apreciar que los mayores períodos de volatilidad están al principio de cada mes, con algunas excepciones. [45]

El problema del ARCH es que necesita que el error al que se está adaptando sea más o menos constante, es por eso que se necesita un modelo algo mejor y es aquí donde entra el modelo de heterocedasticidad condicional autorregresiva generalizada (GARCH)(q,p), el cual a grandes rasgos usa un ARMA en lugar de un AR.

Debido a las condiciones expuestas grandes ventajas del mismo, se ha decidido usar un ARIMA como modelo paramétrico en la realización de este trabajo.

Tal y como se mencionó anteriormente. Cuando la serie temporal no es estacional, supone varios problemas para los modelos, estos son debido a que dejan de tener una media y varianza constantes a lo largo del tiempo, tendencia, p.e Figura 7.

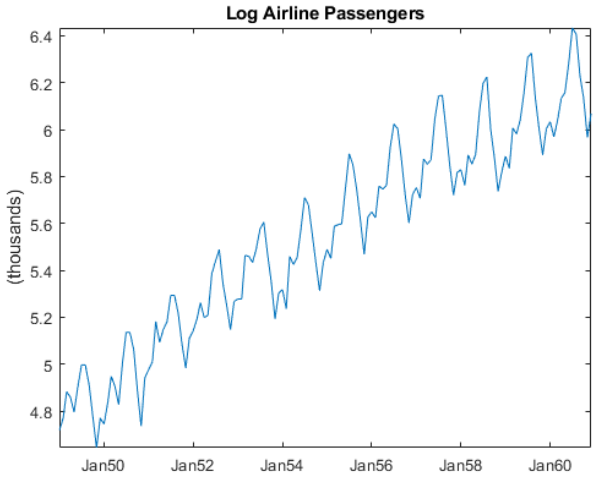


Figura 8: Ejemplo de una serie temporal no estacional donde la media es claramente creciente, aquí no se pueden usar ni MA, AR o ARMA. [11]

Sin embargo, en la Figura 7 exceptuando la media creciente la serie es estacional y es aquí donde entra el modelo (ARIMA), para saber la constante de crecimiento en este caso. Se aplica la diferenciación de orden d=1 a la serie de la figura 7 y queda el resultado mostrado en la figura 8.

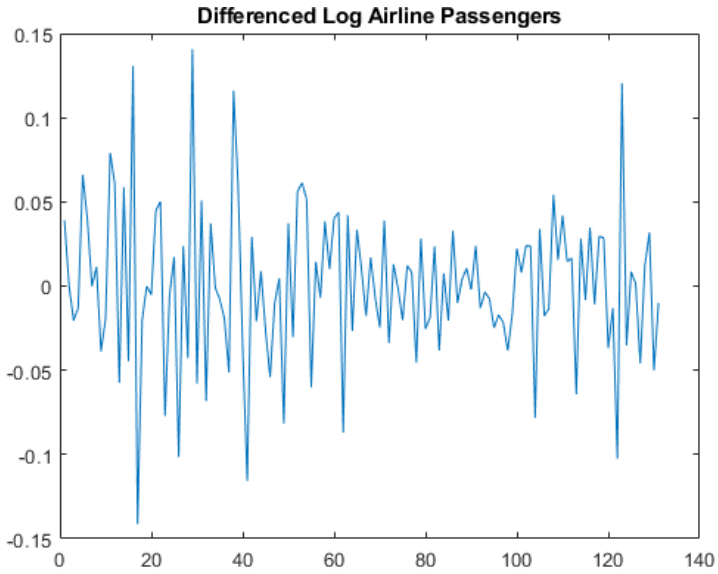


Figura 9: La diferencia entre las series es estacionaria. z_t describe dichas proyecciones [12].

Además, se tiene el ARIMA estacional (SARIMA) el cual aparte de todo lo anterior ya nombrado, también tiene en cuenta los patrones estacionales internos.

La forma de un SARIMA es $(p, d, q)(P, D, Q)m$ añadiéndose $(P, D, Q)m$ al modelo.

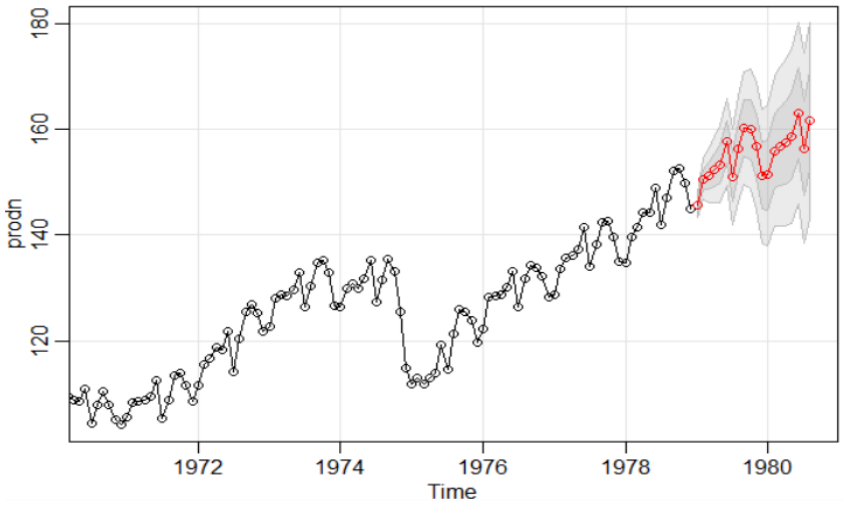


Figura 10: Proyección de SARIMA en los nacimientos entre 1978 y 1980 [13]

Lo más importante a destacar es el factor estacional m que tiene en cuenta la cantidad de muestras tomadas en un periodo definido como estación y lo tiene en cuenta junto con el resto de parámetros.

1.2.4 Métodos de Machine Learning no paramétricos.

*"*Los métodos no paramétricos son buenos cuando tienes muchos datos y no tienes conocimiento previo, y cuando no quieres preocuparte demasiado por elegir las funciones correctas." (Intelligence: A Modern Approach, 1994, p757)

Los métodos no paramétricos buscan ajustarse mejor a los datos de entrenamiento en la construcción de la función de mapeo, mientras mantienen cierta capacidad de generalizar a datos no existentes. Como tal, pueden adaptarse a una gran cantidad de formas funcionales.

Algunos ejemplos de métodos no paramétricos son:

- K-nearest Neightbors
- Árboles de decisión como C4.5
- Support Vector Machines (SVM)
- ANN (Artificial Neural Networks)

Ventajas:

- Flexibilidad: Capaz de adaptarse a una gran cantidad de formas funcionales.
- Poder: No hay suposiciones (o suposiciones débiles) sobre la función subyacente.
- Rendimiento: puede generar modelos de mayor rendimiento para la predicción.

Desventajas:

- Más datos: Normalmente se requieren muchos más datos de entrenamiento para estimar la función de mapeo.
- Más lento: mucho más lento para entrenar ya que a menudo tienen muchos más parámetros para entrenar, además de la dificultad para ajustarlos de los mismos.
- Sobreajuste: hay más riesgo de sobreajuste (Overfitting) en los datos de entrenamiento y es más difícil explicar por qué se hacen predicciones específicas.

1.2.5 Redes neuronales

Las ANN (Artificial Neural Networks) son modelos computacionales inspirados en el procesamiento de información realizado por el cerebro humano. El Perceptrón, exhibido en la Fig. 11, es la forma más simple de una ANN.

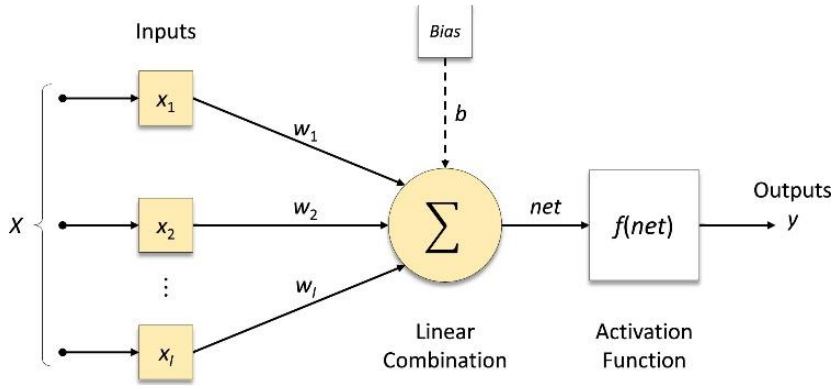


Figura 11: Estructura de un perceptrón.

La neurona única comprende l entradas de datos $xi \in X$. El i^{th} elemento de X, eventualmente proporcionado por las neuronas adyacentes, está asociado con un peso sináptico w_i. Este peso puede asumir un valor negativo o positivo que refleja la importancia de la entrada. La combinación lineal de entradas con pesos, agregada un umbral (sesgo) $b \in \Re$, da como resultado el valor neto $net = \sum_{i=1}^{l} wixi + b$.Este valor se envía a una función de activación f que establece la salida y de la neurona.

El sesgo tiene como objetivo corregir, aumentar o disminuir el valor neto. Esta corrección contribuye a lograr un resultado de $f(neto)$ más cercano al esperado. Además, podemos aplicar otros tipos de función de activación.

Un proceso de aprendizaje con un número finito de iteraciones ajusta los pesos sinápticos del Perceptrón. El aprendizaje se realiza mediante la regla de corrección de errores conocida como el algoritmo de convergencia de Perceptrón [14]. Este algoritmo busca un vector de peso w, de modo que se satisfagan las dos igualdades de la función de paso.[2]

Posteriormente se ideó el algoritmo de aprendizaje de Backpropagation [15] que sirve para combinar ANN con más de dos capas. La figura 12 muestra la estructura de un MLP (Multilayer perceptron) de tres capas. Los MLP tienen nuevas características como la capacidad de memorizar secuencias de datos y un mayor aprendizaje y adaptabilidad a los mismos.

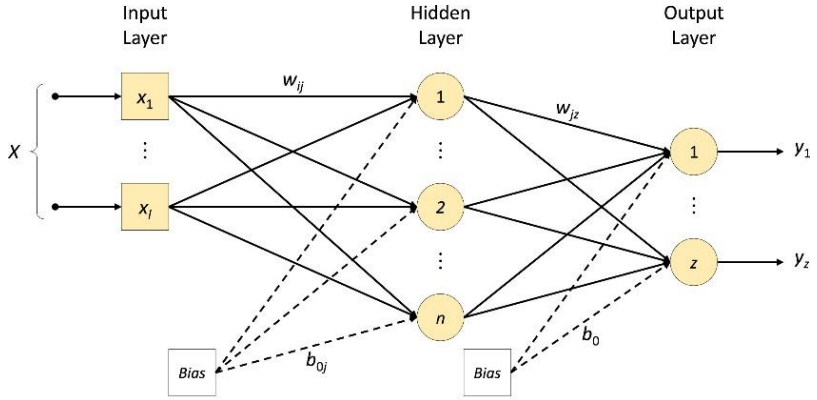


Figura 12: Estructura de un MLP de 3 capas, siendo éstas la entrada, la capa oculta y la salida.

Un MLP puede tener una o más capas de neuronas entre las capas de entrada y salida. Estas capas intermedias funcionan como una caja negra e interactúan entre sí combinando características siendo completamente invisibles al observador de su funcionamiento. Los modelos MLP son más difíciles de interpretar, sin embargo, poseen importantes ventajas como la de poder manejar grandes volúmenes de datos y adecuarse mejor a los mismos, lo cual, al mismo tiempo es su limitación ya que normalmente si no se posee un extenso volumen de información, el MLP no se logrará adaptar adecuadamente y por lo tanto el modelo no será adecuado.

Podemos clasificar los modelos Perceptrón y MLP como redes neuronales feed-forward (de avance) porque en ellas las señales de neurona a neurona fluyen solo en una dirección: de entrada, a salida, el problema de este tipo de redes es que no son buenas memorizando información y es por eso que han surgido nuevas propuestas como las RNN (Recurrent neural network).

En las RNN las conexiones entre las neuronas forman un ciclo y las señales pueden moverse en diferentes direcciones creándose así un proceso de retroalimentación a corto plazo. Este tipo de redes son especialmente útiles para clasificar y predecir voz, o donde la entrada y salida de los datos son dependientes, como en las series temporales. El problema que presentan este tipo de redes frente a las series temporales es que su capacidad de memorización es demasiado corta y no tiene en cuenta datos que poseen más correlación como se ya se ha demostrado en los modelos de predicción paramétricos. Aun así, son ampliamente utilizadas y existen varias versiones mejoradas como las RNN bidireccional (BRNN) [17] y RNN autorregresivo no lineal con entradas exógenas (NARX) [18].

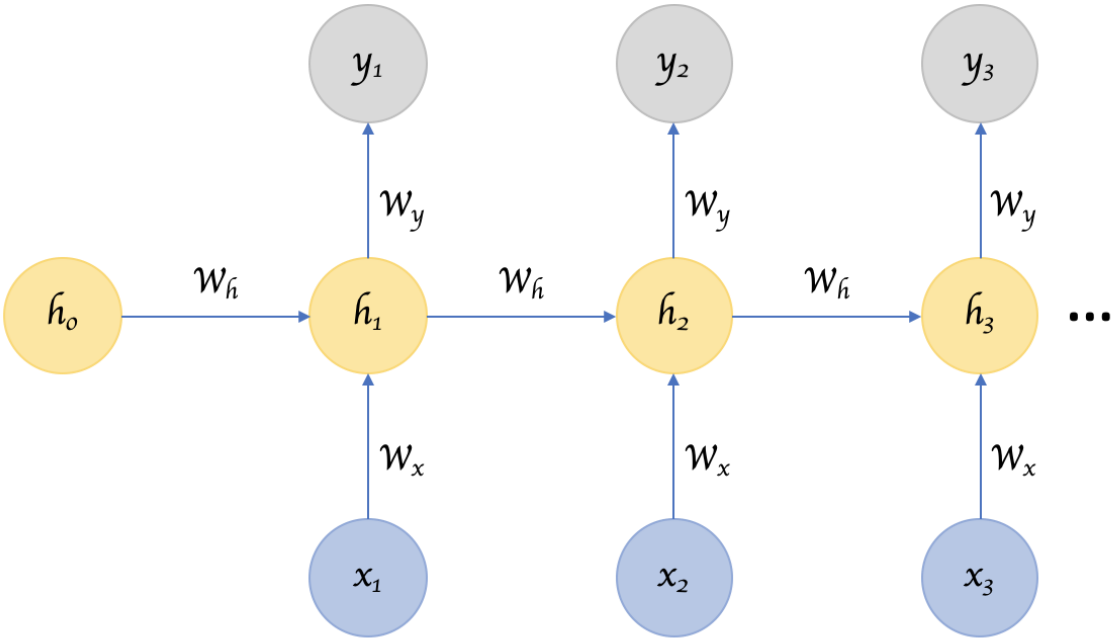


Figura 13: Representación visual de una RNN con un estado oculto que debe transportar información pertinente de un elemento de entrada en la serie a otras partes de la red. [16]

Aunque teóricamente poderosos, los modelos recurrentes mencionados anteriormente fueron ampliamente considerados difíciles de entrenar debido al problema del gradiente de fuga [19], que grosso modo consiste en que a medida que se van agregando más capas intermedias a la ANN que utilizan ciertas funciones de activación, las funciones de gradiente de pérdida se acercan a cero, lo que hace que la red sea difícil de entrenar.

La solución más simple a esto pasaba por cambiar la función de activación de una sigmoide a una RELU o rectificador, el problema de esto es que se pierde información en la entrada.

La segunda solución es usar redes residuales las cuales proveen de conexiones directas a las capas posteriores evitando así las funciones de activación y, por tanto, evitando también el problema del gradiente de fuga, tal y como se muestra en la figura 14.

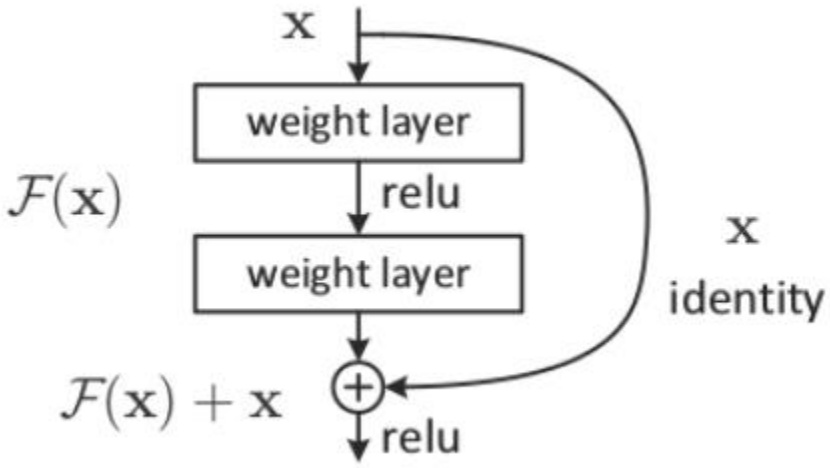


Figura 14: representación de un desvío residual.

Finalmente en 1997, fueron introducidas las redes Long Short Term Memory (LSTM) [20] cuyo esquema se puede ver en la figura 15, las cuales solucionan el problema del gradiente de fuga y, además son capaces de memorizar a más largo plazo que las RNN ya que evitan el problema de la dependencia a largo plazo lo que las hace extremadamente buenas para las series temporales además de ser un esquema que aplica muy bien el algoritmo de "Backpropagation"el cual es un algoritmo que dada una red neuronal artificial y una función de error, el método calcula el gradiente de la función de error con respecto a los pesos de la red neuronal el cálculo del gradiente se realiza hacia atrás a través de la red, con el gradiente de la capa final de pesos calculado primero y el gradiente de la primera capa de pesos calculado el último. Esto siempre en redes con varias capas.

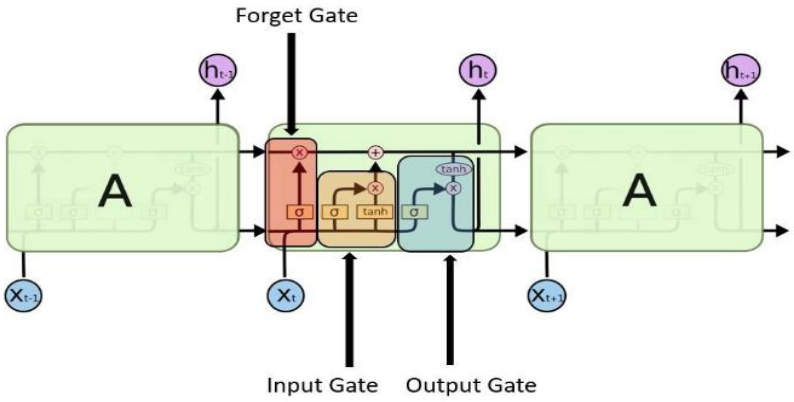


Figura 15: Representación esquemática de una red LSTM [21]

Tal y como se aprecia en la figura 15:

1. Puerta de entrada (Input Gate): decide qué valor de la entrada deben actualizarse. La función sigmoide decide qué valores dejar pasar 0,1. y la función $tanh$ pondera los valores que se pasan decidiendo su nivel de importancia que varía de -1 a 1.
2. Puerta de olvido (Forget Gate). Esta puerta decide qué información se debe desechar o guardar. La información del estado anterior se pasa por una función sigmoidea y se obtienen valores entre 0 y 1, cuanto más cerca del 0 más probabilidades hay de olvidar y cuánto más cerca del 1 más de mantener el dato.
3. Estado de la celda (Cell State): Ahora se debería tener suficiente información para calcular el estado de la celda, para ello el estado se multiplica por el resultado obtenido en la puerta de entrada después de haber pasado la función sigmoidea, dando la posibilidad de obtener valores del estado de la celda si se el valor de la puerta de entrada es cercano a 0. Posteriormente se coge el valor de la entrada y se hace una suma puntual con los valores que la red neuronal encuentra relevantes, que son los que van a determinar el nuevo estado de la celda.
4. Puerta de salida (Output Gate): Esta puerta decide cual debería ser el próximo estado oculto, es decir, a comunicarse con la siguiente neurona, este estado contiene información sobre las entradas anteriores y además se usa para realizar predicciones. Se pasa el estado oculto por una función sigmoide y luego se pasa el estado de la celda recién modificado en el paso anterior por la función $tanh$. Se multiplica la salida de $tanh$ con la salida sigmoidea para decidir la información del estado oculto, y la salida es el estado oculto, por tanto se transfiere a la siguiente neurona el nuevo estado de la celda obtenido en el paso tres y el nuevo estado oculto.

En la Figura 16 se muestra la eficacia de una una red LSTM frente a un problema de series temporales real.

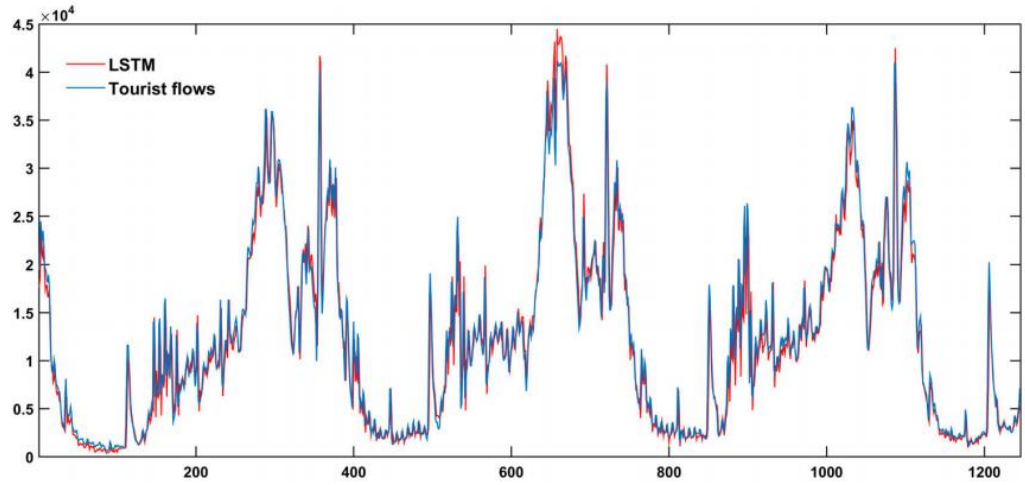


Figura 16: Demostración de la eficacia del modelo frente a un problema real [22].

Además, se tienen las LSTM Bidireccionales las cuales son una extensión de las LSTM "tradicionales" que pueden mejorar el rendimiento en problemas de clasificación/predicción de secuencias (como son las series temporales).
La idea detrás de las mismas es sencilla, se duplica la primera capa recurrente en la red para que ahora haya 2 capas, una al lado de la otra tal que se le proporciona la secuencia de datos original a la capa de entrada y una copia invertida a la segunda capa.
Por lo tanto, mientras que la LSTM "tradicional" solo conserva información del pasado, la LSTM bidireccional al ejecutar sus entradas de dos maneras, puede llegar a conservar información del "futuro", haciéndolas mejores para entender el contexto en el que se están ejecutando.

Tal y como se muestra en esta situación hipotética
"*Digamos que tratamos de predecir la siguiente palabra en una oración, en un nivel alto lo que verá una LSTM unidireccional*
Los muchachos fueron a ...
E intentaremos predecir la siguiente palabra solo por este contexto, con LSTM bidireccional podrá ver información más adelante, por ejemplo
LSTM unidireccional:
Los muchachos fueron a ...
LSTM Bidireccional: (usando información del futuro)
... y luego salieron de la piscina
Puede ver que usar la información del futuro podría ser más fácil para la red entender cuál es la siguiente palabra." [23]

Por último, un tipo de red ampliamente usado para este tipo de problemas son las autorregresivas exógenas no lineales (NARX) y también las autorregresivas no lineales (NAR)

"NARX es una arquitectura neural recurrente ampliamente utilizada para modelar datos secuenciales. Se argumenta que las redes NARX no solo son equivalentes a RNN en el sentido de la capacidad de modelado, sino que tampoco sufren la desventaja de las RNN del problema del gradiente de fuga. Las principales ventajas de las redes NARX son la convergencia rápida junto con una buena capacidad para los problemas con las dependencias a largo plazo" [24]

En la Fig. 17 se muestra una arquitectura de una red NARX general.

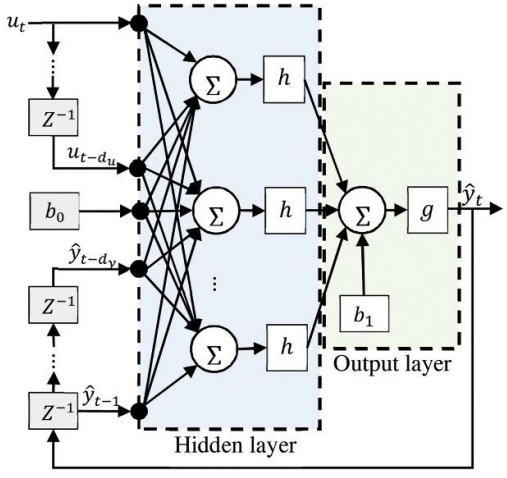


Figura 17: Arquitectura de una red NARX.[24]

Sin embargo, en términos de largas dependencias de tiempo, hay indicios de que las LSTM tienen mejores resultados en las predicciones [25]

Es por ello que en este trabajo no se le dará importancia y se trabajará con la estructura LSTM.

Capítulo 2
Objetivos, fases y desarrollo del proyecto.

Como objetivo principal, se debe intentar predecir el número de piezas de determinado tipo en los almacenes de TITSA para así intentar ahorrar costes en la compañía. Para ello, se han seguido una serie de pasos que normalmente se dan en este tipo de problemas. Dichos pasos se describirán a continuación con el mejor detalle posible. Se debe tener en cuenta que las predicciones a buscar por la compañía son dinámicas y, por lo tanto, los modelos deberían ser adaptables y con predicciones a corto plazo.

Además, se deben intentar mostrar todos los resultados de forma que cualquier usuario externo logre entender los mismos, es decir, con gráficas, información recurrente y simple etc.

2.1 Obtención de los datos, observación de los mismos, representación y tratamiento.

2.1.1 Observación inicial de la muestra.

En primera instancia, se proporcionó como conjunto de datos de base para el examen de los mismos 8 archivos .csv (archivos de valores separados por comas) de un máximo de 3000 registros para un simple examen visual y ver los valores que más peso debieran tener. Cada archivo representaba una pequeña base de datos con algo de relación con el problema en cuestión.

Se tiene:

- *CierresAlmacén*: Tabla que registra mes a mes el conteo de las piezas en los distintos almacenes de la compañía, así como otras características relevantes, como el precio, la familia de la pieza, el Almacén en el que está etc.
    - Las características más relevantes son la Fecha del conteo, la Descripción de la pieza (el nombre de la pieza), la Familia de la pieza, el precio, el Almacén y, por supuesto, el número de existencias
- *Movimientos*: Tabla que registra diariamente los movimientos que hay entre los almacenes, para ello, lleva registro de las piezas al final del día entre cada almacén y del número de que se desplaza o sale de los mismos.
    - Las características más importantes de esta tabla son las mismas que las de la anterior, con el añadido de que ahora contamos con información diaria, y con una nueva variable llamada Vehículo, que representa para cada guagua un ID único que significa que al cierre dicha guagua estaba en ese almacén, dicho ID es un número de 4 cifras entero y único.
- *PartesAvería*: Tabla que registra las fechas de todas las averías con la descripción de las mismas, así como el taller donde se realizó el parte.
    - Las características más relevantes de esta tabla son: la Fecha del parte, el vehículo por el cual se realizó el parte, el taller, así como el Diagnóstico, el cual es un reporte por parte del chófer o mecánico del problema que ha surgido.

- *Vehículos*: Tabla con muchísimas características de cada vehículo.
    - Al ser una tabla con casi 100 características describiendo cada vehículo, realmente ninguna es más importante que otra, sin embargo, esta cuestión se abordará con más profundidad en el apartado 5.0.
- *Kilómetros*: Tabla que va registrando distintas medidas de kilómetros en los vehículos con el ID descrito anteriormente.
    - En esta tabla solo hay 3 características las cuales son importantes que son, la fecha de la medida, el vehículo y los kilómetros medidos.
- *Vehículos_Km_litros*: Esta tabla es como la anterior solo que contiene información adicional, como medidas de C02, N20, etc.
    - Además de las características mencionadas en la tabla anterior, añadir los litros de consumo totales en el margen de kilómetros de medición entre fecha y fecha.
- Por último, tenemos 2 tablas que sirven para añadir información pero que por sí solas no sirven de nada.
    - *VehículosSeries*: la cual nos indica las series de vehículos existentes y que han existido en TITSA, ya que los mismos se compran por lotes.
    - *AlmacénAhora*: Sigue la misma estructura de *CierresAlmacén* pero solo muestra los almacenes en su estado actual, es decir el último día de registros, se va actualizando constantemente en los servidores de Titsa, en nuestro caso, es igual a los últimos registros de *CierresAlmacén* ya que no tenemos acceso a los servidores sino que tenemos una copia hecha a 29 de mayo de 2020.

Una vez examinados los datos y obtenidas las características más importantes ya se podía montar la base de datos creando las relaciones entre las mismas quedando el esquema tal y como se muestra en la siguiente figura:

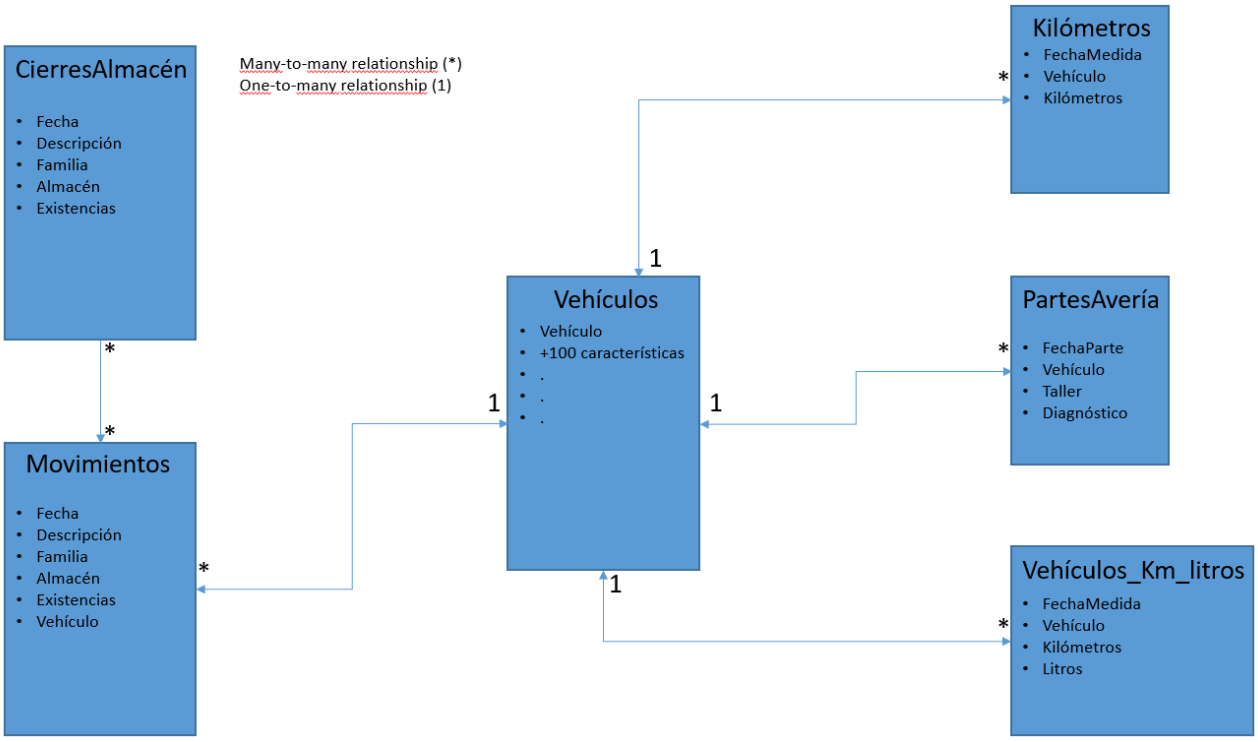


Figura 18: Esquema final de la base de datos proporcionada

Nota: Las características descartadas de las bases de datos se han hecho en base al puro reconocimiento básico, es decir, se han eliminado aquellas variables que eran obviamente inútiles o que eran reiterativas y no aportaban información, (*Vehículo – Matrícula*) siendo matrícula la descartada, ya que reincidía con la misma información también se han eliminado características completamente inútiles como los nombres de los conductores, ID de documentos etc.

Igualmente es necesario un examen mucho más profundo de los datos para obtener información de los mismos y ver con que se construye el modelo.

2.1.2 Exploración y representación del conjunto total

Una vez hecha la observación inicial ya se puede proceder a un examen más minucioso. Se recuerda que el objetivo es intentar pronosticar el número de cierto tipo de piezas para lo cual, es importante decidir que pieza o piezas vamos a utilizar en este caso, ya que debe ser una pieza abundante a lo largo de los años, es decir, que se use con frecuencia y sea importante para la compañía y que, a poder ser esté presente en varios modelos y/o marcas, y además que sea una pieza con cierto valor con respecto a la cantidad, para que así el modelo sea lo más útil posible a la compañía.

Se ha decidido este criterio debido a que tal y como se mencionó en los Antecedentes uno de los problemas es la gran diversidad de piezas que hay, en concreto, aproximadamente 23000 tipos de piezas distintas.

El conjunto total de los datos abarca según la base de datos desde los años 80, 90 o 2000, por lo que, para ponerlo todo en conjunción y que quede un histórico fiable sin demasiado ruido en la serie temporal (ya que las piezas de los años 90 es extremadamente probable que no se vuelvan a usar a día de hoy ni se pidan más), se tomó como fecha inicial común el 1 de enero del año 2008, hace aproximadamente unos 12 años.

Las tablas pues, pasan a tener muchos más registros que los 3000 de muestra, ahora se tienen:

- *CierresAlmacén*: 2 millones 100 mil registros aproximadamente.
- *Movimientos*: 5 millones de registros aproximadamente.
- *PartesAvería:* 320 mil registros aproximadamente.
- *Vehículos:* 1810 registros exactamente (recordemos que describe cada vehículo y este tiene un identificador único).
- *Kilómetros:* 164 mil registros aproximadamente.
- *Vehículos_Km_litros:* 45 mil registros aproximadamente.

Debido a que la empresa usa habitualmente la herramienta PowerBI [26] para este tipo de tareas, se utilizó la misma en conjunción con otras las cuales se irán describiendo a continuación.

2.1.3 Representación en PowerBI.

PowerBI es un programa desarrollado por Microsoft extremadamente enfocado en representaciones de grandes conjuntos de datos, especialmente para el ámbito empresarial por lo que es fácil de manejar para alguien que no sea entendido en el tema y, además permite generar gráficas fácilmente interpretables lo cual es uno de los objetivos de este proyecto. Además, permitirá realizar un escrutinio más meticuloso de los datos y observar variables que a simple observación son imposibles de ver como la evolución de ciertas cantidades en el tiempo, precios, en los almacenes por separado, y lo más importante, nos permite, comparar.

El interés se debe centrar en las piezas, es por eso que es ahí donde va a caer el mayor peso del análisis y, por consecuente de la representación.

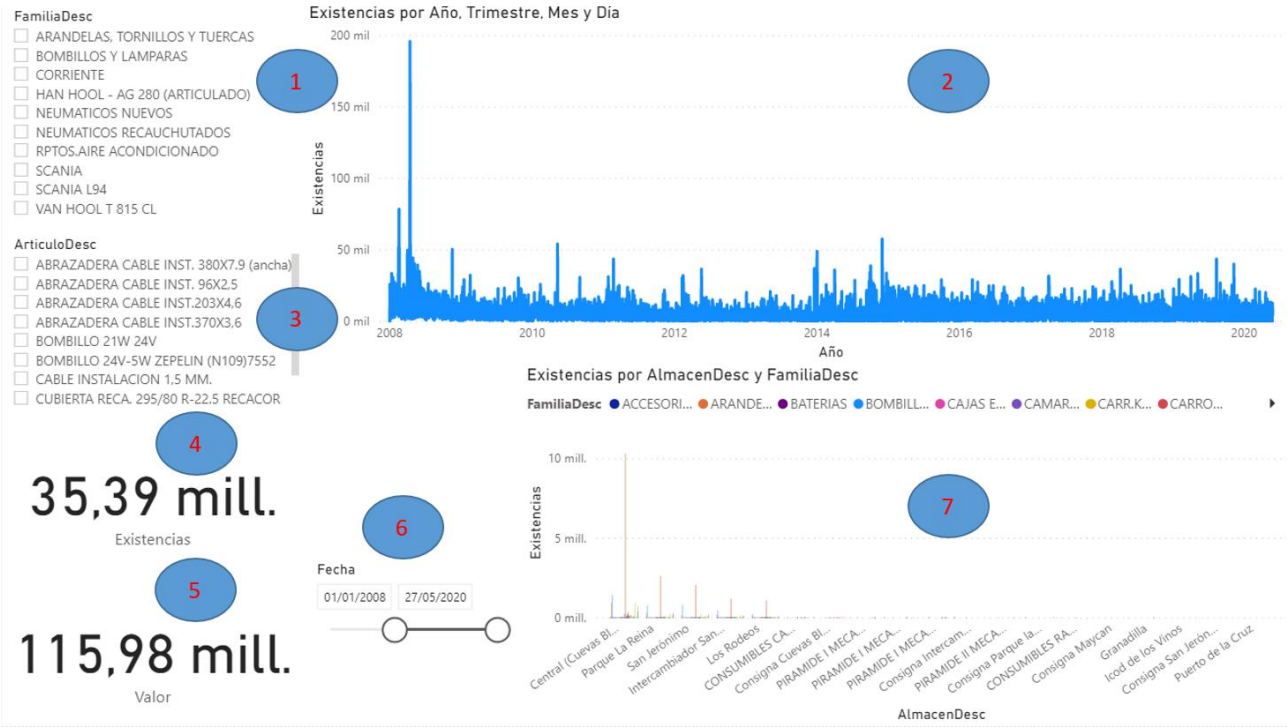


Figura 19: Estructura utilizada para la representación de la base de datos Movimientos en PowerBI por numeración se puede observar: 1: Filtros por familias de piezas, 2: Gráfico de líneas que muestra las existencias comprendidas (Eje Y) en el periodo mostrado en el eje X, 3: Filtro por pieza individual, 4: Contador de todas las piezas que existen y que se representan en el gráfico en (2), 5: Contador con el valor total de las piezas arriba mostradas, 6: Filtro para las fechas, como antes se mencionó se toman desde el 1 de enero de 2008, 7: Existencias según almacén.

El programa tiene la ventaja de ser extremadamente dinámico, tanto es así que muestra los resultados de cada barra con solo poner el ratón encima y de mostrar los resultados al momento aún con una gran cantidad de datos tal y como se muestra en la siguiente figura: (Nota: estos resultados tienen filtros aplicados explicados en la siguiente página).

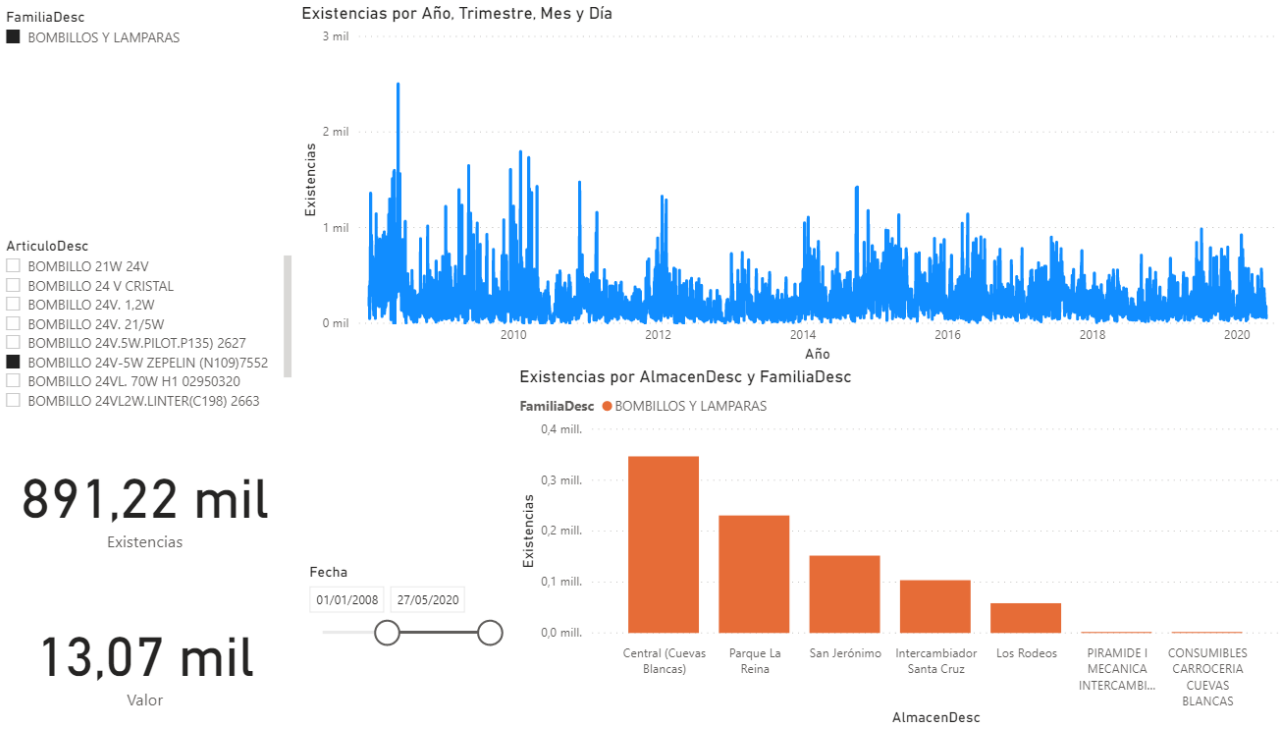


Figura 20: Representación exacta de la misma estructura anterior, pero ahora hemos seleccionado que solo se muestren las piezas de la Familia "Bombillos y Lámparas" y en concreto los Bombillos 24V-5W ZEPELIN (N109)7552

Gracias a las observaciones en PowerBI se descubrieron una gran cantidad de errores en la base de datos, como valores nulos, piezas que no estaban contabilizando correctamente, campos con nombres repetidos, pero con distintos valores ortográficos, así como campos totalmente inútiles para el análisis. Sin embargo, lo más importante a tener en cuenta fueron 3 factores:

- En las piezas se estaban contabilizando los litros de combustible, lubricantes, bonos y demás útiles de personal como camisetas, que son completamente inservibles y, además trastocan los números ya que, solo los combustibles suponían casi el 95% de los datos de la tabla.

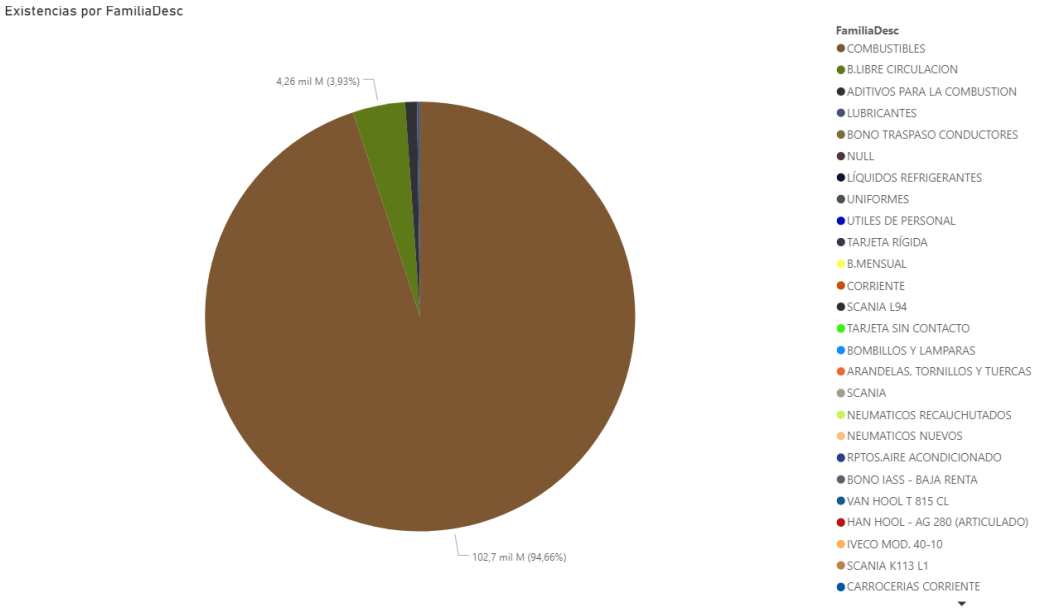


Figura 21: Familias de la tabla movimientos sin ningún tipo de filtro, como se puede observar, hasta llegar a la primera familia que contiene piezas hay 11 familias que nos son completamente inútiles.

- Además, también se descubrió que, aunque hubiera una gran variedad de piezas (unas 23000 tal y como se mencionó anteriormente) muchas de las mismas, ya estaban en desuso o simplemente eran piezas con uso demasiado puntual y, por lo tanto, imposibles de predecir ya que su histórico es extremadamente limitado e irregular debido a que su número es extremadamente bajo a lo largo de los años, es por ello que se llegó a la conclusión de que lo más útil a modelar y representar eran aquellas 10 familias con más piezas en total y, a su vez en cada familia las 10 piezas más comunes, quedándonos un top 100 de piezas con mayor número de existencias.

  El problema que presentaba esto es que las piezas más comunes son extremadamente baratas y puede que para la compañía no suponga demasiado rentable hacer un modelo de predicción sobre las mismas porque se compran en lotes extremadamente grandes no de 1 en 1 (los tornillos) así que se llegó a la conclusión de que también era relevante tener un top 100 precios de las piezas, realizando así un baremo entre la cantidad de piezas y el precio de las mismas se podría elegir una pieza abundante pero a la vez lo suficientemente cara.

- Por último, era interesante elegir una o varias piezas con series temporales con cierta regularidad, es lógico que probablemente ninguna tenga una estacionalidad marcada o tendencia, pero por lo menos que no deje de existir de un año para otro y vuelva a aparecer, es decir, una pieza con un uso más o menos regular desde el 1 de enero de 2008.

- Es por ello que se eligieron:
  - Familia: Neumáticos nuevos (Cubierta nueva 275/70 R-22.5 Metropolitano/urbano) con un valor total de 665.950 mil euros y 50.520 existencias.

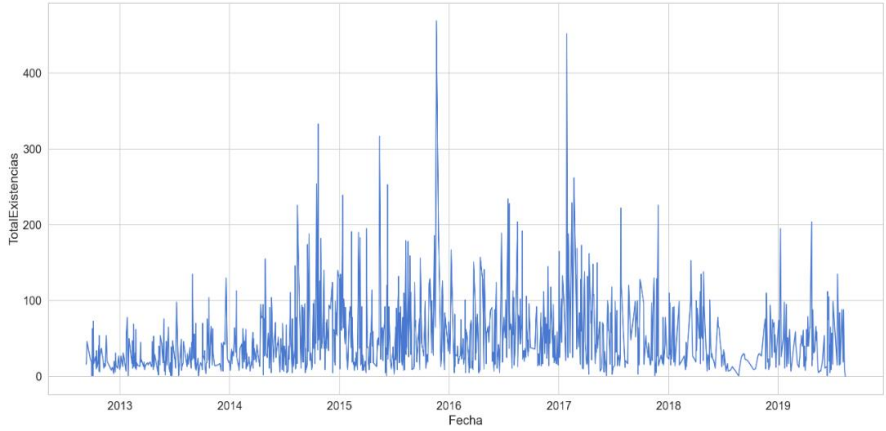


Figura 22: Existencias de neumáticos desde que hay registros (2012)

  - Familia: Bombillos y lámparas (Bombillo 24V-5W ZEPELIN (N109)7552 con un valor total de 13.070 euros y 891.220 existencias.

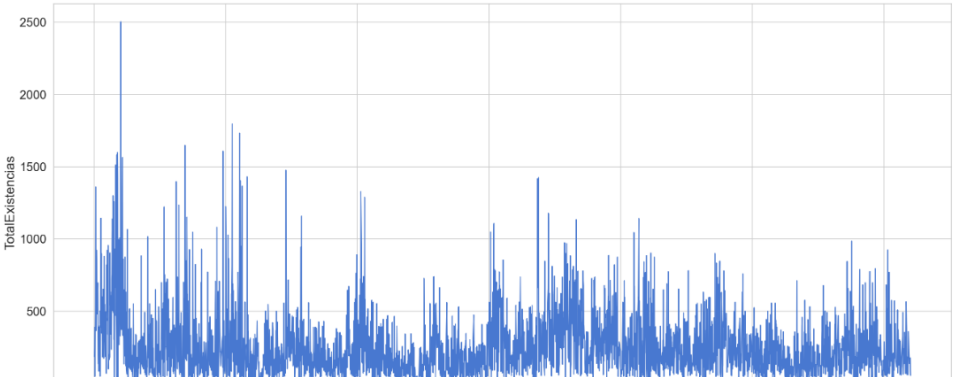


Figura 23: Existencias de Bombillos desde 2008.

2.1.4 Pre Tratamiento de los datos

Una vez elegidas las variables a modelar en un futuro era importante elegir un workbench (banco de trabajo) adecuado para el posterior trabajo.

Para empezar, hay que hablar sobre los archivos .csv, ya que, en general son adecuados cuando hablamos de poca información (unos cuantos miles de registros) o la misma no se encuentra en muchos de los mismos, en este caso tenemos unos archivos .csv de los cuales el más grande pesa 1GB, que, aunque no es nada comparado con proyectos de Big data a gran escala, siempre conviene más trabajar con herramientas más orientadas a este tipo de problemas y es aquí donde entra Hadoop [27], Sqoop [28] y SQLite [29].

Hadoop a grandes rasgos es un framework que permite trabajar con grandes volúmenes de datos gracias a su propio sistema de archivos el Hadoop Distributed File System (HDFS), sin embargo, en este caso, se va a usar junto a Sqoop, el cual es una pequeña herramienta diseñada para Hadoop que su principal función es la de transferir datos entre HDFS y bases de datos relacionales, para convertir los .csv a .sql y así usar el gestor de base de datos de SQLite con una mejora de rendimiento bastante considerable.

Una vez realizada la conversión se tendrá la base de datos en SQLite donde se podrán realizar consultas en SQL a mucha mayor velocidad que si se importaran los archivos .csv directamente. Todo este proceso parecerá algo confuso y tedioso, pero ahora cobrará sentido.

A continuación, ya se podrá proceder al tratamiento en sí de los datos, para ello es esencial contar con un lenguaje de programación que cuente con las herramientas necesarias para ello y por eso que se ha elegido Python [30], ya que cuenta con una librería para el manejo de datos extremadamente eficiente, Pandas [31]. Además, Python, cuenta con otras librerías que se usarán durante el proceso Numpy[32] para el manejo de los vectores, arrays y dimensiones, Matplotlib[33] y Seaborn[34] para las representaciones y muchas más para los modelados.

Aquí es donde entra el propósito de usar SQLite, debido a varios motivos:

- Si se quieren hacer combinaciones de varias tablas con pandas (utilizando un join), es mucho más complejo que con SQL y menos intuitivo, además de que el pandas guarda en memoria una copia de los todos datos, lo que desemboca en una pérdida de rendimiento considerable. Con SQlite sin embargo, se puede hacer la consulta directamente a la base de datos y el lenguaje recibirá solo los datos que interesan, para ello se aprovechará una característica del pandas que permita importar datos mediante una simple conexión a la base de datos y una consulta en SQL.
- Al hacerse la consulta primero en SQL en vez de importar directamente todo el csv, se pueden seleccionar las características que interesan, es decir, en la tabla vehículo de casi 100 características lo más probable es que no interesen todas, y como ya se mencionó antes, el pandas hace una copia de los datos.
- Por último, se facilita enormemente la limpieza de los datos ya que muchos de los valores que no interesan se pueden descartar en la propia consulta (combustibles, lubricantes, etc etc).

Aún y con todo esto, se debe hacer un pequeño tratamiento de los datos, siempre antes de crear el modelo.

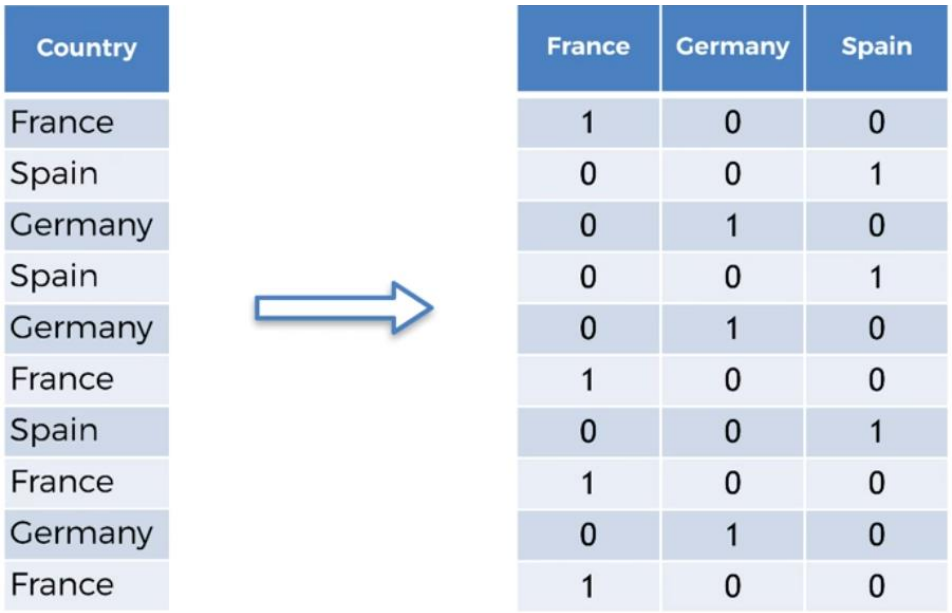
2.2 Tratamiento de los datos

Una vez se tienen los datos en el programa, se deben seguir una serie de pasos para que, en primer lugar, no haya errores en el modelo y, en segundo lugar, para que el modelo se ajuste lo mejor posible a las condiciones existentes.

Para ello se usarán distintas funciones de la librería de Python scikit-learn [35] que están diseñadas específicamente para este tipo de propósito.

En primer lugar se debe comprobar si algún valor está perdido o a nulo (null), para ello se puede usar la función *columna.isnull()* de pandas que simplemente devuelve un booleano sobre la columna en caso de que haya alguno, y, en caso de haberlo, *columna.isnull().sum()* para ver el número de nulo que hay en la columna. En este caso, no se ha obtenido ningún valor nulo, pero, en el caso de haberlo obtenido ya que se está trabajando con valores numéricos una buena opción sería rellenar los huecos con la media del resto de valores siempre y cuando no haya demasiados valores nulos, otra opción lógicamente es borrar aquellos registros con valores nulos, pero no se recomienda. Para llenar aquellos valores nulos con la media del resto de la columna se puede usar la función de pandas *fillna(column.mean())*.

En segundo lugar, se deben tener en cuenta aquellas características que sean datos categóricos, países, ciudades etc. En este caso, el nombre de la Familia de los objetos, el nombre de los objetos en sí etc. Es decir, aquellas características no representadas por números sino por palabras. Cada variable dentro de la característica se transforma en nuevas columnas que representan lo que son, pero con un valor numérico codificado, para ello normalmente se sigue un sistema de codificación en el que si hay n-variables distintas se crearán n-columnas distintas cada una con el nombre de variable transformada.



Country
France
Spain
Germany
Spain
Germany
France
Spain
France
Germany
France

France	Germany	Spain
1	0	0
0	0	1
0	1	0
0	0	1
0	1	0
1	0	0
0	0	1
1	0	0
0	1	0
1	0	0

Figura 24: Ejemplo de codificación de los datos categóricos.

Esto se puede hacer fácilmente con una función de la librería, denominada sklearn.preprocessing llamada LabelEconder y después ajustarlo a los valores numéricos del resto de la tabla con otra denominada OneHotEncoder, sin embargo, debido a dificultades surgidas durante la realización del trabajo, al final solo se trabajó con fechas y variables numéricas, ya que el modelo necesitó mucho más trabajo del esperado (para más información mirar el apartado dificultades encontradas).

Una vez realizados estos pasos, hay que separar los datos en un set de entrenamiento y un set de validación, (Training set y Test set). El training set contiene los datos con los que el modelo va a entrenarse y con los que va a comparar su función de pérdida a medida que aprende, es por ello que el Training set debe tener un porcentaje mucho mayor de datos que el Test set, el cual sirve para comprobar el ajuste final del modelo. En este trabajo se ha elegido un 90% de datos para el Training set y un 10% para el Test set.

Obtenidos los sets, hay un paso extremadamente importante y es el de normalizar los datos. La normalización es una técnica que consiste en cambiar los valores numéricos de las columnas de un dataset a una escala común, sin distorsionar así su valor, esto es muy importante debido a que sin esto habría variables con mucho más peso que otras por un mero hecho de cantidad tal y como se muestra.

Tabla 1: Ejemplo de datos en la tabla:

Edad	Salario
35	58000
50	83000

Quedando después

Tabla 2: Ejemplo de datos normalizados:

Edad	Salario
-0.7071068	-0.7071068
0.7071068	0.7071068

Para la normalización de los datos que servirán al modelo se ha usado la función de *sklearn.preprocesing* RobustScaler, la cual tiene la ventaja frente a otros normalizadores de que es más resistente a los valores atípicos (outliers).

Por último, es prioritario aclarar la forma de los datos de entrada y de entrenamiento.

Tabla 3:Formato de los datos de entrada en el proyecto

Fecha	Existencias
YYYY-MM-DD	Número en formato flotante

Tabla 4: Formato de entrada de los datos en el entrenamiento

Fecha	TotalExistencias	Día de la semana	Día del mes	Mes
YYYY-MM-DD	Número en formato flotante	Número entero (0-6)	Número entero(0-30)	Número del mes (0-11)

Por último, pero no menos importante, se debe hacer un tratamiento especial a las series temporales y es el conocido como "ventana deslizante" especialmente útil para las LSTM que es el método elegido para el modelo no paramétrico. Este método sirve para transformar el problema en uno de aprendizaje supervisado sin cambiar necesariamente la entrada del problema. En la figura 25 se puede ver un esquema de cómo funciona el método.

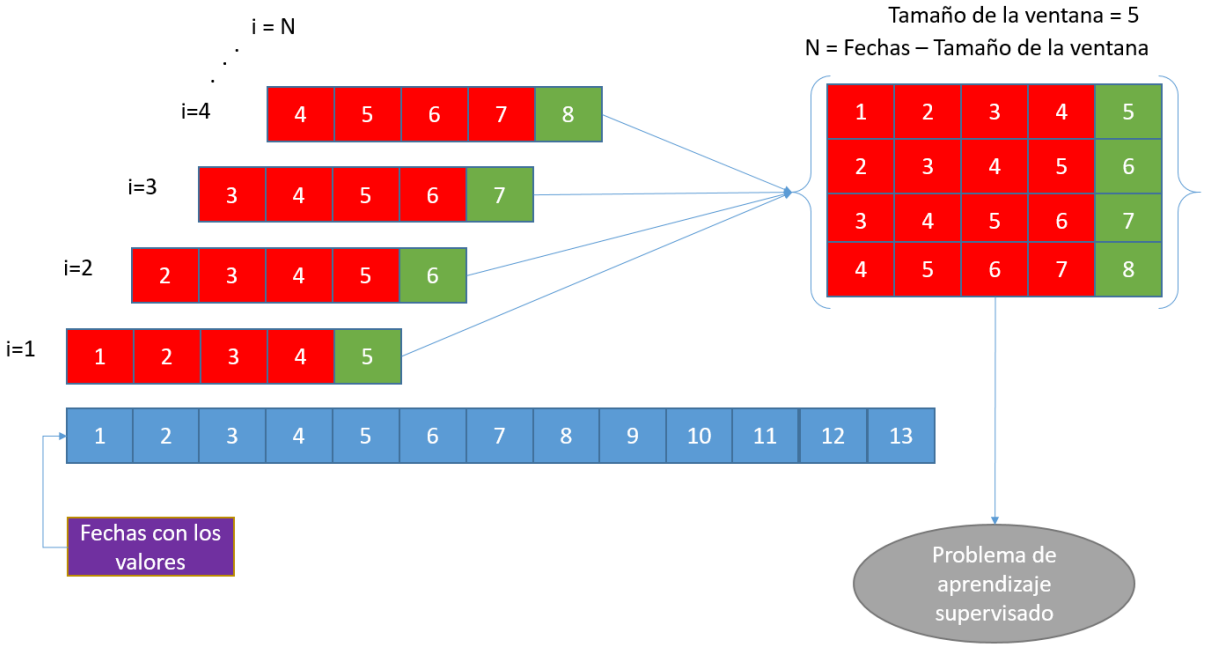


Figura 25: Funcionamiento de la ventana deslizante.

Se tienen como entrada todas las fechas ordenadas con sus valores correspondientes (Azul), sin embargo, esto no es un problema de aprendizaje supervisado, para convertirlo en uno, se debe tener una variable de entrada y una variable de salida o lo que es lo mismo, una función de entrada Y, y una función de salida f(X).

Una ventana no es más que una serie de valores del conjunto de entrada originales que almacenan o "predicen" (Rojo) un último valor (Verde), teniendo así un problema de aprendizaje supervisado, el problema es que al ser una serie temporal se debe abarcar de principio a final y es por eso que se realiza un bucle desde el principio de las fechas originales hasta el final restándose eso si el propio tamaño de la ventana ya que no podemos "predecir" el futuro, el tamaño de la ventana puede variar y ser más grande o más pequeña, la diferencia serán los valores que se tendrán en cuenta para la "predicción" del último valor de la ventana.

Por último, la ventana se "desliza" debido a que tal y como se mencionó anteriormente se debe hacer un problema de aprendizaje supervisado de toda la serie temporal y no solo de 1 trozo, por lo que, se crearán varias ventanas o lo que es lo mismo, la misma ventana se irá desplazando y se irán añadiendo los valores a un nuevo contenedor con 1 dimensión más que la serie temporal original.

Este nuevo contenedor será el que se usará para el modelado pues, este sí, representa un problema de aprendizaje supervisado.

Véase el código en el Apéndice A.

Una vez acabado esto, se podrá proceder a realizar el modelado.

2.3 Modelos, eficacia y predicción de cara a futuro.

Para tener una predicción lo más ajustada posible, se ha de intentar tener distintos modelos para así tener varias referencias de la metodología utilizada. Es por ello que se ha decidido utilizar un modelo paramétrico y otro no paramétrico, aunque, está claro que la dificultad para realizar uno no paramétrico es mayor, en un principio la empresa estaba interesada en la realización de uno. Es por ello que se usará el modelo paramétrico como base para la realización del resto de modelos no paramétricos.

2.3.1 Modelo paramétrico, ARIMA.

Una de las ventajas de los modelos paramétricos tal y como se mencionó en el estado del arte, es que son mucho más fáciles de implementar que los no paramétricos, por eso y porque el ARIMA se ajusta bastante al tipo de serie temporal planteada se usará el mismo para modelarlas.

Para ello, se ha utilizado en su defecto el lenguaje de programación R [36], en este caso en específico se ha decidido no usar Python ya que en R hay una librería especialmente diseñada para este tipo de modelos, la librería forecast, la cual contiene una función denominada *auto.arima()* que además busca los parámetros óptimos para el modelo.

Al igual que con Python, se debe hacer un tratamiento de los datos, en R, al igual que en Python se proveen de dichas herramientas para el tratamiento de los mismos, sin embargo, en este caso, se pueden saltar los pasos de normalizar los datos y la ventana deslizante puesto que la función ya los normaliza de forma automática y la ventana deslizante solo es necesaria para el modelo no paramétrico.

Al trabajar con series temporales el training set debe ser un sector de los datos, pero su vez, estar ordenado, es decir, que su periodo sea desde el principio hasta un punto X donde se decida que empiece el test set. Al haber decidido que el training set tenga un volumen del 90% de los datos con respecto del total en sendos casos queda:

- Bombillos: Training set (01-01-2008 / 17-12-2018) Test set (18-12-2018 / 27-05-2020)
- Cubiertas: Training set (09-10-2012 / 26-11-2018) Test set (27-11-2018 / 12-08-2019)

Como siempre se debe realizar procedimiento, es decir entrenar el modelo con el training set y posteriormente probar su eficacia frente al test set, en este caso, lo que se hará será que los modelos predigan en las fechas de los test para ver si coinciden los resultados (ver como se ajusta al test) para finalmente hacer predicciones de cara a un futuro desconocido. Las predicciones se harán hacia un máximo de 30 días puesto que sino el intervalo de confianza aumenta demasiado y además a la empresa le interesa predicciones a corto medio plazo.

En el caso de los Bombillos.

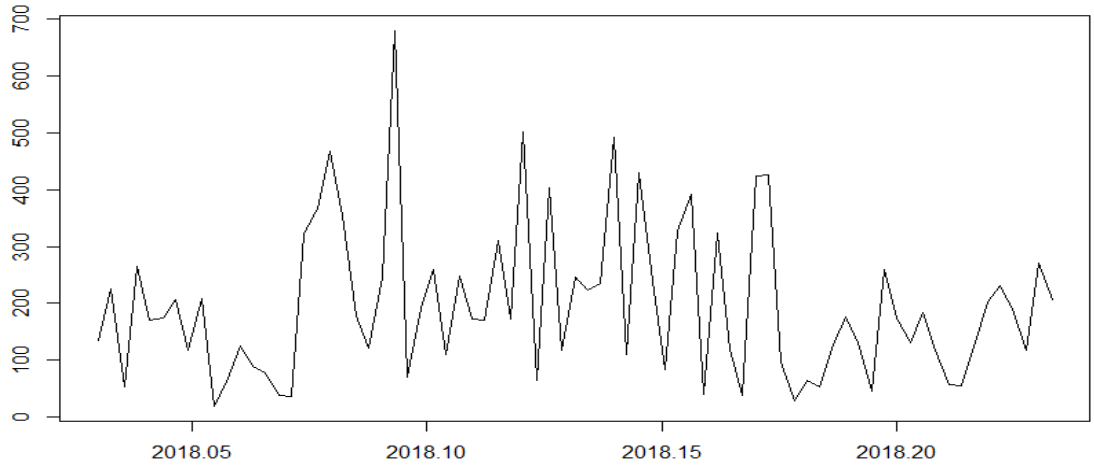


Figura 26: Test set de los bombillos

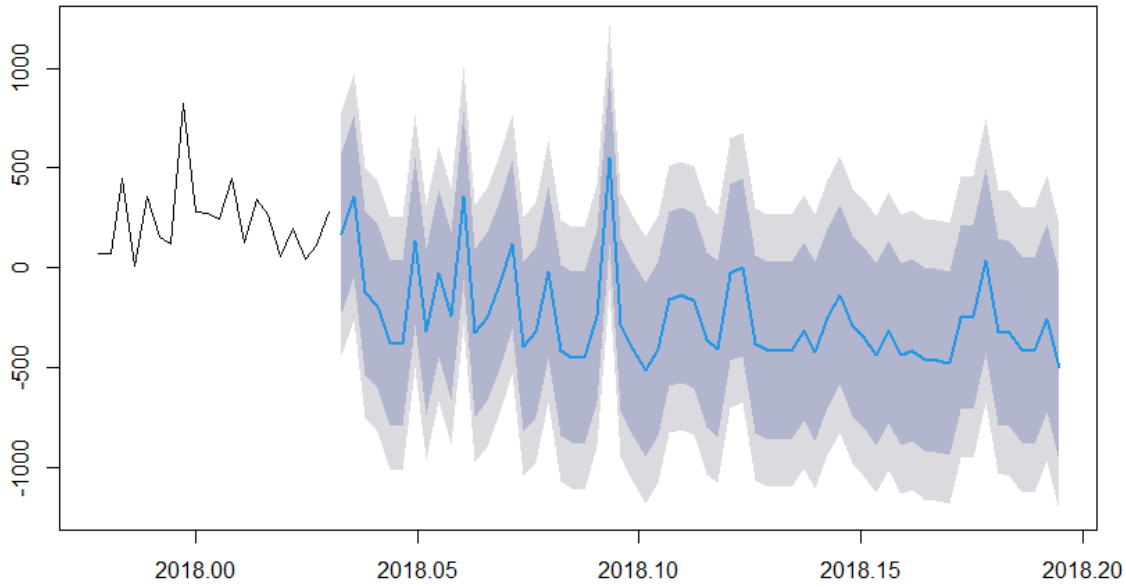


Figura 27: Arima sobre los datos que abarcan el principio del test, exactamente los primeros 30 días (figura 26), como se puede observar el intervalo de confianza es extremadamente grande aún con el horizonte de predicción corto, por lo menos, vemos que se encuentran dentro de los datos reales del test set.

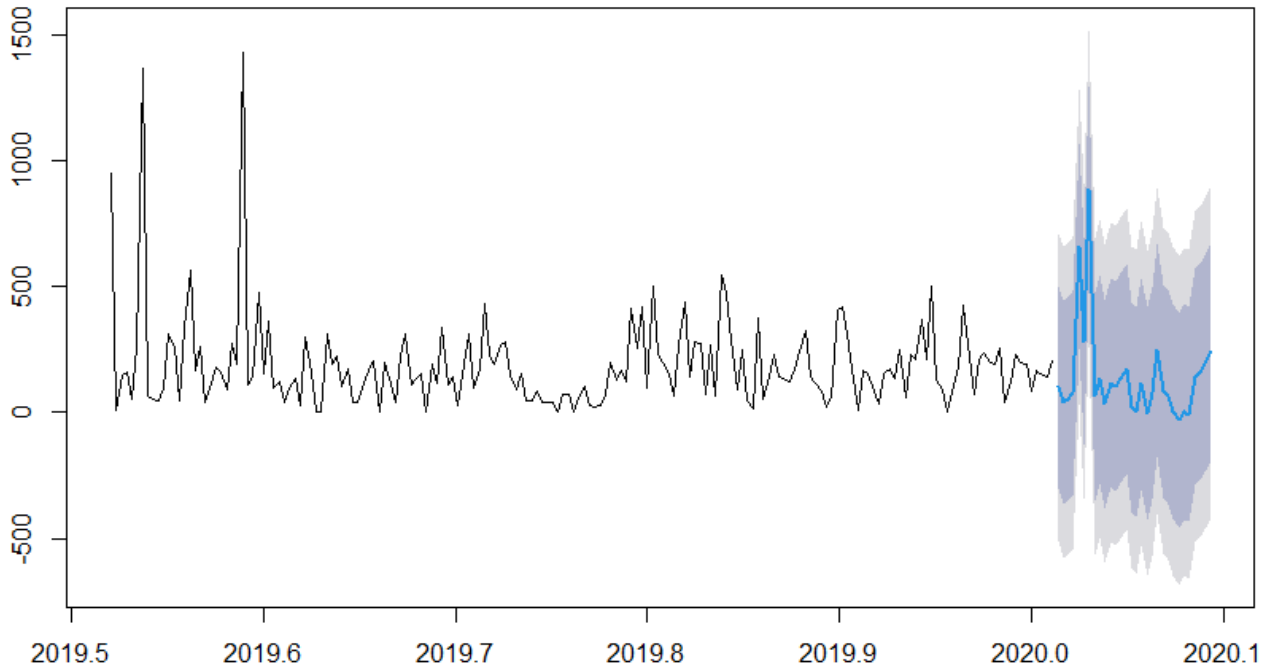


Figura 28: Arima hacia el futuro desconocido de los bombillos, es imposible conocer los valores cómo es lógico, pero por lo menos se puede observar que se sigue la misma tendencia, desgraciadamente los intervalos de confianza son bastante grandes, lo que quiere decir que no hay demasiada seguridad.

En el caso de las cubiertas:

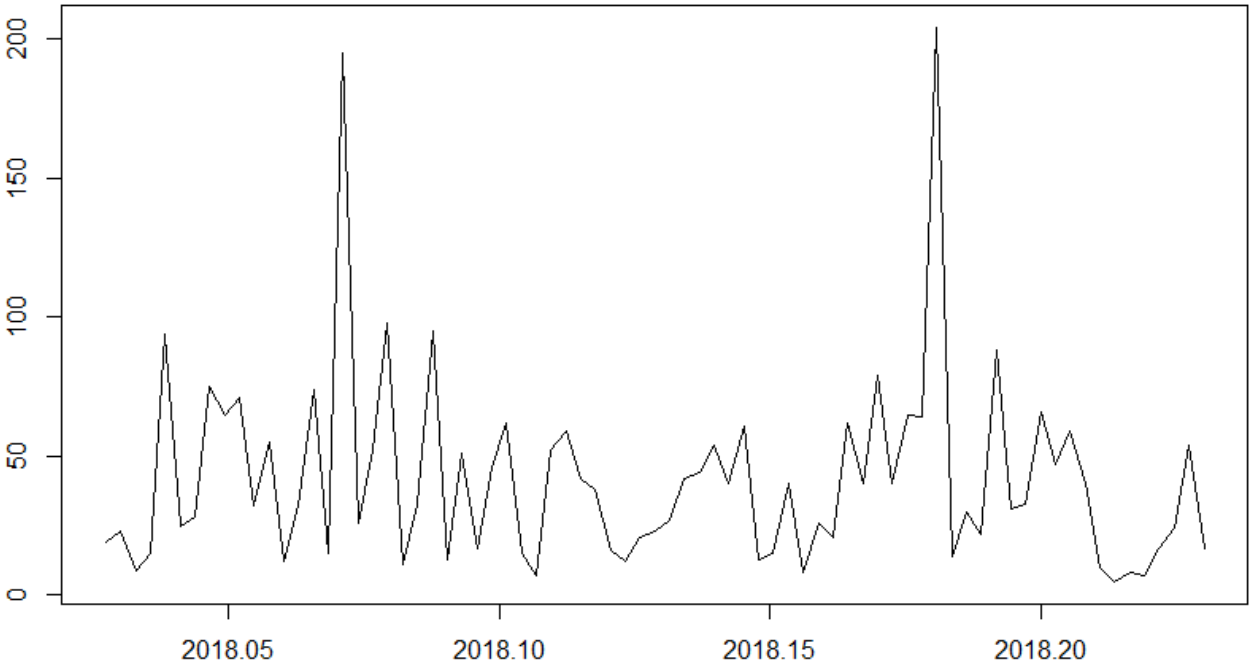


Figura 29: Test set de las Cubiertas.

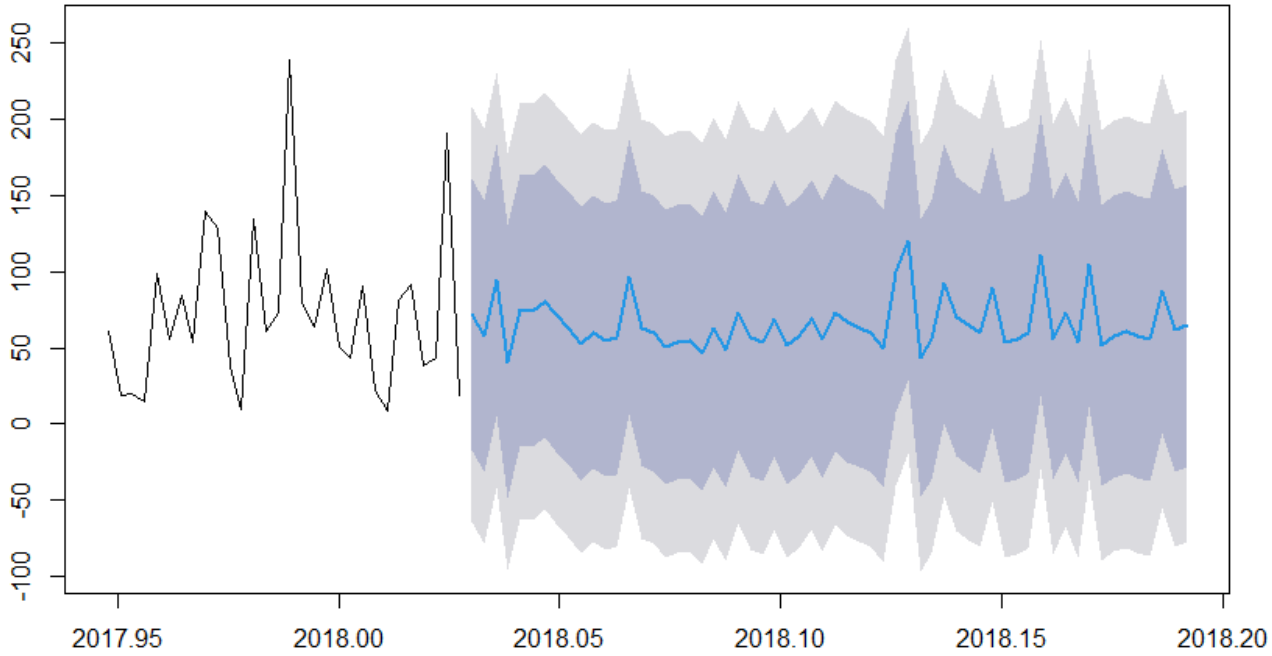


Figura 30: Arima sobre los datos que abarcan el principio del test, exactamente los primeros 30 días (figura 29), en este caso, vemos que la media directa de la predicción ha tenido más éxito ya que no da valores negativos sin embargo tenemos un intervalo de confianza muy grande.

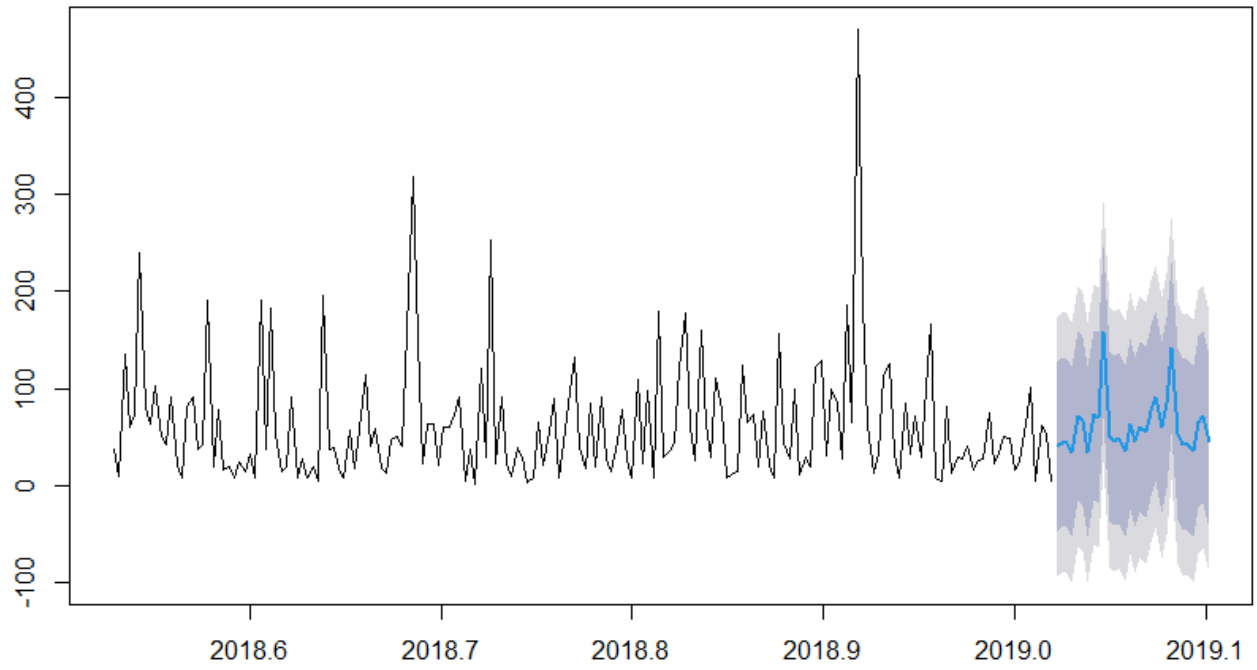


Figura 31: Arima hacia el futuro desconocido en las cubiertas, aunque sea 2019, los últimos registros de las cubiertas datan de este año y no se tiene más información. Al igual que antes, se ve que sigue una cierta tendencia general basada en el histórico anterior.

Como conclusión, recalcar que este es el modelo que va a servir de base, es decir, se va a intentar mejorar mediante modelos no paramétricos los resultados de este, ya que como se puede observar, aunque los resultados son prometedores, distan de ser un modelo perfecto.

2.3.2 Modelo no paramétrico, LSTM Bidireccional.

En este caso se trabajará en todo momento en Python, para volver a usar todas las características disponibles anteriormente descritas en el tratamiento de datos.

Para la implementación de la LSTM Bidireccional, se usará Tensorflow 2.0 [37] junto con Keras [38] disponibles en Python que, además al tener soporte para GPU permite acelerar el proceso ya que como bien indica su nombre, Tensorflow está diseñado de tal forma que los vectores se conviertan en tensores y la arquitectura hardware de las GPU [39] está mejor diseñada para este tipo de tareas. Keras es una biblioteca que funciona teniendo como back-end a Tensorflow y permite abstraerse todavía más a la hora de la creación de las redes neuronales. Como GPU de entrenamiento se usará una GeForce GTX 970 con tecnología CUDA [40] utilizada para este tipo de tareas de Machine Learning.

A continuación, en la figura 32, se presenta un esquema de la red neuronal con sus capas de procesamiento.

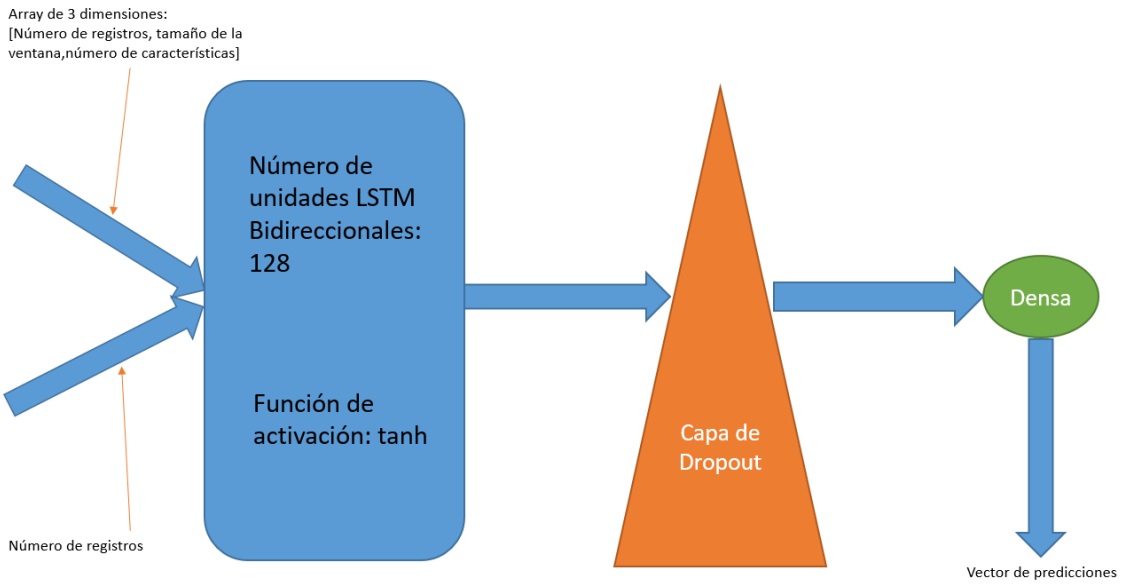


Figura 32: Esquema de la red LSTM Bidireccional utilizada para este proyecto.

Como se puede apreciar en la figura, en primer lugar, consta como entrada de una matriz compuesta por: un array tridimensional que, tal y como se ve en la figura se compone del número total de registros de la serie temporal, del tamaño de la ventana definido en el tratamiento de datos como segundo lugar (ya que la LSTM irá analizando por "trozos", y, por último del número de características totales que tengan nuestros datos, en este caso son 4 (Existencias, Día de la semana, Día del mes y mes) ya que la fecha es el índice y no se toma. Y también se compone de otro array que simplemente contiene la información de los registros.

A continuación, se tiene una capa con 128 unidades LSTM Bidireccionales con función de activación tanh, el número de neuronas ha sido determinado por distintos factores como son: tiempo de entrenamiento, overfitting, rendimiento en el aprendizaje, función de pérdida etc.

Además, se ha añadido una capa de Dropout al 20% para intentar reducir el overfitting, aunque en esto se profundizará a continuación.

Por último, se tiene una capa Densa, de 1 sola neurona ya que la salida obtenida va a ser representada por sólo 1 número en cada iteración, guardándose en las sucesivas iteraciones en un vector de predicciones que son las Existencias. Véase el código para la creación en el Apéndice A.

A la hora de entrenar una red LSTM Bidireccional hay que tener en cuenta el overfitting del mismo, ya que, sino, el modelo no está aprendiendo de nuestros datos, sino que se está ajustando a los mismos o lo que es lo mismo, los está memorizando.

Es por ello que no se deben añadir demasiadas neuronas ni capas de neuronas (aparte de por una cuestión de rendimiento) y siempre es aconsejable añadir DropOut a la red. El DropOut lo que hace a grandes rasgos es aleatorizar un poco la salida de la capa anterior, o incluso de la propia capa si se lo añadimos a la misma, ignorando las salidas de ciertas neuronas aleatoriamente, es decir, con una capa de DropOut de un 20%, ignoraremos en cada iteración del entrenamiento un 20% de las salidas de la capa de la LSTM, haciendo así a la red que no sea tan fácil de memorizar la información.

Una vez hecho esto se debe decidir durante cuanto hay que entrenar a la red, normalmente si se entrena demasiado poco, hay underfitting,lo que significa que la red no ha aprendido lo suficiente o no se ha podido ajustar a los datos, si se entrena demasiado, hay overfitting que es como se mencionaba antes, que la red no sabe salir a un ámbito superior a su aprendizaje.

Para saber en qué punto se debe parar de entrenar a la red, se dispone de las funciones de pérdida (train loss) y de validación (validation loss). La red cuando se entrena intentará disminuir la función de pérdida, en este caso se ha usado la función de pérdida del error cuadrático medio (MSE), sin embargo, puede llegar un punto en el que la red esté aprendiendo a memorizar y no aprendiendo a generalizar los datos. Para ello se usa la función de validación de la pérdida, si esta es mayor que la función de pérdida quiere decir que el modelo está empezando a "overfittear" y no se debe seguir entrenando, por tanto, el punto en el que el modelo debe parar de entrenar, se produce cuando la función de pérdida sea extremadamente baja y menor a la de validación, o cuando la función de pérdida empiece a igualarse a la función de validación ya que el aprendizaje es inútil llegado el caso.

En el caso de los datos a analizar se puede ver que:

En los Bombillos:

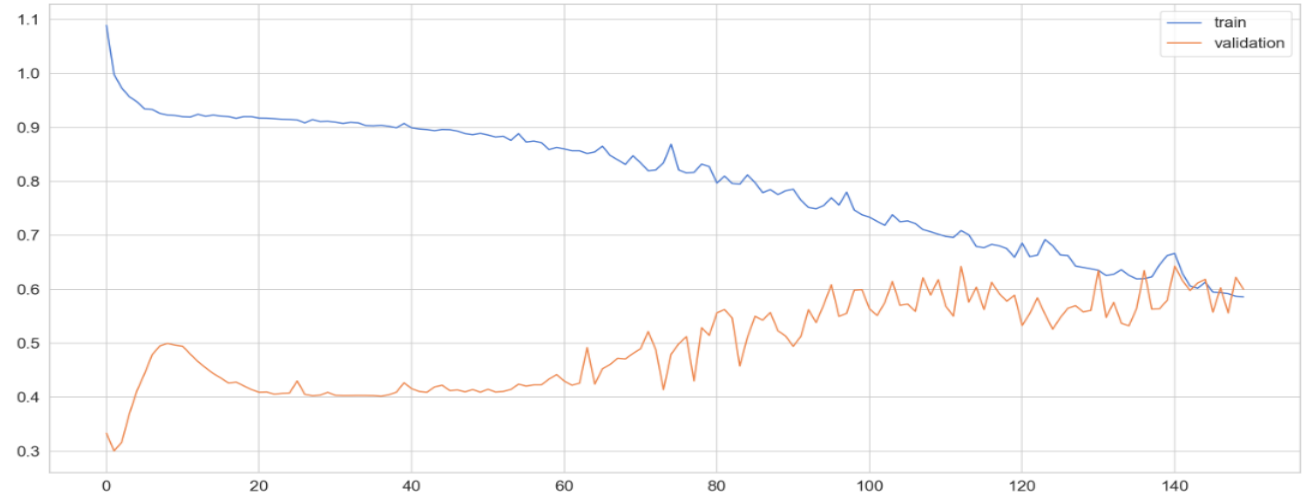


Figura 33: Número óptimo de epochs en los bombillos (sesiones de entrenamiento), ronda las 140 que es cuando se empieza a producir el Overfitting.

En las Cubiertas:

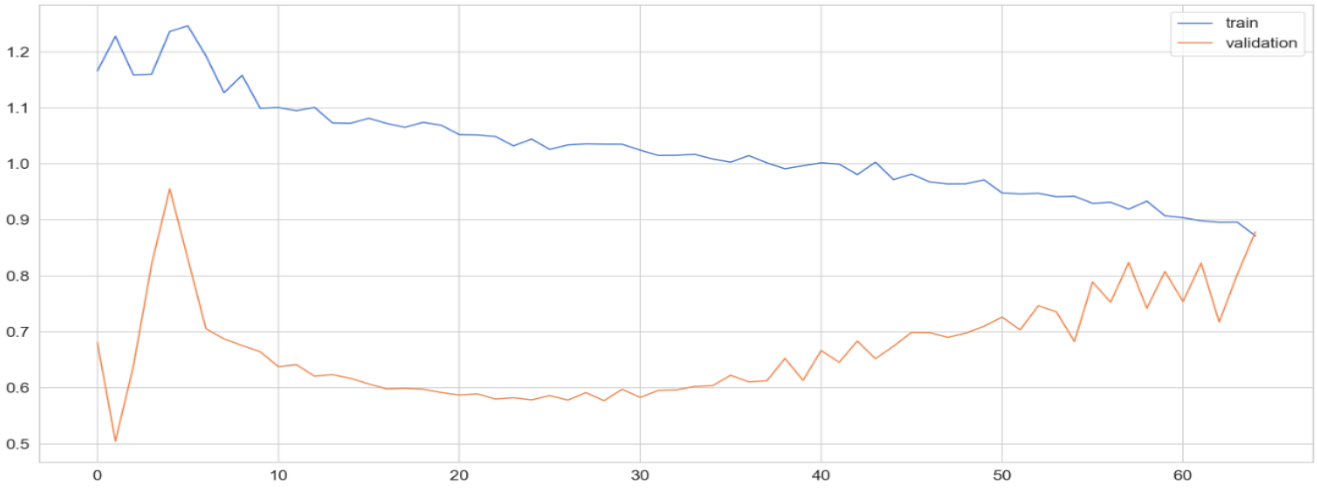


Figura 34: Número óptimo de epochs en las cubiertas. El óptimo son 65.

Nota: Véase el código para el entrenamiento en el Apéndice A siendo N 65 o 140 en este caso.

Normalmente, cuando un modelo suele estar bien ajustado a los datos, la función de pérdida suele rondar en torno a valor menores a 0.05. En estos modelos, la mejor función de pérdida obtenida en los entrenamientos ha sido 0.60 aproximadamente en el caso de los bombillos y 0.90 en el caso de las

cubiertas esto puede deberse a diversos factores como la presencia de demasiado ruido en los datos reales que nuestro modelo no es capaz de identificar, falta de datos etc. Al estar usándose como media el error cuadrático medio, los picos en los datos que el modelo no sea capaz de identificar provoca que se sume de forma extrema al error. Para ello, lo mejor, es ver la predicción del modelo frente a los datos reales del test, tal y como se hizo antes con el modelo ARIMA.

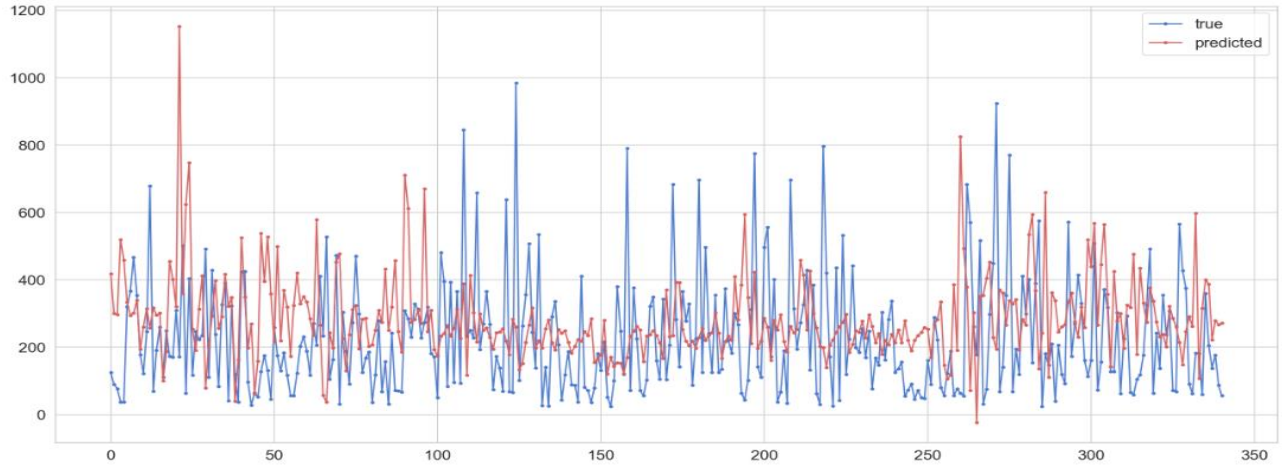


Figura 35: Ajuste del modelo a los datos reales en los bombillos, como se observa no es capaz de detectar los picos de los mismos lo cual puede considerarse como ruido en la serie, este puede ser uno de los motivos de la alta pérdida tal y como se acaba de mencionar.

En este caso, en el eje X aparece el número de muestras que están guardadas en el test set representando las fechas comprendidas entre el 17-12-2018 y 27-05-2020.

Una vez realizada la comparación de la predicción del modelo con los datos reales, se puede hacer una predicción hacia el futuro desconocido. Para ello se puede utilizar un método recursivo en el que se predice un primer valor a partir del histórico, y a partir de ese valor se predicen los siguientes añadiéndose a la lista de predicciones las cuales se van a usar en futuras predicciones El problema de este método es que cuanto más lejano sea el horizonte de predicción, más grande será el intervalo de confianza y menos confiable la predicción. Es por ello que basaremos las predicciones según las características y así de dicho sea de paso aprovecharemos la potencia de las redes neuronales ya que en un futuro se podría decidir introducir más características de aprendizaje y de retroalimentación. Tal y como se hizo anteriormente, se realizará una predicción a 30 días.

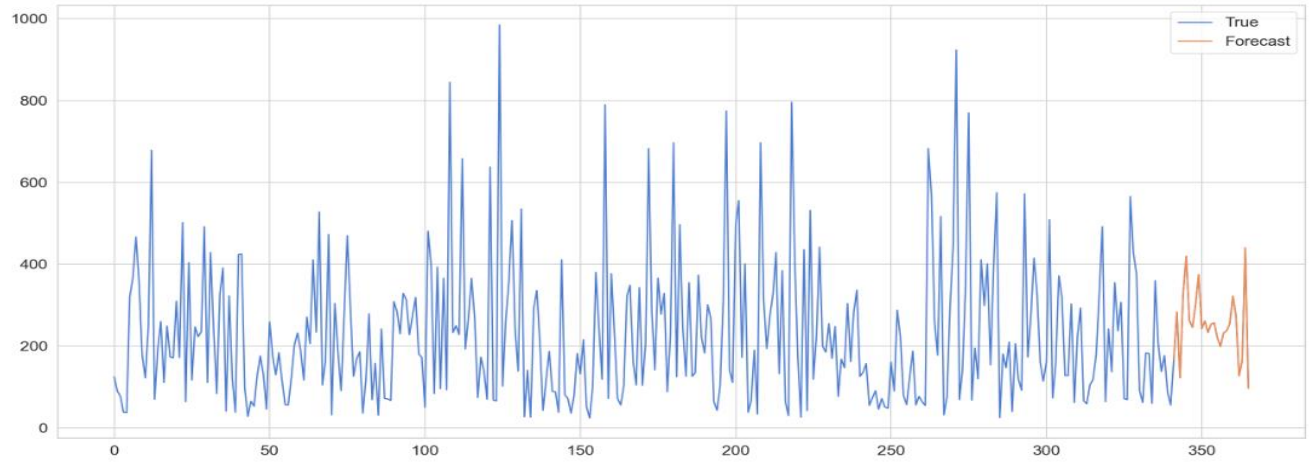


Figura 36: Predicción a 30 días de los bombillos. No se ven anomalías con respecto al histórico

Ahora con las cubiertas:

Datos reales frente a la predicción:

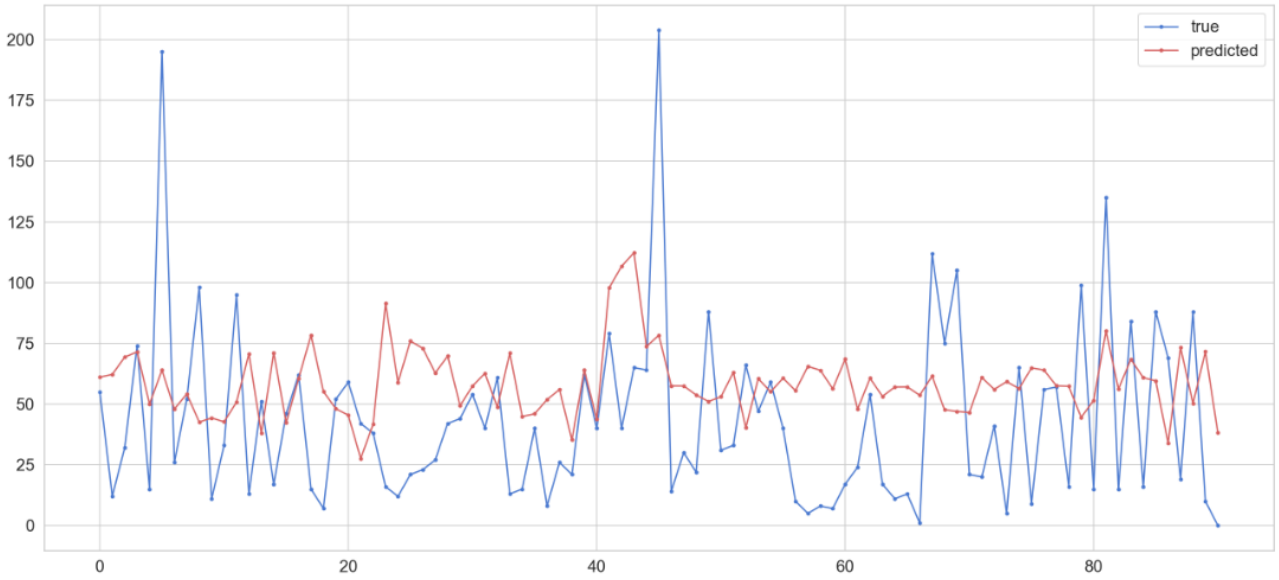


Figura 37: Ajuste del modelo a los datos reales en las cubiertas, Como se observa no lo hace tan bien como en el caso anterior, tal vez pueda deberse a la menor cantidad de muestras de las mismas.

En el eje X aparecen el número de muestras que están guardadas en el test set representando las fechas comprendidas entre el 26-11-2018 y 12-08-2019

Predicción hacia el futuro:

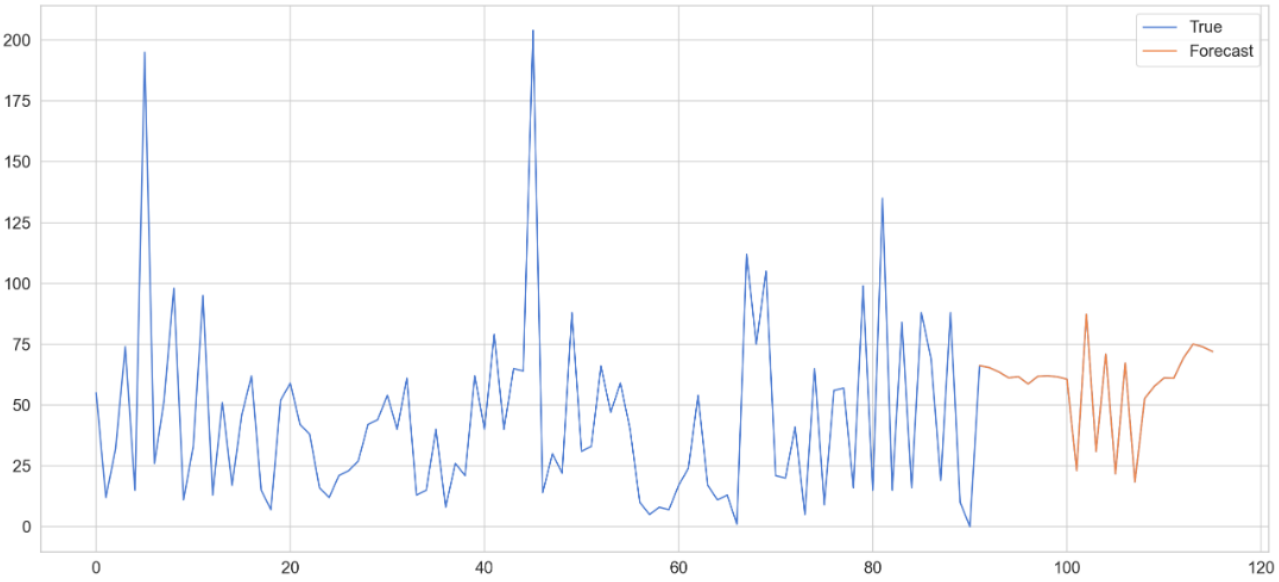


Figura 38: Predicción a 30 días de las cubiertas. A diferencia del caso anterior si vemos como quizá el modelo no detecta las anomalías que se producen regularmente en la serie temporal, aunque no se puede estar seguro de que la predicción sea correcta o no, al final está en una media.

2.4 Dificultades encontradas de cara al desarrollo del proyecto.

Obviamente al ser una disciplina nueva con crecimiento exponencial como pueda ser la Inteligencia Artificial y todo lo relacionado con los problemas de Big Data atañe ya dificultades de por sí como son, la falta de información o la que se encuentre sea extremadamente técnica, demasiado general etc. Es por ello que, en un principio, se contaba con la ayuda del departamento de Data Science de TITSA para múltiples tareas: tratamiento de datos, modelado, representación etc. Sin embargo, llegó el Covid-19 y como a tantos otros, perjudicó en gran medida a este proyecto.

La llegada del Covid-19 causó un retraso inimaginable en términos de tiempo en el proyecto por varios factores:

- Falta de acceso a los datos: Al estar en un estado de alarma, era totalmente imposible acceder a los datos que estaban localizados en el departamento de Data Science de TITSA y en ese momento solo se disponía de la pequeña muestra de 3000 registros por tabla. Sin embargo, a los 2 meses, gracias a un contacto se pudieron enviar de forma telemática.

- Realización casi completamente individual en un proyecto colaborativo: Es obvio que un TFG debe realizarse de forma individual, pero, en este caso, estaba pensado para que el alumno contara con ayuda de diferentes campos día a día en la compañía, sobre todo en lo que a matemáticas y estadística avanzada se refiere, además, se perdió el contacto con varios expertos en Inteligencia Artificial que podían haber asesorado el proyecto y haber acelerado y mejorado el proceso. Sin embargo, dentro de lo malo, se contaba con bastante tiempo para el estudio individual de estas técnicas y, aunque, no es lo mismo, se considera traspasada esta barrera de conocimiento, al menos lo suficiente para haber realizado el proyecto de forma exitosa. También, los tutores de forma telemática aportaron conocimientos de gran utilidad.

- Realización de los modelos no paramétricos: Uno de los grandes problemas surgidos a la hora del desarrollo fue el de ajustar adecuadamente la red neural LSTM. Las redes neuronales tienen un gran problema y es que, a diferencia de los modelos paramétricos, estas funcionan como una caja negra y es "imposible" saber que pasa dentro cuando se están entrenando, además, cada red tiene miles de parámetros para poder ajustarse y llegar a ser óptima, número de neuronas, capas, tipo de neuronas, función de activación y distribución de los datos y un larguísimo etc. Una de las tareas más arduas de este trabajo fue el de conseguir un modelo LSTM funcional que no se limitara a hacer la media de la serie temporal, hasta llegar a ese modelo LSTM final, hubo 6 o 7 modelos anteriores descartados.

- Limpieza de los datos: Los datos de TITSA contenían gran cantidad de información inútil, vacía e incluso errores los cuales se tardaron semanas en localizar y eliminar.

- El confinamiento: Aunque no sea un factor directo del desarrollo se considera que el estado de alarma afectó de forma altamente negativa al mismo debido al factor psicológico del mismo, aunque el alumno tenga más tiempo libre este no es aprovechado de igual manera debido a dicho factor, en cambio hay que agradecer que más allá de lo psicológico, no hubo consecuencias graves.

Capítulo 3
Conclusiones y líneas futuras

Cabe destacar que una de las características de este proyecto es su gran amplitud y desarrollo potencial en el futuro. Lo aquí presentado es un prototipo, que, aunque funcional puede ser mejorado exponencialmente e incluso ser usado para múltiples compañías con el mismo problema planteado. Es un problema bastante común dentro del marco de las empresas no sólo de transporte sino con una estructura que envuelva operaciones logísticas de almacenaje.

Debido entre otros a la gran dificultad que supuso el tener que hacer el proyecto de forma individual, algunas características del trabajo han quedado fuera y no ha dado tiempo a desarrollarlas, la más importante es la predicción basada en el aprendizaje de todas las propiedades disponibles en los datos de TITSA y aunque, no tan importante, una interacción más amigable con el usuario.

Es por ello que en líneas futuras sería extremadamente interesante ir incluyendo cada vez más características al modelo para su entrenamiento, un ejemplo sería utilizar todas las características en la tabla de vehículos que pueden servir para entrenar los modelos, y, no solo eso, sabiendo el tipo de averías que más suelen ocurrir, se podría también extraer de ahí una característica más extremadamente útil para el análisis y entrenamiento, aunque, para ello habría que usar técnicas de procesamiento del lenguaje natural para extraer de los partes de avería palabras clave, resúmenes de las mismas, conclusiones, y así poder crear un set de datos de entrada adecuado para el entrenamiento. Esto no es nada descabellado ya que existen múltiples algoritmos para la extracción y análisis de dichas palabras clave como puedan ser los n-gramas [41], TF-IDF [42] etc…

Está claro que las piezas actúan de forma bastante irregular en el tiempo y por eso el modelo no es capaz de predecir ese ruido, pero, probablemente añadiendo todos los factores comentados anteriormente el ruido se minimice a algo predecible. Además, lo ideal sería crear un modelo mejor a partir de un estudio más profundo del número de neuronas, capas o incluso del tipo de arquitectura.

Respecto a la visualización e interacción con el usuario, podría implementarse una interfaz gráfica de usuario para que las consultas no tengan que hacerse manualmente, sino que puedan hacerse por usuarios novatos con el ratón seleccionando los filtros, y así que el entrenamiento de los modelos para cada pieza sea mucho más dinámico. Y, para la representación, se podría usar alguna librería de representación dinámica o incluso el PowerBI para los modelos en vez de las librerías estáticas como matplotlib o seaborn.

Como conclusión final, me gustaría añadir una pequeña opinión personal y, aunque ha sido difícil trabajar en este proyecto por otro lado ha sido bastante placentero debido a la gran cantidad de cosas aprendidas durante el camino, y a la sensación de recompensa tras el arduo trabajo, además, pienso que es algo que puede tener muchas aplicaciones no solo a futuro sino a un presente inmediato y probablemente una vez tenga tiempo me dedique a intentar mejorarlo para tener las características descritas anteriormente o por lo menos parte de ellas y así sentar las bases para un modelo práctico bastante interesante el cual, una vez más, demuestre que la Inteligencia Artificial está dejando cada vez más atrás a los algoritmos tradicionales y que no debemos luchar contra ella sino adaptarnos a la misma y aprender a usarla.

3.1 Summary and Conclusions

It should be noted that one of the characteristics of this project is its great scope and potential for future development. What is presented here is a prototype, which, although functional, can be improved exponentially and even be used for multiple companies with the same problem. It is a fairly common problem within the framework of companies not only in transport but with a structure that involves storage logistics operations.

Due among others to the great difficulty that supposed to make the project of individual form, some characteristics of the work have been left out and it has not given time to develop them, the most important is the prediction based on the learning of all the available properties in TITSA's data and although, not so important, a more friendly interaction with the user.

In future lines it would be extremely interesting to include more and more characteristics to the model for its training, for example, in the table of vehicles we have absolutely all the information of each vehicle in almost 100 characteristics that could serve to train the models, and, not only that, knowing the type of breakdowns that most often occur, we could also extract from it a feature more extremely useful for analysis and training, although, for this it would be necessary to use natural language processing techniques to extract from the breakdown parts keywords, summaries of them, conclusions, and thus be able to create a set of input data suitable for training. This is not unreasonable since there are multiple algorithms for the extraction and analysis of these keywords such as n-grams [41], TF-IDF [42] etc...

It is clear that the pieces act quite irregularly in time and that is why the model is not able to predict that noise, but, probably adding all the factors mentioned above the noise will be minimized to something predictable. In addition, ideally a better model would be created from a deeper study of the number of neurons, layers or even the type of architecture.

Regarding visualization and user interaction, a graphical user interface could be implemented so that queries do not have to be done manually, but can be done by novice users with the mouse by selecting the filters, and so that the training of the models for each piece is much more dynamic. And, for the representation, you could use some dynamic representation library or even PowerBI for the models instead of the static libraries like matplotlib or seaborn.

Finally, I'd like to share a personal opinion, and even if it's been difficult to work on this project on the other hand it's been quite enjoyable due to the amount of things learned along the way, and the feeling of reward after the hard work, too, I think it's something that can have many applications not only in the future but also in the immediate present and probably once I have time I'll try to improve it to have the characteristics described above or at least part of them and thus lay the foundation for a practical model quite interesting which, once again, shows that artificial intelligence is increasingly leaving behind traditional algorithms and that we should not fight against it but adapt to it and learn how to use it.

3.2 Presupuesto

3.2.1 Coste del proyecto

Para la realización de este proyecto han sido necesarias aproximadamente 350 horas de trabajo efectivo. Siendo la remuneración para alguien con una formación universitaria superior en ingeniería informática o del software en España situada un promedio de 15 € [43] la hora. Todos los materiales resueltos por la Universidad de La Laguna sin coste añadido para el proyecto excepto el ordenador personal de trabajo el cual tiene un precio aproximado de 1000€, la realización del proyecto resulta en un total de 6250€.

Siendo desglosados:

Número de horas trabajadas:	350
Coste por hora:	15€
Ordenador:	1000€
Total:	6250€

Apéndice A: Algoritmos

Creación de la ventana deslizante

```
def create_dataset(X, y, time_steps=1):
    Xs, ys = [], []
    for i in range(len(X) - time_steps):
        v = X.iloc[i: (i + time_steps)].to_numpy()
        Xs.append(v)
        ys.append(y.iloc[i + time_steps])
    return np.array(Xs), np.array(ys)
```

Creación de la red neuronal

```
model = keras.Sequential()
model.add(
    keras.layers.Bidirectional(
        keras.layers.LSTM(
        units=128,
        input_shape=(X_train.shape[1],X_train.shape[2])
        )
    )
)
  model.add(keras.layers.Dropout(rate=0.2))
  model.add(keras.layers.Dense(units=1))
  model.compile(loss='mean_squared_error', optimizer='adam')
```

Entrenamiento de la red neuronal.

```
model.fit(
    X_train,y_train,
    epochs=N,
    batch_size=24,
    validation_split=0.1,
    shuffle=False)
```

Bibliografía

[1] TITSA <<Titsa.com>> 2020. [Internet]. Disponible en https://www.titsa.com/index.php

[2] Sabino Parmezan, A.R, Souza, M.A.V., Batista, E.A.P.A.G. (2019). Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model. Information Sciences. 484, 302-337.

[3] De Goojier, G. J., Hyndman J. R. (2006). 25 years of time series forecasting. 444

[4] Ahmed N.K., Atiya, A.F., Gayar N.E., El-Shishiny H (2010). An empirical comparison of machine learning models for time series forecasting. Econometric Reviews. 594-621.

[5] Lemke, C., & Gabrys, B. (2010). Meta-learning for time series forecasting and forecast combination. Neurocomputing, 73(10-12), 2006-2016.

[6] Xu, S., Chan, H. K., & Zhang, T. (2019). Forecasting the demand of the aviation industry using hybrid time series SARIMA-SVR approach. *Transportation Research Part E: Logistics and Transportation Review*, *122*, 169-180.

[7] Chandra, Rohitash & Ong, Yew & Goh, Chi-Keong. (2017). Co-evolutionary multi-task learning with predictive recurrence for multi-step chaotic time series prediction. Neurocomputing.

[8] Montgomery, D. C., Jennings, C. L., & Kulahci, M. (2015). Introduction to Time Series Analysis and Forecasting (Wiley Series in Probability and Statistics) (2.ª ed.). Wiley-Interscience.

[9] Gardner, E. S. (1985). Exponential smoothing: The state of the art. *Journal of Forecasting*, *4*(1), 1-28.

[10] ¿Qué es Overfitting y Underfitting? (2017) [Internet]. Disponible en:

https://www.aprendemachinelearning.com/que-es-overfitting-y-underfitting-y-como-solucionarlo/

[12] Specify Multiplicative ARIMA Model [Internet]. Disponible en:

https://es.mathworks.com/help/econ/airline-passenger-data-model.html

[13] Foo K. (2018) Seasonal lags: SARIMA modelling and forecasting [Internet] Disponible en: https://medium.com/@kfoofw/seasonal-lags-sarima-model-fa671a858729

[14] Lippmann, R. (1987). An introduction to computing with neural nets. *IEEE ASSP Magazine*, *4*(2), 4-22.

[15] Rumelhart D.E., Hinton G.E., Williams R.J (1986). Learning internal representations by error propagation

Parallel Distributed Processing: Explorations in the Microstructure of Cognition, 318-362.

[16] Venkatachalam M. (2019) Recurrent Neural Networks, [Internet] Disponible en:

https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce

[17] M. Schuster and K. K. Paliwal (1997), "Bidirectional recurrent neural networks," in IEEE Transactions on Signal Processing, vol. 45, no. 11, pp. 2673-2681.

[18] Siegelmann H.T., Horne B.G., Giles C.L. (1997) Computational capabilities of recurrent NARX neural networks, 208-215.

[19] Bengio Y., Simard P., Frasconi P (1994). Learning long-term dependencies with gradient descent is difficult, in *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157-166.[20] Hochreiter S., Schmidhuber J. (1997) Long short-term memory, 1735-1780.

[21] Understanding LSTM Networks (2015) [Internet] Disponible en:

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[22] Zhang, B., Li, N., Shi, F., & Law, R. (2020). A deep learning approach for daily tourist flow forecasting with consumer search data. *Asia Pacific Journal of Tourism Research*, *25*(3), 323-339.

[23] What's the difference between a bidirectional LSTM and an LSTM? (2017) [Internet] Disponible en:

https://stackoverflow.com/questions/43035827/whats-the-difference-between-a-bidirectional-lstm-and-an-lstm

[24] Heidari, E., Daeichian, A., Sobati, M. A., & Movahedirad, S. (2020). Prediction of the droplet spreading dynamics on a solid substrate at irregular sampling intervals: Nonlinear Auto-Regressive eXogenous Artificial Neural Network approach (NARX-ANN). *Chemical Engineering Research and Design*, *156*, 263-272.

[25] G. Abbas, M. Nawaz and F. Kamran, (2019) "Performance Comparison of NARX & RNN-LSTM Neural Networks for LiFePO4 Battery State of Charge Estimation," *2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, Islamabad, Pakistan, 2019, pp. 463-468,

[26] PowerBI <<powerbi.microsoft.com>>[Internet] Disponible en: https://powerbi.microsoft.com/es-es/

[27] Hadoop <<hadoop.apache.org>> [Internet] Disponible en: https://hadoop.apache.org/

[28] Sqoop <<sqoop.apache.org>> [Internet] Disponible en: https://sqoop.apache.org/

[29] Sqlite <<sqlite.org>> [Internet] Disponible en: https://www.sqlite.org/index.html

[30] Python <<python.org>> [Internet] Disponible en: https://www.python.org/

[31] Pandas <<pandas.pydata.org>> [Internet] Disponible en: https://pandas.pydata.org/

[32] Numpy <<numpy.org [Internet] Disponible en: https://numpy.org/

[33] Matplotlib <<matplotlib.org>> [Internet] Disponible en: https://matplotlib.org/

[34] Seaborn <<seaborn.pydata.org>> [Internet] Disponible en: https://seaborn.pydata.org/

[35] Scikit-learn<<scikit-learn.org>> [Internet] Disponible en: https://scikit-learn.org/stable/

[36] R página <<r-project.org>> [Internet] Disponible en: https://www.r-project.org/

[37] Tensorflow <<www.tensorflow.org [Internet] Disponible en: https://www.tensorflow.org/

[38] Keras <<keras.io>> [Internet] Disponible en: https://keras.io/

[39] Dsouza J. (2020) What is a GPU and do you need one in Deep Learning? [Internet] Disponible en:

https://towardsdatascience.com/what-is-a-gpu-and-do-you-need-one-in-deep-learning-718b9597aa0d#:~:text=Why%20choose%20GPUs%20for%20Deep,can%20process%20multiple%20computations%20simultaneously.&text=Additionally%2C%20computations%20in%20deep%20learning,GPU's%20memory%20bandwidth%20most%20suitable.

[40] CUDA [Internet] Disponible en: https://es.wikipedia.org/wiki/CUDA

[41] N-Gramas [Internet] Disponible en: https://en.wikipedia.org/wiki/N-gram:

[42] TF-IDF [Internet] Disponible en: https://en.wikipedia.org/wiki/Tf%E2%80%93idf

[43] Average Software Engineer Salary in Spain (2020) [Internet] Disponible en:
https://www.payscale.com/research/ES/Job=Software_Engineer/Salary

[44] Emmanuel J., (2015) Forecasting: Exponential Smoothing, MSE [Internet] Disponible en:
https://www.youtube.com/watch?v=k_HN0wOKDd0

[45] Yelkenci T., (2014) Time series analysis on EURUSD [Internet] Disponible en:
https://www.slideshare.net/TezerYelkenci/time-series-analysis-on-eurusd