2020

KSCHOOL

Comparación de modelos de Machine Learning para la predicción del precio de la electricidad.

MASTER DATA SCIENCE

JOEL DE LA CRUZ FUERTES

Contenido

1. Introducción

1.1. Introducción a la temática elegida

Para el presente proyecto se ha decidido realizar un estudio comparativo entre diferentes modelos de ML[1] para conocer el precio de la electricidad para la tarifa del mercado regulado por el gobierno. Para entender parte de los procesos y variables elegidos, habrá que poner en contexto como se decide el precio.

Primero habrá que conocer las dos tarifas existentes:

- Tarifa 2.0A o Mercado Libre: En el mercado libre, el consumidor elige una tarifa y pagará una cantidad establecida al mes, sin variaciones por el mercado u horarias. Las tarifas quedan establecidas por cada empresa y puede haber de muchos tipos, como tarifas con horas gratuitas. En definitiva, se paga lo que pone en el contrato.
- Tarifa 2.0DHA o Mercado regulado: En el mercado regulado, el consumidor pagará una tarifa variable. El precio de esta tarifa queda marcado por la subasta del mercado regulado. Esta tarifa se conoce como **PVPC**[2]. Se encuentra regulada por el Gobierno, por lo que todo aquel que quede suscrito a esta pagará lo mismo que el resto. Su cálculo queda establecido mediante el RD 216/2014

1.1.1. Cálculo del PVPC

El cálculo de la tarifa PVPC consiste en el sumatorio de tres condicionantes:

1. **Coste de adquisición de la energía:** Precio horario del mercado mayorista de electricidad entre oferta y demanda. Importante para las empresas comercializadoras.
2. **Peaje de acceso y cargos**: Se trata del coste de las redes de transporte y distribución de la energía.
3. **Gestión comercial**: Coste de la gestión de los clientes con derecho al PVPC. Se encuentra fijado administrativamente.

Tanto la gestión comercial como el peaje de acceso y cargos son valores relativamente fijos. La gestión comercial viene fijada administrativamente, y el peaje de acceso y cargos dependerá de la situación de la red en ese momento.

Por tanto, se puede ver que el objetivo principal es ser capaces de predecir cual será el valor de la subasta del mercado.

El Mercado eléctrico español se organiza en torno a un proceso de subastas y de operación del sistema (mercados diarios, intradiarios, resolución de restricciones técnicas, servicios complementarios y de gestión de desvíos).

Debido al marco tan competitivo del mercado de la electricidad, los distintos agentes están forzados a seguir los movimientos del precio del mercado al contado o mercado de entrega inmediata tanto a largo, como a medio y corto plazo para operar de forma eficaz en el mercado. En particular, las predicciones a corto plazo del precio son un factor clave tanto para generadores como para compradores. A la hora de construir un modelo para predecir el precio

[1] ML: Machine Learning
[2] Precio Voluntario del Pequeño Consumidor

de la electricidad es necesario identificar todas aquellas variables o factores que pueden afectar a la predicción.

1.1.2. Tarifas PVPC

Existen tres tipos de tarifas que ofrecer al cliente:

a) Tarifa general: Se trata de la tarifa por defecto. En esta tarifa no hay cambios drásticos en el precio diario, aunque sí que existe variación horaria.
b) Tarifa de discriminación horaria: Tarifa con dos tramos claramente diferenciados:
    a. Un primer tramo nocturno, con un precio mucho más barato.
    b. Un segundo tramo diurno con un precio más elevado.
c) Tarifa de vehículo eléctrico o Supervalle: Se trata de una tarifa muy similar a la discriminación horaria, pero con variaciones en el periodo nocturno.

1.1.3. Motivación

La motivación de este proyecto viene dada por dos motivos principales.

El primero es conocer un poco más en detalle la eficiencia de los modelos cuando se trata de una serie de datos que, a pesar de seguir una línea temporal, depende de otros factores externos como son la generación de electricidad y la demanda de energía.

El segundo es que este proyecto podría ser útil para aquellas empresas que tengan sus beneficios en productos derivados del precio de la electricidad, como pueden ser empresas que dediquen su actividad al mercado mayorista de la energía. También se puede emplear en pequeñas empresas que quieran tener controlados sus gastos mensuales.

Para nuestro proyecto, nos vamos a centrar en la predicción del precio de la Tarifa con discriminación horaria.

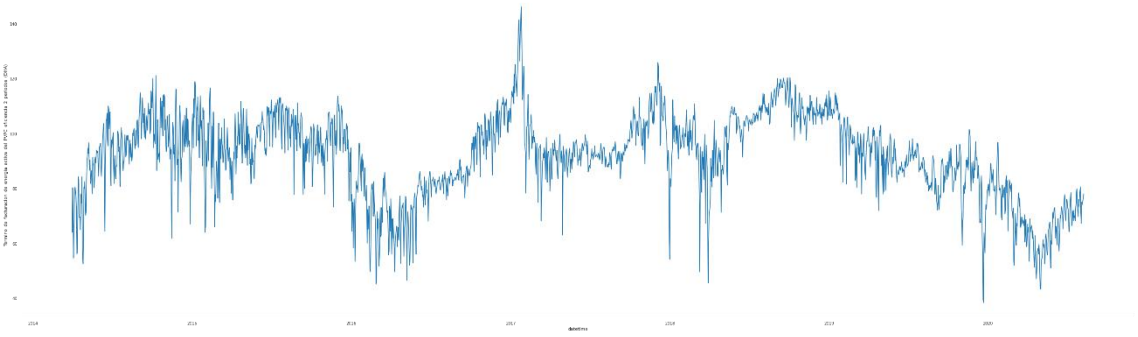

Ilustración 1 Target de nuestro modelo.

1.2. Estado del arte

Debido a la facilidad en la obtención del precio de la electricidad, se trata de un campo ya estudiado en cierta parte. Existen algunos estudios que analizan el comportamiento de las redes neuronales en la predicción, como (Shu Fan, 2006) que estudia como predecir mediante la combinación de un modelo Bayesiano con un SVM[3] para el corto plazo, o (Yongli Zhu, 20) que explica el comportamiento de las redes LSTM[4]. Estos han analizado más el comportamiento de estas redes con muestras pequeñas. Además, ninguno trata el mercado español.

[3] SVM: Support Vector Machine
[4] LSTM: Long Shot Term Memory

También existen otros documentos españoles que sí tratan el mercado español, como (Ugarte, 2017) o (Mate, 2015), pero en ellos no se tratan comparativas de modelos, sino que aplican una técnica de ML y analizan los resultados.

La diferencia entre este proyecto y los anteriores es que en ninguno de ellos se explica el procedimiento completo. Simplemente muestran una tabla con los resultados. Lo que aquí se pretende es analizar más en detalle todos los pasos y su contribución. También se busca encontrar técnicas de Feature Engineering que nos permitan mejorar los resultados. Por último, se quiere realizar una comparación entre diferentes modelos, tanto de series temporales como regresiones, para observar que comportamientos se producen y cuál sería la mejor opción. Por último, se busca un sistema de visualización que permita ver claramente las diferencias entre cada modelo y los resultados obtenidos.

2. Descripción de los datos

2.1. Datos E-SIOS[5]

La página web de E-SIOS ofrece todos los datos relativos acerca de la Red Eléctrica de España (REE). Para obtener estos datos disponen de una API. En ella se puede enviar un correo para obtener un token y acceder a todos los datos que se quiera.

En el código del proyecto se puede ver los pasos necesarios para obtener la información de cada indicador. Los indicadores son los códigos que representan cada variable. A continuación se va a indicar una lista con los indicadores empleados para el estudio del proyecto (remarcadas en negro aquellas que se han acabado empleando como definitivas):

- **1014: PVPC en dos tiempos**
- 1013: PVPC en un tiempo
- 1293: Demanda real
- 10229: PVPC en un tiempo
- 10230: PVPC en dos tiempos
- **600: Precio marginal mercado diario**
- **10027: Previsión de demanda eléctrica**
- 10010: Generación programada de energía eólica
- 10008: Su desglose muestra la energía programada por tipo de producción del Carbón.
- **612: Precio marginal mercado intradiario sesión 1**
- **613: Precio marginal mercado intradiario sesión 2**
- **614: Precio marginal mercado intradiario sesión 3**
- **615: Precio marginal mercado intradiario sesión 4**
- **616: Precio marginal mercado intradiario sesión 5**
- **617: Precio marginal mercado intradiario sesión 6**
- **618: Precio marginal mercado intradiario sesión 7**
- 542: Generación prevista Solar
- 460: Calendario de la demanda diaria eléctrica peninsular según la previsión
- **369: Demanda programada corrección eólica**
- **370: Demanda programada corrección solar**
- 541: Previsión de la producción eólica nacional peninsular
- 805: Precio medio horario componente mercado diario

[5] E-SIOS: Sistema de Información del Operador del Sistema

- **92: Generación Biogás**
- **91: Generación Biomasa**
- **79: Generación ciclo combinado**
- **95: Generación consumo bombeo**
- **88: Generación derivados de petróleo o carbón**
- **90: Generación energía residual**
- **96: Generación enlace baleares**
- **82: Generación eólica terrestre**
- **81: Generación gas natural**
- **87: Generación gas natural cogeneración**
- **71: Generación hidráulica UGH**
- **72: Generación hidráulica no UGH**
- **77: Generación hulla-antracita**
- **78: Generación hulla sub-bituminosa**
- **74: Generación nuclear**
- **86: Generación océano y geotérmica**
- **93: Generación residuos domésticos**
- **94: Generación varios**
- **84: Generación solar fotovoltaica**
- **85: Generación solar térmica**
- **89: Generación subproductos minería**
- **73: Generación turbinación bombeo**

2.2. Datos Yahoo! Finance

Como objetivo adicional, vamos a comprobar si de alguna forma se puede incluir el efecto socioeconómico de la bolsa en el precio de la electricidad.

Para ello vamos a obtener los valores de cierre de la bolsa y vamos a intercalar entre ellos. No es el método más ajustado, pero para una primera aproximación nos puede valer.

Para obtener los datos de Yahoo! Finance existe una librería en Python que es **pandas_datareader** que te permite obtener la información de distintas páginas web, entre ellas Yahoo! Finance, y automáticamente lo pasa a un dataframe de pandas. En este caso, como prueba, vamos a emplear los datos de:

- REE.MC: Red eléctrica española
- %5EIBEX: IBEX35

2.3. Datos WorldBank

También se pueden incluir otros datos socioeconómicos de la página del WorldBank. Tiene una API gracias a la cual se puede obtener la información de otros indicadores que se encuentran en su base de datos.

En este caso la información es trimestral, por lo que no se trata de una información muy fiable para nuestro modelo, al tener un objetivo horario, pero se va a probar a añadirlo a ver si mejora un poco la predicción, ya que no se trata de una variable principal.

Los indicadores empleados del WorldBank son:

- Consumer price index (2010 = 100) (FP.CPI.TOTL)
- Time required to get electricity (days) (IC.ELC.TIME)
- Inflation, consumer prices (annual %) (FP.CPI.TOTL.ZG)
- Employment in industry (% of total employment) (modeled ILO estimate) (SL.IND.EMPL.ZS)

2.4. Vacaciones nacionales

Se ha decidido crear una variable booleana que nos indica cuando el día empleado para la predicción es un día festivo o no. Para ello se han usado los festivos nacionales, no los de cada Comunidad Autónoma. Los días considerados como festivo nacional son:

- 1 de Enero -> Año nuevo
- 6 de Enero -> Reyes - Epifanía del Señor
- 10 de Abril -> Viernes Santo
- 1 de Mayo -> Fiesta del Trabajo
- 15 de Agosto -> Asunción de la Virgen
- 12 de Octubre -> Día de la Hispanidad
- 8 de Diciembre -> Inmaculada Concepción
- 25 de Diciembre -> Navidad

2.5. Datos PIB[6]

Se han extraído los datos trimestrales del PIB de la página web del diario económico Expansión. Para extraer estos datos se ha empleado la librería de **pandas**, ya que tiene un comando que permite leer una página web en html y extraer las tablas de dicha página web.

Al igual que en el caso de los datos de WorldBank, se trata de información trimestral, por lo que es posible que no sea útil a la hora de predecir. Aun así, vamos a comprobarlo en los modelos.

2.6. Datos AEMET[7]

Por último, se han añadido los datos de las temperaturas medias en el país. Para ello se han usado los datos de AEMET. En su API tiene un apartado que, utilizando el token que ellos proporcionan de forma gratuita, te permite acceder a gran cantidad de datos climatológicos.

En el apartado de valores climatológicos, hay una opción que permite extraer los principales datos climatológicos para cada estación en el periodo seleccionado. En nuestro caso, nos

[6] PIB: Producto Interior Bruto
[7] AEMET: Agencia Estatal de Meteorología

interesaría saber la temperatura del país, pero como es bastante complicado obtener esos datos, vamos a sacar una media para la temperatura de cada estación para cada día.

Si observamos como vienen los datos, se ve que para cada estación nos da su temperatura máxima y mínima, además de las horas a las que se producen. Por tanto, como simplificación para nuestro proyecto, lo que se ha decidido hacer es agrupar, para cada hora del día, la temperatura máxima y mínima y hacer la media. Una vez se tiene este dato, se vuelve a hacer una agrupación de las temperaturas, pero para cada día, obteniéndose así también la media de la hora del día. Por último, aproximamos la hora a su valor puntual más cercano, para que cuadre con nuestros datos de E-SIOS.

3. Metodología

3.1. Exploración y manipulación de los datos.

Lo primero de todo será explorar los datos que tenemos. Nuestro objetivo es ser capaces de predecir con un número de días de antelación cual será el precio de la electricidad, por lo que tendremos que manipular los datos recabados para mejorar nuestro modelo.

Además, como también queremos comprobar si se adapta a un modelo de serie temporal, vamos a pasar como índice la columna "datetime", que contiene tanto la fecha como la hora del precio de la tarifa elegida.

3.1.1. Target

Nuestra variable objetivo o *target* es el **Termino_de_facturacion_de_energia_activa_del_PVPC_eficiencia_2_periodos_(DHA)**. Como se puede ver en la Ilustración 1 Target de nuestro modelo., la serie temporal no tiene una tendencia clara, existiendo un pico importante en 2017 con un precio bastante alto (mayor de 140 €/MWh).

3.1.2. Features

Vamos a crear algunas columnas nuevas basadas en los datos del dataframe.

1) One Hot Encoding de las variables categóricas.

Lo primero que vamos a hacer es pasar todas las variables categóricas que tenemos en nuestro dataframe (son todas relativas a la fecha, como hora, tiempo, día, etc.) a variables booleanas. Para ello vamos a usar la técnica de One Hot Encoding, que consiste en que, para cada valor de la columna categórica, se va a crear una nueva columna que le dé 1 si se corresponde con el valor de esa fila, y 0 si no.

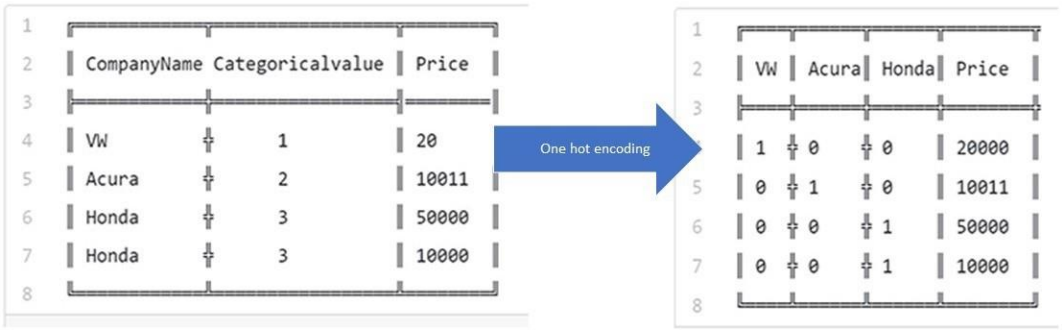

Ilustración 2 Ejemplo de One Hot Encoding. Fuente: https://stackoverflow.com/questions/44961724/one-hot-encoding-categorical-features-to-use-as-training-data-with-numerical-fea

2) Técnica de *cyclical feature.*

Una técnica buena de Feature Engineering es la conocida como "cyclical feature". Esta técnica tiene su base en que las distancias entre ciertas variables son siempre iguales. Pongamos el ejemplo de las horas del reloj, todas ellas se encuentran separadas a la misma distancia de la anterior y de la siguiente, y son "cíclicas", por lo que se podría colocar estas horas en un círculo, asignarle un ángulo a cada hora, y trabajar con el seno y el coseno de esta variable.

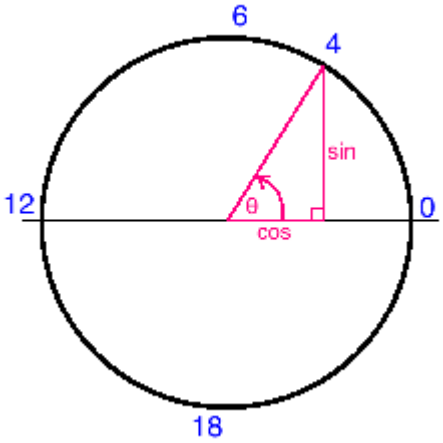

Ilustración 3 Ejemplo de cyclical feature. Fuente: http://blog.davidkaleko.com/feature-engineering-cyclical-features.html

En nuestro caso podemos trabajar con las horas, los meses y los días de la semana con esta técnica, ya que son variables que cumplen los requisitos. Con la variable "Hour" no hay problema, ya que es numérica, pero con el mes y el día de la semana no se puede hacer, así que primero habrá que realizar un Label Encoding, que consiste en sustituir cada valor de la variable con un valor numérico ordenado, y después aplicar nuestra técnica.

3) Rellenar los valores nulos

Al obtener la información de distintas fuentes, existen gran cantidad de valores nulos en nuestro dataset. Por ello, habrá que idear diferentes métodos de rellenar estos valores, ya que no podemos descartar ninguno pues se trata de una serie temporal y necesitamos todos los valores.

a) Valores trimestrales

Para los valores obtenidos del WorldBank y del PIB, al tratarse de datos trimestrales, no se tienen los datos actualizados para el presente trimestre. Por tanto, como no se puede desechar la fila, como único recurso queda utilizar el último valor del que se tiene constancia para cada columna.

b) Valores de bolsa

Los datos que obtenemos de Yahoo! Finance no tienen una hora concreta. Simplemente tenemos el valor de apertura y el valor de cierre, los cuales nosotros asumimos que son las 9 h y las 18 h (el mercado de la bolsa cierra a las 17:30 pero lo aproximamos a las 18 para ajustarlo a nuestros datos). Por tanto, habrá que interpolar los valores entre apertura y cierre aunque no sea el valor real, pero por lo menos se tendrá una pequeña variación de valores.

c) Valores nulos en el precio del mercado intradiario

Existen unos pocos valores nulos para esta variable. Al ser muy pocos, vamos a interpolar linealmente para rellenar los huecos.

d) Valores nulos en la generación:

Existen gran cantidad de valores nulos en las variables de Generación. Esto puede ser debido a que no se tiene programado ningún valor de generación, por lo que vamos a rellenar con 0.

e) <u>Valores para la temperatura</u>

Para la temperatura vamos a realizar la misma consideración que para los valores en bolsa. Para nuestros datos obtenidos de AEMET, nosotros hemos sacado la temperatura mínima del día para una hora en concreto y la temperatura máxima del día para una hora en concreto, por lo que será necesario interpolar linealmente entre ambas horas para obtener la variación nacional de temperatura durante cada hora.

3.2. Análisis de los datos

Una vez que hemos terminado de realizar los procesos de Feature Engineering que hemos considerado, vamos a realizar un análisis de las variables que tenemos en nuestro dataset.

Una técnica bastante útil es comprobar el grado de correlación entre la variable objetivo y el resto de variables, aunque, como ya se sabe, correlación no implica causalidad. También es bastante útil conocer la correlación existente entre las variables que se van a emplear, puesto que variables correlacionadas pueden dar problemas en la predicción.

Ilustración 4 Matriz de correlaciones entre variables y objetivo.

Podemos extraer aquellas variables que tienen una mayor correlación con la variable objetivo:

Feature	Valor de correlación
Demanda_programada_P48_total	0.571216
Precio_mercado_SPOT_Diario	0.545557
Precio_mercado_SPOT_Intradiario_Sesion_1	0.536012
Precio_mercado_SPOT_Intradiario_Sesion_2	0.529253
Precio_mercado_SPOT_Intradiario_Sesion_3	0.556013
Precio_mercado_SPOT_Intradiario_Sesion_4	0.635150
Precio_mercado_SPOT_Intradiario_Sesion_5	0.763426
Precio_mercado_SPOT_Intradiario_Sesion_6	0.682560
Generacion_programada_P48_Consumo_bombeo	0.410226
Generacion_programada_P48_Enlace_Baleares	-0.446626
Generacion_programada_P48_Gas_Natural_Cogeneracion	0.327380
Generacion_programada_P48_Hulla_antracita	0.306885
Generacion_programada_P48_Hulla_sub-bituminosa	0.349222
Generacion_programada_P48_Solar_termica	0.354795
Hour	0.486404
sin_hour	-0.645296
cos_hour	-0.498386
Generacion	0.551228
Demanda	0.569896

Tabla 1 Tabla de correlaciones con la variable objetivo

Además, también es importante echarle un vistazo a la distribución del dataset, para comprobar valores nulos, outliers[8], etc.

Para ello, como nuestro dataset tiene una gran cantidad de columnas (más de 130), vamos a emplear una herramienta muy útil para obtener un informe con todos los datos estadísticos necesarios para conocer nuestros datos.

Esta herramienta es una librería de Python llamada **pandas_profiling**, la cual tiene una función que es **profileReport**. Esta función nos permite crear un informe en formato html con las principales características estadísticas de cualquier dataframe. Se pueden crear dos tipos de informes, uno completo con un estudio exhaustivo del dataframe, y uno más liviano con un análisis básico de distribuciones de las variables y del dataframe general.

En nuestro caso vamos a crear los dos, el completo y el reducido. No se recomiendo utilizar el completo, ya que el peso del mismo es bastante grande y da problemas a la hora de usarlo. Como las correlaciones las hemos visto anteriormente, puede ser suficiente con el uso del informe reducido.

Vamos a poner un ejemplo de los datos que tenemos del mismo para la variable objetivo. Si se desea ver el resto, se puede acudir a dicho informe que viene incluido en el proyecto.

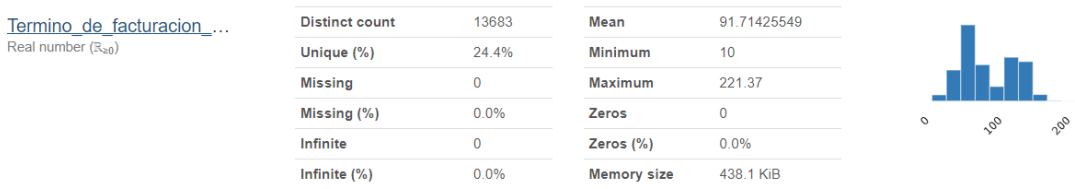

Ilustración 5 Resumen de la variable objetivo

[8] Outliers: datos atípicos en una variable.

Statistics | Histogram(s) | Common values | Extreme values

Quantile statistics

Minimum	10
5-th percentile	38.53
Q1	60.37
median	78.525
Q3	130.37
95-th percentile	153.5385
Maximum	221.37
Range	211.37
Interquartile range (IQR)	70

Descriptive statistics

Standard deviation	39.0453733
Coefficient of variation (CV)	0.4257285096
Kurtosis	-1.234992334
Mean	91.71425549
Median Absolute Deviation (MAD)	35.00868547
Skewness	0.2793347959
Sum	5141868.02
Variance	1524.541176

Ilustración 6 Datos estadísticos de la variable objetivo

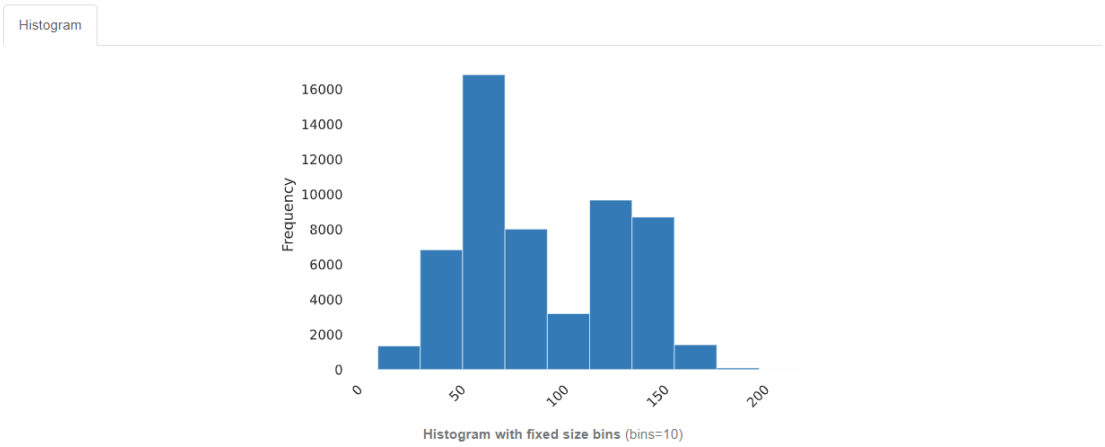

Ilustración 7 Histograma de la variable objetivo

Como podemos observar, es lógico encontrar dos distribuciones normales dentro de nuestro target. Esto es debido a que esta tarifa tiene dos etapas, una etapa diurna donde la electricidad es más cara, y una etapa nocturna donde el precio baja considerablemente. Por ello, se ven dos distribuciones normales dentro de la misma variable.

Empleando aquellas variables que nos indican la hora deberíamos ser capaces de poder representar bien estas variaciones horarias.

3.3. Modelos de Machine Learning

3.3.1. Métricas

Las métricas nos permiten saber la bondad de nuestros modelos. Existen numerosas métricas, incluso cada uno puede crear la suya propia, dependiendo de si se trata de un problema de regresión o de clasificación. Para nuestro caso, una regresión, hemos decidido emplear las siguientes métricas:

- RMSE[9]: Se trata del error cuadrático medio. Nos permite conocer cuál es el error esperado en nuestras medidas. Consiste en elevar al cuadrado el error para evitar que se anulen, así como para penalizar los errores grandes, y después hacer la raíz para pasarlo a nuestras unidades.

[9] RMSE: Root Mean Squared Error.

$$\text{RMSE}(y, \hat{y}) = \sqrt{\frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2}.$$

- R2: El coeficiente R2 nos permite conocer la variabilidad en la explicación de nuestro modelo por las variables independientes.

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

- MAPE[10]: Se trata del porcentaje del error medio absoluto. Nos permite tener un porcentaje del error en nuestra predicción.

$$\text{MAPE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} \frac{|y_i - \hat{y}_i|}{y_i}$$

3.3.2. XGBoost

El primer modelo que se ha empleado para nuestra predicción es un Gradient Boosting, concretamente la librería XGBoost. Este modelo permite ensamblar varios modelos que dan un resultado, y emplear los mismos para obtener la respuesta final.

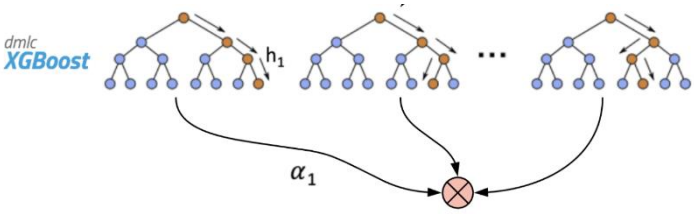

Ilustración 8 Ejemplo de ilustración de XGBoost. Fuente: https://eng.uber.com/productionizing-distributed-xgboost/

Para obtener los resultados, se ha empleado también la función GridSearchCV. Esta función nos permite realizar dos acciones:

- Cross Validation de los datos: Esta técnica consiste en que, para obtener el modelo, no se entrena el algoritmo con los datos y listo, sino que se itera varias veces cogiendo un pequeño trozo del dataset y empleándolo como test para una validación auxiliar.

[10] MAPE: Mean Absolute Percentage Error.

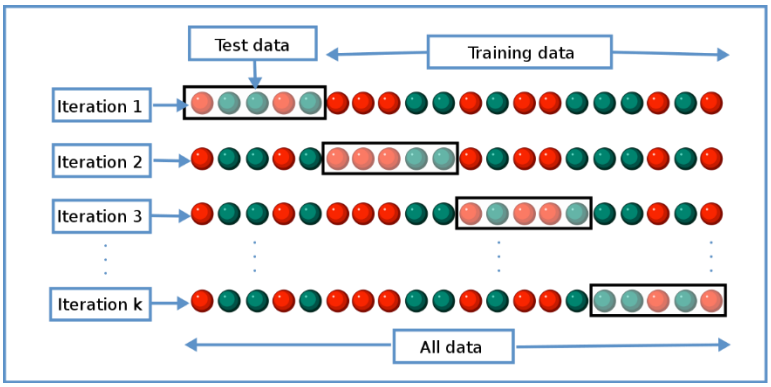

Ilustración 9 Explicación de Cross Validation. Fuente: https://en.wikipedia.org/wiki/Cross-validation_(statistics)

- También permite realizar una iteración del algoritmo para entrenar distintos hiperparámetros y ver cuál es el mejor resultado. Simplemente hay que pasar una lista con los valores que se quiere probar para el hiperparámetro deseado.

Se han realizado dos modelos distintos, aunque el proceso seguido en los dos es el mismo. La única diferencia radica en la forma de dividir los datos para obtener el resultado:

a) Modelo XGBoost con los datos separados en train y test de manera aleatoria.

Con este modelo se quiere comprobar si con un "*random split*" nuestro modelo funciona mejor. Para ello, se va a seleccionar un 80% de train 20% de test, y dentro de ese 80% de train dividiremos un 80% para train puro y otro 20% para validación del algoritmo. Esta validación es simplemente como comprobación.

b) Modelo XGBoost con los datos separados en train y test en el tiempo.

Con este segundo modelo buscamos observar si, tratándolo como un algoritmo de serie temporal, es posible que obtengamos una predicción precisa para los siguientes X días deseados.

Para nuestro proyecto, vamos a dejar todo en función de los días, de tal forma que, simplemente eligiendo en la primera celda el número de días que se quiere predecir con antelación, se puede obtener el resultado de ambos modelos corriendo el resto de celdas.

3.3.3. Serie temporal

La segunda aproximación que queremos realizar es observar si con un algoritmo de serie temporal obtenemos una predicción fiable. Para ello, estos algoritmos se encargan de observar las tendencias y las variaciones estacionales (ya sea variación diaria, semanal, mensual, etc.) de la serie temporal e intentan ajustar el modelo.

El uso de estos algoritmos es relativamente sencillo, ya que lo único que necesitan es una fecha y un valor que predecir. El problema viene a la hora de elegir los parámetros adecuados. Se debe realizar un estudio estadístico de la tendencia, la heterocedasticidad, etc.

Por falta de tiempo, no se ha podido realizar un estudio tan intensivo acerca de los componentes de estos algoritmos. Por ello, estos modelos se van a considerar que podrían ser mejorables con un estudio estadístico de los mismos y la posterior confirmación de la validez de los modelos.

a) Facebook Prophet

Facebook Prophet es un algoritmo diseñado por Facebook y empleado para predicción de series temporales. Su uso es bastante sencillo, puesto que simplemente tienes que iniciar el modelo y ajustarlo con tus datos. Previo al ajuste, es necesario darle el formato adecuado a tu dataframe, que consiste en dos columnas, una con el nombre "*ds*", que contiene las fechas, y otra con el nombre "*y*" que contiene el valor.

En nuestro proyecto vamos a entrenar el modelo con todos los días menos el número de días que se quiere predecir. Así, si queremos predecir los últimos 7 días, separaremos la última semana de nuestro dataset y la emplearemos para comprobar cual es la métrica de nuestro modelo.

b) SARIMAX[11]

SARIMAX es un algoritmo que permite el estudio de series temporales. Realiza un estudio de la media móvil, de la componente autoregresiva, de componentes exógenos y de componentes estacionales. A través de estos factores, consigue ajustar un modelo que se adapta a las tendencias y estacionalidades de nuestra serie.

En nuestro caso vamos a emplear la función **auto_arima** de la librería **pmdarima** para así obtener automáticamente los parámetros deseados para nuestro modelo, de otra forma habría que ir probando manualmente con modelos más simples (MA, AR) hasta encontrar dichos parámetros e incluirlos al final en nuestro algoritmo.

Para este modelo lo único que necesitamos es un índice en formato *datetime* y una columna que tenga el valor deseado de nuestra serie temporal. Además, en nuestro caso, le añadiremos una variable exógena como son las vacaciones nacionales.

3.3.4. Red neuronal

El último modelo implementado para predecir nuestra variable objetivo son las redes neuronales. Se trata de algoritmos de inteligencia artificial que tratan de predecir, a través de un entrenamiento recursivo de los datos de training, el objetivo.

Estos algoritmos son ampliamente empleados en tareas de simulación de los sentidos humanos, como pueden ser modelos de detección visual. Pero su versatilidad nos permite emplearlos también en tareas de clasificación o regresión.

Existen numerosos tipos de redes neuronales, pero nosotros nos vamos a centrar en dos de ellas:

a) Red neuronal LSTM

Las redes LSTM son redes especiales por su forma de trabajar. Estas redes se basan en módulos con varias capas internas en las que se actualiza la información almacenada. De esta forma, no solo tiene en cuenta la información nueva que le entra, sino que emplea también la información anterior.

[11] SARIMAX: Seasonal Autoregressive Integrated Moving Average Exogenous model

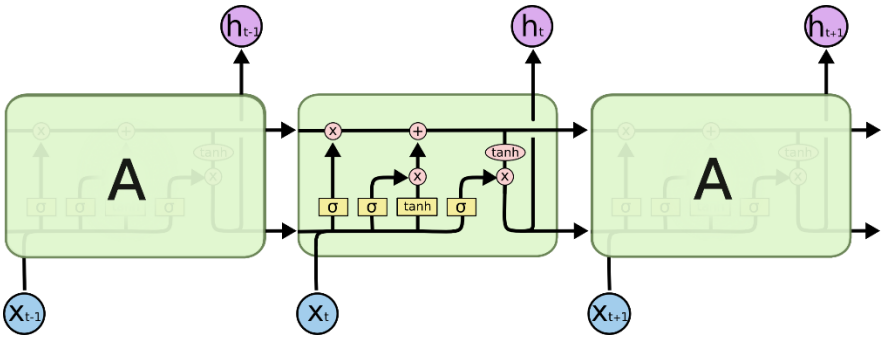

Ilustración 10 Esquema de un módulo LSTM. Fuente: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

En nuestro caso, la información de entrada va a ser un array correspondiente a 7 días anteriores a la predicción, siendo 7 días también el número de días a predecir. Se ha decidido emplear solamente 7 días ya que por recursos y tiempo de entrenamiento no hay otra opción.

b) DNN: Dense Neural Network

Las redes neuronales densas son redes cuyas capas están totalmente conectadas entre sí, de forma que los inputs de cada capa se relacionan con todas las neuronas de la capa siguiente.

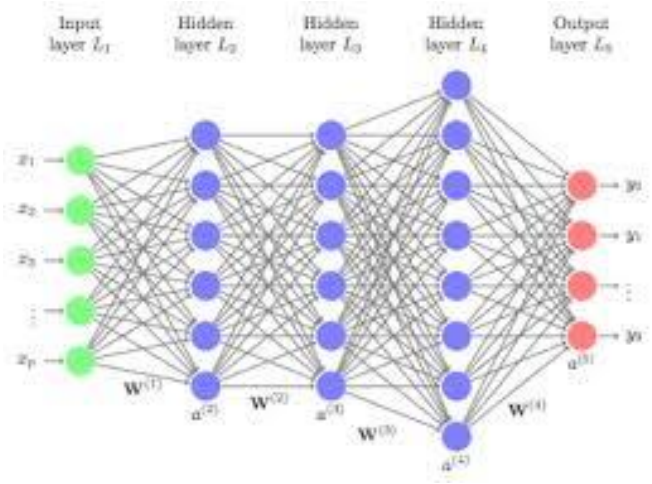

Ilustración 11 Estructura de una DNN. Fuente: http://uc-r.github.io/feedforward_DNN

Con este tipo de red neuronal lo que tratamos de obtener es el resultado de un modelo de regresión. Nuestro objetivo con esta red es conseguir que, a través de la introducción de las variables seleccionadas, obtengamos un resultado lo más parecido al buscado.

Para decidir que resultado queremos obtener, vamos a emplear la misma técnica que para el XGBoost. Lo que haremos es que, para cada fila, vamos a sustituir el valor de la variable objetivo por aquel que se encuentra X días desplazado en el tiempo. Es decir, si se quiere predecir con 7 días de antelación, lo que vamos a hacer es sustituir el valor del target por el de 7 días en el futuro, de tal forma que con los datos del día X podemos predecir el valor del día X+7.

4. Conclusiones

A continuación, se van a comentar los resultados obtenidos para cada uno de los modelos, así como un comentario final acerca del proyecto.

4.1. XGBoost

Los resultados obtenidos para nuestro regresor con XGBoost son bastante buenos. Para nuestro proyecto, vamos a emplear el resultado obtenido para las próximas 4 semanas, pero se podría elegir el número de días que se quiera y entrenar el modelo de nuevo únicamente cambiando el número de días en la primera celda.

Para todos los modelos con este algoritmo se ha realizado el mismo procedimiento. Se han separado los datos en train y test de la forma correspondiente, y se ha llevado a cabo una búsqueda de los mejores parámetros para dicho modelo. Para ello se ha empleado la función **GridSearchCV**, la cual, además de permitirnos buscar estos parámetros, también nos permite realizar un Cross Validation de nuestro modelo. Los parámetros que se han empleado son:

Parámetro	Valores
objective	reg:squarederror
colsample_bytree	0.3, 0.4
learning_rate	0.1, 0.15, 0.2
max_depth	5, 7, 9
alpha	40
reg_alpha	0.1
n_estimators	200
early_stopping_rounds	15
cv	5
scoring	neg_mean_squared_error
n_jobs	-1 (implica que usará todos los recursos)
error_score	raise

Tabla 2 Valores empleados en nuestro modelo

La explicación para cada uno de estos parámetros se puede encontrar en el siguiente enlace.

Una vez se ajustan los parámetros óptimos para el modelo, solo hay que entrenarlo con los datos de train y comprobar el resultado.

a) Modelo XGBoost con los datos separados en train y test de manera aleatoria.

Para este primer modelo se ha obtenido los siguientes parámetros como los óptimos:

```
Out[11]: XGBRegressor(alpha=40, base_score=0.5, booster=None, colsample_bylevel=1,
                      colsample_bynode=1, colsample_bytree=0.4, early_stopping_rounds=15,
                      gamma=0, gpu_id=-1, importance_type='gain',
                      interaction_constraints=None, learning_rate=0.2, max_delta_step=0,
                      max_depth=7, min_child_weight=1, missing=nan,
                      monotone_constraints=None, n_estimators=200, n_jobs=0,
                      num_parallel_tree=1, objective='reg:squarederror', random_state=0,
                      reg_alpha=0.1, reg_lambda=1, scale_pos_weight=1, subsample=1,
                      tree_method=None, validate_parameters=False, verbosity=None)
```

Ilustración 12 Parámetros del mejor modelo (1)

Para este modelo, los resultados obtenidos son:

RMSE	R2	MAPE
7.162363802848708	0.9564336722878534	8.49905917272538

Tabla 3 Resultados para el modelo con los datos separados aleatoriamente

La gráfica que se obtiene es la siguiente comparando las predicciones con los valores reales:

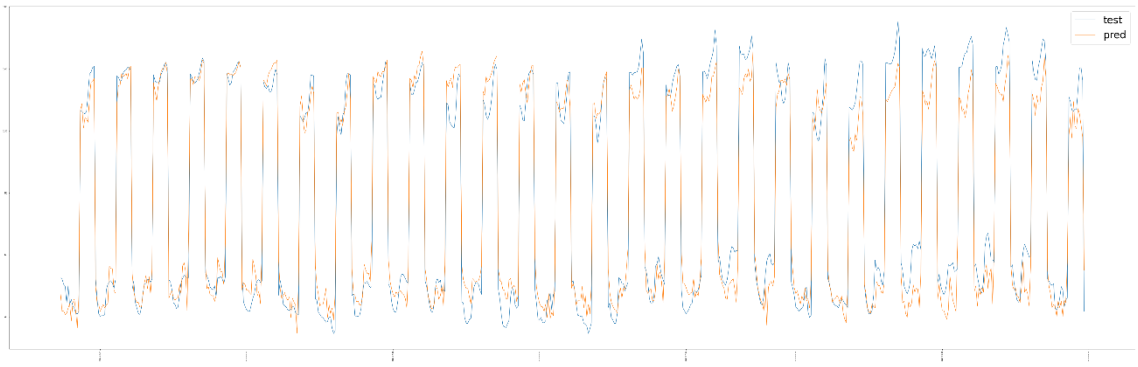

Ilustración 13 Comparación de resultados (1)

b) Modelo XGBoost con los datos separados en train y test ordenados en el tiempo.

Para este modelo hemos obtenido los siguientes parámetros:

```
Out[33]: XGBRegressor(alpha=40, base_score=0.5, booster=None, colsample_bylevel=1,
                      colsample_bynode=1, colsample_bytree=0.4, early_stopping_rounds=15,
                      gamma=0, gpu_id=-1, importance_type='gain',
                      interaction_constraints=None, learning_rate=0.2, max_delta_step=0,
                      max_depth=7, min_child_weight=1, missing=nan,
                      monotone_constraints=None, n_estimators=200, n_jobs=0,
                      num_parallel_tree=1, objective='reg:squarederror', random_state=0,
                      reg_alpha=0.1, reg_lambda=1, scale_pos_weight=1, subsample=1,
                      tree_method=None, validate_parameters=False, verbosity=None)
```

Ilustración 14 Parámetros del mejor modelo (2)

Para este modelo, los resultados obtenidos son:

RMSE	R2	MAPE
12.162310883446125	0.874376641108828	11.328203942961974

Tabla 4 Resultados obtenidos para el mejor modelo con los datos ordenados

La gráfica que se obtiene es la siguiente comparando las predicciones con los valores reales:

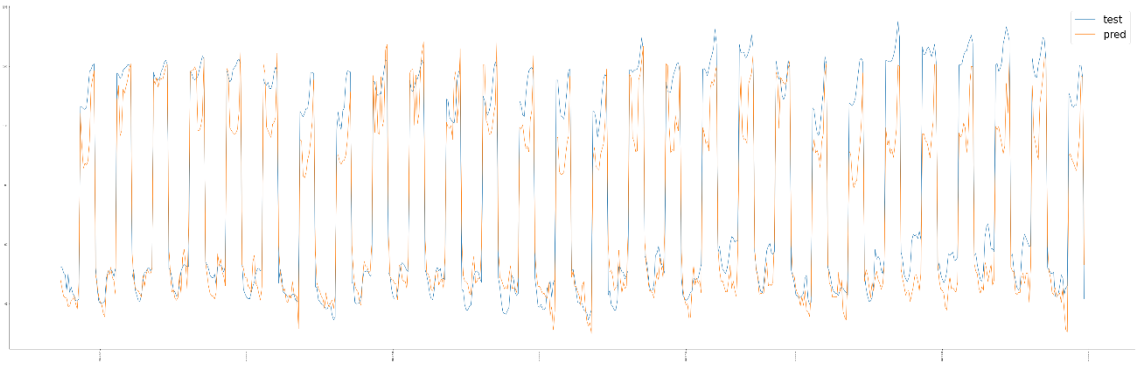

Ilustración 15 Comparación de resultados (2)

4.2. Serie temporal

Los modelos de serie temporal son modelos relativamente sencillos de mostrar. Simplemente hay que ajustar los datos al modelo con los parámetros deseados, y realizar la predicción para el número de días deseado.

En nuestro caso se ha realizado la predicción con dos modelos distintos detallados a continuación.

a) Facebook Prophet

Para nuestro modelo de Facebook Prophet se han seguido los pasos del apartado indicado más atrás. Una vez que se ha ajustado el modelo a los datos, se ha realizado un Cross Validation de los datos para un tramo de 730 días, en periodos de 180 días con un horizonte igual al número de días elegido, en este caso 28. De esta validación, la cual nos da métricas para la comprobación entre los 2 días y 20 horas hasta los 28 días indicados, obtenemos los siguientes resultados:

	MSE	RMSE	MAE	MAPE	MDAPE[12]	Coverage
Valor máximo	276.28	16.62	13.17	17.36 %	13.07 %	85.93 %
Valor mínimo	111.70	10.57	8.11	9.91 %	6.50 %	67.33 %

Tabla 5 Resultados tras el cross validation de nuestro modelo

Una vez se ha validado el modelo, se procede a obtener los resultados finales para nuestro set de test, que corresponde con los últimos 28 días. Los resultados obtenidos en las métricas son:

RMSE	R2	MAPE
17.550550631625185	0.7384108697787883	23.431270959223202

Tabla 6 Resultados obtenidos para el mejor modelo con los datos ordenados

Si queremos conocer los componentes de nuestro modelo:

[12] MDAPE: Median Absolute Percentage Error

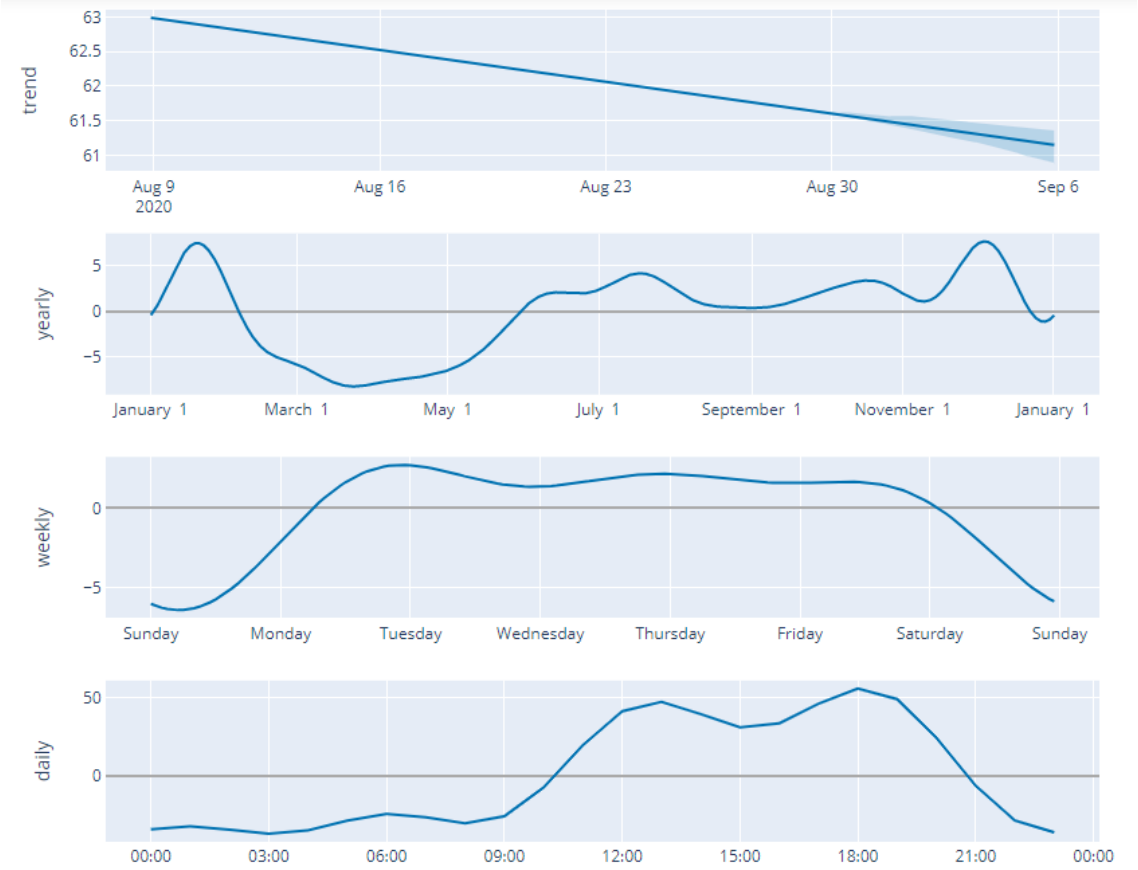

Ilustración 16 Componentes del modelo de Facebook Prophet

Por último, mostramos la comparación entre los resultados de la predicción y el set de test:

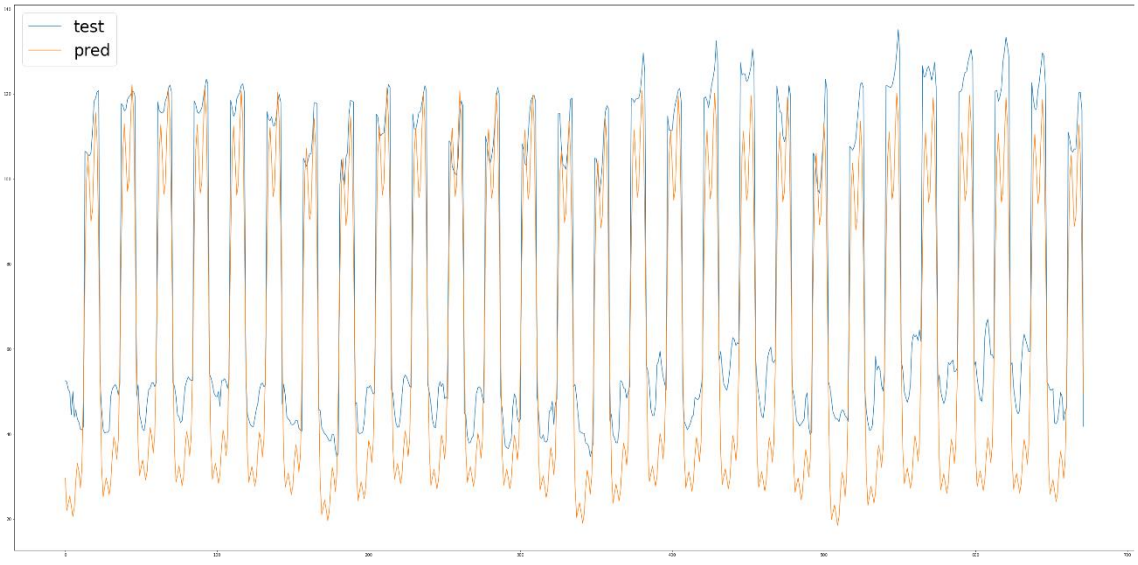

Ilustración 17 Comparación de los resultados de nuestro modelo de FP

b) SARIMAX

Nuestro modelo SARIMAX consta de los siguientes parámetros principales:

- **order = (2, 1, 2)**

- **AR[13] = 2:** Nos indica que nuestro modelo va a considerar 2 lags[14], por lo que considerará dos posiciones anteriores al valor actual.
- **Differences = 1:** Este parámetro es empleado para mejorar la estacionariedad de los datos. Estos deben cumplir unos requisitos que en su gran mayoría no se cumplen, por lo que, aplicando este parámetro el cuál realiza la diferencia entre el valor actual y el valor situado en el lag indicado, en este caso 1, mejoramos la calidad de los datos y nos permite aplicar este modelo.
- **MA[15] = 2:** Parecido al AR, pero en este caso tiene en consideración todos lags hasta el orden indicado, 2 en este caso, por lo que tendrá en cuenta los 2 valores anteriores.

- **seasonal_order = (2, 0, 2, 12)**
  - **Periodicidad = 12:** Nos indica que se trata de una periodicidad mensual.
  - En este caso lo que conseguimos es captar los patrones estacionales para los mismos parámetros explicados anteriormente.

Una vez ajustados los datos al modelo, comprobamos un resumen del mismo:

Out[22]:

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	55726
Model:	SARIMAX(2, 1, 2)x(2, 0, 2, 12)	**Log Likelihood**	-133746.304
Date:	Sat, 05 Sep 2020	**AIC**	267512.607
Time:	14:21:41	**BIC**	267601.889
Sample:	04-01-2014	**HQIC**	267540.434
	- 08-08-2020		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
x1	0.0772	0.017	4.607	0.000	0.044	0.110
ar.L1	0.8985	0.018	51.279	0.000	0.864	0.933
ar.L2	0.0297	0.017	1.781	0.075	-0.003	0.062
ma.L1	-0.7896	0.017	-45.912	0.000	-0.823	-0.756
ma.L2	-0.2090	0.017	-12.160	0.000	-0.243	-0.175
ar.S.L12	-0.0007	4.01e-05	-17.979	0.000	-0.001	-0.001
ar.S.L24	0.9992	4.12e-05	2.42e+04	0.000	0.999	0.999
ma.S.L12	0.0225	0.002	11.206	0.000	0.019	0.026
ma.S.L24	-0.8529	0.002	-533.225	0.000	-0.856	-0.850
sigma2	7.0932	0.021	344.984	0.000	7.053	7.134

Ljung-Box (Q):	1941.02	**Jarque-Bera (JB):**	106654.94
Prob(Q):	0.00	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.48	**Skew:**	0.07
Prob(H) (two-sided):	0.00	**Kurtosis:**	9.78

Ilustración 18 Resumen de las principales características del modelo

En la imagen siguiente se puede comprobar como se ajustan los datos al entrenamiento del modelo:

[13] AR: AutoRegressive
[14] Lag: valor anterior de la serie temporal
[15] MA: Moving Average

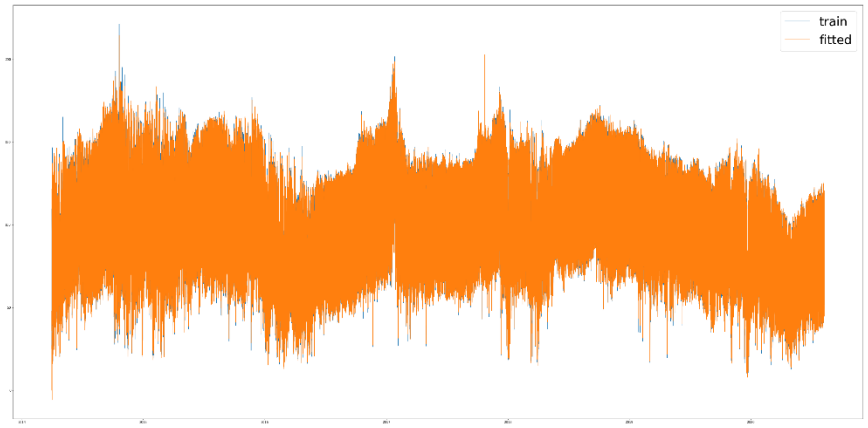

Ilustración 19 Datos ajustados del modelo SARIMAX

Las métricas obtenidas para este modelo son:

RMSE	R2	MAPE
6.15316473655467	0.9678459794054348	7.686197056522472

Tabla 7 Resultados obtenidos para el mejor modelo con los datos ordenados

Por último, graficamos los resultados predichos y los datos de test:

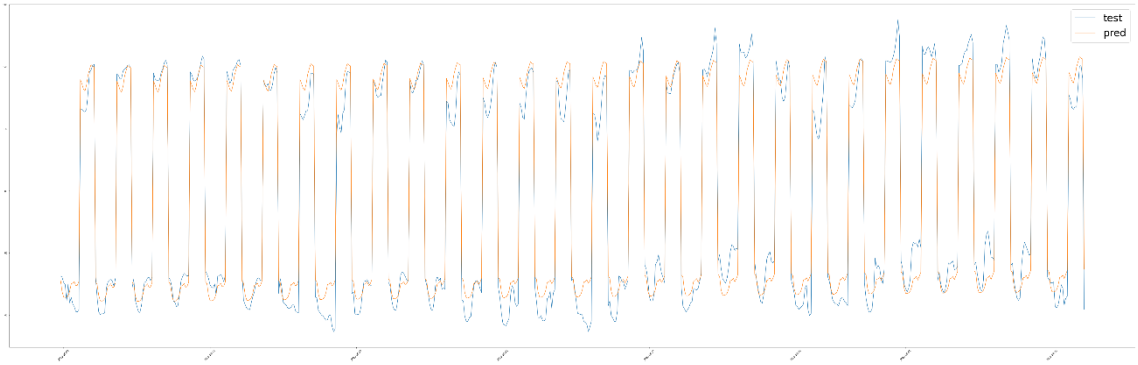

Ilustración 20 Comparación de los resultados de nuestro modelo SARIMAX

4.3. Red neuronal

El último modelo empleado es un modelo de redes neuronales. Se van a probar dos tipos de modelos, una red neuronal LSTM y una red neuronal DNN.

a) Red neuronal LSTM con varias capas

Este modelo, como se ha indicado antes tendrá como input la cantidad de días a predecir introducido en un array. Las características de esta red neuronal son:

Parámetro	Valores
Capas LSTM	3
Unidades por capa	100, 50, 20
Dropout	0.1
Activación	relu
loss	mse
optimazer	Adam
epochs	20
batch_size	100

Validation_split	X_test, y_test

Tabla 8 Características de la red neuronal

Con estas características, los resultados obtenidos son:

RMSE	R2	MAPE
10.09269087976934	0.9161526647016454	11.508015260921695

Tabla 9 Resultados obtenidos para el mejor modelo con los datos ordenados

La gráfica para la predicción y los valores reales es:

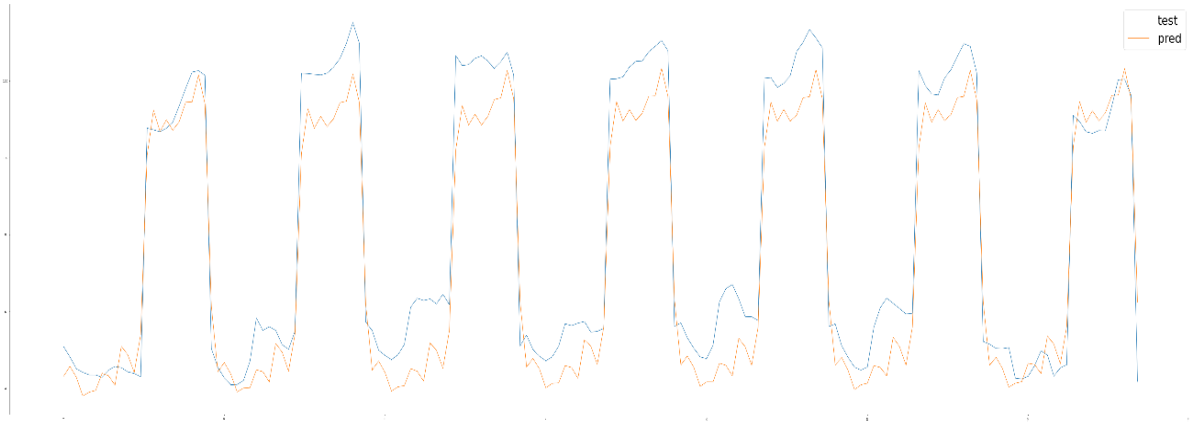

Ilustración 21 Comparación entre predicción y test para LSTM.

b) Red neuronal DNN con los datos aleatorios

En este caso el input es distinto a la red LSTM. Las columnas empleadas para el input se encuentran en el notebook de jupyter. Para nuestro modelo, las características son las siguientes:

Parámetro	Valores
Capas Dense	10
Unidades por capa	4096, 2048, 1024, 512, 256, 128, 64, 32, 16, 1
Dropout	None
Activación	relu
loss	mse
optimazer	Adam
epochs	30
batch_size	100
Validation_data	X_test, y_test
shuffle	True

Tabla 10 Características de la red neuronal

La explicación de este modelo es que se trata de una red de 10 capas, yendo desde 4096 neuronas la primera capa hasta 1 la última, sin eliminar neuronas, con activación *relu*, con función de pérdida el *mean squared error*, con el optimizador *Adam*, con 30 vueltas (*epochs)*, haciendo ciclos de 100 muestras hasta completar la vuelta (*batch_size*) y con una fracción para validación del 0,2.

Con estas características, los mejores resultados obtenidos son:

RMSE	R2	MAPE
6.124169	0.9681483084358267	7.427702844142914

Tabla 11 Resultados obtenidos para el mejor modelo con los datos ordenados

La gráfica para la predicción y los valores reales es:

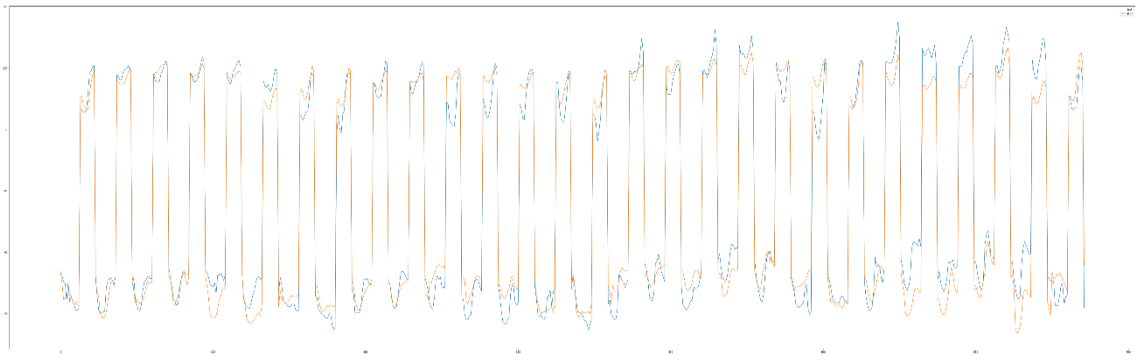

Ilustración 22 Comparación entre predicción y test

c) Red neuronal DNN con todos los datos ordenados

El input para esta red neuronal será el mismo que para el caso b). Para nuestro modelo, las características son las siguientes:

Parámetro	Valores
Capas Dense	9
Unidades por capa	2048, 1024, 512, 256, 128, 100, 50, 25, 1
Dropout	0.1
Activación	relu
loss	mse
optimazer	Adam
epochs	50
batch_size	30000
Validation_split	0.2
shuffle	False

Tabla 12 Características de la red neuronal

La explicación de este modelo es que se trata de una red de 9 capas, yendo desde 2048 neuronas la primera capa hasta 1 la última, eliminando el 0,1 de los resultados (Dropout), con activación *relu*, con función de pérdida el *mean squared error*, con el optimizador *Adam*, con 50 vueltas (*epochs)*, haciendo ciclos de 30000 muestras hasta completar la vuelta (*batch_size*) y con una fracción para validación del 0,2.

Con estas características, los resultados obtenidos son:

RMSE	R2	MAPE
18.864706	0.6977695451887873	25.859156250953674

Tabla 13 Resultados obtenidos para el mejor modelo con los datos ordenados

La gráfica para la predicción y los valores reales es:

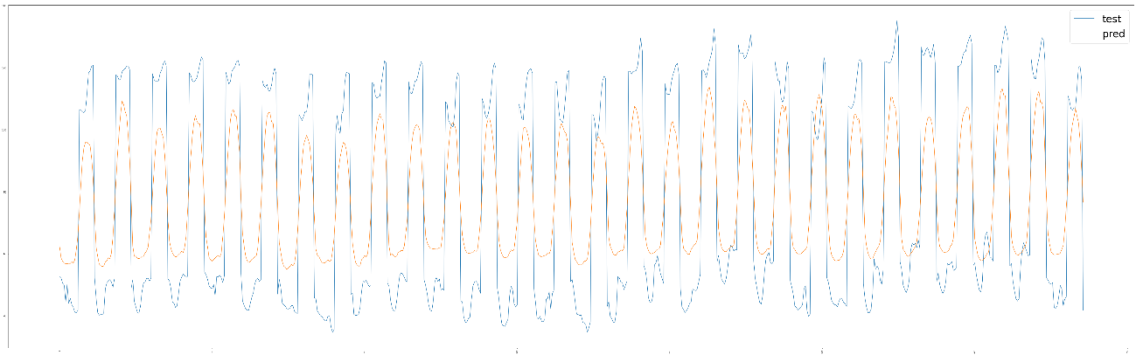

Ilustración 23 Comparación entre predicción y test.

4.4. Conclusiones finales

En resumen, las métricas para cada uno de los modelos que se han ido explicando son:

Modelo	**RMSE**	**R2**	**MAPE**
XGBoost datos aleatorios	7.16	95.64 %	8.50 %
XGBoost datos ordenados	12.16	87.44 %	11.33 %
Facebook Prophet	17.55	73.84 %	23.43 %
SARIMAX	6.15	96.78 %	7.69 %
LSTM multicapa	10.09	91.62 %	11.51 %
DNN datos aleatorios	6.12	96.81 %	7.43 %
DNN datos ordenados	18.86	69.78 %	25.86 %

Tabla 14 Resumen de las métricas para cada uno de los modelos

Con todo esto podemos llegar a unas cuantas conclusiones.

Primero, tratando los modelos de Gradient Boosting, se observa que son modelos que funcionan bastante bien. A pesar de que pueda parecer que funciona bien por el hecho de tener bastantes variables, vemos como un punto muy importante en este modelo es incluir la fecha de la medida, especialmente la hora. Por tanto, se puede decir que estos modelos también permiten detectar en cierta medida las tendencias estacionales. Cabe destacar como, cuando se emplea la totalidad de los datos ordenados, el modelo empeora. Esto puede ser debido a un overfitting que empeore las predicciones. Merece la pena destacar también como el agregar información que a priori parece irrelevante como puede ser los datos del IBEX o de la Red Eléctrica Española en bolsa, nos ayudan a mejorar parte del modelo, siendo esta información adicional.

Si observamos los modelos de series temporales, podemos ver una gran diferencia entre ellos. Esto puede ser debido ya que para el modelo de Facebook Prophet no hemos modificado ningún hiperparámetro, simplemente hemos dejado que el modelo se ajuste automáticamente. Por ello, se observa que este modelo es mucho peor que nuestro SARIMAX, el cual se ajusta muy bien a los datos, obteniéndose unas métricas más que aceptables. Por tanto, nos hace indicar que la estacionalidad es un factor muy importante a la hora de realizar predicciones.

Por último, tenemos los modelos de redes neuronales. Estos modelos son algo más complejos, ya que están formado por distintas capas de “neuronas” que crean relaciones entre los datos. Estos modelos tienen una componente aleatoria muy grande, por lo que su uso para tareas de predicción más sensibles como puede ser predecir un solo número puede resultar en fluctuante. Esto se puede ver en que al entrenar un mismo modelo dos veces, en una primera

prueba puede conseguir unos resultados aceptables y en la siguiente, con los mismos hiperparámetros, puede variar bastante. Por tanto, aquí se han mostrado los mejores resultados para dicha red, con esos parámetros.

Analizando los resultados de las redes neuronales, se puede ver que existen limitaciones para este ejercicio. Principalmente, para el modelo LSTM, buscamos una predicción de una gran cantidad de horas, concretamente 7 x 24 = 168 valores, por lo que, si queremos emplear aún más valores como input, el modelo requiere de demasiados recursos. Otra de las limitaciones es el tiempo de entrenamiento. Los resultados obtenidos para la red LSTM no son correspondientes con el tiempo de entrenamiento. Estas tardan alrededor de 45 minutos en entrenar y los resultados son similares que los obtenidos con los otros modelos, por lo que no merece la pena emplearlos. Si analizamos las DNN, son modelos más simples, con un menor tiempo de entrenamiento, y con resultados similares tanto al XGBoost como al modelo SARIMAX. Se han llegado a obtener resultados mejores que con estos modelos anteriores, pero al ser un modelo tan volátil no se puede considerar como un modelo mejor.

Resumiendo, los modelos más complejos como son las redes neuronales no suponen una gran ventaja para este ejercicio. De hecho, el mejor modelo obtenido de estas redes es bastante similar al mejor modelo obtenido tanto con el modelo de serie temporal como el modelo de Gradient Boosting. Por tanto, quedaría descartado el uso de esta tipología. Si atendemos a los otros dos tipos de modelos, podríamos suponer que tanto el modelo SARIMAX como el modelo XGBoost con una separación de los datos de forma aleatoria sería lo óptimo. Ambos modelos tienen unas métricas parecidas, y predicen bastante bien para periodos largos en el tiempo.

Por simplicidad, si se tuviese que decidir por uno de ellos, sería el modelo SARIMAX, ya que simplemente sacando los datos del precio para la tarifa con discriminación horaria se puede montar el modelo y obtener los resultados, cosa que para mejorar el modelo XGBoost hasta el punto de alcanzar el modelo SARIMAX no es posible. Este último necesita de datos complementarios para obtener métricas similares, pudiendo ser mejores o peores dependiendo del modelo entrenado.

5. Comentarios generales del proyecto

El proyecto en sí ha supuesto un gran reto. No solo conseguir la información ha sido complicado y enrevesado por la variedad de fuentes y de procedimientos, sino que se ha decidido adentrarse todo lo posible en el estudio de cada uno de los modelos. Por supuesto, se puede hacer un estudio mucho más detallado de los mismos, pero por falta de tiempo se ha decidido llegar hasta este punto.

Para este proyecto ha habido que dedicar bastantes horas, ya que la parte de modelado ha sido un ensayo y error en los que cada prueba llevaba bastante tiempo, especialmente en el apartado de redes neuronales. Además, para el estudio de los modelos de GB[16] ha sido necesario recabar bastante información para poder llegar al nivel de las series temporales.

Aun así, se han obtenido unos resultados muy buenos. Observando los resultados de los proyectos mencionados anteriormente, se han obtenido resultados muy ajustados al mínimo, mejorando dichos estudios. Además, se ha hecho una comparativa bastante útil entre los modelos GB y los de series temporales como SARIMAX, viendo que en ambos casos se puede llegar a estudiar las tendencias estacionales y las periodicidades hasta llegar a niveles parecidos.

Finalmente, el resultado es la capacidad de recrear unos modelos muy útiles para tareas de predicción de tarifas horarias de la luz. Cualquier empresa cuyos productos sean derivados del precio de la electricidad sería capaz de adelantarse a este, incluso en el periodo de 1 mes. Esto permitiría ajustar sus productos y tener una planificación bastante exacta para adelantarse a sus competidores.

[16] GB: Gradient Boosting

6. Visualización

A continuación, se va a realizar una explicación para poder visualizar los resultados con la aplicación construida.

Se trata de un formato sencillo. Cuando se corren los notebooks de los modelos, se generan unos archivos en formato csv que contiene los resultados de los mismos. Se ha montado una estructura en un cuaderno de jupyter mediante la librería **streamlit** que permite una visualización interactiva de los mismos. Simplemente habrá que correr el Notebook situado en la carpeta *Visualization*, de tal forma que, al correr las 2 celdas del notebook, solamente hace falta abrir la conexión en una pestaña a parte con la ruta:

- **Network URL**: http://192.168.1.43:8501

La ruta anterior puede variar, se puede comprobar con los resultados del notebook si es el mismo enlace, sino simplemente hay que usar el que aparezca en el notebook.

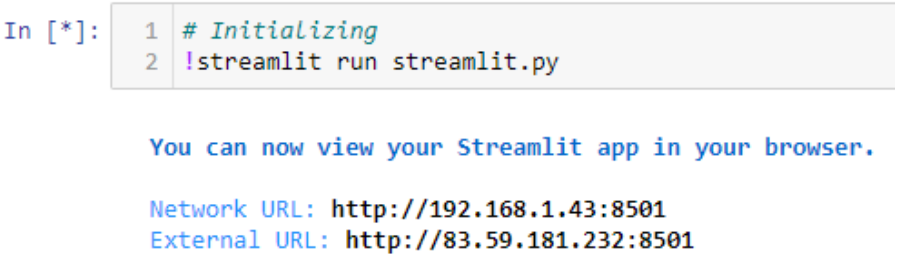

Ilustración 24 Link a la visualización

Una vez hemos abierto esta ruta, lo primero que nos aparece en pantalla será la descripción del proyecto, incluida en la ruta de github donde está alojado el proyecto. Una vez dentro, podemos desplazarnos por el menú de la izquierda para elegir el modelo deseado:

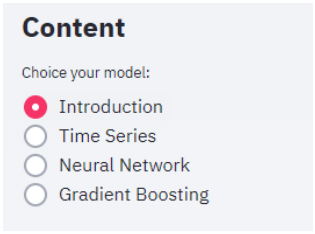

Ilustración 25 Menú de opciones en Streamlit

Una vez elegido el modelo que se quiere comprobar, solo queda elegir el modelo que se quiere comprobar, sobre el cual aparecerán las métricas descritas anteriormente.

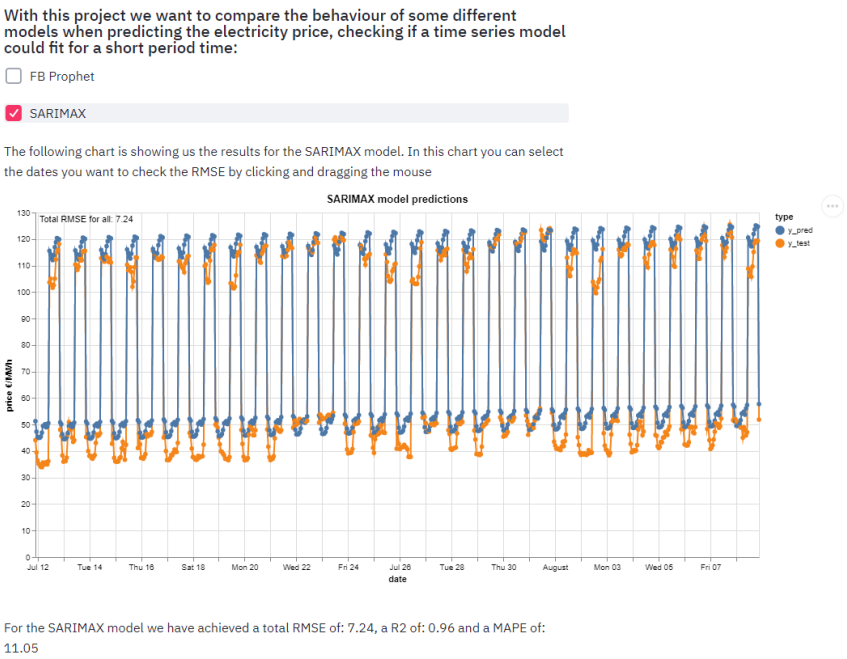

Ilustración 26 Ejemplo de visualización de resultados

Por último, explicar que para cada modelo se han desplegado dos gráficas. Aunque son los mismos datos, la gráfica de abajo permite seleccionar el rango de fechas que se desee y en la gráfica de arriba aparecerá solamente ese rango señalado. Además, en el gráfico superior se puede hacer zoom sobre la serie y aparece el valor del RMSE para ese rango de datos.

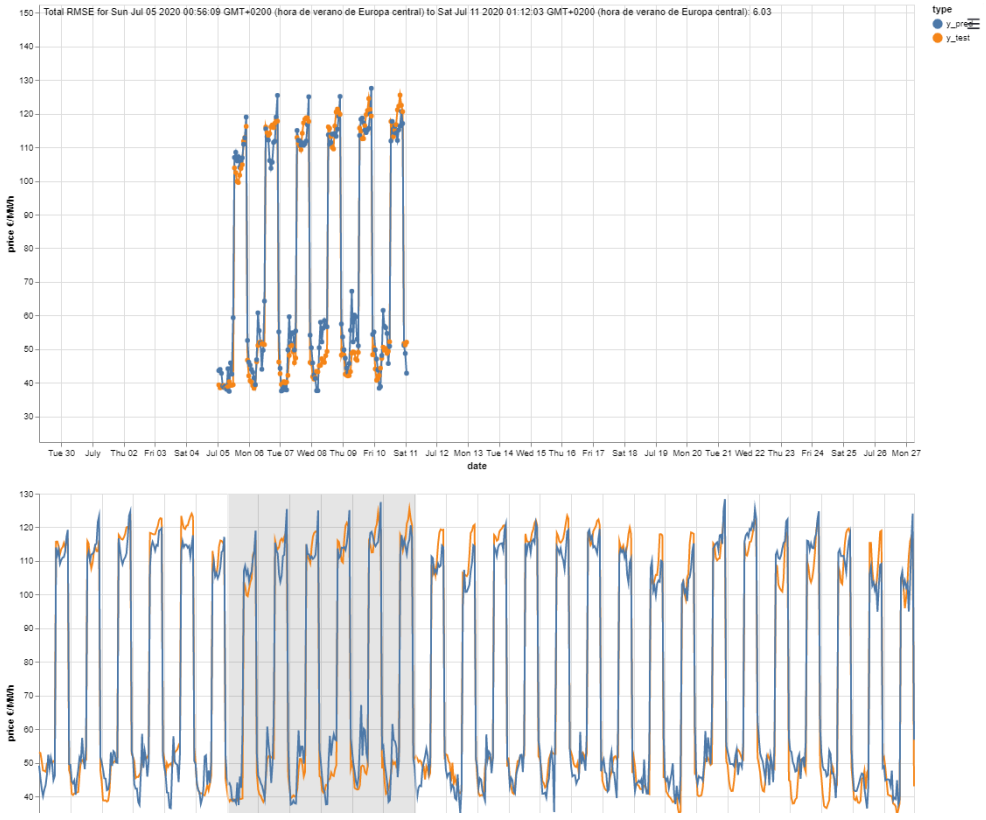

Ilustración 27 Ampliación de la visualización de los datos

Referencias

Herrá, J. Q. (2019). *DESARROLLO DE UN MODELO DE PREDICCIÓN DEL PRECIO DE LA ENERGÍA ELÉCTRICA PARA EL MERCADO A PLAZO MEDIANTE REDES NEURONALES.* Valencia.

JAIN, A. (1 de 3 de 2016). *Analytics Vidhya*. Obtenido de Analytics Vidhya: https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

Mate, P. G. (2015). *MODELO DE PREVISIÓN DEL PRECIO DEL MERCADO DIARIO DE LA ELECTRICIDAD EN EL CORTO PLAZO .* Madrid.

Ping-Huan Kuo, C.-J. H. (2018). *An Electricity Price Forecasting Model by Hybrid Structured Deep Neural Networks.* Ganzhou.

Shu Fan, J. R. (2006). *An Integrated Machine Learning Model for Day-Ahead Electricity Price Forecasting.*

Ugarte, A. R. (2017). *Predicción de precios de energía eléctrica utilizando árboles dinámicos.* Madrid.

Yongli Zhu, R. D. (20). *Power Market Price Forecasting via Deep Learning .* Chicago.