

Informe - Trabajo Práctico Machine Learning

Descripción del problema

Rossmann es una tienda de retail de origen alemán que opera más de 3000 tiendas en 7 países europeos. Actualmente, se está intentando predecir las ventas diarias de cada tienda para los meses de agosto y septiembre de 2015, para ello, contamos con información histórica desde principios de 2013.

Las ventas diarias de cada tienda están influenciadas por muchos factores, tales como promociones, competidores y su cercanía, escuelas, feriados y factores estacionales, y conocerlas podría ser fundamental para poder hacer más eficiente la toma de decisiones de diferentes aspectos de negocio como el stock de los diferentes productos, la cantidad de empleados o los días que debería abrir o no la tienda, entre otros.

Preprocesamiento

En primer lugar se realiza un análisis al dataset “train” y “test”, donde se lleva a cabo un preprocesamiento relacionado al tipo de dato de algunas de las variables, como por ejemplo, el pasaje de la variable “Date” a tipo fecha para luego poder hacer Feature Extraction y generar nuevas variables a partir de ésta. Se realiza en esta etapa porque aun no conlleva una necesidad de procesamiento tan grande como lo hará cuando esté unido al dataset de “Stores”. Los features nuevos son: “Year”, “Month”, “Day”, “DayOfYear”, “DayOfWeek”, “Semana”, “IsWeekend”, “PrimeraQuincena”, “SegundaQuincena”, “ComienzaAnio”, “TerminaAnio”, “Quarter”, “YearWeek”. Todas estas nuevas variables entendemos pueden llegar a tener poder predictivo para el problema en cuestión que luego corroboramos en la evaluación de los modelos. Desde este primer momento notamos algunos aspectos en los datos que podrían señalar variables importantes para la predicción, como por ejemplo que las tiendas cerradas no tienen ventas o que las tiendas con promociones tienden a vender más, entre otras cosas. Algo muy importante en este paso fue deshacernos de la columna “Customers” ya que correlaciona con las ventas y es un dato que no se puede conocer para predecir.

En cuanto al dataset de “Store”, se analizan las variables relacionadas a Promo 2. Se genera un nuevo feature a partir de la columna “Promo2Interval”, el cual es una variable booleana que toma el valor 1 si el mes analizado se encuentra dentro del intervalo de Promo 2: “Promo2Timing”. Esta variable detecta si el mes es el de la Promo2 pero también si la promo 2 ya había comenzado en ese momento.

Dentro de la variable “CompetitionOpenSinceYear” se encuentran algunos valores outliers (<1990), los cuales se deciden convertirlos en NA para luego ser imputados. Se procede de esta forma ya que entendemos que pueden ser valores que han sido cargados erróneamente.

Se concatenan los datasets y se realiza la conversión de las variables no numéricas a numéricas ya que la mayoría de los modelos que se analizarán tienen como condición o bien funcionan mejor utilizando únicamente variables numéricas.

Gestión de valores faltantes

Previo a trabajar con valores faltantes se ha decidido separar el dataset en entrenamiento y validación para evitar cometer filtraciones de un conjunto al otro al imputar valores. Por lo tanto, se trabajan por separado.

Una de las variables que cuenta con valores faltantes es “CompetitionDistance”, la cual se decide asignarle un valor muy alto a los faltantes (outlier) dado que entendemos que esto implica que no tiene competencia cercana. Lo mismo se hizo con “MesanioCompetition”, ya que es la fecha desde la cual se tiene competencia.

En cuanto a “YearWeekPromo2”, se optó por imputar la mediana a los faltantes.

Análisis exploratorio de datos

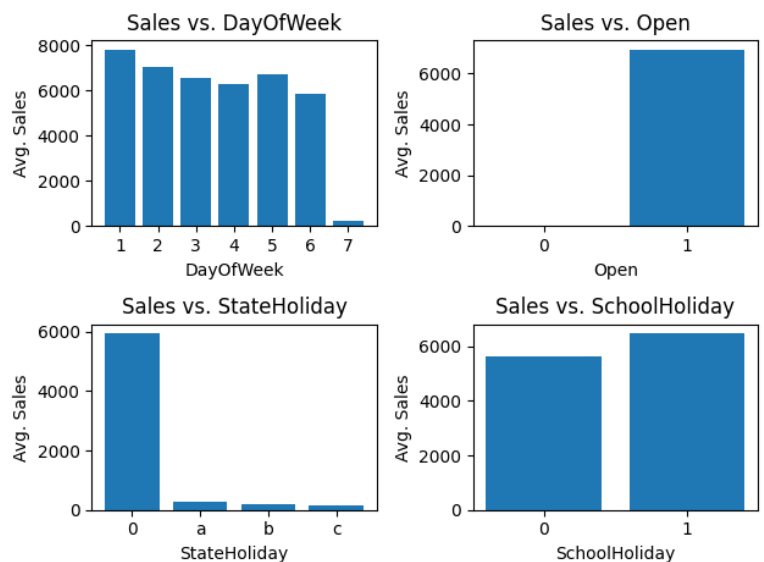
Con la finalidad de descubrir patrones de interés en los datos y en base a eso efectuar la creación de nuevas variables en el feature engineering, se estudió la variable a predecir, ventas, bajo diferentes dimensiones.

Variables que podrían disminuir las ventas

Lógicamente, la principal variable que marca una disminución en las ventas es “Open” que marca si la tienda abre o no, ya que en caso de que la tienda se encuentre cerrada las ventas de la misma son cero, lo cual se chequea que sea siempre así para comprobar que no haya alguna fila que contradiga esto.

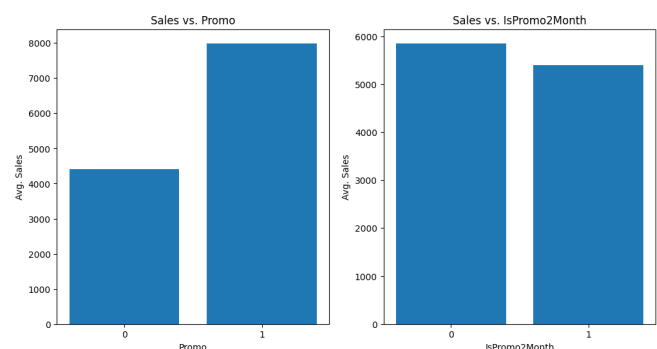
El hecho de que la tienda esté o no cerrada termina influyendo también en otras dimensiones, como la del día de semana o la de los feriados, ya que las tiendas suelen cerrar los domingos y feriados.

Una variable que sospechamos que podría acompañar una disminución de las ventas pero no fue así, fue la de los feriados escolares. Por el contrario, como se puede ver en el gráfico, el promedio de ventas es mayor cuando se produce un feriado de esta clase. Se piensa que puede existir un efecto de correlación entre las ventas de Rossman y que los chicos tengan feriado en la escuela.



Efectividad de las promos

Evaluando la efectividad de las promos, descubrimos que la promo 1 es mucho más efectiva que la promo 2 en cuanto a su efecto en ventas, lo cual tiene sentido ya que es una promo más efímera de corta duración que podría generar una mayor sensación de urgencia en los consumidores, es por esto que esta promo es la que utilizamos posteriormente en la ingeniería de atributos.



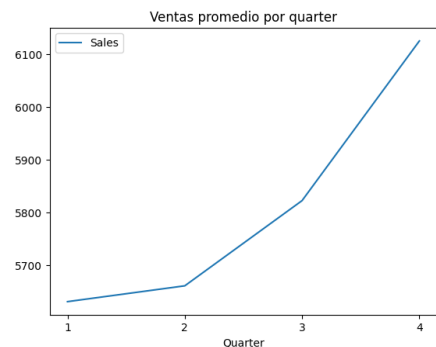
Estacionalidad

A partir del análisis exploratorio de datos descubrimos que el dataset tiene una estacionalidad marcada, siendo el mes de diciembre el mes con mayor cantidad de ventas promedio probablemente explicado por las fechas festivas. En base a esto, notamos que a lo largo de los años las ventas siguen un patrón similar.



Realizando el mismo análisis pero promediando solo por quarter, descubrimos que existen diferencias entre las ventas promedio de los quarter, sobre todo entre la primera mitad del año y la segunda.

A partir de la ingeniería de atributos, nuestro objetivo es la creación de las variables relacionadas con la fecha como para que el modelo pueda capturar todos estos efectos estacionales mencionados.



Feature engineering

Se llevó a cabo un proceso de ingeniería de atributos con la finalidad de proveer la mayor información posible a los modelos, este proceso incluyó la creación de nuevas variables, la desagregación de algunas y también la inclusión de variables externas que consideramos que podrían ser influyentes para explicar las ventas. También se pensó cómo adaptar dichas variables para que puedan ser captadas con facilidad por los modelos que más pensábamos usar: árboles de decisión y sus variantes que los ensamblan.

Variables temporales

En cuanto a las variables relacionadas con la temporalidad, a partir de la variable "Date" que viene dada en el dataset, se efectuó la creación de diferentes variables, entre ellas:

- Día del año: Representación numérica del día del año (1 a 365)
- Día de la semana: Representación numérica del día de la semana.
- Quarter: La división del año en trimestres
- IsWeekend: Una variable dummy que indica 1 si es sábado o domingo, y cero en caso contrario.

El objetivo de estas variables fue desagregar la variable fecha lo más posible para captar cualquier efecto temporal que pueda suceder en las ventas.

Variables relacionadas a la competencia

En este tipo de variables el objetivo fue resumir en 2 variables toda la información sobre la competencia. Se utilizó el año y mes de apertura de la competencia para determinar si en cada año y mes la competencia estaba presente o no, de modo que sea una variable 1 cuando tuviera competencia presente y 0 cuando no. Esta variable es "HasCloseCompetition", mientras que conservamos la distancia a dicha competencia. Se descartan las variables de apertura de competencia originales más adelante, ya que no esperamos que aporten información nueva al modelo.

Variables sobre las promociones

Se crearon dos variables relacionadas con la promo1, ya que es la que mayor efecto tiene en las ventas, según lo observado.

- PromoDaysCount: Es un contador de los días de promo1 consecutivos, con la esperanza de poder captar el efecto decreciente que tienen las ventas a medida que pasa la promo.
- RecentPromo: Es una variable dummy que marca con un 1 los dos días posteriores a la realización de una promo1, con la esperanza de poder capturar el efecto decreciente de las ventas durante estos días.

Euribor

Como parte de la búsqueda de variables relevantes fuera del dataset que tengan un poder predictivo sobre las ventas en las tiendas Rossmann, se decide incorporar el EURIBOR (Euro Interbank Offered Rate). Es una variable macroeconómica que fija una tasa de interés a la cual los bancos de la Unión Europea se prestan dinero entre ellas. Si bien se trata de una tasa interbancaria, es muy relevante para el consumidor a la hora de tomar decisiones de compra: mayores tasas de interés en la economía incentivan el ahorro, mientras que menores tasas incentivan el consumo. Por eso entendemos que esta variable podría tener poder predictivo y se decidió incorporarla al dataset. Si solo incorpora ruido, el modelo la seleccionará.

Incorporamos a nuestro dataset la tasa diaria real para el conjunto de entrenamiento. Para los fines de semana, donde no hay información de dicha tasa, se asignó el promedio de cada mes. Sin embargo, para el conjunto de validación decidimos imputar el promedio ya que para predecir valores nuevos de ventas, esta tasa es desconocida. De haber incorporado la tasa real en el conjunto de validación estaríamos cayendo en data leakage dado que es falso conocer de antemano la tasa Euribor y por lo tanto, no sería generalizable.

Métrica de interés

La métrica propuesta para utilizar como guía es el RMSPE (Root Mean Square Percentage Error):

$$RMSPE = \sqrt{\frac{1}{N} \sum \left(\frac{y - \hat{y}}{y} \right)^2}$$

Esta métrica es útil para el problema ya que tiene en cuenta los errores relativos y no absolutos, lo cual elimina el problema de las diferentes escalas de las ventas a lo largo del tiempo y hace que los errores que en términos absolutos fueron pequeños, sean evaluados desde su término relativo que puede ser grande. Además, el hecho de que eleve los residuos al cuadrado hace que la fórmula penalice de mayor forma los errores grandes porcentualmente. Sin embargo la fórmula tiene un problema para el dataset actual, ya que las ventas de una tienda pueden ser 0 y el error tendería a infinito. Por ello es que tanto en la competencia de Kaggle como para este trabajo, se ignorarán los casos en que las ventas reales sean 0. Si bien esto puede parecer una simplificación excesiva del problema original (ya que también se debería querer acertar cuando las ventas vayan a ser 0), pensándolo un poco esto no es así. La vasta mayoría de las veces que una tienda no tiene ventas es cuando está cerrada, por lo que la sola proyección de cuándo abrirán las tiendas es

suficiente para predecir estos días de 0 ventas casi a la perfección. Para el resto de los casos, está el modelo propuesto.

Selección de conjuntos de validación y entrenamiento

Respecto a la decisión fundamental para cualquier problema de machine learning que representa seleccionar el conjunto de validación que va a ser una guía para la toma de decisiones nos enfrentamos a la primera decisión de que ya que el dataset es una serie temporal, no se iba a recurrir a técnicas como cross validation, sino a técnicas que tuvieran en cuenta dicha temporalidad. Para esto probamos dos alternativas:

- Tomar las últimas 6 semanas previas al testeo (misma ventana de tiempo que el test set)
- Tomar agosto y septiembre (mismos meses que el test set) del año 2014, por el efecto estacional que tiene el dataset

Encontramos que el test de validación más útil era la primer opción de tomar las últimas semanas, ya que la validación daba consistentemente resultados más parecidos a los que luego arrojaba Kaggle para el testeo, es por esto que utilizamos las últimas semanas del dataset como validación y todo el resto como entrenamiento.

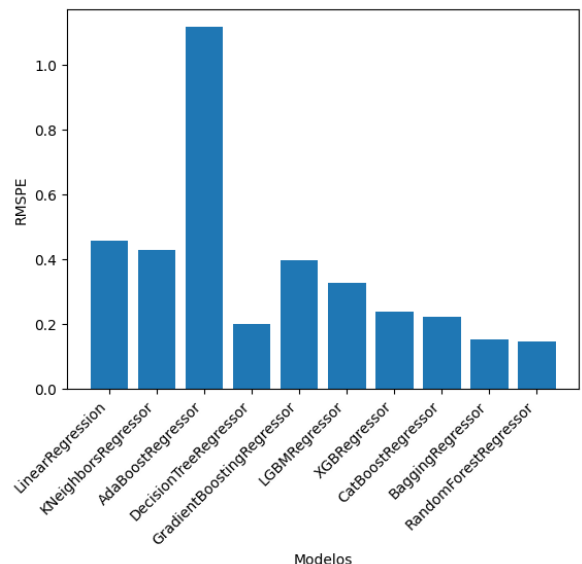
Esta decisión tiene fundamentos en que los datos más recientes son los que más reflejan la realidad de lo que vamos a predecir en el testeo. Si nuestro set de testeo fuese en noviembre y diciembre, que son meses mucho más marcados por la estacionalidad, seguramente hubiese sido mejor tomar los mismos meses del año pasado para validar.

Prueba de modelos

El primer paso que se dió con el fin de probar modelos con los cuales enfrentar el problema de predicción, fue la prueba de distintos modelos out-of-the-box para así poder chequear cuales son aquellos que minimizan la métrica de interés, el RMSPE.

En línea con los resultados obtenidos, decidimos implementar una optimización de hiperparametros para dos de los modelos con un menor RMSPE, XGBoost y Random Forest. Ambos algoritmos mencionados son basados en árboles, solo que Random Forest es un algoritmo de bagging, mientras que XGBoost es un algoritmo de boosting.

Bootstrap Aggregating (bagging) es una clase de algoritmos que trabaja mediante la creación de múltiples modelos en paralelo con bootstrap sobre la muestra, para luego promediar las predicciones de todos los modelos y encontrar una predicción general precisa. Por otro lado, XGBoost tiene fundamentos en el boosting, que se basa en la creación de modelos de una manera secuencial donde los modelos van perfeccionando su predicción entre sí, corrigiendo los errores de los modelos pasados.



Optimización de hiperparámetros

Para llevar a cabo la optimización de hiperparámetros se decide utilizar la técnica de random search ya que entendemos que se explora mejor el espacio sampleando más puntos y es más robusto a la búsqueda de hiperparámetros irrelevantes.

Para realizar esto, se generan funciones adicionales. Para la función asociada a evaluar los distintos puntos se establece un límite de tiempo, o bien terminar con todo el sampleo, lo que ocurra primero de los dos. Esto se realiza de esta manera dado el costo computacional que esto conlleva y para aprovechar los tiempos muertos como las noches, en que nadie necesita la capacidad de cómputos.

En primer lugar, se realiza una búsqueda de hiperparámetros para el modelo que se obtuvo aplicando Random Forest y luego, para el modelo final elegido: XG Boost.

Random Forest

Se establecieron rangos dentro de los cuales simular cada uno de los hiperparámetros. A partir de estos rangos, el script genera la cantidad que uno quiera de sets de hiperparámetros obtenidos de forma aleatoria. Con cada set se probaba un modelo hasta abarcar todos los sets simulados o, como se estableció antes, hasta que terminara el tiempo asignado a esta tarea. Se fueron acotando los valores posibles según los resultados obtenidos hasta llegar a la siguiente tabla de boundaries, para la cual se simularon finalmente 20 modelos de Random Forest.

Hiperparámetro	Breve descripción	Lower bound	Upper bound
n_estimators	Cantidad de árboles	500	800
criterion	Método para generar particiones	squared_error, friedman_mse o poisson	
max_depth	Profundidad máxima de cada árbol	50	150
max_features	% de features a muestrear	0.55	0.8
max_samples	% de las rows a muestrear	0.55	0.8
min_samples_split	Mínimo número de samples en un nodo para generar una nueva partición	60	95

Debido a que en Random Forest el riesgo de overfitting se tiene solo si cada árbol overfittea individualmente (ya que son paralelos), usamos árboles de una profundidad mayor, intentando usar menos samples y menos features, para que los árboles fueran distintos entre sí. Con este proceso se llegó al siguiente ranking de modelos. Como se puede apreciar, llegan a 18,5% de error en validación, lo cual no parecía demasiado prometedor.

...	n_estimators	criterion	max_depth	min_samples_split	max_features	bootstrap	max_samples	random_state	n_jobs	rmspe_train	rmspe_val
18	745	squared_error	115	78	0.701972	True	0.797706	42	-1	0.209108	0.185386
4	747	squared_error	105	65	0.732595	True	0.672776	42	-1	0.206354	0.185660
5	545	squared_error	147	62	0.771711	True	0.574924	42	-1	0.208401	0.185952
9	715	squared_error	109	74	0.761552	True	0.663190	42	-1	0.210186	0.186753
0	571	squared_error	140	73	0.754014	True	0.558661	42	-1	0.212418	0.187691
1	554	squared_error	102	70	0.671780	True	0.584370	42	-1	0.212038	0.187911
8	517	squared_error	96	65	0.567810	True	0.795750	42	-1	0.201576	0.187982
6	585	squared_error	108	94	0.790496	True	0.649655	42	-1	0.217612	0.188793
15	577	squared_error	143	72	0.609088	True	0.778807	42	-1	0.208665	0.189103

Xgboost

De manera similar a como se encaró el problema con Random Forest, se establecieron diferentes rangos para los principales hiperparámetros de Xgboost y se realizó un random search para la definición de dichos hiperparametros, obteniendo así 57 modelos diferentes.

Hiperparámetro	Breve descripción	Lower bound	Upper bound
n_estimators	Cantidad de árboles	500	1400
learning_rate	ETA, tasa de aprendizaje	0.01	0.3
max_depth	Profundidad máxima de cada árbol	4	10
colsample_bytree	% de features a muestrear	0.8	0.9
subsample	% de las rows a muestrear	0.7	0.9
gamma	Reducción mínima del error para la partición del árbol	0.1	0.4

Como se puede ver el rango de hiperparámetros es bastante amplio, pero buscamos llegar al límite entre que el modelo ajuste bien pero no llegue a overfittear, hecho que puede suceder al usar un algoritmo xgboost. Nuestra idea fue hacer modelos de mucha cantidad de árboles poco profundos, para encarar el problema con muchos “tocones” por modelo. Iterando sobre este rango de hiperparámetros, conseguimos una buena cantidad de modelos útiles para predecir, que se comportaban mejor en validación que los observados en Random Forest.

Ensamble de ensambles

Pensamos en combinar las nociones de Boosting y Bagging, haciendo Bagging de varios modelos hechos con Boosting para mejorar las predicciones. Para esto se tomó el ensamble paralelo (a similitud de Random Forest) de varios modelos XGBoost distintos. Este camino se vio reforzado por el hecho de que observamos que los mejores modelos XGBoost que obtuvimos tenían distintas features como las más relevantes. Esto nos llevó a pensar que cada modelo en realidad estaba captando cosas diferentes a su manera, y que podría ser buena idea ensamblarlos.

Bajo este criterio, como último paso para la predicción, tomamos los mejores modelos de los 57 que entrenamos, quedándonos así con 10 modelos, que fueron aquellos que tuvieron menos de 0.162 de rmspe en validación.

	n_estimators	learning_rate	max_depth	colsample_bytree	subsample	gamma	n_jobs	random_state	rmspe_train	rmspe_val
52	1387	0.262299	6	0.855580	0.840756	0.217642	-1	42	0.177601	0.143819
21	1268	0.203399	10	0.825222	0.730904	0.296638	-1	42	0.084127	0.152590
2	1209	0.060663	10	0.886189	0.884134	0.186740	-1	42	0.146061	0.155407
4	955	0.180648	8	0.855588	0.860360	0.387464	-1	42	0.156079	0.156121
44	791	0.190010	10	0.858730	0.884524	0.341369	-1	42	0.120871	0.157959
35	1068	0.278482	7	0.804264	0.812517	0.205062	-1	42	0.169025	0.158663
39	628	0.114325	10	0.846826	0.759753	0.273269	-1	42	0.153072	0.160478
13	1044	0.211845	10	0.852330	0.881415	0.123130	-1	42	0.103402	0.160964
8	1046	0.253455	8	0.882563	0.817608	0.397064	-1	42	0.146487	0.161033
43	554	0.199681	7	0.897627	0.710786	0.101154	-1	42	0.188851	0.161823

Estos modelos fueron re-entrenados con todo el dataset, incluyendo las semanas de validación, y predijeron cada observación del set de testeo. Una vez que tuvimos las 10 predicciones distintas para cada observación, las promediamos, bajo el fundamento

conocido como Majority Voting, que tiene por detrás los modelos de ensamble como los que venimos usando.

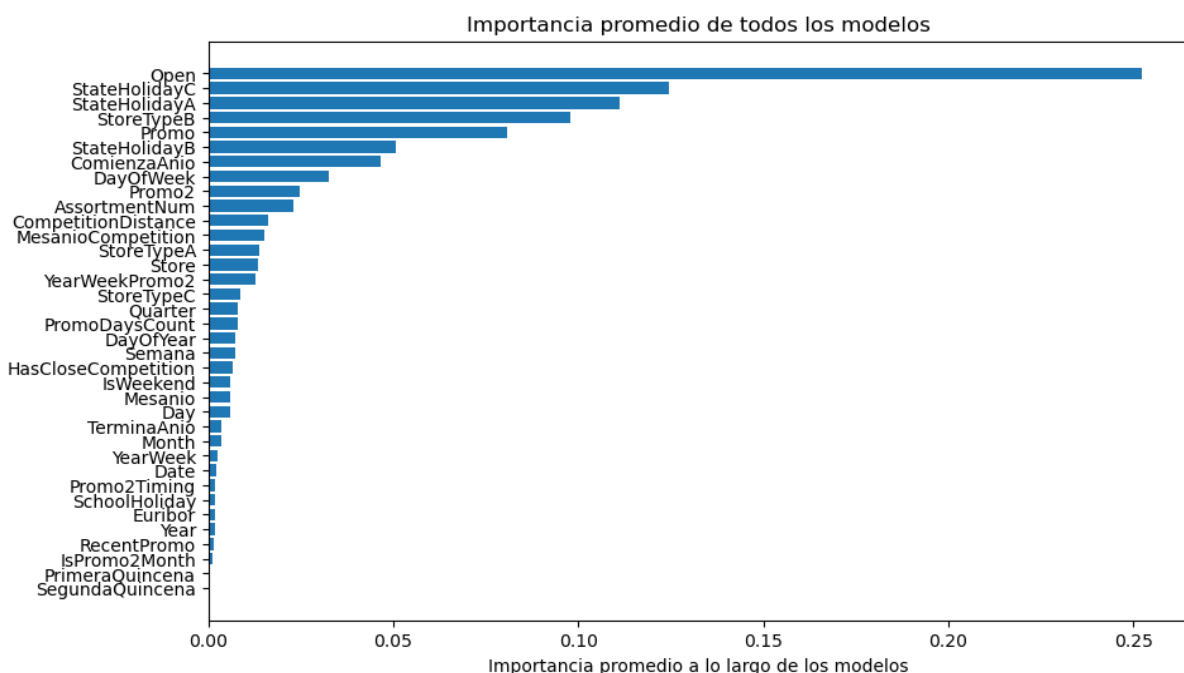
Resultados

Validación

Los resultados de validación variaron entre 0.18 y 0.19 de RMSPE en random forest y fueron un poco mejor en xgboost consiguiendo entre 0.14 y 0.16 en la mayoría de los modelos. Estos últimos resultados de validación de xgboost son similares a el puntaje obtenido en kaggle para los modelos individuales.

Feature importance e Interpretación

Dado que utilizamos muchos modelos para predecir y conseguir la media de las predicciones, calculamos la feature importance promedio de estos 10 modelos utilizados y los resultados fueron los siguientes:




Tiene sentido que la variable que más feature importance tenga es Open, ya que marca una clara diferencia entre las muestras que no tuvieron ventas por estar cerrados y las que tuvieron al menos una. El tipo de feriado también parecería ser un factor influyente para los modelos, así como algunas variables creadas en el feature engineering que surgen de la fecha como "Day Of Week" o "Comienzo año" que permiten al modelo captar la estacionalidad que hay en las ventas en diferentes fechas. Por último, las variables que marcan una promoción como "Promo" o "Promo2" también tienen bastante importancia en el modelo, lo cual nos da a entender que las promos están siendo efectivas ya que tienen algún impacto sobre las ventas.

Dentro de la interpretación de estas features, se podría tomar la decisión de eliminar aquellas variables que aportan muy poco al modelo final. Esto tendría sentido desde el punto de vista de que al final, quitando esas variables se perdería muy poco poder predictivo y quedaría un modelo más simple. En definitiva, sería como aplicar el principio de la navaja de ockham a nuestro modelo, pensando que entre dos modelos que predicen "igual", el mejor es el más sencillo.

Kaggle

Al subir este promedio de predicciones de muchos modelos a Kaggle, obtuvimos un puntaje privado de 0.14 en RMSPE y un puntaje público de 0.12, puntaje que no está mal cuando se lo compara con el puntaje del ganador de la competencia, que es de 0.10, es decir, 2pp más cerca de los valores reales.

Submission and Description	Private Score ⓘ	Public Score ⓘ	Selected
 kaggle.csv Complete (after deadline) · now	0.13973	0.11957	<input type="checkbox"/>

Problema de ML adicional

Se genera un problema adicional distinto al inicial, en donde se buscará predecir si la cantidad de clientes que ingresan a una determinada store en un día determinado supera un valor, el cual se estableció en 750 clientes. Se tomó este valor dado que es aproximadamente el promedio de clientes que ingresan a una store un día determinado.

El móvil de generar un problema de estas características podría ser de interés para Rossman ya que podría predecir cada día, cuáles serán las tiendas más importantes para su negocio global y ver cómo varían estas para los fines de semana o los feriados escolares. Responder un problema así podría ayudar a saber qué tiendas necesitan reforzar en supply de stock o incluso en contrataciones temporales para cubrir picos inesperados y atender a los clientes de la mejor forma posible.

Por lo tanto, este nuevo problema se trata de un problema de clasificación donde la variable a predecir es categórica y toma solamente dos valores: 1 en caso que supere dicho valor, o bien 0 en caso contrario.

En este caso, la métrica que se utiliza para evaluar los modelos “out of the box” es el ROC-AUC (Área bajo la curva ROC). Por lo que se intentará maximizar dicho valor.

Se decide utilizar esta métrica ya que es independiente de un determinado threshold y además no incide en el resultado qué tan balanceadas están las clases, como sería el caso de la métrica Accuracy que depende de que las clases estén balanceadas para ser una buena medida.

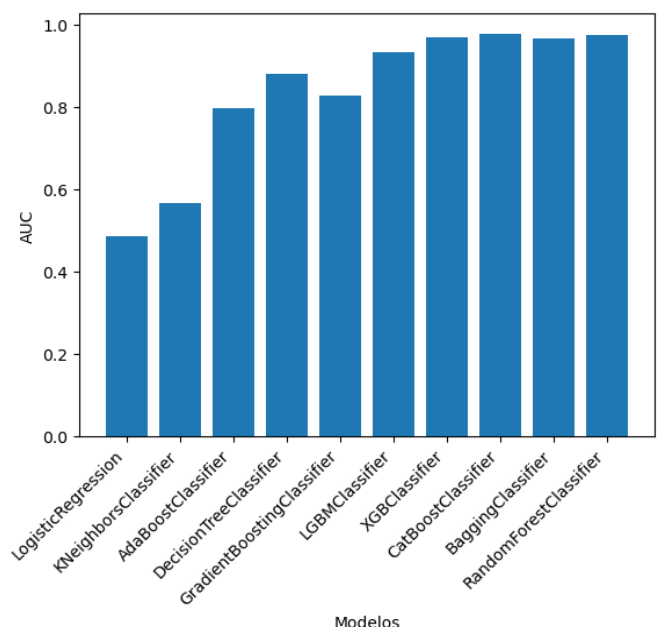
Estos modelos decimos que son “out of the box” ya que no se realiza ningún análisis exploratorio de datos o ingeniería de atributos acorde a este problema.

Se toma el mismo modelo que se definió para el problema principal. Es decir, tiene el mismo preprocesamiento e ingeniería de atributos que se utilizó en el problema de regresión.

Se entrenan los modelos mencionados debajo usando el conjunto de entrenamiento y se evalúan usando el conjunto de validación definido. Se definió como conjunto de validación el mismo utilizado para evaluar los modelos en el problema principal.

Se evalúan los siguientes modelos “out of the box”, algunos simples y otros ensamblados como Boosting, Bagging y Random Forest, y así poder comparar los valores de AUC:

- Regresión Logística
- KNN
- AdaBoost



- Decision Tree
- Gradient Boost
- Light GBM
- XG Boost
- Cat Boost
- Bagging
- Random Forest

Se puede concluir que el mejor modelo “out of the box” se obtuvo entrenando los algoritmos Cat Boost y RandomForest dado que se obtuvo en el conjunto de validación un valor de AUC de 0.97.

Dado que no se ha efectuado ningún análisis exhaustivo para resolver este problema específico y se ha utilizado el mismo modelo del problema principal, concluimos que se obtuvo un valor muy razonable de AUC.