

INFORME SOBRE EL TRABAJO REALIZADO POR EL EQUIPO PYSHARKS

EN EL RETO ATMIRA STOCK PREDICTION

Este trabajo se encuadra dentro del reto **Atmira stock prediction**, dentro del datathon **Cajamar UniversityHack 2021**

Se procedió a realizar una tarea de predicción de unidades vendidas, tomando como base un conjunto de datos de las ventas de los productos en una tienda online entre las fechas 01/06/2015 y el 30/09/2016 respecto a los datos a modelar (archivo **Modelar_UH2021.txt**), y entre 01/10/2016 y 31/12/2016 en los datos a estimar (archivo **Estimar2.txt**), siendo la variable a predecir las unidades vendidas por cada registro cuyo identificador único es el id de producto y la fecha.

Para ello se llevaron a cabo los pasos frecuentes en todo proceso de aprendizaje automático supervisado: preprocesamiento y exploración de las variables involucradas en el conjunto de datos, selección de las variables más relevantes para la predicción, selección y entrenamiento de modelos utilizando los datos del fichero de modelado. Se hicieron varias iteraciones de este proceso, las cuales nos permitieron ir mejorando cada etapa y obteniendo nuevas vías por las cuáles experimentar, aprender y mejorar. Finalmente se hizo la predicción utilizando los datos del fichero de estimación y el mejor modelo obtenido. Dentro de los cuadernos de trabajo se explica de manera más detallada todo el proceso realizado y las opciones exploradas.

El objetivo del reto fue construir un modelo de predicción que logre obtener las mejores predicciones posibles midiendo los resultados a través de la siguiente métrica propuesta por la competición:

$$(0.7 * rRMSE) + (0.3 * (1 - CF))$$

Se procede a continuación a realizar una descripción de los diferentes pasos llevados a cabo:

Preprocesamiento y exploración de las variables

Nos encontramos con un dataset de modelado con un tamaño de 4045022 registros o filas y 11 variables o features más un target (unidades_vendidas) y un dataset de estimación de 218.263 registros o filas y las mismas 11 variables o features.

En primer lugar, procedimos a analizar los valores nulos en el dataset, hallando lo siguiente: 5.844 registros nulos en la variable categoría dos, 874165 registros nulos en la variable antigüedad y 2642911 registros nulos en la variable precio.

En segundo lugar, exploramos las variables del dataset a través de dos perspectivas: frecuencia o presencia de los valores de cada variable en el dataset de modelado, y comportamiento de cada una de estas variables frente a la variable respuesta. Encontramos de esta forma información relacionada con las visitas a los productos en la web, la antigüedad de estos en catálogo, categorías de clasificación, de estado de stock, periodos de campaña y de nivel de demanda, y de unidades vendidas (la información a estimar). Al analizar el comportamiento de las variables explicativas frente a la variable respuesta (frente a su media, así evitamos distorsiones por la desigual proporción de los valores en el conjunto de datos), obtuvimos algunas observaciones esperables: cuando el stock se rompía las ventas eran inexistentes, en situaciones de campañas de producto y mayor demanda las ventas eran mucho mayores, las ventas eran mayores cuando los precios eran bajos y descendían enormemente cuando se encarecen, y respecto a la antigüedad los productos más comprados no eran ni demasiado nuevos ni demasiado viejos. No obstante, también obtuvimos otras informaciones no tan obvias aparentemente: respecto a las categorías de producto las ventas solían concentrarse en un pequeño grupo de ellas, y aunque los productos frecuentemente visitados eran vendidos fácilmente, los números más altos de visitas no tenían las mejores ventas.

Debido a que las ventas de productos suelen relacionarse con estacionalidad o influencia temporal, nos dispusimos a representar algunas variables (incluida la variable respuesta: las unidades vendidas) en función de su evolución temporal a lo largo del periodo de tiempo contemplado en el dataset, con el fin de observar si esta influencia se daba en nuestros datos. De esta forma, centrándonos en la variable respuesta, observamos que esta parecía estar en cierto modo influida por el tiempo. Por consiguiente, nos dispusimos a extraer distintos aspectos de la información temporal (año, mes, días, semanas del mes, días de la semana) de la variable fecha que poseía el dataset. Representando estos aspectos temporales respecto a la variable respuesta, vimos una prometedora influencia sobre ésta, por lo que decidimos incluir la información temporal analizada en el modelo.

Analizado todo lo anterior, se procedió a aplicar las conclusiones extraídas al preprocesamiento y tratamiento de los datos. Primeramente eliminamos los duplicados e introducimos las variables temporales ya comentadas. Posteriormente imputamos los nulos de las variables precio y antigüedad según el valor anterior más cercano al registro, y los de la variable categoría_dos por la media de esta. Finalmente dummificamos las variables categóricas (cada categoría de la variable categórica se convierte en una nueva variable donde 1 indica presencia y 0 ausencia) y normalizamos las variables cuantitativas para obtener mejor precisión en el proceso de estimación del modelo. A consecuencia de este tratamiento, el conjunto de modelado se redujo a 2040036 registros y aumentó en 38 variables.

Selección de características

La selección de características se llevó a cabo en dos pasos:

1. Se procedió a aplicar el método de selección de características Recursive Feature Elimination (RFE), ofrecido por la librería de python sobre machine learning Scikit learn. RFE utiliza un estimador externo (usualmente un algoritmo de predicción) que realiza ponderaciones en las variables. El objetivo es seleccionar características considerando recursivamente conjuntos de variables cada vez más pequeños. En cada iteración, RFE irá descartando las variables menos importantes hasta quedarse, en principio, con las más óptimas. Con su atributo ranking pudimos ver la posición que el algoritmo RFE les había dado a nuestras variables en cuanto al peso o importancia en la estimación, output que utilizamos en el siguiente paso de selección. Como estimador se usó el regresor del algoritmo catboost.
2. Aplicado el RFE, utilizamos una función de elaboración propia que combinaba características de distinta posición en el ranking RFE, ponía a entrenar y testear el modelo con esta combinación de características, y devolvía una métrica junto con las características que habían dado lugar a esta. Esto se hizo con el objetivo de identificar si existía alguna información importante proporcionada por alguna variable no identificada como rank 1 en el paso anterior. En esta función se usó la métrica Relative Root Mean Square Error (rRMSE) debido a su relación directa con la métrica a optimizar en el reto y a su más fácil aplicación en los algoritmos. De esta manera, la configuración final de características a seleccionar fue aquella que mostró un menor rRMSE.

Realizado estos dos pasos, obtuvimos las variables que mejor rendimiento aportan al modelo, relacionadas estas fundamentalmente con la información temporal (año, mes, semana del mes, día de la semana y día), las visitas a los productos en la web, la antigüedad de los productos, la alta demanda, algunas categorías de producto y la rotura del stock.

Selección y entrenamiento del modelo

Seleccionadas las características, continuamos con la selección del modelo o algoritmo de aprendizaje supervisado, realizando una comparación entre varios respecto a su rendimiento en la precisión de la predicción. Debido a su escalabilidad, complejidad y fama de buenos resultados, decidimos comparar los modelos Extreme Gradient Boosting, Catboost, Random Forest, Redes Neuronales y tratar el problema como serie temporal.

Esta selección de modelos se llevó a cabo realizando los siguientes pasos:

1. Un entrenamiento con cross-validation con los parámetros por default de cada modelo para tener un punto de partida y una métrica inicial a mejorar.
2. Usando optimización de hiper parámetros Grid Search con cross-validation sobre cada uno de los modelos considerados, comparamos su rendimiento usando de nuevo la métrica rRMSE, elegida otra vez en esta fase del modelado debido a su más fácil aplicación y relación directa con la métrica del reto.

Los hiper parámetros a optimizar fueron, por modelo:

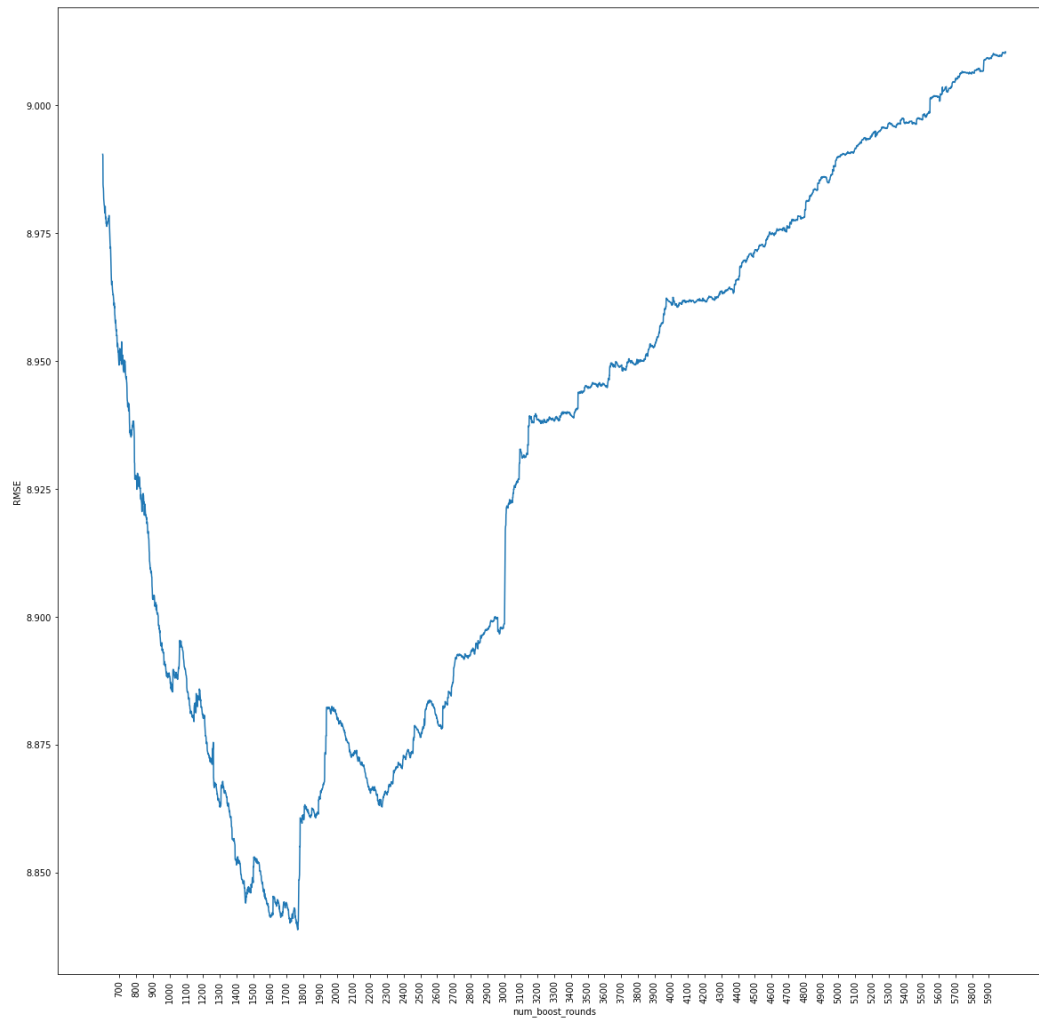
- **Extreme Gradient Boosting:** min_child_weight, eta, subsample, colsample_bytree, max_depth, gamma, num_boost_round, early_stopping_round.
 - **Catboost:** depth, learning_rate, iterations.
 - **Random Forest:** max_depth, min_samples_leaf.
 - **Redes Neuronales:** Número de inputs, capas ocultas, función de activación y optimizador.
3. Del paso anterior se obtuvo que los dos mejores modelos fueron el **Extreme Gradient Boosting** y el **Catboost**. Se investigó cómo poder optimizar aún más los resultados y se encontró que en el **Extreme Gradient Boosting** se puede utilizar la API Nativa en lugar de la de scikit-learn para mejorar el performance de los entrenamientos y los resultados, por lo que nos decidimos por este algoritmo para hacer un proceso de optimización de hiper parámetros aún más exhaustivo sobre este modelo.

En los anexos se pueden encontrar todos los cuadernos de trabajo realizados con cada uno de los enfoques, así como las bitácoras de resultados que arrojó cada modelo y la comparativa entre los resultados obtenidos.

La optimización aplicada, en principio, arrojó un buen resultado (rRMSE=2.485). No obstante, persiguiendo mejorar aún más la métrica decidimos aplicar algunos cambios. De esta forma, realizamos pruebas exhaustivas sobre los parámetros más importantes. Se decidió ir optimizando los parámetros en pares de acuerdo a su importancia, con el fin de no generar un Grid demasiado amplio y computacionalmente costoso, y también con el fin de conocer cuál es el impacto de cada variable en las métricas finales.

También decidimos en esta fase cambiar el tratamiento de nulos de las variables categoria_dos y antigüedad por una interpolación de valores. Todo ello dio lugar a una mejora apreciable en la métrica (rRMSE=2.4098).

Después de obtener la mejor combinación de parámetros posibles, obtuvimos esta gráfica en donde se muestra de manera clara cómo varía el rRMSE en función del número de ensembles:



Con los resultados obtenidos, el performance, la flexibilidad y las múltiples variantes que nos ofrece el XGBoost, es el algoritmo por el que hemos optado.

Predicción

Finalmente procedimos al proceso de entrenamiento del modelo, usando para ello el conjunto de datos de modelado. Entrenado los datos, llevamos a cabo el proceso de predicción sobre los datos del conjunto de estimación. Tras esto, construimos el archivo respuesta que la competición exige para la evaluación del modelo.

Anexos

En la carpeta anexos se pueden encontrar todos los cuadernos de trabajo con las distintas vías exploradas, algunas bitácoras de resultados y comparativos entre los resultados obtenidos, así como el archivo que contiene el modelo generado listo para importarse y usarlo para hacer predicciones.