RETO ATMIRA STOCK PREDICTION - MINERVA

El trabajo desarrollado para el reto Atmira Stock Prediction ha consistido en las siguientes etapas:

- Primeras pruebas usando los datos en el formato original para establecer un baseline.
- Análisis exploratorio para entender cómo se comportan los distintos atributos.
- Generación de características para enriquecer los modelos.
- Estudio de rendimiento con un modelo simple y distintas características.
- Hiperparametrización de modelos más complejos en búsqueda del resultado óptimo.
- Stacking de distintos modelos.

Para el desarrollo del trabajo se siguieron inicialmente varias líneas de exploración que han conducido a distintos resultados y conclusiones:

1. Entrenar un único modelo usando todas las instancias disponibles anteriores al test.
2. Entrenar un único modelo usando sólo las instancias que corresponden al mismo periodo del conjunto de test pero del año anterior.
3. Entrenar N modelos distintos agrupando los productos en clusters según el comportamiento de la serie temporal de unidades vendidas.

Finalmente, la primera de estas alternativas ha sido la que ha obtenido mejores resultados, tanto en nuestro proceso de validación interno como en las entregas realizadas sobre el conjunto de test a lo largo del concurso.

Análisis exploratorio

En el notebook EDA Features se incluye el primer análisis exploratorio al completo, del cual se hace aquí un resumen de lo más destacado.

Dado que hay más de 4000 productos, es difícil hacer una exploración individual de las series temporales de cada uno. Por ello, nos ayudamos de la agrupación en clusters o de análisis de las medias. Analizando las medias de visitas y unidades vendidas de cada día descubrimos algo importante a tener en cuenta a la hora de crear los modelos predictivos. Como se puede observar en las imágenes que se adjuntan, hay una subida importante de la media de visitas a la web a partir de febrero de 2016. Sin embargo, esta subida de visitas no resulta en un incremento del número medio de unidades vendidas (excepto en los períodos especiales como las rebajas o el Black Friday). Por tanto, esto nos hizo darnos cuenta de que es importante normalizar las visitas de acuerdo a la media en cada periodo. Así podemos evitar que el regresor modele una correlación positiva excesivamente fuerte entre el número de visitas y el número de ventas.

Por otro lado, también se observa una clara estacionalidad semanal en las series temporales. Las mayores ventas y visitas corresponden al fin de semana, mientras que los valores más bajos al martes/miércoles. Es por ello que resulta fundamental que cada instancia tenga información acerca del día de la semana en el que se encuentra, aparte de la época del año. Esta información se puede incorporar al modelo usando directamente la serie temporal de visitas, o codificando el día de la semana.

El último paso del análisis exploratorio comprende el análisis de la importancia de los atributos a la hora de predecir las unidades vendidas. Haciendo este estudio con un regresor tipo Random Forest se observa que el atributo más importante son las visitas, tal y como muestra la imagen que se adjunta

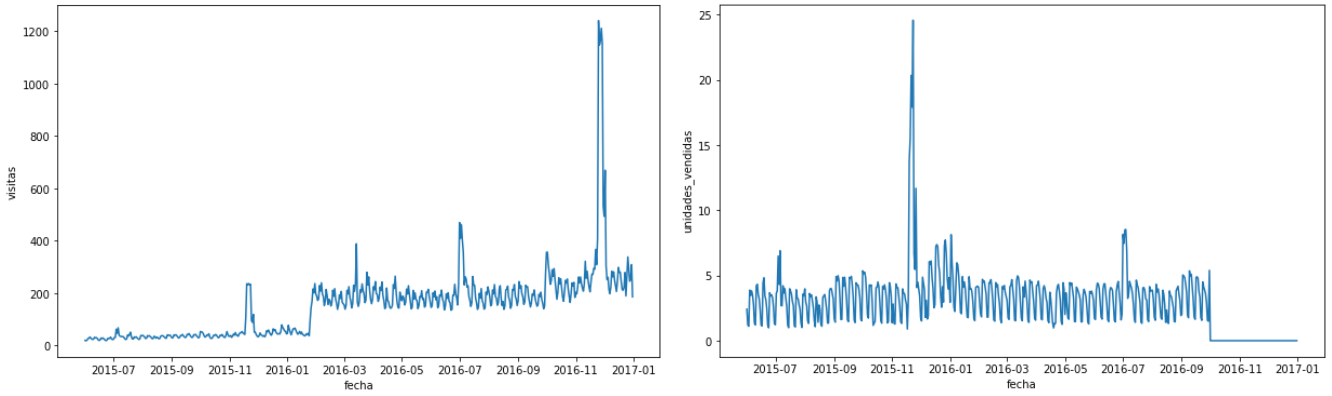

Fig 1. Media diaria de visitas y unidades vendidas

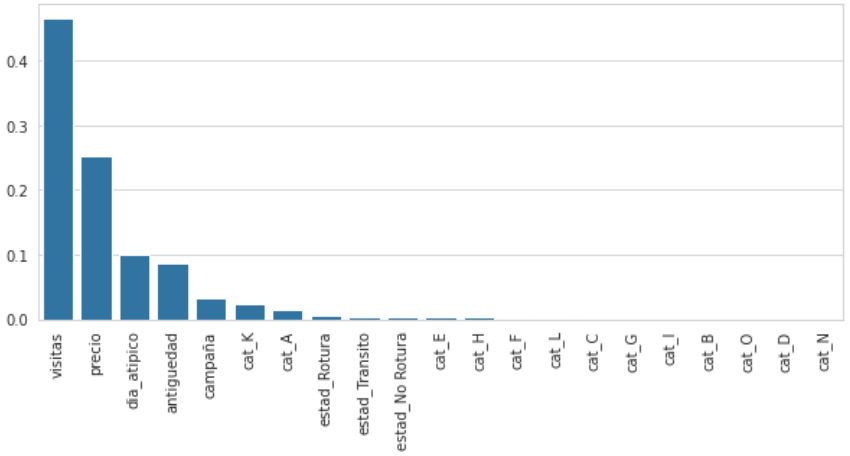

Fig 2. Importancia de atributos

Manipulación de variables

Tras el estudio de la importancia de los atributos, decidimos centrar los esfuerzos en generar características de la serie temporal visitas para agregarlas a cada instancia. Por otro lado, había otras variables categóricas que tenían poca importancia y que aumentaban significativamente el número de columnas de tipo one-hot. Dado que una gran cantidad de variables puede no ser conveniente por problemas de sobreajuste, decidimos eliminar las columnas categoria_uno, categoria_dos, y estado.

En resumen, se toman las variables iniciales (visitas, precio, antigüedad, campaña, dia_atipico) y se intenta enriquecer la información generando nuevas variables.

En primer lugar, a cada instancia le añadimos columnas con información de una ventana cercana de la serie temporal de visitas. En concreto, se añaden las siguientes características usando el valor de visitas normalizado de acuerdo a la media y desviación estándar de todos los valores del dataset, separando en los dos períodos mencionados anteriormente (a partir del 25 de enero de 2016 incrementa significativamente la media de visitas):

- Diferencia de visitas con respecto al valor del día anterior y de 7 días antes.
- Media de visitas de los últimos 3 y 7 días.
- Mediana de las visitas de los últimos 3 y 7 días.
- Desviación estándar de los últimos 3 y 7 días.

Dada la clara estacionalidad de la serie y la importancia de la época del año en la que estemos, se deciden añadir características asociadas a la fecha. En concreto se añaden dos nuevas variables:

- El día del año (número de 0 a 365).
- El día de la semana codificado en formato one-hot.

Todos estos conforman los 21 atributos que se usaron en la entrega de la primera fase:
`[visitas, precio, antiguedad, campaña, dia_atipico, visitas_diff_-1, visitas_diff_-7, visitas_mean_3, visitas_mean_7, visitas_median_3, visitas_median_7, visitas_std_3, visitas_std_7, DiaDelAño, Lunes, Martes, Miercoles, Jueves, Viernes, Sabado, Domingo]`

En la segunda fase, se intentaron analizar más en detalle las predicciones y observamos dos aspectos importantes:

- Para valores normalizados muy altos de visitas (aquellos que se alejan mucho de la media), los resultados eran muy variables para valores similares. Para intentar paliar este problema, se decidió añadir dos columnas más con valores relativos de las visitas:
    - El número de visitas normalizado de acuerdo al máximo histórico de visitas de ese producto.
    - El número de visitas normalizado de acuerdo el máximo de visitas de ese día teniendo en cuenta todos los productos.

- Se observó que cuando se daban valores anormalmente altos de unidades vendidas, solía ocurrir el estado Tránsito ese mismo día o al día siguiente. Por tanto, decidimos añadir 3 variables más al estudio: la variable estado_transito con el valor de ese día (0/1), y otras dos variables que indican el número de días desde el último estado tránsito y el número de días hasta el siguiente. Finalmente, estas 3 variables no fueron utilizadas puesto que no mejoraba los resultados en nuestro proceso de validación interno.

Selección del modelo

Tras realizar el análisis exploratorio y decidir cuáles eran las variables que más influyen en nuestro objetivo (que es predecir las unidades vendidas) pensamos que quizás sería interesante buscar un período similar al que debíamos predecir para así poder validar nuestros modelos y tener una idea sobre cómo de buenos eran nuestros resultados. El período a estimar comprendía los meses de Octubre y Diciembre de 2016, un período en el que encontramos fechas señaladas como el Black Friday. Pudimos observar que el comportamiento de la variable “unidades vendidas” era muy parecido entre este período y el período de verano que incluye las rebajas. Por tanto, decidimos usar este período (Junio-Julio 2016) para hacer la validación de nuestros modelos y buscar la mejor configuración y dejaremos los últimos dos meses (Agosto-Septiembre 2016) como test para la elección del modelo final. Teniendo un conjunto de validación y otro diferente para test, nos aseguramos de no sobreajustar el modelo al conjunto de validación.

Lo primero que hicimos fue establecer un baseline con un modelo Random Forest con los parámetros por defecto, usando el dataset con los 21 atributos especificados anteriormente. Una vez que teníamos este baseline, decidimos parametrizar varios modelos sobre el conjunto de validación. Los modelos que elegimos y sus mejores resultados se detallan a continuación:

Modelo	**Mejor resultado (Validación Junio-Julio 2016)**
XGBRegressor	2,50
LGBMRegressor	2,69
CatBoostRegressor	2,71
RandomForestRegressor	2,75

Para encontrar la mejor parametrización de los modelos, hemos usado una búsqueda aleatoria (random search). En total hemos entrenado más de 600 modelos con diferentes parámetros. El espacio de

búsqueda del random search para cada modelo están detallados en el script selección_modelo.py. En la siguiente tabla se detalla el espacio de búsqueda y la mejor parametrización encontrada para el XGBRegressor, el cual hemos usado como modelo de la entrega de la fase local.

Parámetro	Espacio de búsqueda	Mejor valor encontrado
booster	gbtree | gblinear	gbtree
n_estimators	10, 20, …, 490, 500	140
eta	(0.0001, 0.5)	0.0108
min_child_weight	(0.001, 1)	0.0547
max_depth	3, 4, 5, …, 49, 50	11
gamma	(0.0001, 2)	0.2543
subsample	(0.01, 1)	0.8530
colsample_bytree	(0.01, 1)	0.8759
lambda	(0.0001, 5)	0.0384
alpha	(0.0001, 5)	1.0571
eval_metric	mae | rmse	mae

Fase 2 - Stacking de modelos y nuevas características

El modelo entregado en la Fase 1 obtuvo una métrica de 4,5175 sobre el conjunto “Estimar”. Después de obtener este feedback decidimos seguir dos vías de investigación, la primera fue hacer un estudio más extenso de las variables que podían tener más importancia y la segunda fue usar un nuevo modelo, en este caso, un stacking de modelos.

El stacking es un tipo de ensemble que consiste en concatenar la salida de varios modelos individuales y utilizar un regresor final que permita calcular la predicción final. De esta forma, el ensemble es capaz de aprovechar los puntos fuertes de cada modelo individual, obteniendo una predicción final más precisa.

Tras varias pruebas sobre el conjunto de validación decidimos usar cuatro modelos como regresores individuales del ensemble. Concretamente usamos un XGBRegressor basado en árboles de decisión, otro XGBRegressor basado en funciones lineales, un LGBMRegressor y un CatBoostRegressor. Estos modelos se han elegido basándose en los buenos resultados obtenidos como modelos individuales e intentando mantener cierta variabilidad entre los modelos individuales del stacking. Así mismo, también analizamos la estructura del stacking, probando tanto varios modelos (MLP, LinearRegression, RF) y diferentes configuraciones. Finalmente nos quedamos con un MLP con dos capas ocultas (32, 16) sin función de activación como Regresor final.

Todos los modelos individuales fueron parametrizados usando el conjunto de validación y dejamos el conjunto de test para la selección del modelo final. La configuración final de modelos está detallada en el script prediccion.py.

Para la entrega intermedia de la segunda fase, decidimos probar el XGBoost con las nuevas características (visitas relativas y tránsito) y el stacking con las características usadas en la fase local. Los resultados de esta entrega, junto a los resultados de la fase local, nos ayudaron a confirmar que los resultados de nuestro conjunto de test interno tienen un comportamiento similar a los resultados finales (datos Estimar.txt). Así mismo, vimos como, efectivamente, tanto el stacking como las nuevas características habían ayudado a mejorar las predicciones.

Entrega (Estimar.txt Octubre-Diciembre 2016)		
	XGB	Stacking Ensemble
21 atributos (primera fase)	4.5175 (1ª fase)	**4.4980**
21 atributos (primera fase) + visitas_rel + tránsito	**4.5133**	-

Test Interno (Agosto-Septiembre 2016)		
	XGB	Stacking Ensemble
21 atributos (primera fase)	2.8248	**2.7586**
21 atributos (primera fase) + visitas_rel + tránsito	**2.8247**	-

A la vista de estos resultados decidimos estudiar en profundidad el comportamiento de los modelos con las nuevas características. Como podemos ver en la siguiente tabla, las visitas relativas ayudan considerablemente a mejorar la predicción, mientras que las características de tránsito no ayudaban, o incluso perjudicaban en el caso del stacking.

Test Interno (Agosto-Septiembre 2016)		
	XGB	Stacking Ensemble
21 atributos (primera fase)	2.8248	2.7586
21 atributos (primera fase) + visitas rel	**2.8004**	**2.7512**
21 atributos (primera fase) + tránsito	2.8246	2.7950
21 atributos (primera fase) + visitas_rel + tránsito	2.8247	2.7813

La siguiente figura ilustra el modelo Stacking final y las características que se han usado:

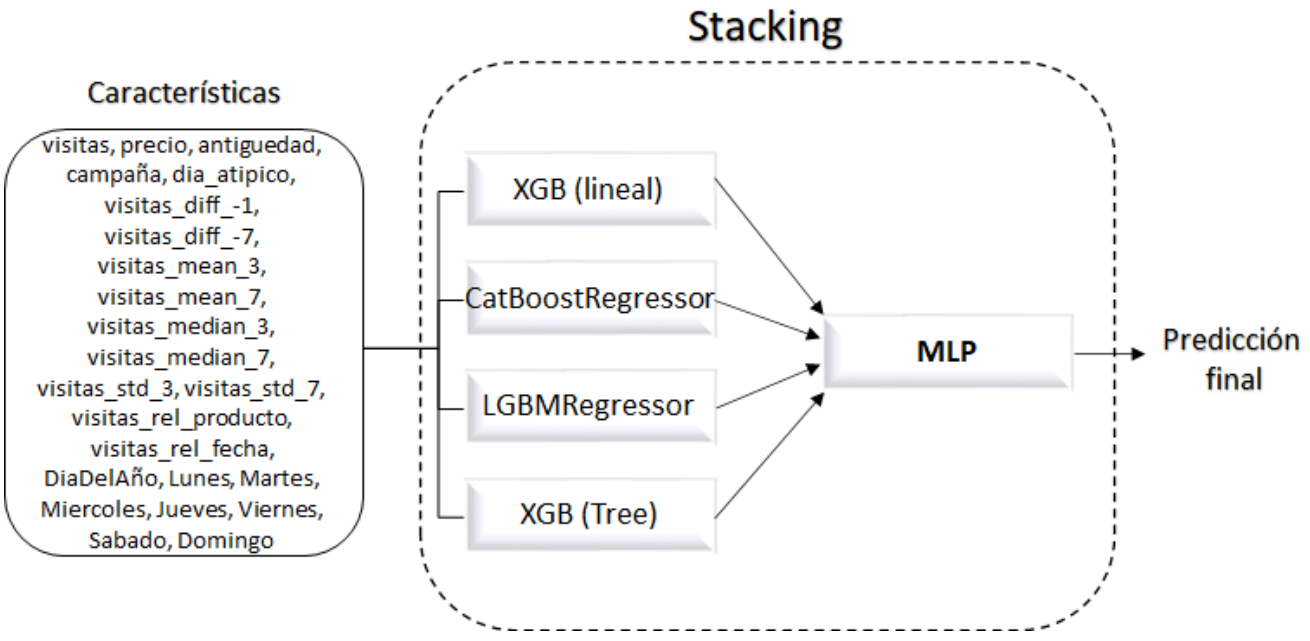

Fig 3. Modelo final