




17 DE JUNIO DE 2022

PREDICCIÓN DE REGISTRO DE ENVÍOS EN UNA EMPRESA DE PAQUETERÍA

TRABAJO FIN DE MÁSTER

DAVID J. LOPEZ BARROSO
KSCHOOL



Contenido

Introducción.....	2
Objetivo.....	3
Entorno	4
Modelo de datos.....	5
Preprocesado y análisis.....	6
Lectura y preprocesado de datos para análisis.....	6
Análisis y preprocesado de los datos	6
Modelos	10
Introducción.....	10
SARIMAX	11
Prophet	12
KNN	13
Random Forest Regressor.....	14
XGBoost	15
LSTM	16
Evaluación de los modelos.....	18
Interfaz web	20
Descripción de la herramienta.....	20
Conclusiones y líneas futuras.....	23
Conclusiones	23
Líneas futuras y mejoras	24

Introducción

Actualmente nos encontramos en la era de la digitalización y la globalización, una era donde se genera una gran cantidad de información a diario (se estima que para 2025 se generarán al día 450 exabytes en todo el mundo). Esto es porque tanto el software como el hardware ha avanzado hasta el punto de poder almacenar y gestionar esta información de cara a conseguir ventajas competitivas para las empresas.

Pero no sólo consiste en almacenar datos, consiste en manipularlos, aprender de ellos, obtener resultados y actuar en consecuencia, y de todo esto se encarga la ciencia de datos.

Los datos no solo permiten conocer la actividad de la empresa, si no que permiten conocer el mercado, alcanzar clientes potenciales de forma estratégica, adaptarse a cambios en el negocio, ahorrar costes, conseguir una mejor calidad de servicio, mejorar las condiciones del personal, mayores retornos de la inversión, mayores ventas, es decir, potenciar el negocio en todas sus líneas. Y puede darse en negocios de diversa índole como el marketing, la publicidad, las finanzas, la industria, la logística o la salud, entre otros.

Entre las empresas dedicadas a la logística, encontramos aquellas que ofrecen servicios de paquetería. Para este tipo de empresas es vital el buen aprovechamiento de sus propios datos, concretamente predecir la actividad futura a corto, medio y largo plazo.

En cuanto al medio y largo plazo, conocer cómo va a aumentar o disminuir el volumen de negocio es muy importante, ya que su conocimiento permite tomar decisiones al respecto como por ejemplo abrir más oficinas, aumentar la plantilla o renegociar contratos con los transportistas.

En este trabajo se centra el foco en el corto plazo, esto es, predecir cuántos paquetes se van a registrar el siguiente día en el sistema (los cuáles serán desplazados el próximo día laborable) de una empresa española con más de 20 años de experiencia en el sector.

De esta manera, se pretenden poner a prueba diversos conocimientos adquiridos en el presente máster correspondientes a diversas áreas como son: la obtención de datos, limpieza y tratamiento de los datos, análisis y visualización, modelos de previsión, evaluación de los mismos y la elaboración de interfaces.

Históricamente, la predicción de series temporales ha sido un problema muy presente, ya que hay gran cantidad de ámbitos que dependen del tiempo de forma ordenada. Estos pueden ser desde las predicciones de ventas de un producto hasta la predicción de flujos de vehículos o la asistencia de pacientes a un hospital a lo largo de la semana, es decir, conocer de forma inteligente qué va a ocurrir en el futuro para poder realizar una buena gestión de los recursos para estar preparados.

Es por eso que ya en el siglo pasado se utilizaban métodos estadísticos elementales y econométricos como ARIMA o GLM. Sin embargo, existen diferentes tipos de modelos que se aplican a series temporales: basados en árboles, en el descenso del gradiente, redes neuronales e incluso clustering. Modelos que, conforme avanzan los años y cada vez existe más potencia de computación informática, puede pasar a un segundo plano en favor de otros y viceversa.

De este modo, en el presente trabajo se quieren probar diferentes algoritmos pertenecientes a las familias recién mencionadas que sirva como prueba para establecer diferencias de rendimiento entre ellos en el problema en cuestión y que sirva al alumno como aproximación a los modelos de regresión aplicados a series temporales.

Objetivo

El objetivo principal del trabajo es el de realizar una herramienta basada en analítica avanzada para la previsión del registro de paquetes el siguiente día en el sistema de una empresa de paquetería española.

La obtención de dicho dato es importante para la empresa de cara a organizar los recursos necesarios tanto de personal como de transporte para el movimiento de los paquetes el día próximo debido a que la planificación se realiza con un día de antelación.

De esta manera, se pretenden poner a prueba diversos conocimientos adquiridos en el máster realizado correspondientes a diversas áreas como son: la obtención de datos, limpieza y tratamiento de los datos, análisis y visualización, modelos de previsión, evaluación de los mismos y la elaboración de interfaces.

Así se pretende hacer una aproximación a la predicción de series temporales realizando un proyecto completo de Data Science y que ponga a prueba varios modelos de diferente índole para este tipo de problemas suponiendo así una aproximación a los modelos de previsión.

Concretamente, los objetivos técnicos del trabajo son los siguientes:

- Obtención de la información.
- Limpieza y tratamiento de los datos.
- Análisis y visualización de los datos.
- Implementación de modelos de previsión.
- Evaluación de los modelos de previsión.
- Interfaz web para la visualización y ejecución del modelo de previsión.

Cabe destacar que, en un principio, el trabajo pretendía realizar la previsión del registro de paquetes en el sistema teniendo en cuenta los orígenes y los destinos de modo que se realizara la previsión entre cada par de oficinas para, a partir de ella, la empresa pueda gestionar los medios de transporte necesarios. Sin embargo, debido a que hasta la fecha no ha sido posible conseguir el maestro de oficinas correctamente (no se mantenían en el histórico de datos bajo la misma nomenclatura y era imposible asociar cada dato a las oficinas en cuestión), se han tomado los datos de manera global.

Actualmente, se está trabajando para la correcta identificación de las oficinas, y el trabajo desarrollado se podría aplicar individualmente para cada par de orígenes y destinos con la misma lógica.

Entorno

Repositorio

El repositorio (https://github.com/dlopbar/TFM_ParcelForecasting) cuenta con la siguiente organización en su interior:

- docs: contiene la presente memoria del proyecto.
- files: deberá contener en su interior todos los archivos guardados necesarios para la ejecución del proyecto (depositados en drive y necesario alojarlos aquí, en caso de necesitar los datos, contactar con dlopbar416@gmail.com).
 - data: se encuentran todos los CSV tanto de datos iniciales como resultado de algunos notebooks que son usados en los siguientes.
 - save_model: se encuentran los archivos de los modelos de previsión guardados para su uso en la interfaz.
- notebooks: se encuentran los cuatro notebooks con el código de la parte técnica del proyecto y una serie de imágenes incluidas en ellos.
- tools: contiene scripts de Python como la app, la query SQL con la que se obtuvieron los datos iniciales y los archivos del entorno "requirements.txt" y "environment.yml".
- readme: contiene una introducción y directrices para la ejecución del proyecto.

Repositorio de desarrollo

La parte técnica del proyecto empezó siendo desarrollada Google Colab. Sin embargo, una vez avanzado se migró a un entorno virtual de Conda usando Visual Studio Code y las extensiones necesarias para Python y Jupyter Notebook (cabe destacar que el módulo fbprophet no fue posible migrar al entorno virtual y se han usado las capturas de Google Colab).

Para replicar el entorno del proyecto y los módulos necesarios existe tanto el archivo "requirements.txt" como el "environment.yml".

Modelo de datos

Los datos disponibles corresponden al número de paquetes registrados en el sistema de la empresa cada día desde 2010 hasta el 31 de marzo de 2022.

Estos datos se obtienen de un *webservice* privado de la empresa conectado con su base de datos. De forma que se obtuvieron a través de dicho servicio y se depositan en una base de datos SQL en una máquina virtual habilitada para ello.

Una vez depositados en la base de datos, mediante una query ejecutada en *SQL Server Management Studio* se obtuvieron y se depositaron en un *CSV* que sirva de punto de partida para el presente trabajo.

"initial_data.csv"

Descripción: histórico de paquetes registrados por día en el sistema entre 2010 y 2022.

Campos:

- Fecha de registro: en formato %dd/%mm/%yyyy.
- Número de paquetes registrados.
- Número del día de la semana.
- Número del día del mes.
- Número de mes.
- Año.

Cabe destacar que el campo de fecha contiene todos los días desde el 1 de enero de 2010 hasta el 31 de marzo de 2022 ya que se creó con la query desarrollada para tener todos los días de la serie temporal, de modo que aquellos días que no tenían registro aparecen con valor *NULL*.

```
2010-01-01;4;6;1;1;2010
2010-01-02;164;7;2;1;2010
2010-01-03;8;1;3;1;2010
2010-01-04;25345;2;4;1;2010
2010-01-05;20130;3;5;1;2010
2010-01-06;104;4;6;1;2010
2010-01-07;22950;5;7;1;2010
2010-01-08;23411;6;8;1;2010
2010-01-09;37;7;9;1;2010
2010-01-10;8;1;10;1;2010
2010-01-11;29974;2;11;1;2010
2010-01-12;30018;3;12;1;2010
2010-01-13;31576;4;13;1;2010
2010-01-14;35759;5;14;1;2010
2010-01-15;28768;6;15;1;2010
2010-01-16;10;7;16;1;2010
```

Muestra datos iniciales

Preprocesado y análisis

Lectura y preprocesado de datos para análisis

Inicialmente se procede a la importación de los datos y su preparación mediante la adición de variables de apoyo para el posterior análisis y visualización. También se generan los datos de calendario, que servirán tanto para el análisis como para los posteriores modelos generados.

Para ello se usa el archivo “*initial_data.csv*” completamente estableciendo como índice del *dataframe* la fecha.

Para los datos de calendario se generan los días festivos nacionales, el *Black Friday*, la Semana *BlackFriday*, la navidad y el período de confinamiento y meses posteriores de COVID.

Todas estas variables se añaden al *dataframe* para cada uno de los días en forma de variable binaria y, además, a partir de los propios datos, se añaden otras como las vísperas de festivo, el día después a festivo, sábados, domingos, fin de semana y laborables, también binarias.

Además, se comprueba si existen días sin valor (NAs), encontrando 48 pertenecientes a días en los que no hubo registros y no existían en el sistema de la empresa..

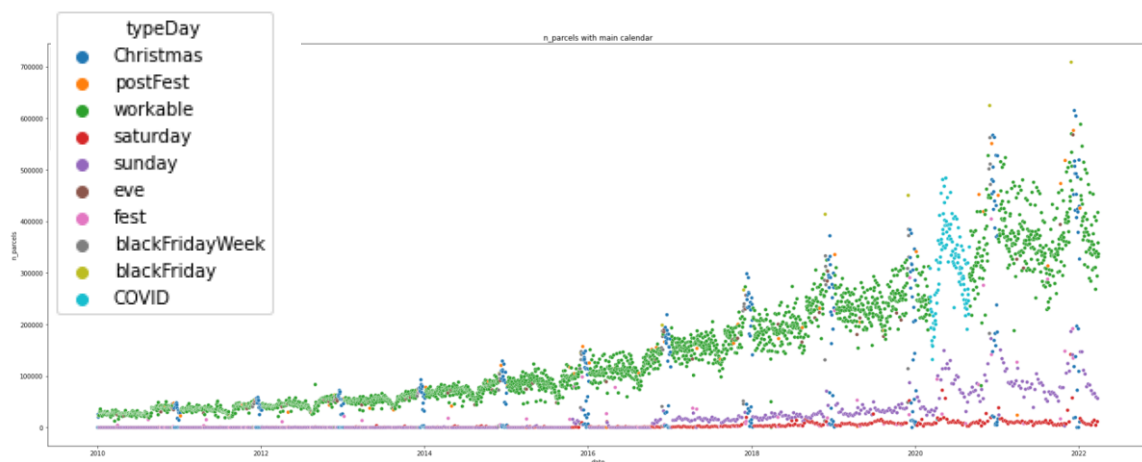
De esta manera, el resultado, como se citó anteriormente, es el de un *dataframe* con los datos suficientes para realizar el análisis de la serie temporal.

Análisis y preprocesado de los datos

En este apartado se procede al análisis de la serie temporal para conocer más acerca del comportamiento de los datos en el pasado de cara a transformar los datos y añadir las variables necesarias que se usarán más tarde para los modelos.

Análisis gráfico

Se procede al análisis de la serie temporal, la cual tiene el siguiente aspecto:



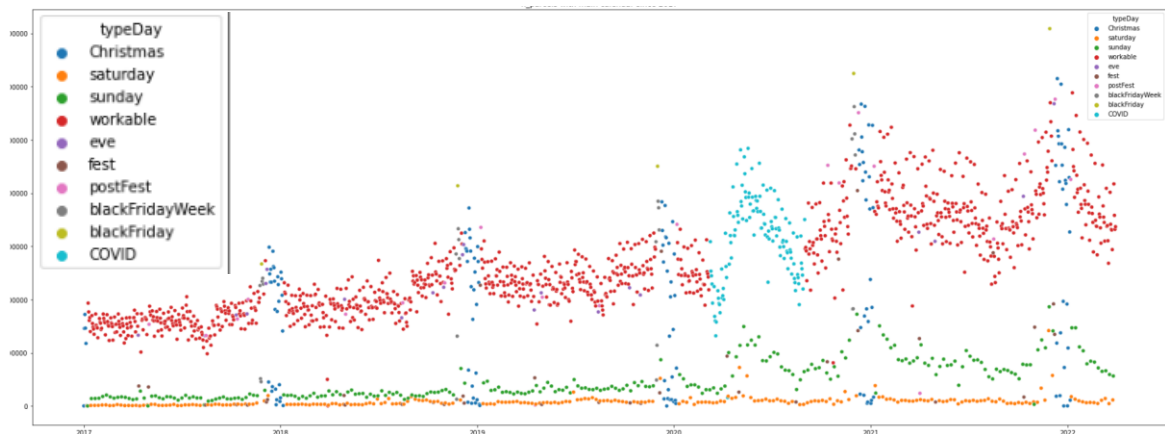
Paquetes registrados y fecha desde 2010 hasta 2022

A priori, se puede observar lo siguiente:

- Existe una tendencia geométrica al no mantenerse el total de paquetes registrados en una banda paralela a lo largo de los días de los diferentes años.

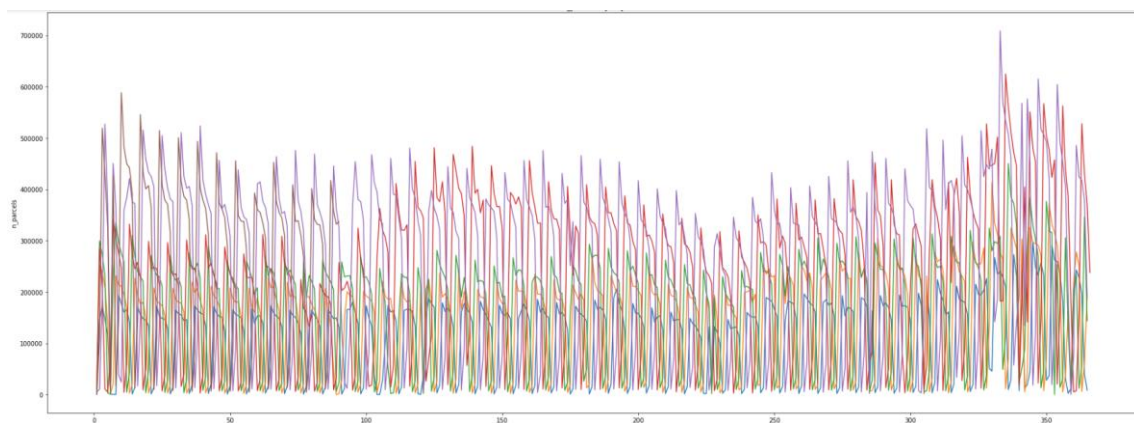
- Existe una estacionalidad anual marcada por una caída en el mes de agosto (valle) y una inestabilidad generalizada con picos superiores a final de noviembre y diciembre debido al *Black Friday* y la navidad.
- Existe una gran inestabilidad en el periodo marcado como COVID en forma de un pico inusual tras el confinamiento.
- Los sábados mantienen una actividad casi nula a lo largo de todo el período mientras que los domingos van aumentando su actividad conforme avanza el tiempo.
- Se observa que no es hasta 2016-2017 cuando los fines de semana (con mayor fuerza los domingos) empiezan a tener registros de paquetes y la inestabilidad a final de año es más evidente.

Debido a la última observación se decide continuar con los datos a partir de 2017.



Paquetes registrados y fecha desde 2017 hasta 2022

Además de la estacionalidad anual ya comentada, parece interesante analizar una posible estacionalidad semanal donde exista un patrón común a lo largo de los días de cada semana.



Paquetes por día del año para cada uno de los años

Observando la gráfica para cada uno de los años por día del año, observamos que hay un patrón que se repite para cada una de las semanas durante todos los años. Igualmente podemos encontrarlo visualizando la suma para cada día de la semana de forma global.

De esta manera se puede observar la forma semanal que tiene la distribución, de forma que el lunes es el día con mayor volumen, el cual decrece hasta el sábado, encontrando un volumen mínimo. Y, por último, el domingo tiene un volumen algo más alto, pero muy bajo en comparación con los días laborables.

Resumiendo, se extraen las siguientes conclusiones.

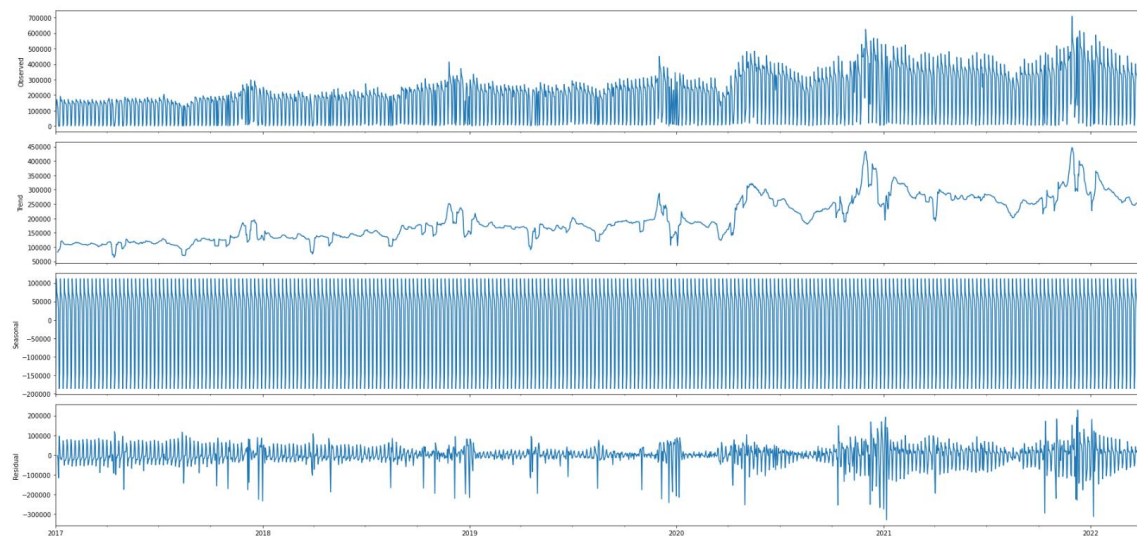
- Existe tendencia anual.
- Existe tendencial semanal (sobre todo con los fines de semana con muy poco volumen).
- Existe una gran inestabilidad a final de año debido al período que incluye en Black Friday y la navidad.
- Existe un período anómalo como es el correspondiente al confinamiento y el verano de 2020 con un aumento muy pronunciado del volumen.

Descomposición de la serie

Las series temporales tienen una serie de características:

- Tendencias: aumento o disminución a largo plazo.
- Ciclos: subidas y bajadas sin un periodo fijo.
- Estacionalidad: patrón determinista de duración fija y conocida de una estación.
- Volatilidad: ruido por factores impredecibles no asignables a las características anteriores.

A continuación, se procede a la descomposición de la serie temporal haciendo uso del módulo *statsmodels*:



Descomposición de la serie temporal

A la vista de la descomposición vemos cómo se detecta la estacionalidad semanal. Sin embargo, la tendencia, a pesar de ser ascendente, es un tanto extraña, de forma que los residuos tienen períodos de altos valores. Esto se puede explicar debido a que este tipo de descomposición sólo trabaja con una estacionalidad, de forma que no se tiene en cuenta la anual. Además, los altos valores residuales se observan a final de año debido al Black Friday y la navidad y, por otro lado, en los días festivos, todo ello ya identificado en las variables de calendario generadas con anterioridad.

Test ADF

El test ADF (prueba de Dickey Fuller aumentada) es una prueba de raíz unitaria para la estacionariedad, es decir, para estudiar la estabilidad de la serie a lo largo del tiempo (media, varianza y covarianza constantes en el tiempo). Si la serie no fuera estacionaria, sería muy difícil modelarla mediante modelos simples (por ejemplo, ARIMA) y se podría buscar alguna transformación de los datos (por ejemplo, un logaritmo) para hacerla estacionaria.

Se obtiene que la serie sí es estacionaria, por lo que es estable a lo largo del tiempo y no es necesario realizar ninguna transformación buscando su estabilidad.

```
Strong evidence against Null Hypothesis
Reject Null Hypothesis - Data is Stationary
Data is Stationary n_parcels
```

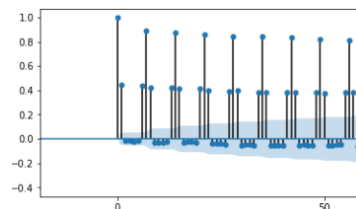
Resultado del test ADF sobre la serie temporal

ACF y PACF

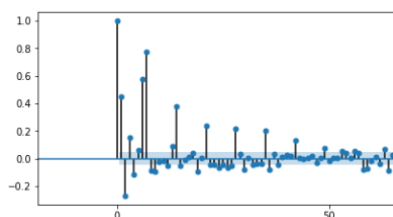
En series temporales, la autocorrelación es un término estadístico que describe la presencia o ausencia de correlación en los datos. Es decir, hace referencia a cuando los valores de una variable en el tiempo no son independientes y las observaciones anteriores influyen en las posteriores

Por un lado, la función ACF mide la correlación entre dos variables separadas por k períodos, mientras que la función PACF mide la correlación entre dos variables separadas por k períodos sin considerar la dependencia creada por los datos intermedios existentes entre ambas.

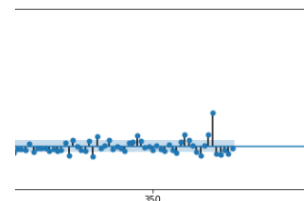
Se realiza este análisis sobre la serie temporal en cuestión usando de nuevo el módulo *statsmodels*.



Resultado ACF



Resultado PACF



Con todo lo anterior, se añaden los *lags* 1, 2, 3, 4, 6, 7, 14, 21, 28 y 365 al *dataframe* con los datos históricos del número de paquetes registrados diarios y con las variables de calendario.

Por último, se genera un nuevo archivo *csv* con los datos de calendario anteriores y las nuevas variables con los *lags* nombrados: "*daytypelags2017_parcels.csv*".

Modelos

Introducción

Una vez tratados y analizados los datos de entrada, se continúa con el modelaje, es decir, construir modelos que resuelvan el problema en cuestión: predicción del número de paquetes registrados en el sistema de una empresa de paquetería el siguiente día.

La predicción de series temporales es un problema común, se puede encontrar en diversos ámbitos como el de la predicción de consumo energético, de ventas de una determinada empresa, de congestión del tráfico, de asistencia a un establecimiento o de llamadas a un servicio de emergencia, por ejemplo.

Existe una gran cantidad de modelos capaces de resolver este problema de analítica predictiva, por lo que en este trabajo se van a probar 6 modelos de diferente índole con el objetivo de compararlos entre ellos y a modo de aprendizaje para el alumno por la diversidad de los mismos. Los resultados se compararán y se establecerá el mejor de ellos, el cual será el elegido para la interfaz final. Los modelos a testear son los siguientes:

- SARIMAX
- PROPHET
- KNN
- RANDOM FOREST REGRESSOR
- XGBOOST
- LSTM

Variables

Por un lado, se tiene la variable target, que no es otra que el número de paquetes registrados cada uno de los días, es decir, la variable que se quiere predecir.

Por otro lado, se tienen las variables exógenas, de las cuales inicialmente se seleccionan las siguientes:

- Variables de calendario: *Fest*, *Saturday*, *Sunday*, *blackFriday*, *blackFridayWeek*, *COVID* y *Christmas*.
- Variables de lags: número de paquetes registrados para los *lag1*, *lag2*, *lag3*, *lag4*, *lag6*, *lag7*, *lag14*, *lag21*, *lag28* y *lag365*.

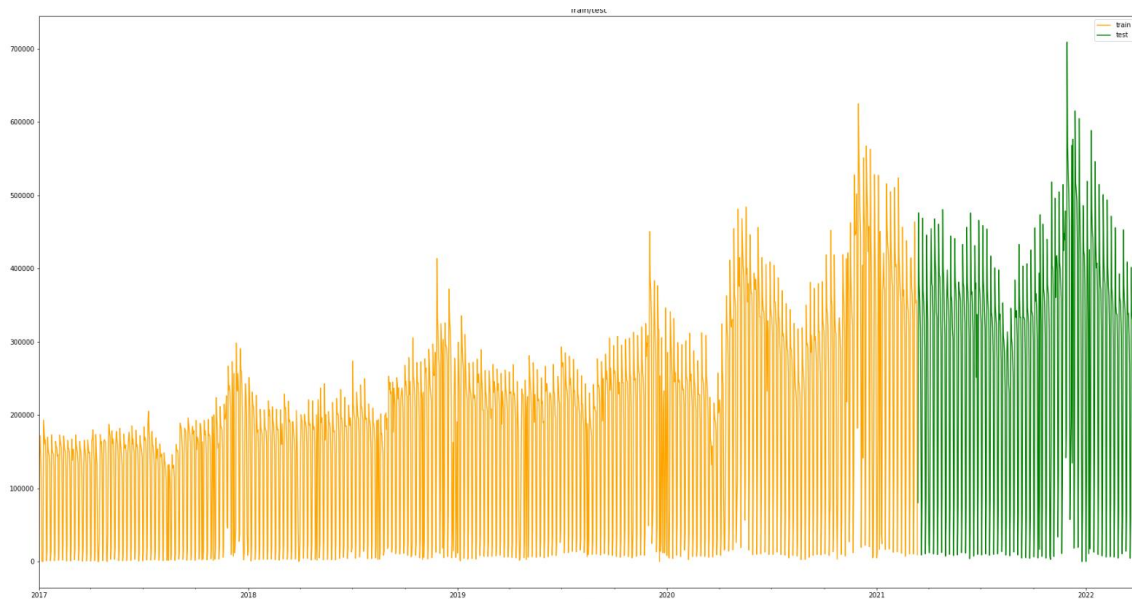
Train y test

Previamente a testear los modelos, es necesario dividir los datos en *train* y *test* de forma que primero se entrenarán con los datos de *train* y posteriormente se evaluarán con los datos de *test*, sobre los cuales se calculará el error asociado para cuantificar su rendimiento.

Anteriormente se decidió usar la serie temporal partiendo desde los años 2016-2017. Por ello, y como tenemos el *lag365*, los datos de *train* van a tener su punto de partida en 2017 de forma completa (con los *lag365* con datos de 2016).

Además, se establece, de forma ordenada, que el conjunto de datos de *train* conste del 80% de la muestra mientras que el conjunto de datos de *test* contenga el 20% de datos restante, quedando así:

- *Train*: desde el 2017-01-01 hasta el 2021-03-13
- *Test*: desde el 2021-03-14 hasta el 2022-03-31.



Datos de train y test

Además, por último, algunos modelos como SARIMAX, KNN o LSTM necesitan un escalado de los datos, por lo que, haciendo uso de la función *MinMaxScaler* de *sklearn* se realiza un escalado de las variables del modelo entre 0 y 1 para los mismos conjuntos de datos de *train* y *test* generados.

Errores

A medida que se van implementando los diferentes modelos de predicción se van a ir recopilando los errores de los mismos para su posterior comparación. Existen una serie de errores típicos que se usan para la evaluación de los modelos (MSE, RMSE, R^2 , MAE, MAPE, etc). Sin embargo, los errores elegidos para este problema son los siguientes:

- RMSE (error cuadrático medio): consiste en comparar los valores reales con los valores predichos a través de la distancia euclídea de los valores.
- MAE (error absoluto medio): se calcula como el promedio de diferencias absolutas entre los valores reales y los valores predichos de forma que todas las diferencias se ponderan de la misma manera.

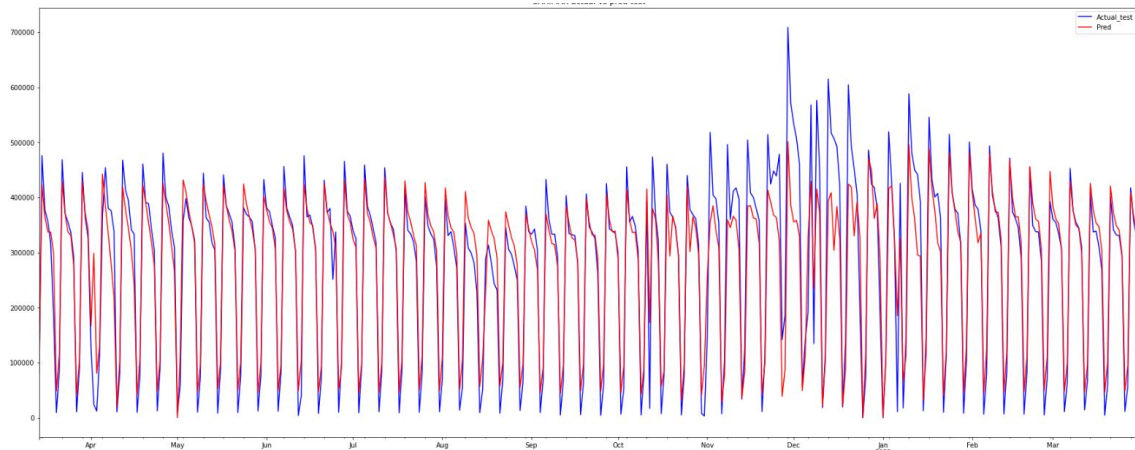
Se eligen esos dos errores debido a, por un lado, que el RMSE da importancia a los valores altos y bajos, lo cual tiene sentido en la serie temporal en cuestión debido a que existen continuamente valores muy altos (período de Black Friday y Navidad) y muy bajos, como los fines de semana, por lo que es interesante ajustarnos a esos valores. Y, por otro lado, el MAE proporciona una visión más general del error a lo largo de toda la serie.

SARIMAX

El modelo SARIMAX (Media Móvil Autorregresiva con Factores Exógenos) es un modelo de regresión lineal que usa un modelo de tipo ARIMA estacional de residuos que permite variables exógenas. Típicamente, los modelos basados en ARMA suelen ser un buen modelo para empezar,

pudiendo alcanzar resultados decentes y sirviendo como modelo de referencia para los problemas de series temporales. Además, se puede usar como modelo de referencia para el resto.

En este caso, primero vamos a usar la función *auto_arima* de la librería *pmdarima* para calcular los parámetros (p, d, q) y, posteriormente se implementa un modelo SARIMAX del módulo *statsmodels* para ser entrenado y testeado.



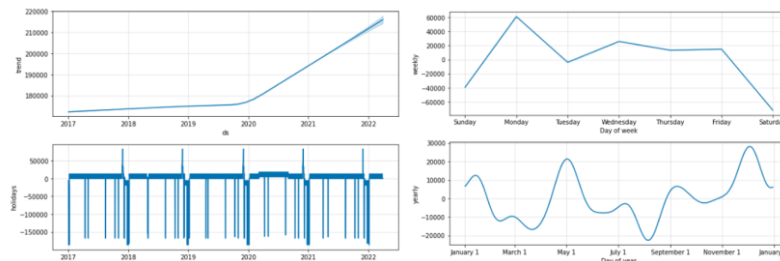
Gráfica valores reales y predicciones de test SARIMAX

Se observa cierta dificultad para acercarse completamente a los valores bajos de los fines de semana, además tiene problemas en los períodos de mayor volumen y volatilidad de final de año porque no incrementa nada sus previsiones.

Prophet

Prophet es un módulo desarrollado por Facebook con su propio modelo de series temporales, el cual permite obtener los componentes de tendencia, múltiples estacionalidades (lo cual es interesante porque en la descomposición que se realizó anteriormente sólo se podía identificar una estacionalidad), además de la adición de variables exógenas como las de los festivos.

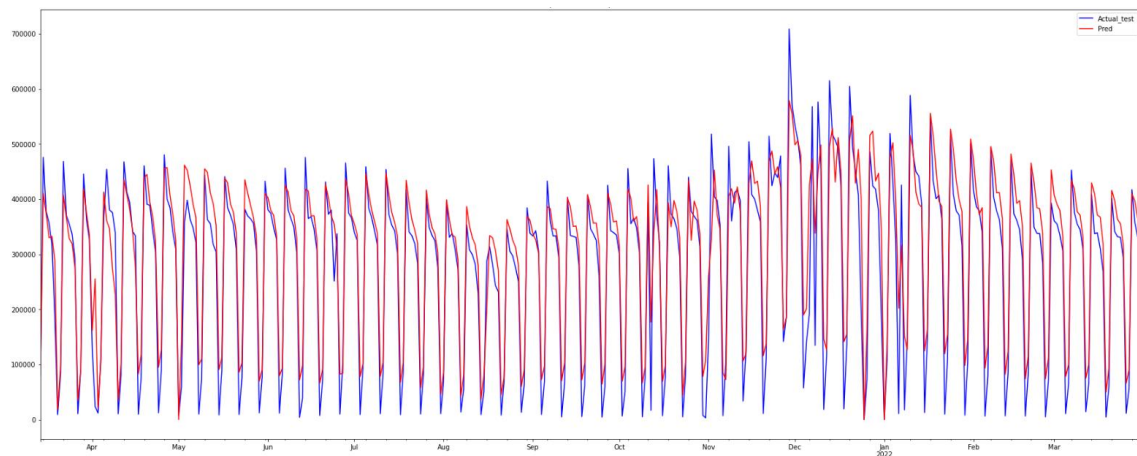
Similarmente a los modelos ARIMA, se descompone la serie temporal en diferentes contribuciones, obteniendo la componente estacional, pero siendo especialmente bueno en ello, para lo cual usa series de Fourier de diferentes órdenes.



Componentes Prophet

En ella vemos cómo la tendencia positiva en todo momento sufre un aumento aún mayor a partir de 2020 como consecuencia del efecto del COVID en la vida de las personas y la transformación que ha sufrido hacia un mundo donde el comercio online se ha visto altamente afectado de forma positiva. Además, se observa tanto la estacionalidad semanal como la anual, la cual tiene sus picos

superiores a finales de año e inferiores en el período de vacaciones de verano.



Gráfica valores reales y predicciones de test Prophet

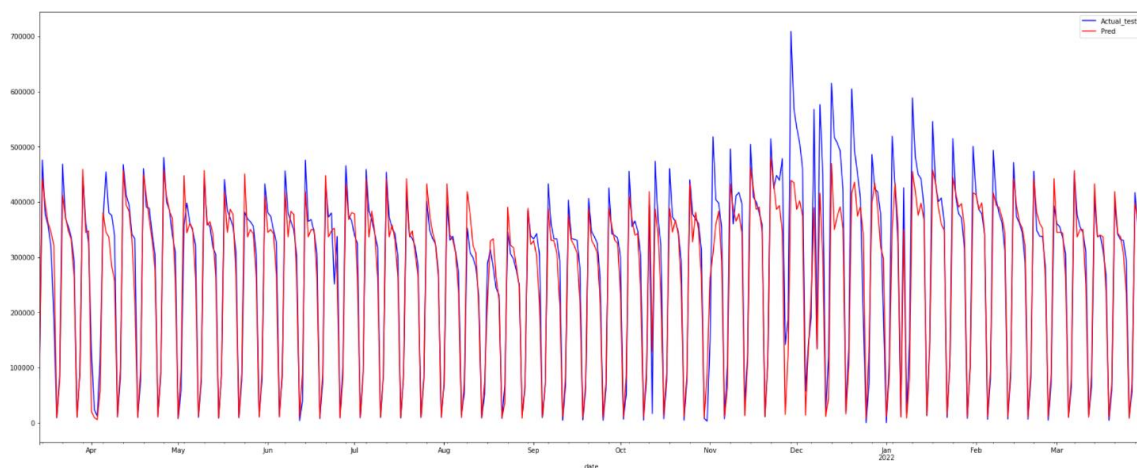
En este caso, se ajusta aún menos a los valores inferiores correspondientes a los fines de semana, probablemente debido a la tendencia resultante después de 2020 que indica un aumento muy pronunciado. Sin embargo, parece ajustarse mejor a los valores superiores

KNN

El algoritmo K-nearest neighbors (kNN) es un algoritmo que pertenece a los algoritmos de aprendizaje supervisado simples y de fácil implementación que pueden ser utilizados para resolver problemas de clasificación y de regresión.

Funciona de una manera intuitiva, de forma que busca las distancias entre unas muestras y otras de forma que selecciona un número determinado de vecinos y se considera el valor que toma la etiqueta para cada uno de ellos y se devuelve como predicción el valor medio de dichos valores.

De este modo, para su implementación se hace uso de la función GridSearchCV perteneciente al módulo sklearn para el cálculo de K. Una vez obtenidos el valor de dicho parámetro se realiza la predicción de los datos de test para testarlo haciendo uso de la función KNeighborsRegressor del mismo módulo.



Gráfica valores reales y predicciones de test K-nearest neighbours

En este caso se puede observar que se tiene muy buen comportamiento con los valores bajos de los fines de semana (similitud propia de los sábados y los domingos), mostrando, por otro lado, un rendimiento peor en períodos de alto volumen y volatilidad ya que tampoco se adapta a los picos de fin de año.

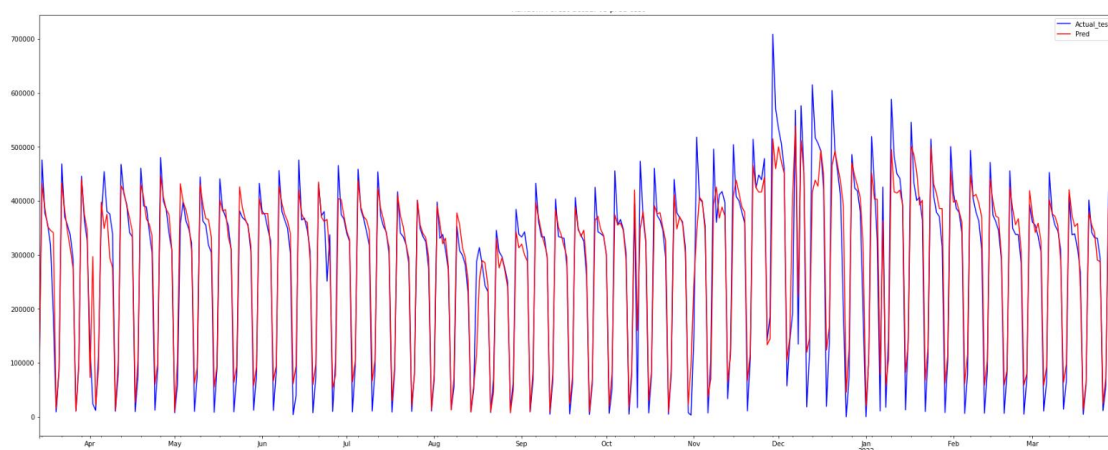
Random Forest Regressor

Random Forest para regresión es un algoritmo basado en la combinación de árboles de decisión clasificatorios que se ajustan en varias submuestras del conjunto de datos y utiliza el promedio para mejorar la precisión de la predicción y controlar el sobreajuste.

Por un lado, entre sus ventajas se encuentran que se puede aplicar para clasificación y regresión, que puede gestionar grandes cantidades de datos con mayor dimensionalidad y que es un método efectivo cuando existe una falta de gran cantidad de los datos manteniendo la precisión. Y, por otro lado, entre sus desventajas se encuentra que no predice más allá del rango de los datos de entrenamiento y que puede sobreajustar los conjuntos de datos que son concretamente ruidosos.

Para su implementación se hace uso de la función *GridSearchCV* perteneciente al módulo *sklearn* debido a la necesidad de probar con diferentes parámetros con el objetivo de encontrar la mejor combinación de los mismos. Existe una gran cantidad de parámetros para los cuales se pueden establecer rangos del tamaño que se desee, de forma que el problema puede hacerse tan grande como se quiera en función del tiempo disponible o la potencia de computación, entre otros. En este caso se ha decidido usar *GridSearchCV* para los parámetros *max_depth*, *n_estimators*, *max_features* y *bootstrap* con MSE como función a minimizar a modo de prueba que permita ejecutarse en un tiempo asequible.

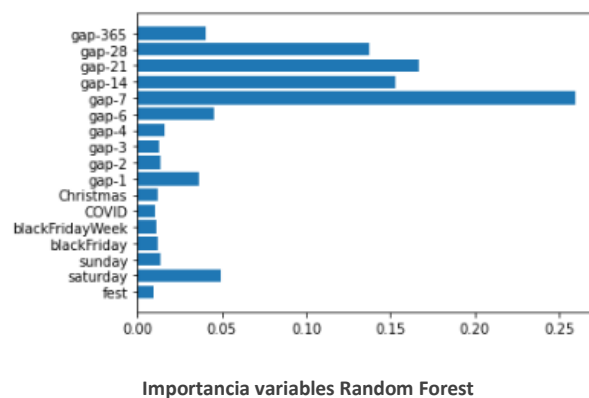
Una vez obtenidos los valores para dichos parámetros y de forma paralela al caso de SARIMAX, usamos los parámetros calculados para implementar un modelo *Random Forest Regressor* del módulo *sklearn* alimentado con los datos y los parámetros para testarlo.



Gráfica valores reales y predicciones de test Random Forest Regressor

En este caso se observa un comportamiento que se aproxima a los valores bajos en algunas ocasiones, pero no en todos los casos, mientras que para volúmenes altos y mayor volatilidad ofrece un mejor rendimiento que SARIMAX o KNN.

Se puede comprobar la importancia de las diferentes variables introducidas en el modelo:



Se observa que las variables con mayor peso son las del *lag7* y múltiplos, además de los sábados (día de la semana con menor volumen de registros).

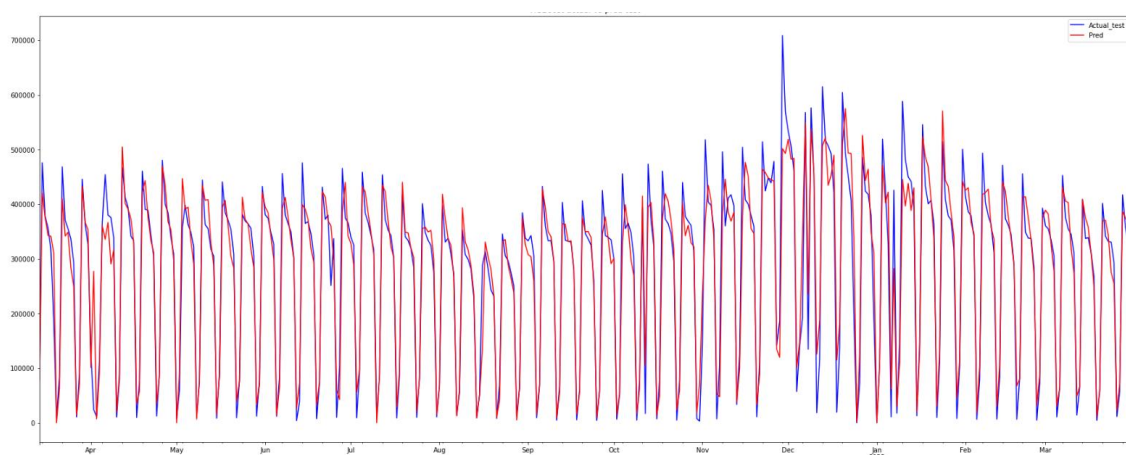
XGBoost

El algoritmo *XGBoost* es uno de los más usados en la actualidad debido a su facilidad de implementación, buenos resultados, diferentes aplicaciones y porque existe para un gran número de lenguajes. Además, se trata de un algoritmo paralelizable, lo que permite usar de forma óptima la potencia de procesamiento disponible haciéndolo idóneo para conjuntos de datos de gran tamaño.

Este es un algoritmo de gradiente de árboles reforzados, siendo *gradient boosting* un algoritmo de aprendizaje que predice con precisión una variable objetivo combinando las predicciones de un conjunto de modelos más simples y débiles.

Como en el caso de *Random Forest*, se va a hacer uso de *GridSearchCV* para luego usar los parámetros obtenidos en el modelo *XGBoost* del módulo homónimo.

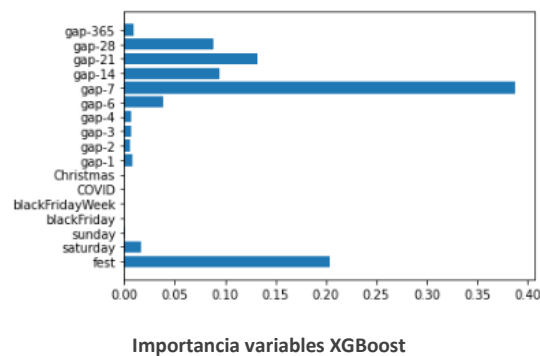
Cabe destacar que, por la propia naturaleza del algoritmo, al igual que *Random Forest* no es necesario el uso de los datos escalados.



Gráfica valores reales y predicciones de test *XGBoost*

Este algoritmo presenta un buen rendimiento tanto en volúmenes bajos (si bien no se ajusta del todo es bastante decente), y además se aproxima también a los altos, presentando a priori los mejores resultados hasta el momento.

En cuanto a la importancia de las diferentes variables introducidas en el modelo, se obtiene lo siguiente:



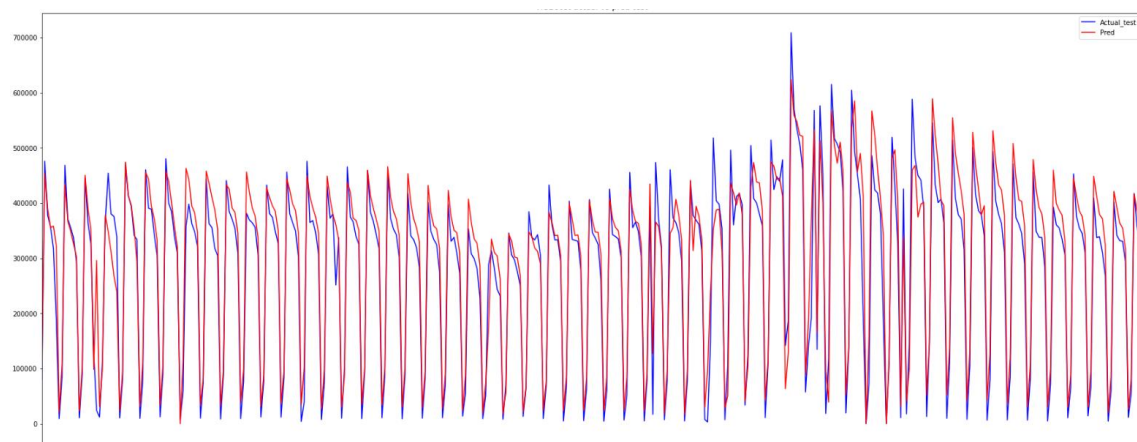
En este caso se puede observar con mayor claridad la importancia de las variables, donde el lag7 tiene una gran importancia seguido de sus múltiplos y, en segundo lugar, aparecen los festivos nacionales.

LSTM

LSTM ("Long-Short Term Memory") es una evolución de las redes recurrentes estándar para problemas de aprendizaje automático donde existe la componente temporal, debido a que su propia arquitectura en forma de celdas y *loops* permiten la transmisión y recordar la información en diferentes pasos.

Esto es interesante porque, una vez crece el tamaño de los datos, existe un gap temporal entre las informaciones de tal forma que una red neuronal recurrente estándar no es capaz de recordar la información. Sin embargo, *LSTM* tiene una arquitectura que le permite almacenar esa información durante grandes intervalos de tiempo.

Para implementar este modelo se hace uso de *keras* y de sus componentes. Keras es un módulo muy importante que proporciona bloques modulares sobre los que se pueden desarrollar modelos complejos de *Deep Learning*. En cuanto a la red implementada, consta de la propia *LSTM*, *dropout* y la capa densa de salida, todas ellas con una configuración típica de sus parámetros.



Gráfica valores reales y predicciones de test LSTM

Se puede observar que tanto en los valores altos como a los valores bajos se aproxima aceptablemente. Sin embargo, muestra diferencias en los valores de los días entre semana conforme estos avanzan.

Evaluación de los modelos

Una vez implementados y testados los distintos modelos descritos anteriormente, es el turno de evaluar el error proporcionado por los mismos en el conjunto de datos de test.

Se obtienen los siguientes resultados:

RMSE	MAE	models
53656.031948	37667.855030	SARIMAX
55870.203534	42357.515211	Prophet
49831.885970	31890.101175	KNN
44943.222224	30531.417441	Random Forest
41747.458484	28268.717799	XGBoost
45027.742670	32911.297351	LSTM

Errores por tipo de modelo en test

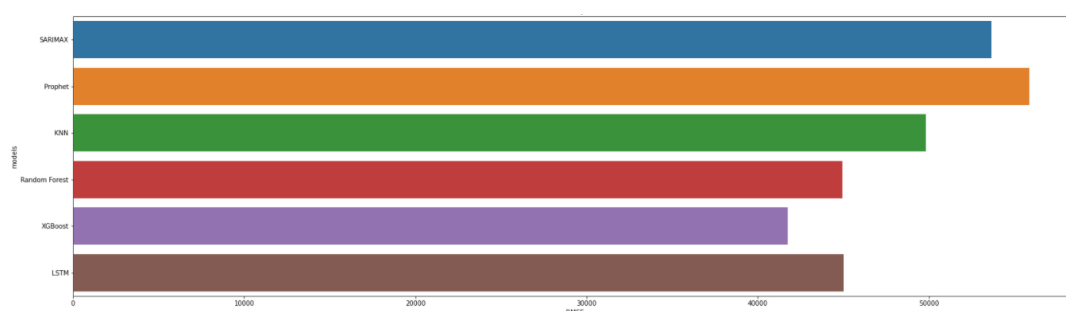


Gráfico RMSE modelos en el conjunto de test

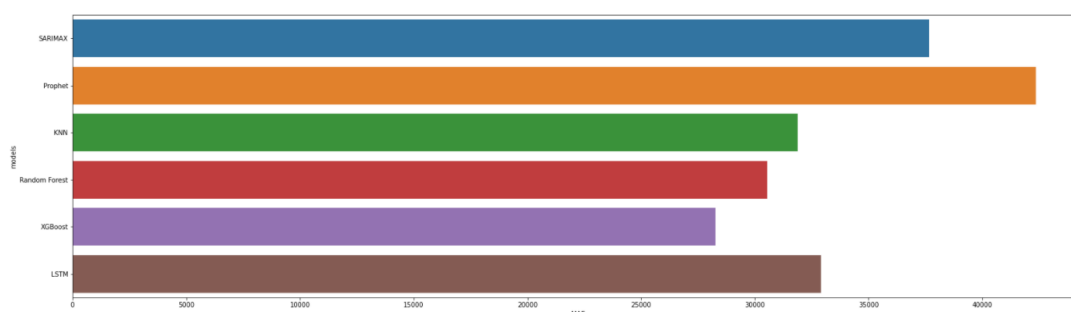


Gráfico MAE modelos en el conjunto de test

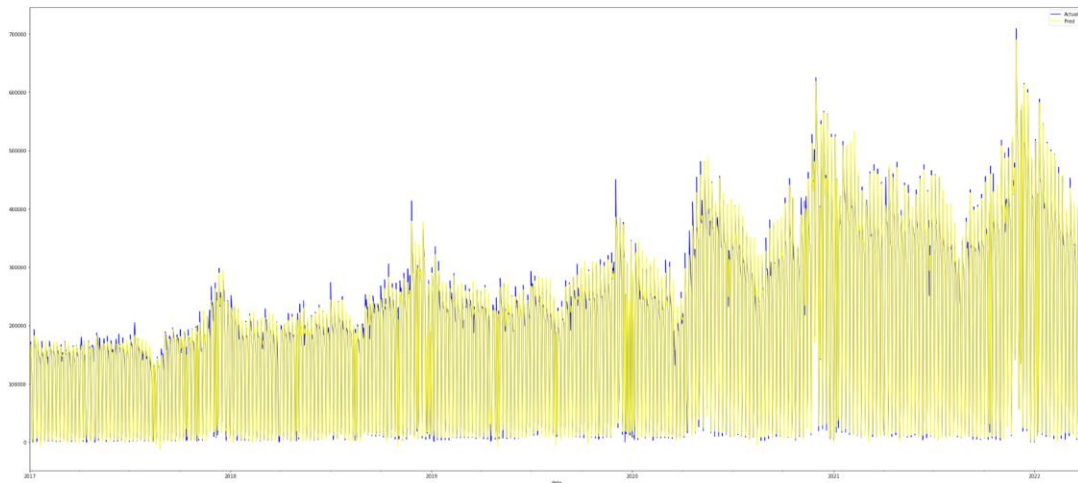
Se puede observar que tanto en el caso del RMSE y el del MAE tanto *Random Forest* como *XGBoost* se diferencian del resto de modelos, consiguiendo *XGBoost* errores inferiores en ambos casos.

Tras estos, aparece *LSTM* y *KNN* con errores situados entre los valores de *Random Forest* o *XGBoost* con los conseguidos por *SARIMAX* y *Prophet*, los cuales muestran los peores resultados.

Cabe destacar que *KNN* obtiene un RMSE mayor que *LSTM*, mientras que *LSTM* obtiene un MAE mayor. Esto es debido a que *KNN* se ajusta mejor a los valores bajos de los fines de semana mientras que en los períodos volátiles de mayor volumen tiene mayores carencias. En el caso de *LSTM*, si bien no es tan exacto con los valores inferiores sistemáticamente afectando al MAE, sí tiene un mejor comportamiento en período de volatilidad, por lo que su RMSE es menor.

Como se ha expuesto, tanto *Random Forest* como *XGBoost* presentan los mejores resultados. De cara a elegir un modelo para prepararlo para usarlo la interfaz, se elige *XGBoost* por tener un error inferior en ambos casos.

De cara a preparar el modelo que va a ser usado en la interfaz, se entrena con todos los datos.



Fit XGBoost con todo el conjunto de datos

Además, se comprueba que el error en el fit con todos los datos es mayor que en el train, por lo que es un indicador de que no hay overfitting presente.

Por último, se guarda el modelo entrenado para ser utilizado en la interfaz haciendo uso del módulo *joblib*.

Interfaz web

Finalmente, para simular una posible herramienta que un cliente pudiera usar, se ha creado una interfaz web que permita tanto ver los datos históricos de la serie temporal como hacer la previsión para el siguiente día.

Como se explicó anteriormente, el último día del histórico de la serie es el 31 de marzo de 2022, por lo que se establece el día 1 de abril de 2022 como el día a predecir para simular una situación real. Para ello se ha generado un archivo CSV con los datos de las variables para dicho día y que es leído por la herramienta a la hora de generar la predicción.

Para ello se ha utilizado el módulo *streamlit*, que permite crear aplicaciones desarrolladas en Python.

Su ejecución puede llevarse a cabo mediante *ngrok*, servicio que permite crear el servidor local en un subdominio para poder visualizarlo fuera de la LAN, a través de internet. Sin embargo, para más sencillez y evitar el uso de dicho servicio se ha configurado a través de LocalTunnel en un entorno local.

```
# run app
!streamlit run app.py & npx localtunnel --port 8501
```

Despliegue de la aplicación desarrollada en Streamlit en un entorno local

Descripción de la herramienta

La herramienta está compuesta por 3 componentes principales.:

- Cuadro de opciones lateral
- Gráfico histórico
- Previsión
 - Datos calendario día previsión
 - Datos previsión y gráfico semanal

Options

Start date

2021/01/01

End date

2022/03/31

Upload plot

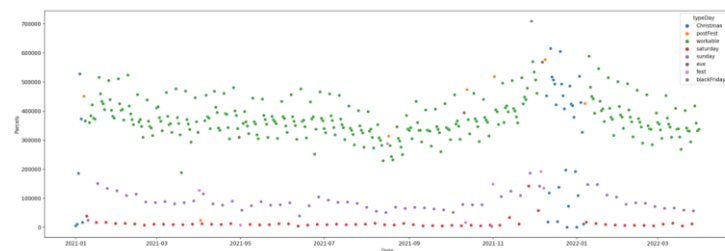
Launch forecasting model

Execute prediction

Parcels Forecasting

Historical data

Number of parcels per day



Aspecto general de la herramienta

Cuadro de opciones lateral

Primero, los desplegables de fechas permiten establecer los límites para visualizar en el gráfico del histórico de datos de la serie temporal. Para ello, se selecciona cada uno de los días en el calendario que aparece al *clickar* en ellos y pulsando posteriormente en “*Upload plot*” se actualiza el gráfico.

Options

Start date

2021/01/01

End date

2022/03/31

Upload plot

Launch forecasting model

Execute prediction

Cuadro de opciones lateral

Por otro lado, el botón “*Execute prediction*” lanza el modelo de previsión para el cálculo del valor para el siguiente día. Además, activa la visualización de dicho dato, como se verá más adelante.

Gráfico histórico

Permite la visualización de toda la serie temporal histórica y puede filtrarse a través del cuadro de opciones lateral como se ha explicado anteriormente. Se trata de un gráfico de puntos donde además se distingue por su color el tipo de día.

Historical data

Number of parcels per day

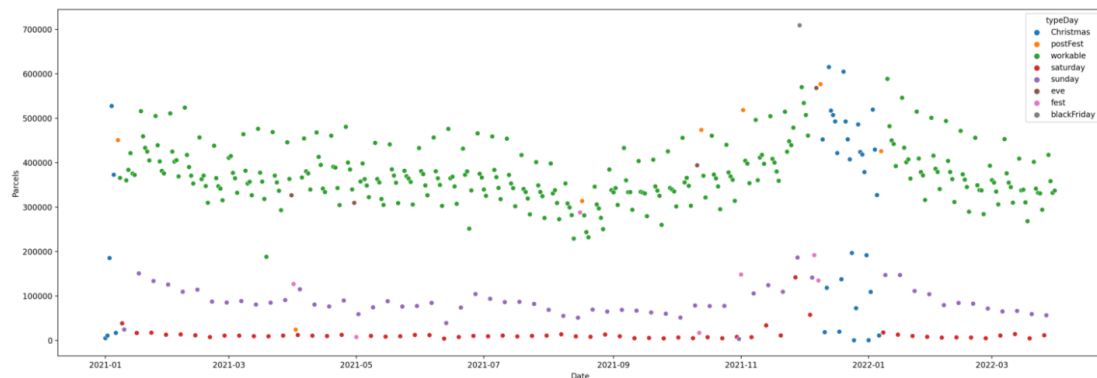


Gráfico histórico serie temporal

Previsión

Primero se visualizan los datos de calendario pertenecientes al día a prever, es decir, el día 1 de abril de 2022.

Predict

Forecast number of parcels for the next day.
The characteristics of the next day are as follows:

2022-04-01

NEXT DAY

No

HOLIDAY

No

WEEKEND

No

BLACK FRIDAY

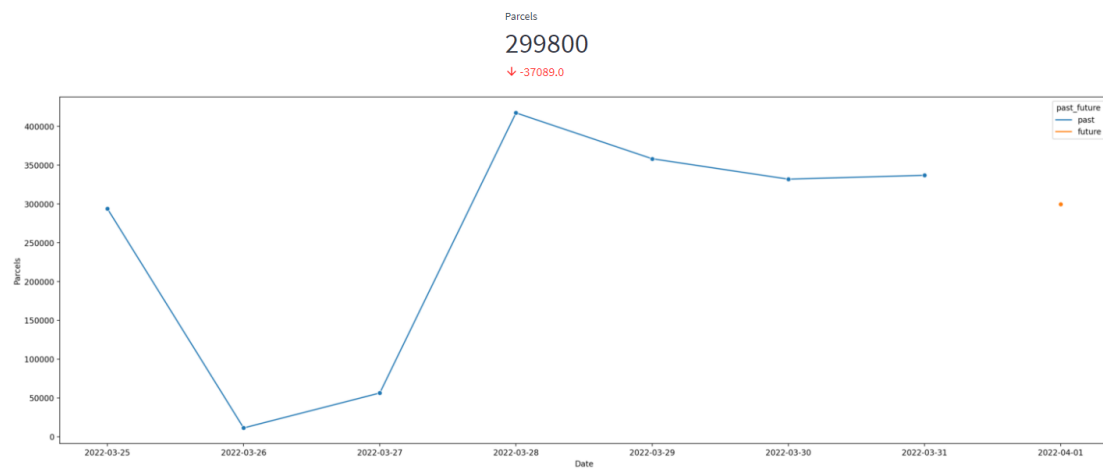
No

COVID LOCKDOWN

Características día a prever

Una vez lanzado el modelo de previsión para el cálculo de dicho día, se activa la visualización de los resultados, donde aparecen tanto el valor calculado (y su diferencia con el día anterior), además de desplegarse un gráfico con los datos de la última semana y el nuevo valor calculado.

The prediction for the next day is



Visualización previsión realizada

Conclusiones y líneas futuras

Conclusiones

Una vez finalizado el desarrollo del trabajo, se pueden entrar las siguientes conclusiones:

- El análisis y predicción de series temporales se trata de un problema con una importancia extrema en el mundo de la ciencia de datos porque hay una gran cantidad de sectores y ámbitos de estudio donde saber qué va a ocurrir en el futuro es muy importante para adaptarse a esos cambios cuanto antes. De esa manera pueden optimizar los recursos necesarios para cumplir con esa previsión y, por ejemplo, poder maximizar las ventas o minimizar el número de trabajadores encargados de llevar a cabo ese trabajo con una mejor planificación.
- Existen una gran cantidad de algoritmos capaces de predecir este tipo de series de datos. En este trabajo se han probado varios de ellos con dos objetivos: el aprendizaje por parte del alumno para el uso de algoritmos de diferente índole de forma general y poder comparar algoritmos de diferente tipo que den lugar de alguna manera a inclinarnos claramente por unos antes que otros. Ambos objetivos se han cumplido:
 - Se han encontrado diferentes escalas de resultados en los algoritmos tratados, encontrando algoritmos como *XGBoost* o *Random Forest* conocidos por su buen rendimiento, proporcionando *XGBoost* los mejores resultados. Son seguidos de *LSTM* que, actualmente con el desarrollo del *Deep Learning*, va ganando enteros, y *KNN* que también presenta resultados de calidad media. Por último, algoritmos estadísticos más tradicionales como *SARIMAX* o *Prophet* quedan a la cola.
 - Personalmente ha sido un reto enfrentarse a un problema de series temporales y los diferentes algoritmos, si bien no se ha profundizado tanto como se podría, ya que el objetivo no era encontrar el mejor modelo hasta el extremo en cuanto a modelos y *feature engineering*, si no trabajar con diferentes de ellos y compararlos en líneas generales.
- En cuanto a la serie temporal en sí y los resultados obtenidos, el período actual es convulso y difícil a los siguientes factores:
 - El “efecto COVID” ha hecho que desde 2020 y durante 2021 el comercio online y, por tanto, las empresas que reparten dichas compras, aumentaran drásticamente. Esto muestra una tendencia aún más alta de la que había anteriormente (año tras año se supera), pero en los últimos tiempos que en 2022 ya se está mostrando que no se mantiene y está sufriendo un estancamiento. Sin embargo, este mismo problema justifica conseguir herramientas analíticas que sepan gestionar esta situación y servir aún más de ayuda a este tipo de negocios para poder adaptarse cuanto antes a la realidad que les rodea.
 - Existen períodos de mucha volatilidad a final de año debido al Black Friday y la navidad que han resultado difíciles de gestionar por los modelos.

Líneas futuras y mejoras

Existen varias líneas futuras de desarrollo:

- Mejoras de los modelos:
 - Uso de otros modelos de *gradient boosting*, *clustering*, redes neuronales, *SVM*, etc.
 - Uso de otras técnicas de validación como *cross validation*.
 - Transformaciones de la serie como, por ejemplo, logaritmos, transformación de *Box-Cox*... que consigan mejores resultados debido a la obtención de series transformadas más fáciles de predecir para los algoritmos.
 - Introducir otras variables como las diferencias entre días en lugar del valor total de paquetes de cada día u otros períodos importantes como, por ejemplo, los períodos de rebajas.
 - Aumento de la intensificación en la búsqueda de parámetros para los modelos.
- Obtención del maestro de oficinas correctamente:
 - Realizar la previsión por cada par de oficinas para gestionar los recursos necesarios para ello mediante un modelo de programación lineal.
 - Crear una interfaz más completa adaptada a esa lógica de pares.