Universidad de La Laguna

Master Universitario en Ciberseguridad e Inteligencia de los Datos

Trabajo de Fin de Máster

Predicción de la demanda de cuotas de socios en un centro deportivo tras la COVID-19

Demand forecasting for membership fees at a sports center after COVID-19

Carlos Domínguez García

La Laguna, 8 de marzo de 2021

D. **José Marcos Moreno Vega**, con N.I.F. 42.841.047-M profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

Dña. **María Belén Melián Batista**, con N.I.F. 44.311.040-E profesora Titular de Universidad adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutora

C E R T I F I C A N

Que la presente memoria titulada:

”*Predicción de la demanda de cuotas de socios en un centro deportivo tras el COVID-19*”

ha sido realizada bajo su dirección por D. **Carlos Domínguez García**, con N.I.F. 42.238.865-D.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 8 de marzo de 2021

Agradecimientos

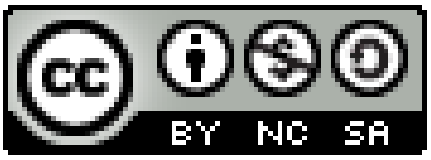
Durante el proceso de este trabajo he tenido el placer de contar con la ayuda de los mejores compañeros que se pueden tener, muchas gracias Erika y Samuel y también incluyo a Ginés que ha estado supervisando el proyecto y guiándonos por el camino acertado en todo momento.

No me puedo olvidar de mis tutores académicos, Marcos y Belén, y su paciencia infinita conmigo cambiando repentinamente el tema del trabajo de fin de máster.

Por supuesto, tampoco puedo olvidarme de mi familia y resto de amigos siempre apoyando.

Finalmente, también se merece una mención toda la gente que comparte contenido educativo de manera accesible para todos, sin duda el conocimiento que tendría ahora sería muy distinto si no fuera por esos materiales.

Licencia



Resumen

Como parte de un proyecto de auditoría de datos a un centro deportivo, esta memoria de trabajo de fin de máster recoge el desarrollo del tratamiento de datos y de los modelos para la predicción de la demanda de cuotas de socios en el tiempo. De esta manera, se pretende asistir en la toma de decisiones de la empresa, lo cual toma especial importancia tras el impacto de la COVID-19.

Con este objetivo, se plantearon diversos procesos para el tratamiento de datos además de diversos modelos. Se realizaron numerosos experimentos, por una parte sin tener en cuenta el impacto de la COVID-19 para estimar la capacidad predictiva de los modelos de manera robusta. Y en una segunda parte, se realizaron predicciones hacia el futuro, terminando con resultados prometedores.

Palabras clave: Predicción, COVID-19, Demanda, Cuotas

Abstract

As part of a data audit project for a sport center, this end-of-master's report includes the work done in data processing and modelling for forecasting the membership demand over time. This is intended to assist in decision making which takes special importance after the impact that COVID-19 has had.

With this objective, several processes for data processing were proposed, as well as several models. Numerous experiments were carried out, on the one hand without considering the impact of the COVID-19 to estimate the predictive capacity of the models in a robust way. And, on the other hand, future predictions were made, ending with promising results.

Keywords: Forecasting, COVID-19, Membership, Dsemand

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

1.1. Motivación

El tratamiento de datos es un proceso clave en los negocios para poder tomar decisiones mejores informadas. En particular, el uso de datos históricos para crear modelos predictivos permite a las empresas tomar decisiones adelantándose a los acontecimientos.

En un tiempo especialmente incierto debido al impacto de la COVID-19, poder realizar predicciones acertadas toma una importancia adicional para ayudar a la toma de decisiones que permita la continuidad de la empresa.

En el presente proyecto se ha trabajado con datos de una empresa local de Tenerife, Radazul Sport Center. Este es un centro deportivo con más de $23{,}000m^2$ de superficie distribuida entre canchas, piscinas, residencia con 12 habitaciones, aparcamientos y zonas comunes. Además, cuenta con un gimnasio totalmente equipado y ofrece una variedad de clases y actividades a los socios.

Previamente, junto a un equipo de la cátedra Big Data, Open Data y Blockchain, se ha realizado un extenso proyecto de auditoría de datos al centro donde:

- Se empezó investigando el software que usa el centro para registrar su actividad.
- Se realizó un estudio de lo que se podría hacer en el centro basado en el uso de datos.
- Se identificó cómo extraer datos relevantes del software.
- Se realizó un estudio de tecnologías RPA (Robotic Process Automation) y se implementó una solución con esta tecnología para la extracción automática de los datos a través de la interfaz gráfica del software.
- Se identificaron unas exportaciones periódicas de datos contables del software que se envían a una asesoría. Además, se identificó el formato de estos datos y se desarrolló un programa para pasar estos datos a un formato Excel para entenderlos mejor.

- Se realizaron cuadros de mando comparando la información exportada del software con la información que da la asesoría para asegurarse que cuadran los datos y que la empresa pueda tener un control propio de esta información.
- Se identificó y se pudo acceder a la base de datos del software, por lo que a partir de este momento pudo empezar a extraer la información del software en cualquier momento, de manera que el control que se le puede ofrecer a la empresa de su actividad esté siempre actualizado.
- Se realizaron cuadros de mando comparando la información que ya se tenía con la extraída de la base de datos para asegurarse de que son correctos.
- Además, en este momento se tenía acceso a una gran variedad de datos del centro, por lo que se pudieron realizar cuadros de mando de diversas materias útiles para la toma de decisiones y sugerir mejoras en los procesos de la empresa para que tenga un mejor control de su actividad.
- Se desarrolló un sistema para enviar resúmenes diarios por Telegram de las actividades de interés.
- También se desarrollaron modelos predictivos para la demanda de cuotas de socios en el tiempo.

Esta memoria de trabajo de fin de máster se centra en desarrollar el trabajo realizado en lo relativo a los modelos predictivos. Para ello se contó con datos de cuotas de socios extraídas de la base de datos y se desarrollaron modelos predictivos con el objetivo de obtener predicciones razonables tras el impacto de la COVID-19.

1.2. Antecedentes

Además de técnicas tradicionales basadas en regresiones sencillas, actualmente también se aplican modelos más complejos en la industria:

- En [1] se provee una recopilación de métodos de predicción de series temporales basadas en redes neuronales debido a su creciente popularidad en la industria.
- En la reciente competición “M5 Forecasting - Accuracy” se pedía a los participantes predecir las ventas de productos de la compañía Walmart. En esta competición el equipo que terminó en el tercer puesto probó diversas soluciones basadas en redes neuronales [2]. Y de estas se terminaron decantando por una solución basada en el modelo DeepAR el cual se explica con más detalle en el capítulo de fundamentos teóricos.
- En 2017 la compañía Facebook publicó como software libre el modelo Prophet que se usa en la propia compañía para la predicción de futuros elementos en una serie temporal donde el factor humano tiene un impacto importante. Este modelo también se explica con más detalle en el siguiente capítulo de la presente memoria.

Capítulo 2

Fundamentos teóricos

2.1. Aprendizaje automático

El aprendizaje automático (más conocido por su nombre en inglés: *Machine Learning*) es un campo que se centra en el estudio de algoritmos capaces de aproximar procesos sin necesidad de ser explícitamente diseñados para tener en cuenta las peculiaridades del proceso a aproximar.

Para lograr este objetivo, estos algoritmos necesitan muestras de datos para mejorar su desempeño en la tarea. En el caso concreto del aprendizaje supervisado, estos algoritmos necesitan los datos dispuestos en pares consistentes en un vector de entrada al modelo y otro vector "etiqueta" que le indica al modelo la salida deseada. Entonces, el proceso de aprendizaje consiste en definir una función de error que indique la diferencia entre lo que da el modelo como salida y la salida real y buscar los parámetros del modelo que minimicen esta función.

Sin embargo, los datos recogidos no suelen ser ideales y contienen mucho ruido. Por otra parte, los modelos de aprendizaje supervisado suelen hacer un buen trabajo aproximando funciones arbitrarias. Esto lleva a un problema habitual en el aprendizaje automático, el sobreajuste, que ocurre cuando se entrena un modelo y este empieza a detectar patrones específicos de los datos usados para entrenar pero que no son patrones generalizables.

Por tanto, a la hora de evaluar el desempeño de un modelo no se debe hacer sobre el mismo conjunto de datos que se ha usado para entrenar. Como el objetivo es que el modelo aprenda patrones generalizables de los datos, una manera de medir este desempeño es aplicando el modelo a un subconjunto de datos de validación que no ha sido usado para entrenar.

Pero usar sólamente dos conjuntos de datos no es suficiente debido a que se van a tomar decisiones para seguir entrenando un modelo tras ver los resultado en el conjunto de validación, por lo que se corre el riesgo de que estas decisiones provoquen que el modelo se sobreajuste a particularidades de estos conjuntos de datos. Por tanto, en general, es una buena práctica usar tres conjuntos de datos y dejar el tercero (conjunto de testeo) para el final del proyecto para tener una medida más fiable del desempeño del

modelo en “el mundo real” cuando se enfrente a datos nuevos.

Estos tres conjuntos de datos se pueden obtener dividiendo un conjunto de datos inicial. Sin embargo, es importante la manera en que se hace esta división. En este proyecto los datos son series temporales y se quiere predecir valores futuros, por tanto, la partición de los datos no puede ser arbitraria, se tiene que respetar el orden en el tiempo y los conjuntos de validación y testeo tienen que ser de datos posteriores al conjunto de entrenamiento para que la evaluación de un modelo sea fiel a como se quiera usar.

Aún así, es importante notar que es posible que un modelo que dé buenos resultados en el conjunto de testeo no tenga un desempeño tan bueno en la práctica o puede que con el tiempo su rendimiento baje. Esto puede pasar porque el modelo ha sido entrenado con unos datos que no reflejan suficientemente bien la realidad o que la realidad ha cambiado con el tiempo. Por tanto es importante monitorizar el rendimiento del modelo y volver a entrenarlo si hace falta.

2.2. Series temporales

Las series temporales son secuencias de observaciones tomadas en intervalos de tiempos regulares y ordenadas cronológicamente. Una serie temporal puede presentar cuatro componentes distintos como se puede apreciar en la figura 2.1:

- La tendencia indica un incremento o decremento de la serie a largo plazo.
- La variación estacional es un movimiento que ocurre durante un tiempo y en una frecuencia fija. Generalmente debido a factores como la estación del año o el día de la semana.
- La variación cíclica es una oscilación de frecuencia no fija que ocurre generalmente durante un período de tiempo mayor a un año.
- El ruido es una variación debido a fenómenos aislados y que no presentan ninguna regularidad.

2.3. Regresión Lineal

La regresión lineal es un modelo con mucha historia dentro del campo de la estadística. Sin embargo, nos centraremos en la regresión lineal desde la perspectiva del aprendizaje automático. Este es un modelo sencillo que representa el mejor hiperplano por el que pasan todos los datos de entrenamiento posibles:

$$f(x) = Wx + b$$

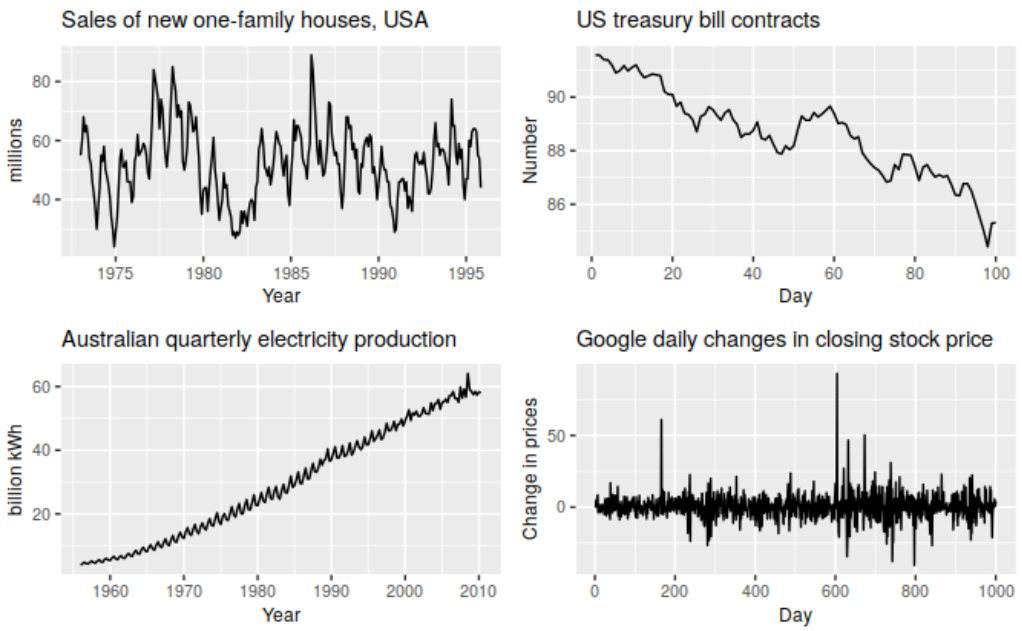


Figura 2.1: Ejemplo de series temporales

- La serie temporal de arriba a la izquierda presenta una variación estacional además de cíclica.
- La serie a su derecha presenta solamente una tendencia decreciente.
- La serie de abajo a la izquierda presenta tanto una variación estacional como una tendencia creciente.
- Finalmente, la última no presenta ningún comportamiento regular.

Imagen perteneciente al libro "Forecasting: Principles and Practice" [3]

Donde x es el dato de entrada, es un vector con tantas dimensiones como atributos tengan los datos. W es una matriz de parámetros que tiene que aprender el modelo. Y b es el término independiente que se usa para que el modelo pueda ser más general. También se puede expresar de la siguiente manera haciendo que el término independiente sea parte de la matriz de parámetros y teniendo como entrada un vector con una dimensión extra que es un uno constante:

$$f(x) = Wx$$

Para entrenar este modelo generalmente se usa la suma de errores al cuadrado (SSE) como función de error a minimizar.

$$SSE = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

Donde los x_i son vectores de entrada al modelo y los y_i son las etiquetas asociadas a cada x_i.

Se puede encontrar una solución analítica para obtener los parámetros:

$$W = (X^\top X)^{-1} X^\top Y$$

Donde X es una matriz con todos los datos de entrenamiento e Y otra con todas las etiquetas.

Sin embargo, esta solución involucra invertir una matriz y esto es muy costoso computacionalmente cuando se tienen muchos datos de entrenamiento y/o muchos atributos de los datos. Por ello normalmente se recurre a métodos que minimicen la función de error de manera iterativa como el descenso del gradiente que se explica en más detalle en el apartado 2.8.1

2.4. ARIMA

ARIMA es un modelo que se ajusta a una serie temporal estacionaria [1] ya sea para entenderla mejor o para predecir futuros valores de esta serie. Aunque también hay series temporales que se pueden transformar a estacionarias. La operación más común para ello es una diferenciación que consiste en calcular la diferencia entre elementos consecutivos. Teniendo en cuenta que el tiempo se trata como discreto y las secuencias se toman en intervalos regulares, esta operación es equivalente a una derivada discreta:

Por definición, la derivada de una función es:

[1] Una serie temporal se dice que es estacionaria si sus propiedades estadísticas no cambian con el tiempo, es decir, presenta una media y varianza constante. Sin embargo, hay series temporales que no son estacionarias que se pueden transformar a una estacionaria aplicando una operación de diferenciación, equivalente a una derivada con tiempo discreto.

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Como estamos en un dominio discreto los valores son los que se tienen, no se puede tomar el límite, h es simplemente la diferencia entre el tiempo de una observación y la siguiente:

$$f'(x) = \frac{f(x+h) - f(x)}{h}$$

Finalmente, como los intervalos entre una observación y otra son regulares, $h = 1$:

$$f'(x) = f(x+1) - f(x)$$

De esta manera podemos ver intuitivamente por qué aplicar esta operación a una serie con tendencia lineal termina con una serie con tendencia constante. De la misma manera, aplicar dos veces esta operación a una serie con una tendencia cuadrática termina con una tendencia constante.

ARIMA es un acrónimo de *Autoregressive Integrated Moving Average* que indica precisamente los tres componentes del modelo:

- **Autoregresión**: es un modelo de regresión lineal que usa como entradas los p valores anteriores de una serie temporal

  $$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots \phi_p y_{t-p} + \epsilon_t$$

  Donde las y_i son los valores de la serie temporal en el momento i, las ϕ_i y c son los parámetros del modelo y ϵ_t es ruido blanco.

- **Integrado**: es la operación contraria a la diferenciación, la cual es una operación que se realiza para hacer que la serie temporal sea estacionaria. ARIMA espera un hiperparámetro d que es el número de veces que se ha llevado a cabo la diferenciación en la serie temporal.

- **Media móvil**: es un modelo en el que se usa como base la media de la serie temporal y para predecir el siguiente punto se ajusta esta media añadiendo una combinación lineal de los q errores anteriores de la serie temporal

  $$y_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \dots + \theta_q \epsilon_{t-q}$$

  Donde μ es la media de la serie temporal, las θ_i son los parámetros de este modelo y los ϵ_i son los ruidos de la serie temporal.

Dados unos valores para los hiperparámetros (p, d, q), ARIMA estima los mejores parámetros mediante máxima verosimilitud. Es decir, se usa una función que mide lo bien que se ajusta un modelo a los datos y se trata de buscar los parámetros que den el mejor ajuste. El concepto de función de verosimilitud se describe en más detalle en el apartado 2.8.4.

2.5. SARIMA

SARIMA es una extensión de ARIMA para tratar con series temporales con variaciones estacionales. Esto es porque la operación de diferenciación puede no hacer que una serie temporal sea totalmente estacionaria si tiene variaciones estacionales. De manera intuitiva, pensando en variaciones estacionales como funciones periódicas, una derivada de un seno da como resultado un coseno, otra función periódica.

Una manera de borrar el efecto de las variaciones estacionales para tener una serie estacionaria es con SARIMA. Para ello se añade a ARIMA otras tres componentes para tratar con la parte estacional $(P, D, Q)_m$. Donde m indica el período de la variación estacional y (P, D, Q) son los parámetros para el modelo autoregresivo, la operación de integrado y el modelo de media móvil que, en vez de considerar la serie temporal elemento a elemento, sólo tienen en cuenta la serie temporal cada m elementos.

2.6. SARIMAX

SARIMAX es una extensión de SARIMA para tener en cuenta también otras variables. Para ello, estas variables tienen que estar también en forma de serie temporal y se incorporan al modelo como otra autoregresión lineal.

2.7. Facebook Prophet

Prophet es un modelo para predecir nuevos elementos en una serie temporal [4]. En concreto, Prophet está pensado para series temporales donde el factor humano juega un papel importante para entender la serie temporal. Por ejemplo, series temporales con variaciones estacionales debido a días festivos o a eventos como la Super Bowl o partidos de fútbol, series temporales con cambios de tendencias debido al lanzamiento de un nuevo producto o cambios de la manera de recoger los datos, etc. Al especializarse en este tipo de series temporales Prophet está pensado para ser fácil de usar porque los hiperparámetros son fáciles de interpretar incluso por analistas no expertos.

Este modelo consiste en la suma de tres funciones:

$$\hat{y}(t) = g(t) + s(t) + h(t)$$

- $g(t)$ se encarga de modelar la tendencia de la serie temporal. Prophet da la opción de que esta función sea lineal o sea logística, esta última función permite especificar un punto de "saturación" a partir del cual la tendencia deja de crecer. Un ejemplo de esto puede ser el número de personas en un recinto debido a la capacidad máxima del recinto. También permite especificar puntos para indicar cambios repentinos en la tendencia y la flexibilidad del modelo para ajustarse a estos cambios.

- $s(t)$ se encarga de modelar las variaciones estacionales como sumas de funciones trigonométricas. Se puede especificar el número de funciones trigonométricas a usar para ajustar una variación estacional: a mayor número, más se ajusta el modelo pero también está el riesgo de sobreajustarse al ruido.

  También se puede especificar si el "modo" de las variaciones son aditivas o multiplicativas. En el primer modo $s(t)$ se suma a la función de tendencia sin más y sirve para series temporales donde las variaciones estacionales son iguales cada temporada, el segundo caso es para cuando se quiere ajustar a una serie temporal donde el efecto de las variaciones estacionales es mayor cada temporada.

- $h(t)$ se encarga de modelar días festivos y eventos como una serie temporal donde estos días tienen un valor y el resto de días es cero.

Para ajustar el modelo a la serie temporal se buscan los parámetros que maximicen la probabilidad a posteriori (*maximum a posteriori estimate*). Esto es parecido a la función de verosimilitud que se maximiza para las variantes de ARIMA pero con un factor llamado probabilidad a priori que introduce "conocimiento" previo en la optimización. De esta manera es como se introduce en el modelo algunos de los hiperparámetros establecidos por los usuarios.

2.8. Redes neuronales

Las redes neuronales son una familia de modelos de aprendizaje automático que están ligeramente inspiradas en los circuitos neuronales biológicos. Por ello la unidad fundamental de estos modelos se llaman "neuronas" y son funciones que dadas unos valores de entrada, realizan unas operaciones sobre estas entradas y propagan el resultado en mayor o menor intensidad. En concreto, las operaciones que realizan sobre las entradas son dos:

- Una **transformación afín** entre las entradas y los parámetros entrenables de la neurona. Es decir, se hace una suma ponderada de las entradas según la "importancia" que tenga cada entrada para la neurona. Además, se añade una constante (también parámetro entrenable de la neurona) cuyo objetivo es ayudar a que la neurona pueda aproximar una función más general.

- Una función, llamada **función de activación**, sobre el resultado de la transformación afín. El uso de esta función pretende representar la propagación de la señal de las neuronas biológicas. Es decir, el resultado de una neurona puede propagarse hacia otras neuronas en mayor o menor intensidad.

  La función de activación a usar en cada neurona es una decisión que se ha de tomar de antemano. Sin embargo, hay funciones que se usan más que otras, una de las más comunes es el "rectificador" ($ReLU$) y simplemente devuelve la parte positiva de la entrada: $ReLU(x) = max(0, x)$.

En la figura 2.2 se puede ver una representación gráfica de una neurona que recibe tres entradas. Estas pueden ser salidas de otras neuronas o los atributos del dato de entrada al modelo. A su vez, la salida puede dirigirse a otras neuronas o ser parte de la salida del modelo.

Por tanto, componiendo neuronas se puede obtener una red neuronal capaz de aproximar funciones más complejas.

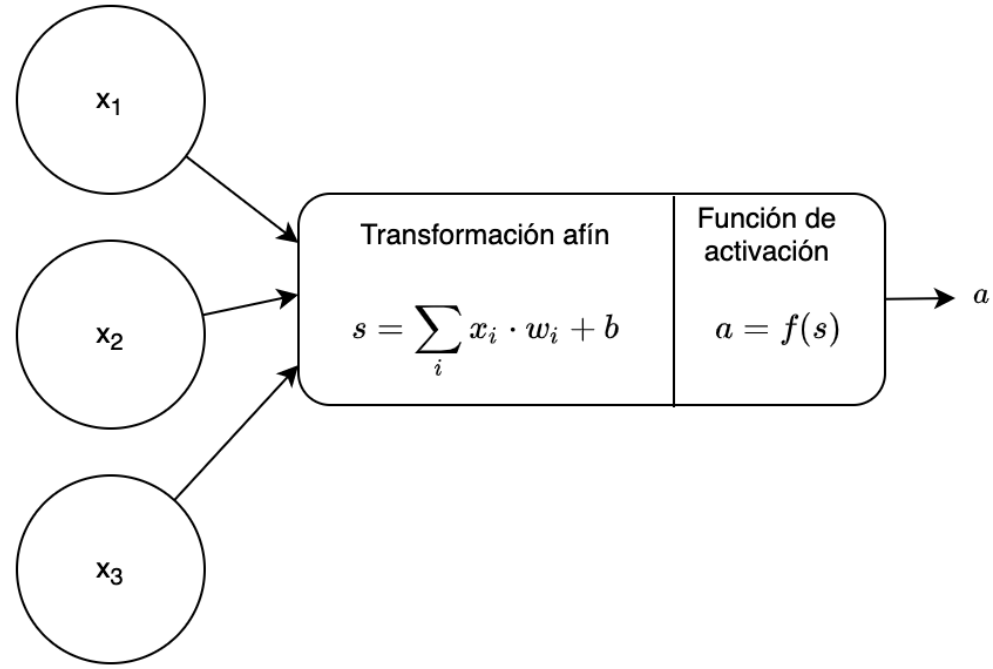


Figura 2.2: Diagrama de una neurona artificial

En esta figura se pueden ver las operaciones que realiza una neurona de manera esquemática. En primer lugar se encuentra el vector de entrada x con tres componentes x_1, x_2, x_3 en este ejemplo. Esta entrada es sometida a una transformación afín con los parámetros entrenables de la neurona y, finalmente, este resultado se pasa por una función de activación.

2.8.1. Red neuronal prealimentada

La red neuronal más sencilla es la prealimentada (*feed-forward* en inglés). Aquí las neuronas se disponen en capas donde las entradas de una capa son las salidas de la capa anterior. Siendo la capa de entrada una excepción pues recibe los datos y los propaga directamente a la siguiente capa.

Toda red prealimentada ha de tener una capa de entrada y una capa de salida. El número de neuronas de estas capas está determinado por el problema a resolver: la capa de entrada tendrá tantos elementos como atributos tengan los datos que se introduzcan al modelo. Y la capa de salida tendrá tantas neuronas como salidas queramos que tenga el modelo.

Por otro lado, el número de capas internas, el número de neuronas en estas capas y la función de activación son decisiones que también se tienen que tomar de antemano

pero son decisiones libres. Es decir, son hiperparámetros que no se establecen durante el proceso de aprendizaje.

Los parámetros que se optimizan durante el proceso de aprendizaje son los que usan las neuronas para la transformación afín.

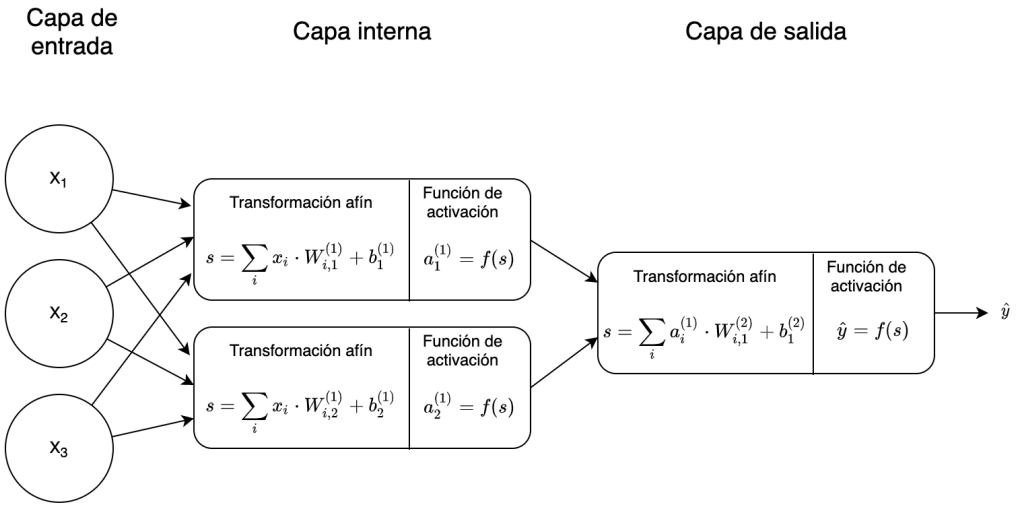


Figura 2.3: Diagrama de una red neuronal prealimentada

En esta figura se puede ver cómo se combinan distintas neuronas para producir una red neuronal prealimentada.

Entrenamiento

Sabiendo cómo funciona esta red neuronal, sólo falta ver cómo es el proceso de aprendizaje. Es decir, cómo obtener los valores de los parámetros de las neuronas a partir de datos.

En primer lugar, cabe destacar dos observaciones:

- Las redes neuronales se pueden expresar como una función compuesta. Por ejemplo, el valor de salida $\hat{y}$ de la red en la figura 2.3, se puede expresar como $\hat{y} = f(s)$, siendo s la transformación afín que se da en esa neurona. Por lo que esto lo podemos expandir a

$$\hat{y} = f(\sum_i a_i^{(1)} \cdot W_{i,1}^{(2)} + b_1^{(2)})$$

  Y, a su vez, las $a_i^{(1)}$ son las salidas de las neuronas de la capa anterior, por lo que podemos seguir expandiendo la función hasta llegar a los valores de entrada.

- Que un modelo aprenda significa actualizar sus parámetros para que sus resultados se asemejan a la realidad. Lo cual se puede expresar como un problema de optimización donde se quiere minimizar el error entre la salida del modelo y lo que se observa en la realidad.

Por tanto, el entrenamiento consiste en:

- 1. **Inicializar los parámetros** de la red de manera aleatoria al comienzo.
- 2. Pasar un conjunto de datos por la red para **obtener las predicciones** para cada una de estas entradas. Cuando el conjunto de datos es muy grande se suele pasar un subconjunto de los datos cada vez, este subconjunto es lo que se conoce como *batch*.
- 3. **Calcular el error** entre los resultados de la red y la realidad.
- 4. A continuación se puede **calcular las derivadas parciales** de la función de error con respecto a cada parámetro de la red. Esto es lo que se conoce como "propagación hacia atrás" o *backpropagation* en inglés. Esta técnica consiste en aplicar sucesivamente la regla de la cadena para hallar las derivadas parciales como es habitual. Lo distinto de esta técnica es que, gracias a la arquitectura de las redes neuronales, el cálculo de las derivadas parciales se puede hacer de manera eficiente, sin necesidad de calcular estas derivadas más de una vez.
- 5. Finalmente, se usa esta información de las derivadas para **actualizar los valores de los parámetros** de manera que se minimice la función de error. Esto suele hacerse con alguna de las variantes del algoritmo descenso del gradiente. La variante base actualizan los parámetros restándole la derivada parcial correspondiente escalada por un factor llamado *learning rate*. Así, los parámetros se actualizan de manera que se minimice la función de error. Y con el *learning rate* se decide cómo de importante es la actualización de los parámetros.

  Este hiperparámetro es muy importante a la hora de entrenar las redes. En primer lugar porque afecta a la rapidez del entrenamiento: un *learning rate* excesivamente pequeño hará que las actualizaciones sean modestas en cada iteración del entrenamiento. Al contrario, un *learning rate* excesivamente grande hará que no se pueda alcanzar un mínimo local.

  Una estrategia común para este hiperparámetro es empezar con un *learning rate* alto e irlo decrementando según se avanza en el entrenamiento. Aún así sigue siendo importante el valor inicial del *learning rate* y el factor de decremento. Otro problema es que tras varias iteraciones el *learning rate* será pequeño y la optimización puede quedar "atrapada" en un mínimo local o en un punto de silla.

  Actualmente, otra estrategia es que el *learning rate* oscile entre unos valores, de esta manera no se tiene un*learning rate* siempre decreciente, por lo que se puede ver como una combinación de intensificación y diversificación del espacio [5].

El entrenamiento es un proceso iterativo donde se repiten los pasos 2, 3, 4 y 5. Cuando se ha pasado por todo el conjunto de entrenamiento se dice que se ha pasado un *epoch*. Generalmente, para saber cuándo parar de entrenar se puede hacer tras un número de *epochs* arbitrario. Otra manera es observando el error del modelo con datos de validación tras cada *epoch*. Cuando el error con datos de validación deja de disminuir durante unos *epochs*, se determina que el modelo se está sobreajustando a los datos de entrenamiento y se para el entrenamiento.

2.8.2. Red neuronal recurrente

Las redes neuronales recurrentes son un tipo de red pensada para tratar con datos secuenciales como series temporales, texto, secuencias de ADN, etc.

Este tipo de red procesa los datos de las secuencias elemento a elemento. Además, en cada momento también se tiene en cuenta información del momento anterior. En la figura 2.4 se puede ver una representación gráfica de este tipo de red que se basa en realizar dos operaciones en cada elemento de la secuencia:

- En primer lugar **calcula lo que se conoce como estado oculto**. Esto es el equivalente a las activaciones de las capas ocultas de una red prealimentada. Y el cálculo también consiste en una transformación afín y en una función de activación. En este caso se añade unos nuevos parámetros para tener en cuenta también la información del estado oculto anterior. En el primer elemento de la secuencia se suele usar como estado oculto anterior un valor arbitrario como ceros.
- A continuación se **calcula la salida de la red** para ese momento. Para ello se usan las mismas operaciones que son habituales en una neurona: transformación afín y función de activación. En este caso la transformación es del estado oculto con unos parámetros entrenables. También se puede devolver como salida el propio estado oculto y si se quiere realizar la operación anterior se puede poner una "capa" de neuronas extra a la salida que se encargue de hacer esta operación.

Un aspecto importante de este tipo de red es que los parámetros son los mismos para todos los elementos de una secuencia, de esta manera estas redes son escalables a un tamaño de secuencia de entrada y salida arbitrario. Otra manera de representar este tipo de red se puede ver en la figura 2.5. Mientras que la figura 2.4 representa una red recurrente de manera compacta, en la figura 2.5 se muestra una red desarrollada para cada elemento de una secuencia. De esta manera también se puede apreciar que el entrenamiento de este tipo de red es muy parecido al de una red prealimentada. En este caso la función de error ha de tener en cuenta todas las salidas de la red. Y, al calcular las derivadas parciales, la propagación hacia atrás también irá hacia atrás "en el tiempo" por la dependencia de cada estado oculto con el anterior.

El mayor inconveniente de entrenar redes recurrentes es el problema del desvanecimiento gradiente. Esto se debe a que, para calcular las derivadas parciales de los primeros parámetros, hay que pasar por toda la red. Es decir, por la regla de la cadena, el cálculo de las derivadas parciales involucra hacer multiplicaciones. De manera que, para hallar la derivada parcial con respecto a los primeros parámetros hay que hacer muchas multiplicaciones en una red profunda[2]. Por tanto, cuando estas multiplicaciones involucran suficientes valores pequeños, la derivada terminará siendo diminuta, haciendo que la actualización del parámetro sea prácticamente irrelevante. Y el efecto final de esto en el modelo es que la red no es capaz de reconocer patrones a largo plazo.

Una manera de mitigar este problema es usando un tipo de red recurrente llamada LSTM.

[2] Entendiendo como red profunda en este caso una red recurrente para secuencias largas.

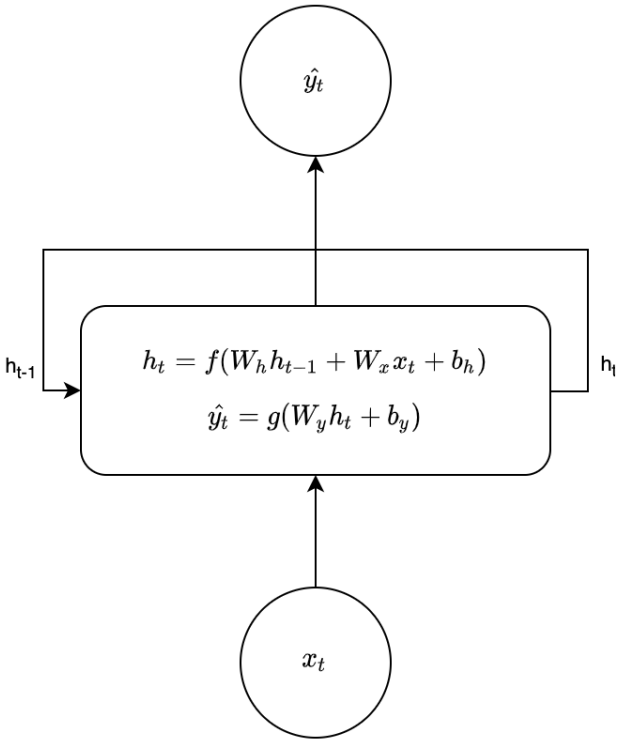


Figura 2.4: Diagrama de una red neuronal recurrente

Representación compacta de una red neuronal recurrente. Esta red toma en cada instante de tiempo un vector de entrada x_t y otro vector que es el estado oculto de la red en el instante anterior h_{t-1}. Para calcular el nuevo estado oculto de la red se aplica una transformación afín seguida de una función de activación. Finalmente, la salida de la red se calcula como la salida de una neurona convencional, usando el estado oculto como entrada.

2.8.3. Long short-term memory (LSTM)

Este tipo de red está basado en las redes recurrentes. Como se puede apreciar a simple vista en la figura 2.6, las redes LSTM difieren de las recurrentes en dos aspectos principalmente:

- Además del estado oculto se usa también un estado de la celda.
- Se realizan más operaciones para cada elemento de la secuencia.

Se puede apreciar mejor el objetivo de estas operaciones en un diagrama que representa el flujo de las operaciones como el de la figura 2.7. Los objetivos son los tres siguientes:

- 1. **Olvidar el pasado irrelevante**: f_t es un vector que ha pasado por una función sigmoide y se multiplica elemento a elemento con el estado de la celda anterior. Como la función sigmoide devuelve valores entre 0 y 1 podemos interpretar este vector como un filtro que quita información irrelevante del estado de la celda.
- 2. **Guardar la información relevante del momento actual**: g_t calcula la información a añadir al estado de la celda. Intuitivamente se puede pensar que esta

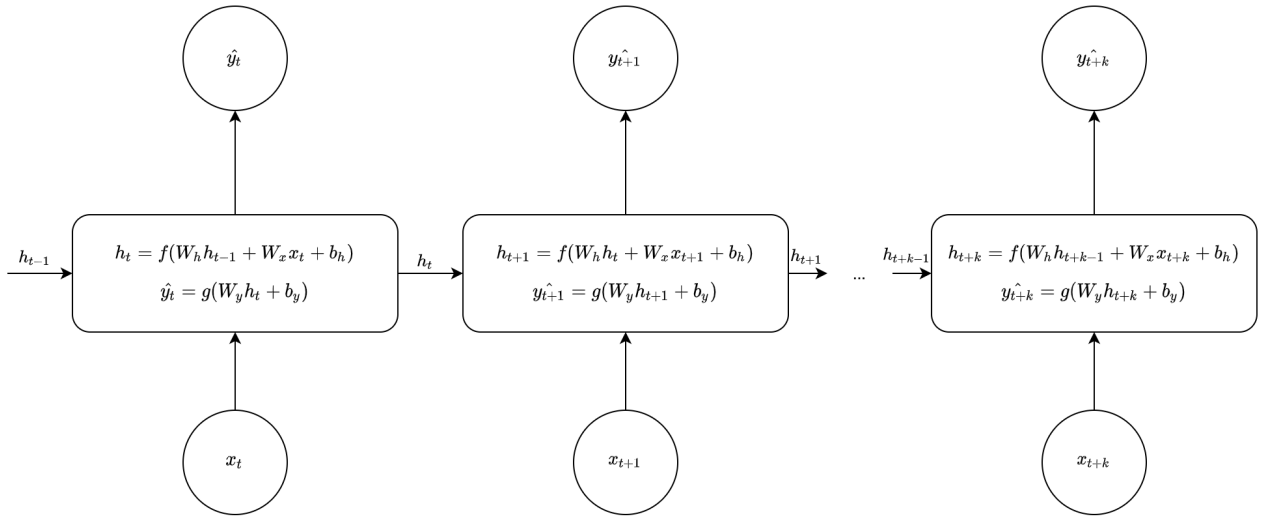


Figura 2.5: Diagrama de una red neuronal recurrente desarrollada

Representación de una red neuronal recurrente desarrollada para cada instante de tiempo.

información se pasa por una tangente hiperbólica como función de activación porque así se tienen los valores variando de -1 a +1, lo cual es ideal para tratarlo más adelante como una diferencia a añadir al estado de la celda. Además, de la misma manera que f_t en la operación anterior, aquí se usa i_t para filtrar la información de manera que sólo queden los elementos más relevantes. Finalmente, este resultado se añade al estado de la celda como una suma elemento a elemento.

- **Actualizar el estado oculto**: finalmente, se ha de calcular el estado oculto que es lo que se usará para el siguiente elemento de la secuencia y como salida de la red en el instante de tiempo actual. Como puede ser interesante no pasar toda la información, primero se usa o_t como filtro para determinar qué partes del estado pasar como salida.

Entonces se puede ver que con las redes recurrentes sencillas el estado de las celdas era único y se sobrescribía en cada momento. Con las LSTM, viéndolo de manera muy simplificada, se calcula el estado como una diferencia del estado anterior.

Dicho de otra manera, con las redes recurrentes sencillas, de manera simplificada, el estado actual era una función del estado anterior:

$$s_t = f(w_t s_{t-1})$$

La derivada parcial del estado respecto al estado anterior tiene un factor w_t que para estados anteriores se multiplica una y otra vez [3]. Esto lleva al problema del desvanecimiento del gradiente si se trata de un factor menor que uno. Y a otro problema llamado explosión del gradiente si el factor es mayor que uno.

$$\frac{\partial s_t}{\partial s_{t-1}} = w_t f'(w_t s_{t-1})$$

[3] Porque las redes recurrentes usan los mismos parámetros para cada instante de tiempo

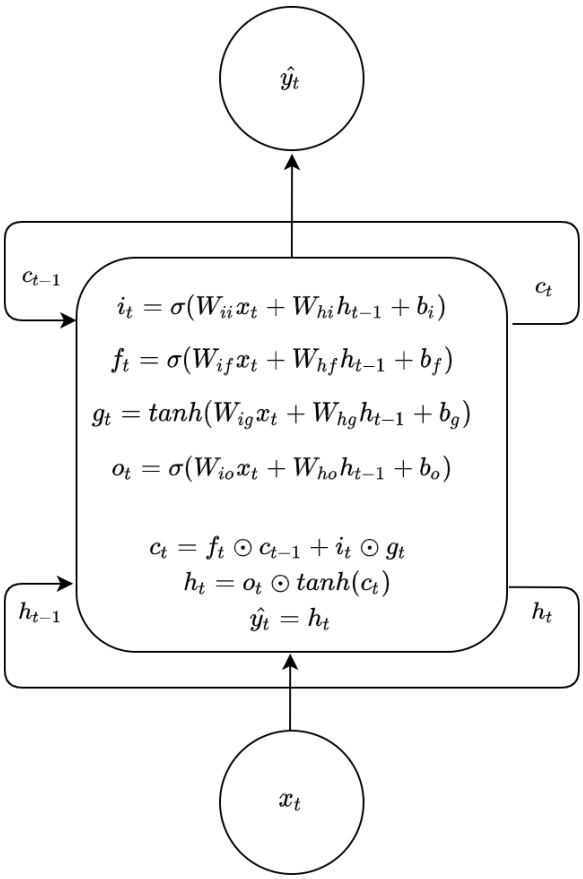


Figura 2.6: Diagrama de una red LSTM

Representación compacta de una red LSTM. Esta red toma en cada instante de tiempo un vector de entrada x_t, otro vector que es el estado oculto de la red en el instante anterior h_t y el estado de la celda anterior c_{t-1}. Para calcular las salidas de la red se aplica una serie de operaciones que se explican en la sección 2.8.3

Por el contrario, de manera muy simplificada, las LSTM y otras variantes de las redes recurrentes pensadas para mitigar el problema del desvanecimiento del gradiente, calculan el estado como un incremento del estado anterior:

$$s_t = s_{t-1} + f(w_t s_{t-1})$$

La derivada parcial del estado respecto al estado anterior tiene ahora una constante que asegura que "fluya" algo de información para actualizar las capas previas aún si el segundo término es pequeño.

$$\frac{\partial s_t}{\partial s_{t-1}} = 1 + w_t f'(w_t s_{t-1})$$

2.8.4. DeepAR

DeepAR es una red LSTM usada para predecir distribuciones de valores en lugar de un único valor para cada elemento de una serie temporal [7]. De esta manera se tiene

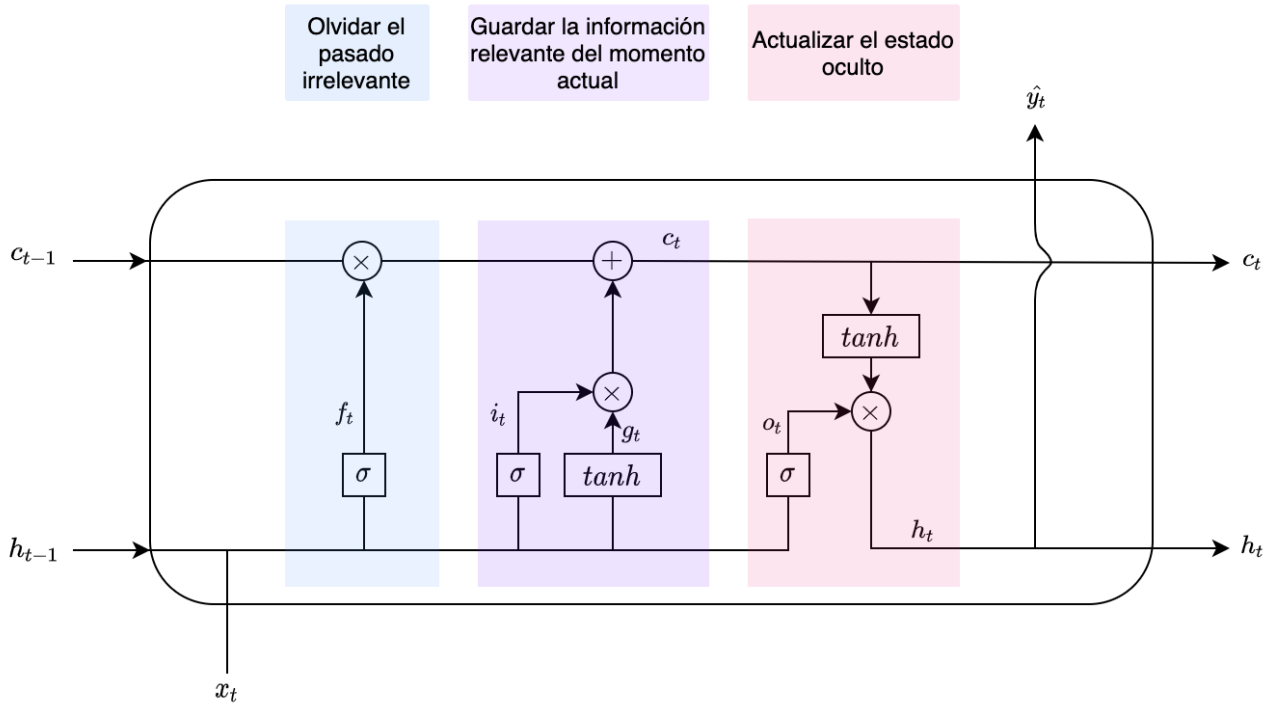


Figura 2.7: Diagrama del flujo de las operaciones de una red LSTM

Representación como flujo de las operaciones de una red LSTM basada en [6].

una distribución de valores para cada momento, por lo que se puede dar un intervalo de predicción en vez de un único punto para que la red pueda expresar variaciones en sus resultados.

Esto se consigue haciendo que la salida de la red sean tantos valores como parámetros tenga la distribución elegida. Por ejemplo, suponiendo una distribución normal, la red dará como salida en cada momento la media y desviación típica de la distribución de la variable objetivo en ese momento.

El otro aspecto a considerar es la función de error porque ahora no se puede comparar directamente la salida de la red con los valores reales. Para esto se toma el concepto de función de verosimilitud de estadística que mide lo bien que se ajusta un modelo a los datos, en este caso el modelo sería la distribución cuyos parámetros devuelve la red. Por tanto, se quiere optimizar la red de manera que las distribuciones que devuelve se ajusten bien a los datos.

Función de verosimilitud

En estadística la función de verosimilitud mide lo bien que se ajusta un modelo a los datos. Dicho de otra manera, dado un modelo, la función de verosimilitud da la probabilidad de que los datos hayan sido observados bajo el modelo. Por ejemplo, asumiendo como modelo una distribución normal $N_{\mu,\sigma}(x)$ de media μ y varianza σ^2, la función de verosimilitud es la probabilidad de observar los datos bajo este modelo. Si asumimos que los eventos que generan estos datos son independientes, se puede definir la función de verosimilitud como el producto de la probabilidad de observar el primer dato x_1, por la

probabilidad de observar el segundo dato x_2, etc.

$$\mathcal{L}(x_1, x_2, ..., x_n|\mu, \sigma) = \prod_i N_{\mu,\sigma}(x_i)$$

Entonces, ahora por conveniencia se puede tomar el logaritmo de la función de verosimilitud para cambiar esos productos por sumas, así se consigue que los valores sean más estables para un ordenador [4]. Finalmente, se usa el negativo de esta función. De esta manera se tiene que los valores menores indican que el modelo se ajusta mejor y así se puede tratar esta función como una función de error.

[4] De manera similar al problema del desvanecimiento del gradiente, aquí se está multiplicando muchas probabilidades (valores pequeños), por lo que el resultado puede ser demasiado pequeño para la precisión que soporta un ordenador.

Capítulo 3

Tecnologías

En este capítulo se detallan las tecnologías principales que se han usado en el proyecto.

3.1. Base de datos

El software que trata la información del centro almacena los datos en una base de datos SQL Server localizado en un ordenador del propio centro. Gracias a una funcionalidad propia de SQL Server, se puede acceder a la base de datos sin necesidad de autenticarse con unas credenciales específicas para ello. En su lugar se usan las credenciales del propio sistema operativo Windows, de manera que una vez iniciada la sesión en el sistema, se puede acceder a la base de datos.

3.2. Tratamiento de datos

Para el tratamiento de datos y entrenamiento de los modelos se ha usado Python como lenguaje de programación por el ecosistema y comunidad que tiene para estas tareas. En concreto se ha hecho uso de las siguientes librerías:

- **Pandas** y **Numpy** para la carga y manipulación de datos.
- **Plotly** para la visualización de datos.
- **StatsModels** para el modelo SARIMAX.
- **fbprophet** para el modelo Facebook Prophet
- **GluonTS** y **Apache MXNet** para las redes neuronales. Se decidió usar esta alternativa en lugar de otras más populares como Tensorflow o Pytorch debido a que GluonTS y Apache MXNet son tecnologías con origen en Amazon al igual que la publicación del modelo DeepAR. Además, aunque GluonTS provee una funcionalidad limitada, ofrece funcionalidades para facilitar el tratamiento de series temporales. Cabe indicar que el modelo DeepAR se llegó a implementar en Pytorch para este

proyecto pero se descartó en favor de GluonTS debido a las dificultades surgidas para implementar el tratamiento de las series temporales para entrenar los modelos.

El código para realizar los experimentos consta de tres scripts, uno para SARIMAX, otro para Facebook Prophet y un último script para las redes neuronales. En estos scripts primero se definen los hiperparámetros para poderlos modificar cómodamente. Entonces, cuando se ejecutan:

- 1. Se leen los datos.
- 2. Se transforman los datos y se dividen en un conjunto de entrenamiento y otro de validación.
- 3. Si el modelo con los hiperparámetros especificados ya ha sido entrenado, se carga del disco duro. En otro caso, se entrena el modelo y se guarda en disco.
- 4. Se usa el modelo para realizar las predicciones del conjunto de validación.
- 5. Se calculan las métricas de los resultados y se realiza la gráfica resultante.
- 6. Se registra el experimento.

3.3. Registro de los experimentos

Para mantener un control de los resultados de todos los modelos entrenados se decidió usar un servicio web llamado Weight & Biases. De esta manera, se puede registrar en un servicio las características del modelo entrenado así como las métricas resultantes como se puede ver en la figura 3.1. Y, además, se puede registrar también la gráfica resultante para visualizar y comparar cómodamente entre los diversos modelos como se puede ver en la figura 3.2.

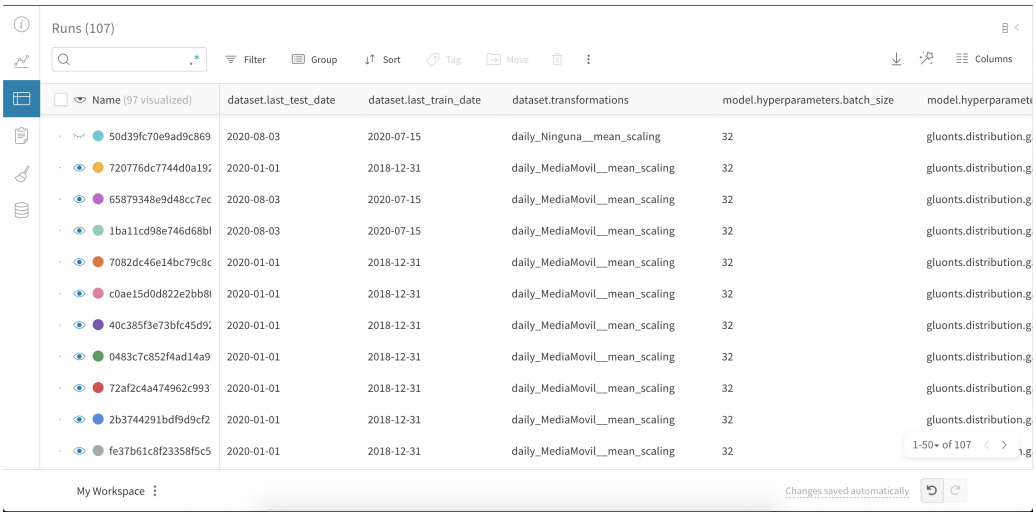


Figura 3.1: Registro de las características de las redes neuronales entrenadas en Weight & Biases

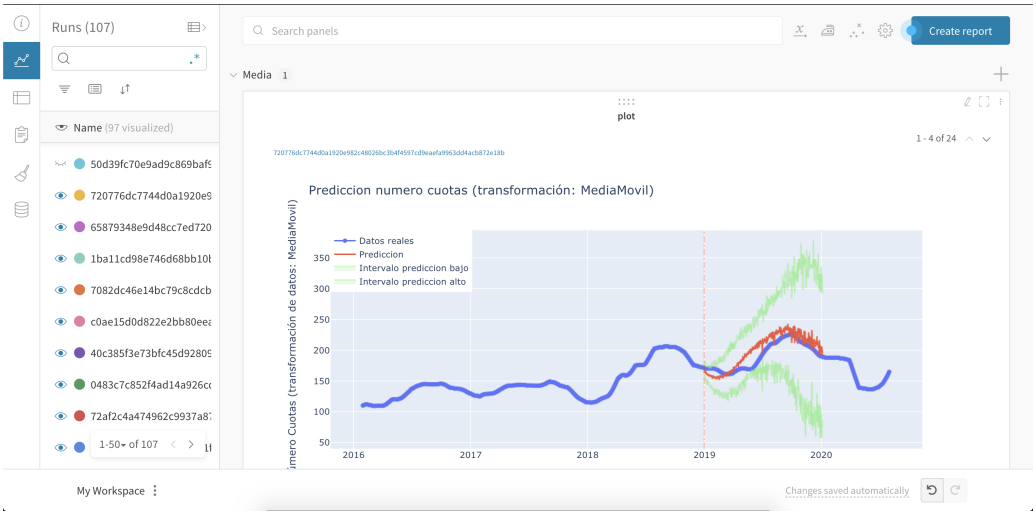


Figura 3.2: Registro de las gráficas resultantes de las redes neuronales en Weight & Biases

Capítulo 4

Desarrollo

4.1. Descripción de los Datos

Para este proyecto se ha trabajado con datos de cuotas de socios. Más concretamente, se trabajó con los 17 tipos de cuotas de interés para la empresa excluyendo, por ejemplo, cuotas para el uso de la piscina o para la asistencia a clases de bailes. Esto es algo que se extraía del software del centro de manera manual como informes. Sin embargo, es información que se encontraba en la base de datos a la que se tenía acceso por encontrarse en el propio centro sin más protección que las credenciales del usuario del ordenador, por tanto, sólo faltaba saber la consulta a hacer a la base de datos. Para ello se hizo uso de SQL Server Profiler, una herramienta pensada para analizar el rendimiento de una base de datos SQL Server. Esta herramienta, de manera conveniente, enseña las consultas que se realizan a la base de datos, por tanto, se usó mientras se generaba un informe de las cuotas de socios para identificar la consulta exacta que se realiza a la base de datos.

Con la consulta identificada, se pudo extraer el número de cuotas de socios de cada día, desde el inicio de 2016 hasta el día 3 de agosto de 2020. Esto constituye una serie temporal con el número de cuotas activas en cada día. Cabe destacar que esta serie temporal se podría descomponer en varias según el tipo de cuota, también se podría añadir datos acerca de los socios que pagaron esas cuotas, otra opción sería trabajar con el importe pagado en lugar del número de cuotas. Sin embargo, debido a la delicadeza de estos datos se ha decidido que para este proyecto se trabaje con el agregado del número de cuotas y sin información de los socios.

Por tanto, se tiene una sóla serie temporal que se puede ver en la figura 4.1. Como se puede apreciar, no es una serie fácil de tratar. En primer lugar, se puede ver un claro impacto en la serie debido a la COVID-19 desde los meses de abril a julio del año 2020. Por otra parte, sólo se cuenta con cuatro años completos de datos útiles donde, además, la serie cambia drásticamente de ser prácticamente constante los dos primeros años a crecer mucho en el año siguiente. Sin embargo, a simple vista se pueden observar patrones en los datos que pueden ser útiles para estimar el número de cuotas de socios en días futuros:

Figura 4.1: Número de cuotas de socios en el tiempo

- Existe una variación estacional a principio de cada mes debido a las cuotas mensuales que no se renuevan con antelación.
- Existe otra variación estacional que comprende los meses de julio, agosto y septiembre.
- La tendencia es constante durante los dos primeros años y creciente en los siguientes.
- Aunque la COVID-19 ha tenido un impacto importante en el negocio, a día 3 de agosto el número de cuotas vuelve a estar en un nivel parecido a los primeros meses del año, que es donde suele encontrarse tras la temporada de mayor número de altas en años anteriores, por lo que no es descabellado intentar realizar estimaciones futuras suponiendo que la situación no vuelve a cambiar drásticamente.

4.2. Descripción del tratamiento de datos

4.2.1. Transformaciones de la serie temporal

Con el objetivo de ayudar a los modelos predictivos a encontrar patrones útiles en los datos, se consideraron varias transformaciones de la serie temporal que se pueden apreciar en la figura 4.2:

Para mitigar la bajada drástica de cada principio de mes:

- **Uso del agregado de los datos por mes**. Sin embargo, esto no se consideró para las redes neuronales pues disminuye considerablemente el número de datos de entrenamiento y las redes necesitan de una gran cantidad de datos para detectar patrones.
- **Suavizado mediante una media móvil de 30 días.**

Para disminuir el efecto de crecimiento tan drástico de la serie temporal:

- **Uso del logaritmo de los datos** porque, si se ve la distribución de los datos en la figura 4.3, hay una ligera cola hacia la derecha, por lo que el logaritmo es una transformación apropiada para hacer estas distribuciones más simétricas al disminuir la distancia entre los valores más grandes y aumentarla entre los más pequeños. Sin embargo, esto no tuvo el efecto deseado en la serie temporal, por lo que se terminó descartando.
- **Uso de los datos diferenciados del agregado del mes**. De nuevo, esto no se consideró para las redes neuronales debido a la disminución del número de datos que supone.

El uso de estas transformaciones se consideró en la práctica como un hiperparámetro más a afinar para mejorar el rendimiento de los modelos.

4.2.2. Variables extra

Además de realizar estas transformaciones, también se añadieron variables extra a todos los modelos excepto Facebook Prophet con el mismo objetivo de facilitar la búsqueda de patrones. Estas variables son creadas a partir de la información temporal y son las siguientes:

- El año empezando por 2016.
- El mes del año escalado por un doceavo para que se encuentre en el rango [0, 1].
- Valores booleanos que indican si el mes de la fecha a predecir corresponde a un mes de invierno, primavera, verano u otoño.
- Otro valor booleano que indica si el mes de la fecha a predecir corresponde a uno de los meses de temporada alta (julio, agosto y septiembre).

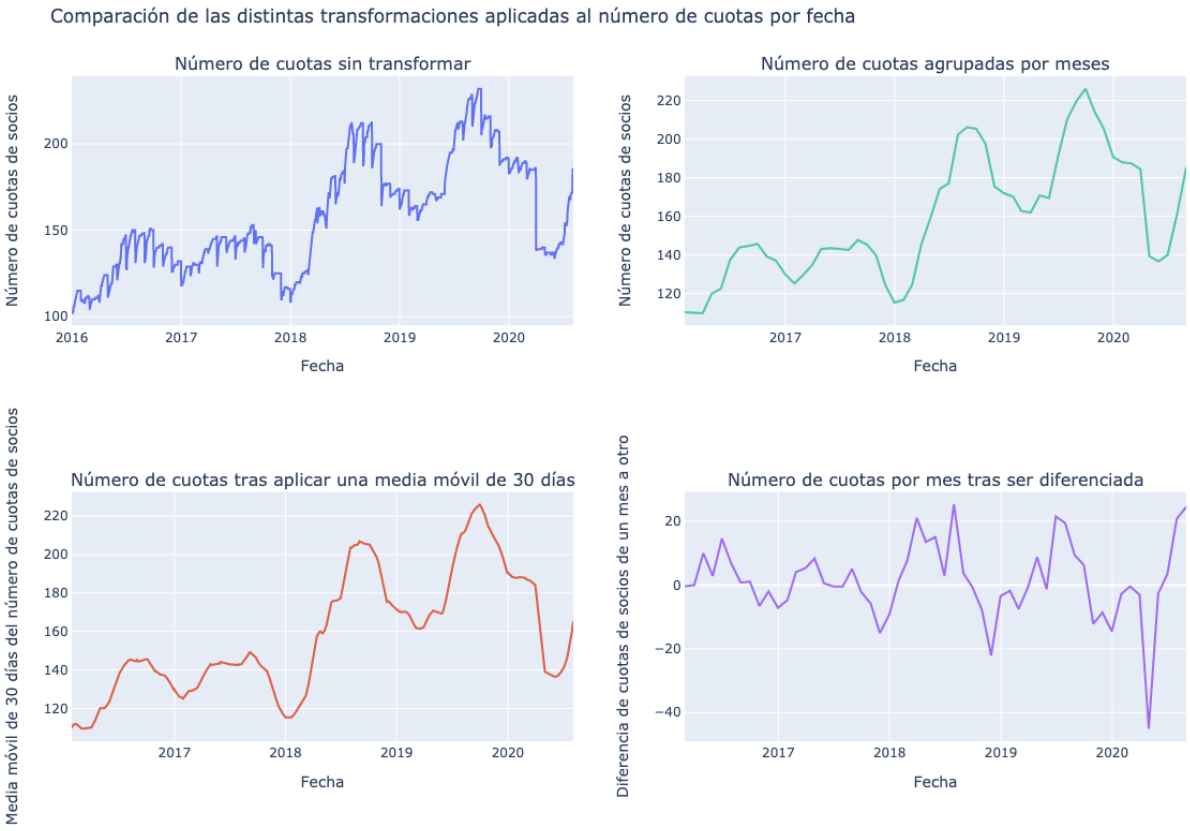


Figura 4.2: Comparación de las distintas transformaciones a la serie temporal de número de cuotas de socios

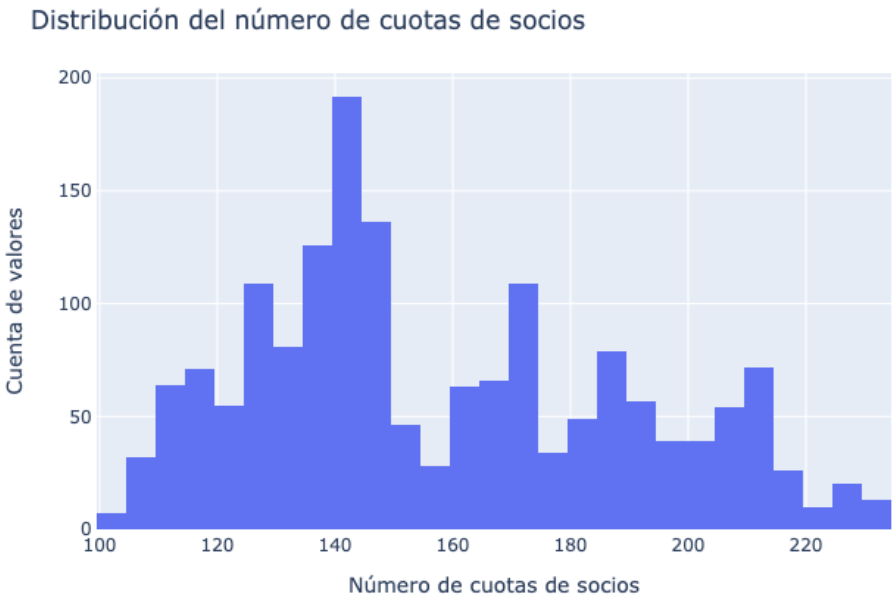


Figura 4.3: Distribución del número de cuotas de socios

4.2.3. Separación del conjunto de datos

Respecto a la separación de los datos en entrenamiento y validación, se realizaron dos fases. La primera sin considerar el año 2020, aquí se tomaron los primeros tres años para entrenar y el año 2019 para validar. El objetivo de esto fue evaluar la capacidad de los modelos ante estos datos sin considerar el impacto de la COVID-19 pues para ello se tienen muy pocos datos. En el caso de las redes neuronales y Prophet, esta fase también sirvió para estimar el efecto general que tendrían los hiperparámetros de una manera más fiable que con un conjunto de validación más pequeño.

En la segunda fase, para entrenar, se tomaron datos hasta el 15 de junio de 2020 y hasta el 3 de agosto para validar, en el caso de datos mensuales sólo se tomó para validar el mes de julio. Cabe notar de nuevo que el conjunto de validación es tan pequeño porque en la práctica se quieren hacer predicciones tras el período de la COVID-19.

4.2.4. Evaluación de los modelos

Para evaluar el rendimiento de los modelos, se han usado las siguientes métricas sobre el conjunto de validación y las predicciones para los mismos días:

- **Error absoluto medio**: es la métrica más sencilla en la que se calcula la diferencia entre la salida del modelo y el valor esperado. Se hace uso del valor absoluto para evitar que diferencias positivas contrarresten parte de las negativas o viceversa.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - f(x_i)|$$

- **Error cuadrático medio**: similar al error absoluto medio pero se usa el cuadrado de la diferencia en lugar del valor absoluto. Con esto se consigue que se de un peso mayor a diferencias mayores.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$$

- **Raíz cuadrada del error cuadrático medio**: la desventaja del error cuadrático medio es que las unidades son difíciles de interpretar. Sin embargo, esto se puede solucionar tomando su raíz cuadrada, de esta manera se tienen las mismas unidades que los valores de entrada: el número de cuotas en el caso de este proyecto.

$$RMSE = \sqrt{MSE}$$

- **Coeficiente de determinación (R^2)**: es una relación entre la dispersión de los datos alrededor del modelo y alrededor de la media. Si el modelo predice todos los valores correctamente entonces el resultado sería 1. Si el modelo es tan desacertado que la media de los datos sería una mejor predicción entonces el resultado sería 0 o negativo. Cabe destacar que la media de los datos sería del conjunto de validación

que en la práctica no se conoce pues imita a un conjunto de datos desconocido al entrenar el modelo.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - f(x_i))^2}{\sum_{i=1}^{n} (y_i - \overline{y})^2}$$

Estas métricas son comunes para evaluar modelos de regresión. Sin embargo, en este proyecto no son suficientes porque los modelos no sólo predicen un único valor, también dan un intervalo de predicción. Además, para la segunda fase de la evaluación se cuenta con muy pocos datos por lo que estas métricas pueden no ser fiables. Por tanto, además de prestar atención a estas métricas para determinar la bondad de un modelo, también se presta especial atención a las gráficas que comparan los datos reales con las predicciones y los intervalos de predicción.

4.3. Descripción de los modelos de predicción y los resultados

4.3.1. SARIMAX

Para SARIMAX sólo se han considerado las dos transformaciones de los datos mensuales pues es muy costoso computacionalmente ajustar este modelo a datos diarios con un período de variación estacional de 365 días.

Los hiperparámetros de SARIMAX son los pasos de autorregresión, integrado y media móvil para la serie completa y la variación estacional, además del período de la variación estacional.

Por tanto, SARIMAX se ha ajustado dejando fijo el período de variación estacional a 12 meses. Y para el resto de hiperparámetros se han probado todas las posibles combinaciones de 0 a 2 valores. La manera de elegir la mejor combinación de hiperparámetros fue mediante el criterio de información de Akaike, esta es una métrica que tiene en cuenta el buen ajuste del modelo a los datos de entrenamiento (mediante la función de verosimilitud) y penaliza esta cantidad según el número de parámetros del modelo para penalizar de cierta manera a los modelos complejos porque pueden sobreajustarse a los datos.

En la figura 4.4 se puede ver las gráficas de resultados de SARIMAX con los mejores hiperparámetros encontrados:

- a) En la esquina superior izquierda se puede ver el mejor SARIMAX ajustado con datos mensuales hasta 2019 y usando como validación datos hasta el año 2020. Los mejores hiperparámetros encontrados son (0, 1, 1)x(0, 1, 1) 12, siendo la primera tripleta los hiperparámetros de autoregresión, integrado y media móvil para la serie completa y la segunda tripleta los hiperparámetros para la variación estacional. Se puede observar que el modelo logra detectar la variación estacional. Además, el

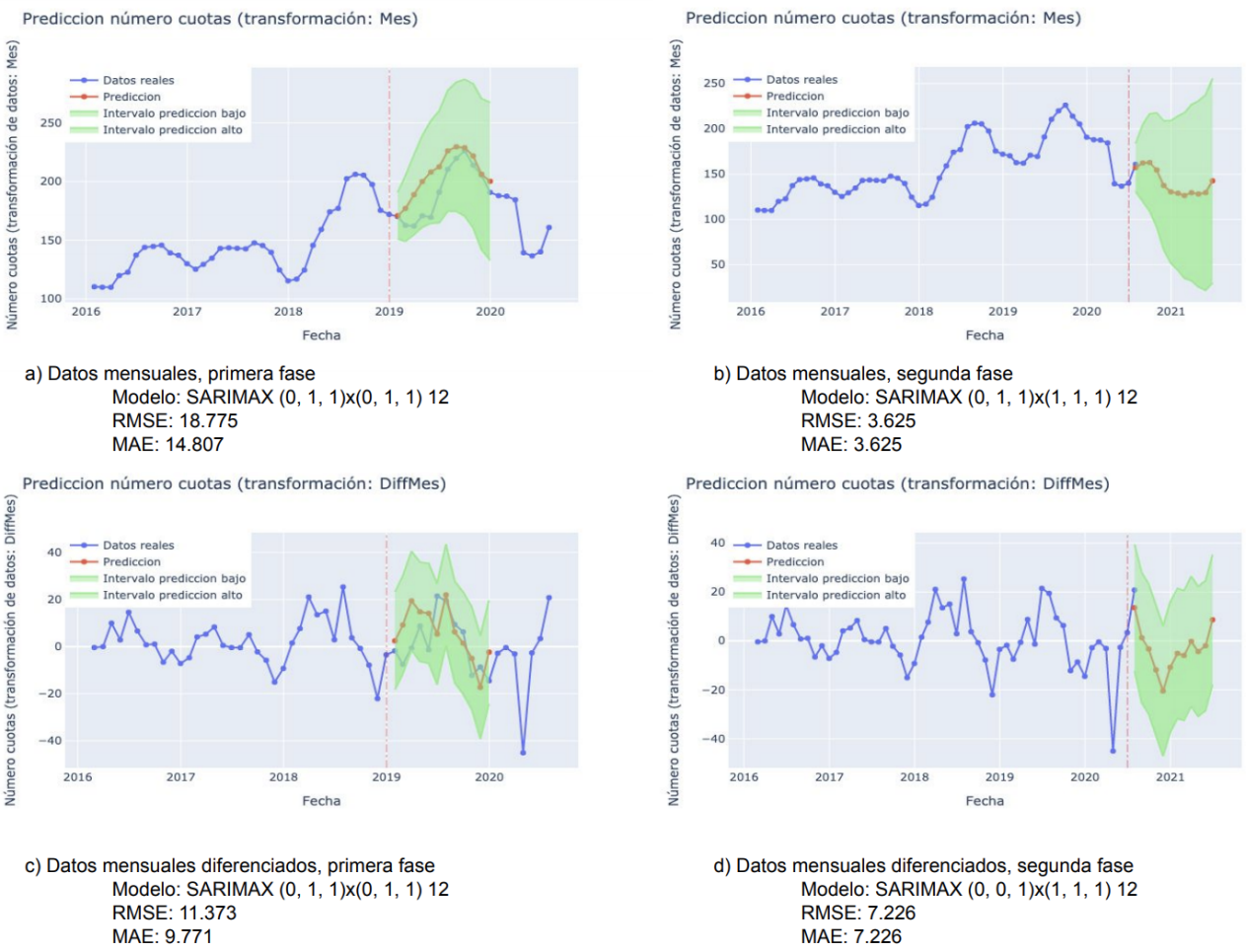


Figura 4.4: Resultados del modelo SARIMAX

número de cuotas reales siempre está dentro del intervalo de predicción aunque el principio del año 2019 fuera muy distinto a lo esperado.

- b) En la esquina superior derecha se puede ver el mejor SARIMAX ajustado con casi todos los datos mensuales, dejando el mes de julio de 2020 para validación. El modelo hace un trabajo excelente: aunque la situación por la COVID-19 tuviese un claro impacto en la serie temporal, la predicción para julio se ha acercado bastante a la realidad con un error de menos de cuatro cuotas. Además, el patrón de las predicciones para el resto de meses es bastante razonable si la situación no vuelve a cambiar drásticamente debido a la COVID-19.

- c) En la esquina inferior izquierda se puede ver el mejor SARIMAX ajustado con los datos mensuales hasta 2019 tras aplicar una operación de diferenciación. El rendimiento es similar al modelo ajustado sin diferenciación: durante los primeros meses de 2019 el valor esperado sigue algo alejado de la realidad pero el valor real siempre se encuentra dentro del intervalo de predicción.

- d) Finalmente, en la esquina inferior derecha se puede ver el mejor SARIMAX ajustado con los datos mensuales hasta junio de 2020 tras aplicar una operación de diferenciación. El rendimiento también es similar al modelo ajustado sin diferenciación.

4.3.2. Facebook Prophet

Debido al elevado número de hiperparámetros que se pueden afinar en este y posteriores modelos no se consideró realizar una búsqueda exhaustiva para la mejor combinación como se hizo con SARIMAX. En su lugar se fueron entrenando los modelos y cambiando los hiperparámetros poco a poco para ver sus efectos, manteniendo siempre un registro de cada resultado en Weight & Biases.

Para la primera parte de las pruebas, los mejores resultados se obtuvieron con datos mensuales y datos diarios suavizados como se puede ver en la figura 4.5. Sin embargo, aunque las predicciones se acerquen a los valores reales, sigue costando predecir bien el principio del año 2019 pues el valor esperado de las predicciones tiene una tendencia ascendente pronunciada. Respecto a esto cabe recalcar que, en la práctica, estos modelos se pueden ir corrigiendo con nuevos datos cuando se observa una diferencia considerable con respecto a la realidad, de manera que según pasa el tiempo se pueden obtener predicciones cada vez más acertadas del final de año.

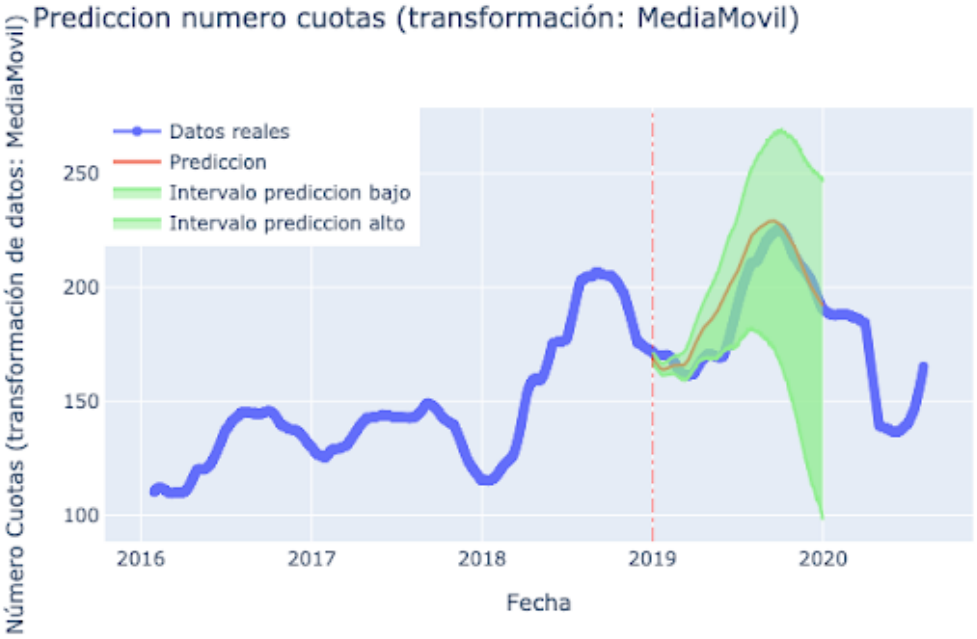
Una desventaja clara en esta fase de evaluación con respecto a SARIMAX es que los valores reales no están siempre dentro del intervalo de predicción. Pero, por otro lado, se ha podido hacer uso de los datos diarios, lo cual con SARIMAX era inviable. Además, los hiperparámetros en Prophet son fáciles de interpretar incluso por analistas no expertos.

De esta parte se obtuvieron dos observaciones:

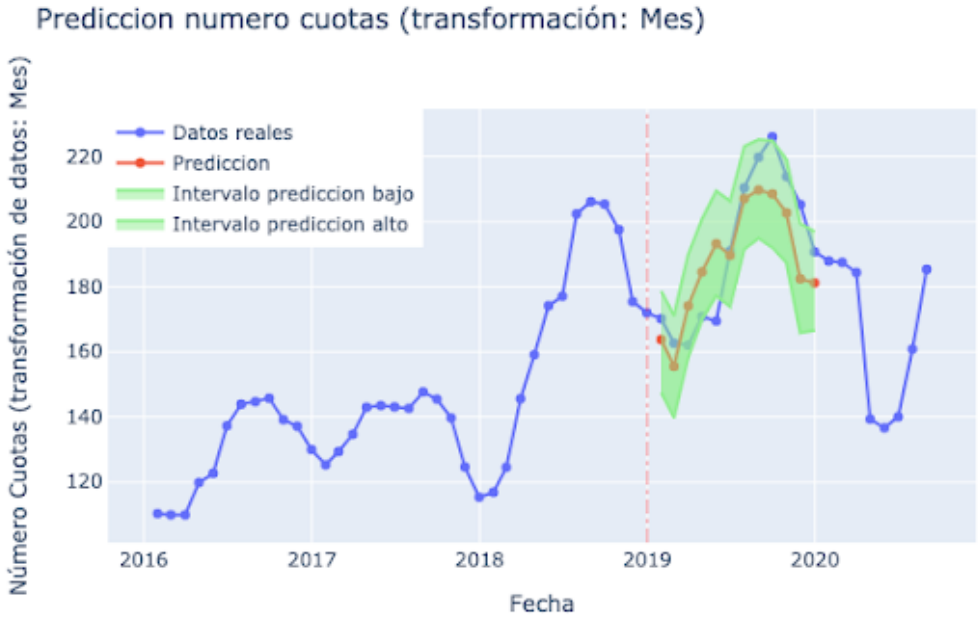
- El tipo de función usada para modelar la tendencia tiene un gran impacto en las predicciones. En caso de usar una función lineal, el modelo estima que al principio del 2019 habrá una subida tan pronunciada como al principio del 2018. En cambio, usando una función logística con 300 cuotas como punto de saturación se tienen unas predicciones menos aceleradas. En la figura 4.6 se puede ver una comparación del efecto de usar una función u otra con datos mensuales.
- Otro hiperparámetro que resultó de ligera ayuda en el caso de los datos mensuales fue el factor de escala de la variación estacional cuyo efecto es modular el ajuste del modelo a las variaciones estacionales, usando un valor menor hace que el modelo no se ajuste tanto a estas variaciones y se obtuvieron predicciones con menos cambios drásticos como se puede apreciar en la figura 4.6.

Con el resto de hiperparámetros no se observaron efectos positivos en los resultados al usar otro valor que no fuera el que está por defecto.

Para la segunda parte de la evaluación, sólo se consideraron datos mensuales y los diarios suavizados. Para los datos diarios suavizados, el mejor modelo es el mismo que en la parte anterior, donde están todos los hiperparámetros por defecto excepto la función logística para ajustar la tendencia. Para los datos mensuales, el modelo resultante es casi el mismo que se usó en la parte anterior, aquí además se tuvo que disminuir el factor de escala de la tendencia, el cual tiene el efecto de hacer que el modelo se pueda ajustar más o menos a cambios de la tendencia.



a) Datos diarios suavizados, modelo Prophet con tendencia logística
RMSE: 11.10
MAE: 8.73



b) Datos mensuales, modelo Prophet con tendencia logística y escalado de variación estacional = 0.5
RMSE: 13.38
MAE: 11.57

Figura 4.5: Mejores resultados del modelo Prophet para la primera parte de la evaluación

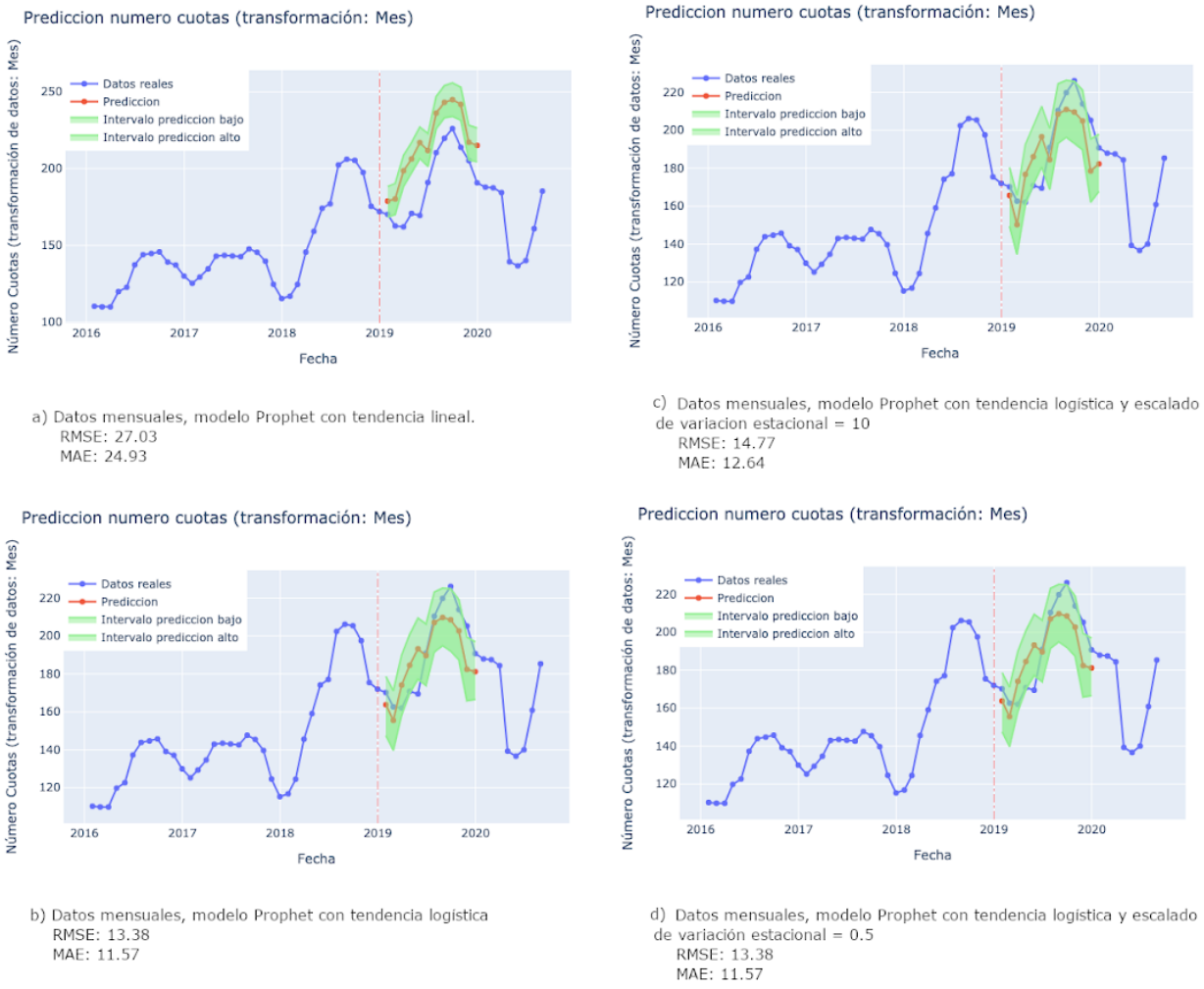


Figura 4.6: Comparación de hiperparámetros de Prophet

Comparación entre la función de tendencia lineal (a) y logística (b). Comparación entre el factor de escala de la variación estacional por defecto (c) y un valor menor (d).

Los mejores resultados se pueden observar en la figura 4.7. La forma de las predicciones parece razonable para el resto de 2020 en ambos casos. Sin embargo, en el caso de los datos mensuales el valor esperado de la predicción se aleja de la realidad en unas 19 cuotas, además, el valor real no está dentro del intervalo de predicción. Por otra parte, para los datos diarios suavizados, los valores reales están dentro del intervalo de predicción pero no parece confiable para futuros días.

4.3.3. Red neuronal prealimentada

La red neuronal prealimentada que se usó está inspirada en el mismo principio que DeepAR: predecir distribuciones en lugar de sólo los valores esperados. Para ello, la red acepta una entrada de tamaño fijo que consiste en tantas entradas como elementos de la serie previos a las predicciones se quieran usar. Esto se denomina "contexto" en la librería GluonTS y la red tiene que dar como salida los parámetros de tantas distribuciones como elementos se quieran predecir.

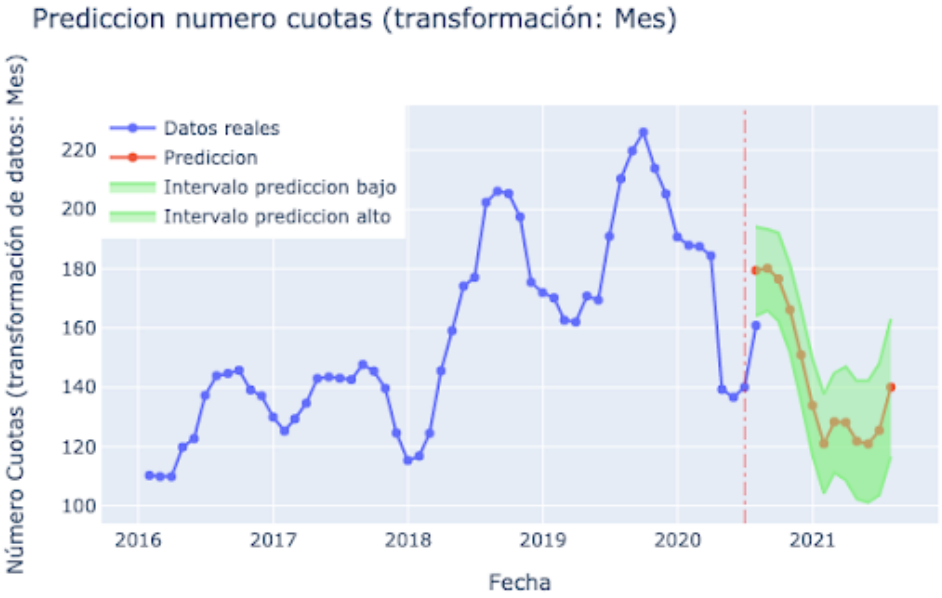
Las redes neuronales tienen una gran cantidad de hiperparámetros que pueden afectar considerablemente al modelo final. Tras realizar numerosas pruebas, la red neuronal prealimentada con los resultados más razonables es una red con dos capas ocultas donde el número de neuronas es creciente de una capa a otra. Esta red toma como entrada los $31 \cdot 3$ elementos previos de la serie de días suavizada para predecir los $31 \cdot 12$ elementos siguientes para tener estimaciones del final de año. Y para el entrenamiento se realizaron 30 epochs, con un tamaño de *batch* de 32 elementos y un *learning rate* fijo de 0.001.

En la figura 4.8 se pueden ver los resultados de esta red para las dos fases de la evaluación. Para la primera parte (a) se puede observar que el modelo tiene un rendimiento similar al de Prophet pues el valor esperado de las predicciones se acerca a la realidad aunque, como punto positivo para la red, todos los valores reales se encuentran dentro del intervalo de predicción. Sin embargo, para la segunda parte de la evaluación (b) el modelo parece ignorar la fecha en la que se encuentra y continúa el patrón ascendente que se dio a finales de agosto, lo cual no parece un comportamiento que se vaya a dar en la realidad.

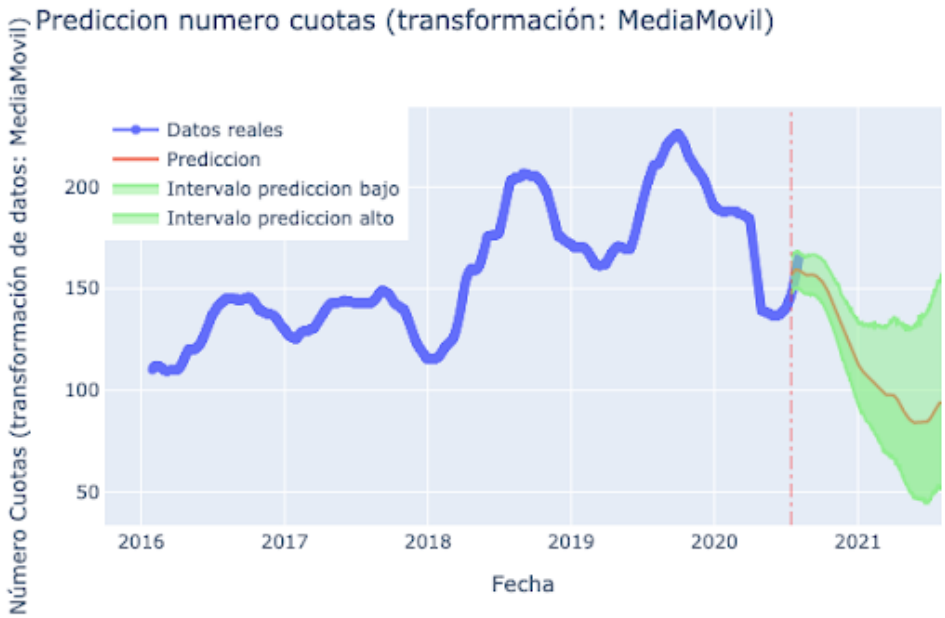
4.3.4. DeepAR

A la complejidad de esta red respecto al número de hiperparámetros, se le suma el considerable coste computacional necesario para el entrenamiento además de otras dificultades debido a la limitada funcionalidad de la librería usada en el momento de realizar el proyecto. Por ello, en este proyecto, no se consideró probar una red con un número excesivo de neuronas (considerando una sóla capa LSTM) ni realizar un alto número de iteraciones para el entrenamiento.

Los mejores resultados observados tras realizar numerosas pruebas se pueden ver en la figura 4.9. Para entrenar la red para la primera parte de la evaluación se tomaron como entrada los $31 \cdot 12$ elementos anteriores a las predicciones y se trató de predecir los $31 \cdot 12$ elementos siguientes para tener estimaciones del final de año. Sin embargo,

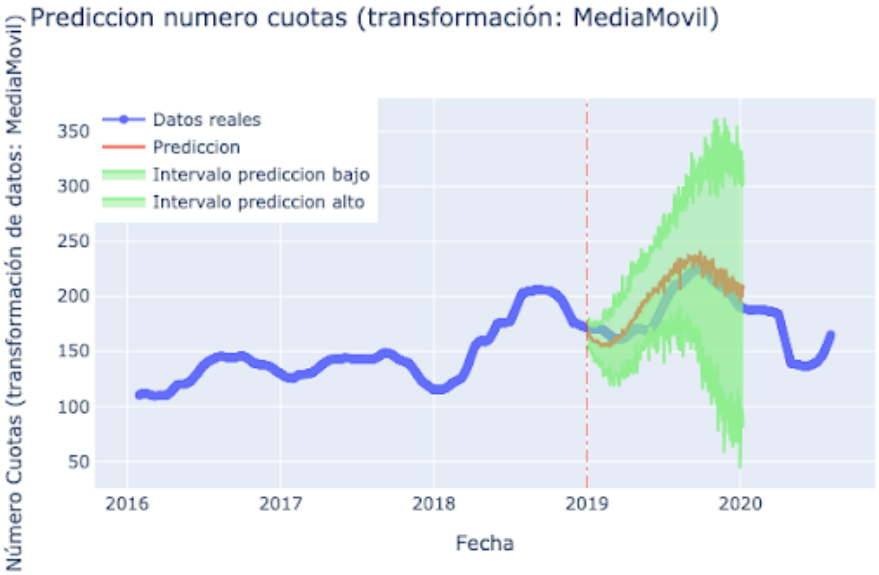


a) Datos mensuales, modelo Prophet con tendencia logística, factor de escala de la variación estacional = 0.5 y de la tendencia = 1
RMSE: 18.59
MAE: 18.59

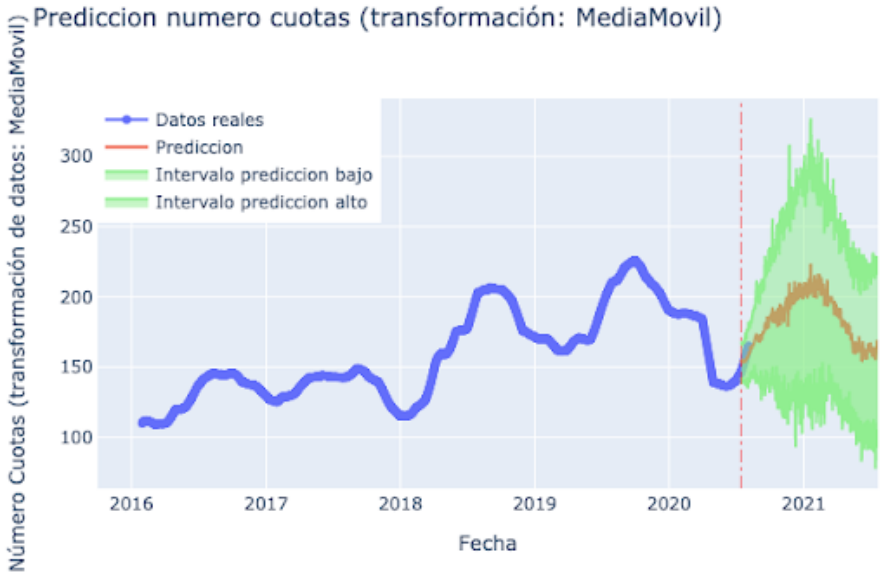


b) Datos diarios suavizados, modelo Prophet con tendencia logística
RMSE:5.67
MAE: 4.92

Figura 4.7: Resultados de Prophet para la segunda parte de la evaluación



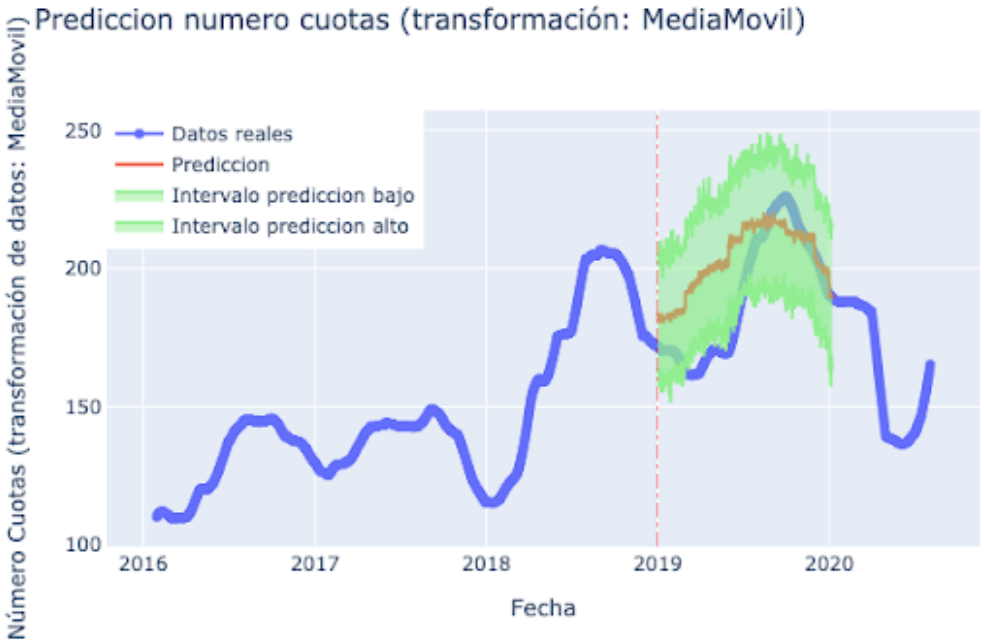
a) Datos diarios suavizados, red neuronal prealimentada.
RMSE: 14.06, MAE: 11.58
nº entradas: 31*3, nº salidas: 31*12
nº neuronas primera capa oculta: 31*6
nº neuronas segunda capa oculta: 31*9
epochs: 30, batch size: 32, learning rate: 0.001



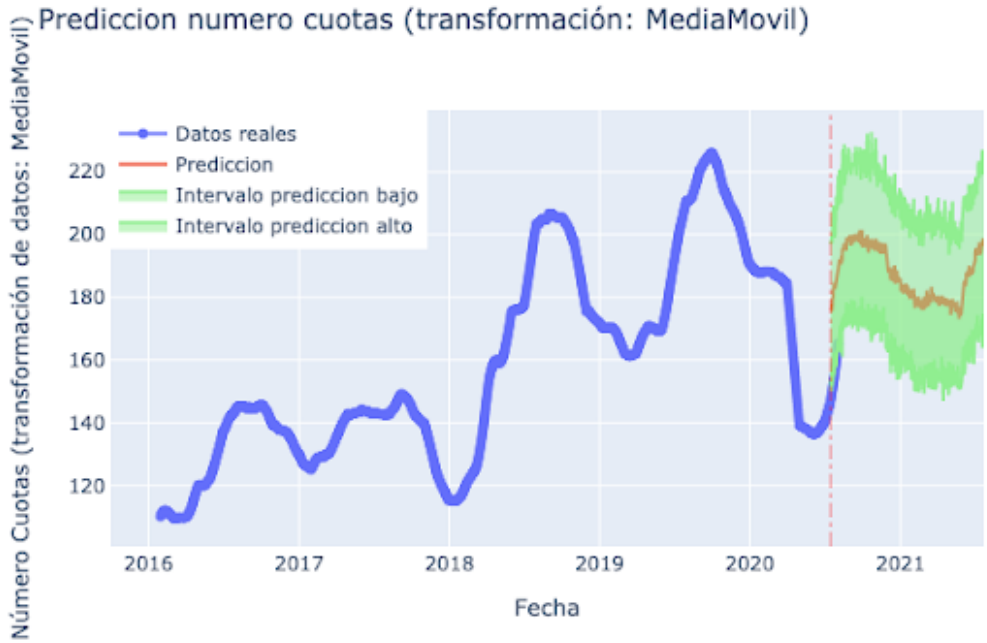
b) Datos diarios suavizados, red neuronal prealimentada.
RMSE: 17.50, MAE: 16.68
nº entradas: 31*3, nº salidas: 31*12
nº neuronas primera capa oculta: 31*6
nº neuronas segunda capa oculta: 31*9
epochs: 30, batch size: 32, learning rate: 0.001

Figura 4.8: Resultados de la red neuronal prealimentada

como se puede ver en la gráfica (a) de la figura 4.9, estas predicciones no resultan ser tan acertadas como las de la red neuronal prealimentada o Prophet, tal vez porque sea difícil capturar todos los patrones útiles de la serie con sólo 50 neuronas. Por otro lado, se puede ver en la gráfica (b) de la figura 4.9 los resultados de la una red similar entrenada para la segunda parte de la evaluación. En este caso se decidió disminuir el número de entradas a $31 \cdot 3$ con el objetivo de que la red tuviese en cuenta los últimos tres meses de los datos para hacer predicciones para que pueda hacer un buen trabajo realizando predicciones tras los efectos de la COVID-19. De esta manera, la forma de los resultados es razonable porque se espera que el número de cuotas se incremente ligeramente en el resto del mes de agosto y septiembre y disminuya hasta el final del año. Sin embargo, el valor esperado de la predicción parece demasiado optimista, muestra de ello es que las métricas dan un error de unas 11 cuotas en este pequeño conjunto de validación.



a) Datos diarios suavizados, DeepAR
RMSE: 18.66, MAE: 15.38
Longitud contexto: 31*12, longitud predicción: 31*12,
nº neuronas: 50
epochs: 20, batch size: 32, learning rate: 0.001



b) Datos diarios suavizados, DeepAR
RMSE: 11.70, MAE: 11.02
Longitud contexto: 31*3, longitud predicción: 31*12,
nº neuronas: 50
epochs: 20, batch size: 32, learning rate: 0.001

Figura 4.9: Resultados de DeepAR

Capítulo 5

Conclusiones y líneas futuras

5.1. Conclusiones

Como parte de un proyecto de auditoría de datos a Radazul Sport Center, en esta memoria de trabajo de fin de máster se recoge el trabajo realizado para el desarrollo de modelos predictivos para la demanda de cuotas de socios.

Este trabajo estuvo motivado por la importancia que tiene la realización de predicciones para la toma de decisiones de una empresa, especialmente tras el impacto de la COVID-19.

Por ello, se plantearon diversos procesos para el tratamiento de datos además de diversos modelos. Se realizaron numerosas pruebas, por una parte sin tener en cuenta el impacto de la COVID-19 para estimar la capacidad predictiva de los modelos con estos datos. Y en una segunda parte, se realizaron predicciones teniendo en cuenta el año 2020 para realizar predicciones hacia el futuro.

De estos modelos, SARIMAX dio unos resultados muy prometedores para datos mensuales. Sin embargo, es un modelo muy costoso computacionalmente para predecir datos con periodicidad diaria. Por ello también se probaron otros modelos, de los cuales Facebook Prophet arroja unos resultados que parecen demasiado pesimistas y DeepAR arroja otros resultados que parecen demasiado optimistas. Por tanto, aunque parezca que por sí solos estos modelos no puedan ser acertados, los resultados de estos dos modelos juntos pueden ser de ayuda en la toma de decisiones.

5.2. Líneas futuras

Hay muchas vías por las que se puede seguir el proyecto, a continuación se enumeran unas posibilidades:

- Descomponer la serie temporal en varias series para cada cuota.
- Usar más datos relacionados con el número de cuotas como podrían ser los accesos de los socios al centro.

- Establecer una métrica que no sólo compare los valores reales con el valor esperado de las predicciones sino que también tenga en cuenta los intervalos de predicción.
- Investigar técnicas para generar datos artificiales en el contexto de series temporales. De esta manera se tendrían más datos con los que entrenar a las redes neuronales.
- Realizar predicciones de otros aspectos de interés para el centro como los ingresos y gastos, las asistencias a las actividades, los accesos al centro, etc.

Capítulo 6

Conclusions and Future Work

6.1. Conclusions

As part of a data audit project for Radazul Sport Center, this end-of-master's report includes the work done to develop predictive models for forecasting the demand for membership quotas.

This work was motivated by the importance that making predictions have for decision making in a company, especially after the impact that COVID-19 has had.

For this reason, plenty of data processing techniques were considered in addition to several predictive models. Numerous experiments were done. On one hand, without taking into account the impact of COVID-19 to estimate the predictive capability of the models with this data. On the other hand, predictions were made taking into account data from the year 2020 to make future predictions.

Of these models, SARIMAX gave very promising results for monthly data. However, it is a computationally expensive model for daily data. Therefore other models were also tested, of which Facebook Prophet gave results that seemed too pessimistic and DeepAR gave other results that seemed too optimistic. Therefore, although it seems that these models cannot be accurate enough by themselves, the results of these two models together can be helpful in decision making.

6.2. Future Work

There are many ways in which the project can be continued, here are some possibilities:

- Decompose the time series into several series for each membership type.
- Use more related data like the access to the sport center.
- Establish a better metric to evaluate the models that not only compares the real values with the expected value from the predictions but also takes into account the

prediction intervals.

- Investigate methods for data augmentation in the context of time series. This way, there would be much more data to feed the neural networks.
- Predict other interesting variables for the center like income, expenses, access to the center, attendances to clases, etc.

Capítulo 7

Presupuesto

En este capítulo se muestran los costes estimados del proyecto. El cual solo se compone de recursos humanos y hardware ya que el software usado es gratuito o ya estaba establecido.

7.1. Costes de Hardware

Tipo	Descripción	Precio
Macbook Pro	Ordenador desde el que se realizó el proyecto	1349€

Tabla 7.1: Costes de hardware

7.2. Costes de Recursos Humanos

Horas de trabajo estimadas	Coste por hora	Total
240h	21€/h	5040€

Tabla 7.2: Costes de recursos humanos

7.3. Costes de Totales

Hardware	Recursos humanos	Total
1349€	5040€	6389€

Tabla 7.3: Costes totales

Bibliografía

[1] K. Benidis, S. S. Rangapuram, V. Flunkert, B. Wang, D. Maddix, C. Turkmen, J. Gasthaus, M. Bohlke-Schneider, D. Salinas, L. Stella, L. Callot y T. Januschowski, *Neural forecasting: Introduction and literature overview*, 2020. arXiv: 2004.10240 [cs.LG].

[2] Jeon. (2020). 3rd place solution in the "M5 Forecasting - Accuracy" competition, dirección: https://www.kaggle.com/c/m5-forecasting-accuracy/discussion/164374 (visitado 10-09-2020).

[3] R. Hyndman y G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2014, isbn: 9780987507105. dirección: https://books.google.es/books?id=gDuRBAAAQBAJ.

[4] S. J. Taylor y B. Letham, "Forecasting at scale", *PeerJ Preprints 5:e3190v2*, 2017. doi: https://doi.org/10.7287/peerj.preprints.3190v2.

[5] L. N. Smith, "No More Pesky Learning Rate Guessing Games", *CoRR*, vol. abs/1506.01186, 2015. arXiv: 1506.01186. dirección: http://arxiv.org/abs/1506.01186.

[6] C. Olah. (2015). Understanding LSTM Networks, dirección: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (visitado 10-09-2020).

[7] V. Flunkert, D. Salinas y J. Gasthaus, "DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks", *CoRR*, vol. abs/1704.04110, 2017. arXiv: 1704.04110. dirección: http://arxiv.org/abs/1704.04110.