UNIVERSIDAD DE BUENOS AIRES

FACULTAD DE INGENIERÍA

75.06 Organización de Datos

Trabajo Práctico 2

Segundo Cuatrimestre de 2020

Grupo 30: Tales from the Data

Integrantes	Padrón
Bobadilla Catalan, German	90123
Briglia, Antonella	90903
Cot, Sofía	103071

Link a Github:
https://github.com/germanbc/Organizacion-de-Datos-2c2020-TP2

Link a la presentación: https://www.youtube.com/watch?v=-sxTeAaTgJ8

Índice

1. Introducción

El presente trabajo práctico consiste en resolver un problema de machine learning donde debemos predecir la probabilidad de éxito de una oportunidad de negocios con el set de datos que trabajamos en el trabajo práctico anterior.

El trabajo práctico se desarrolló en el contexto de una competencia de Kaggle, donde nos brindaron un set de entrenamiento y otro para realizar las pruebas con los algoritmos que elegimos para resolver el problema. Dados estos sets de datos, utilizamos el set de entrenamiento para realizar un procesamiento y generar un modelo el cual luego entrenamos utilizando diferentes algoritmos y realizamos la predicción. Esta predicción fue subida a Kaggle en diferentes etapas, ajustando el modelo y los algoritmos, donde intentamos generar una nueva predicción que mejore el puntaje obtenido con anterioridad.

2. Preprocesamiento de datos

Como primera medida decidimos quedarnos solamente con las oportunidades ganadas y perdidas, dado que las oportunidades con estado “Qualification”, “Proposal” y “Negociation” son demasiado pocas y no conocemos la probabilidad que tienen de terminar en ganadas o perdidas.

Luego eliminamos las columnas con todos sus valores nulos, como Last_Activity y Actual_Delivery_Date. También eliminamos las columnas que tienen todos sus valores en cero o “none”, como Submitted_for_Approval y Prod_Category_A. Finalmente eliminamos las columnas de Opportunity_Name (por ser redundante con Opportunity_ID) y Sales_Contract_No. Esta última la eliminamos para evitar el problema de target leakage, dado que mejora muchísimo el resultado pero es información que no vamos a poseer al momento de hacer una predicción real.

Finalmente rellenamos los datos faltantes de Total_Taxable_Amount con los datos de Total_Amount. Y convertimos esos montos a una única moneda (a dólares) con la ayuda de la información del tipo de cambio que se encuentra presente en las columnas de ASP, ASP_(converted) y ASP_Currency. Lo mismo hicimos con la columna Price.

3. Feature engineering

Dado que en el set de entrenamiento cada fila no corresponde necesariamente a una oportunidad, sino a un producto de una oportunidad, desde nuestro punto de vista existen dos tipos de columnas en el set de entrenamiento: las que tienen características de una oportunidad y las que tienen características de un producto. Un ejemplo del primer caso es el territorio, el cual es único para todos los productos dentro de una oportunidad. Y un ejemplo del segundo caso son las TRF, las cuales pueden tener diferentes valores para diferentes productos dentro de una misma oportunidad.

Esto hace que estos dos tipos de columnas tengan un trato diferente a la hora de entrenar. Por ejemplo, en el primer caso, si elimino los duplicados por Opportunity_ID no estoy perdiendo información en esa columna, simplemente me estoy quedando todas las oportunidades, sin repetir, y cual es su territorio. Sin embargo al hacer lo mismo en el segundo caso me estoy quedando solo con una TRF de un producto, cuando la oportunidad tenía muchas TRF distintas. Esto nos obliga a que el feature tenga más elaboración. Es por ello que lo primero que hicimos fue distinguir unas columnas de otras.

Figura 1

Estas son las columnas que hacen referencia a características de los productos: ID, Product_Family, Product_Name, ASP, ASP_(converted), Planned_Delivery_Start_Date, Planned_Delivery_End_Date, Month, Delivery_Quarter, TRF, Total_Amount, Delivery_Year

Todas las otras columnas del set de entrenamiento que no son las mencionadas arriba, tienen el mismo valor en todos los productos de una oportunidad. Sin olvidar este detalle es que utilizamos diferentes columnas de diferentes maneras.

En cuanto a la búsqueda de los mejores features para obtener el mejor puntaje en Kaggle nos ayudamos del conocido algoritmo Random Forest (con Grid Search Cross Validation) por ser rápido y dar buenos resultados para la mayoría de los sets de datos. Fuimos probando de uno en uno todos los features que se nos ocurrían, y quedaban los que mejoraron el resultado de GridSearchCV y Kaggle.

A continuación vamos a mencionar, en orden, solamente los features que fuimos encontrando y agregando, los cuales hicieron mejorar nuestro puntaje.

3.1. Features

1º Territorio

Nuestro primer paso fue hacer una predicción con un solo feature. Decidimos iniciar con uno poco original pero que parecía que no podía faltar: el territorio. Simplemente tomamos la columna correspondiente en el set de entrenamiento y la codificamos con Target Mean Encoder. El resultado obtenido en Kaggle con ese único feature es de 0.68564, lo que nos parecía bastante bueno dada la poca elaboración hasta el momento.

2º Región

El segundo feature probado fue el de Región, haciendo lo mismo que hicimos con Territorio. Si veíamos que su score del Grid Search Cross Validation era superior, crearíamos la predicción para subir a Kaggle. En caso contrario sería descartado (al menos por el momento). En este caso pensamos que este particular feature podría ser redundante con el anterior, ya que parecen obtener información parecida. No pensamos que podría empeorar el modelo, pero tampoco mejorarlo mucho. En efecto el score en Kaggle presentó una leve mejora: 0.67906. Adelantandome al resto de los resultados, (casi) todas las predicciones que presentaban una mejora en su puntaje con GridSearchCV, también presentaron una mejora en el puntaje en Kaggle. Todas excepto una que mencionaremos más adelante.

3º Aprobación especial

La metodología de incorporación de features descripta en el paso anterior la seguimos aplicando con todos los demás. El tercer feature que dio resultados positivos fue el de Aprobación especial. Con las columnas "Pricing, Delivery_Terms_Quote_Appr" y "Pricing, Delivery_Terms_Approved" creamos el feature Aprobacion_Especial, el cual tiene los datos categóricos de "NecesitaObtuvo", "NecesitaNoObtuvo" y "NoNecesitaNoObtuvo", haciendo referencia a su necesidad y obtención de una aprobación especial de su precio total y los término de la entrega. La mejora de el puntaje en Kaggle que nos generó fue un poco superior que la del segundo paso: 0.65521.

4º Brand

Asi como Territorio y Región, esta columna no requirió de ningún tratamiento en especial ya que corresponden a características de la oportunidad, y no de productos particulares. Y como todas las columnas categóricas de este TP, la codificación que usamos por darnos mejor resultado fue la de Target Mean Encoder. Para nosotros la gran mejora en el score que nos dio (0.56756) fue una sorpresa dado que teníamos dudas de sus beneficios ya que tenía varios registros con valores nulos.

5º Opportunity_Type y Account_Owner

Estos fueron los siguientes dos features que mejoraron la predicción. Ambos obtenidos directamente del set de entrenamiento y codificados. La mejora obtenida en estos casos fue mucho menor que la del anterior, paso. Particularmente la del segundo fue muy fina. Los resultados fueron 0.55441 y 0.55237 respectivamente.

6º TRF promedio

Como su nombre lo indica, es el TRF promedio de la oportunidad. Antes se había probado, sin ver mejora, la suma total de todos los TRF de cada oportunidad (así como el Total Taxable Amount es la suma de todos los Total Amount). Sin embargo, con el promedio la mejora en el score de Kaggle fue muy grande (0.48445), pero a diferencia de lo que había ocurrido con Brand, en este caso era esperable. Esto lo podíamos prever por lo visto en el TP1 donde nos dimos cuenta que las cantidades perdidas tenían un número más elevado de TRF promedio.

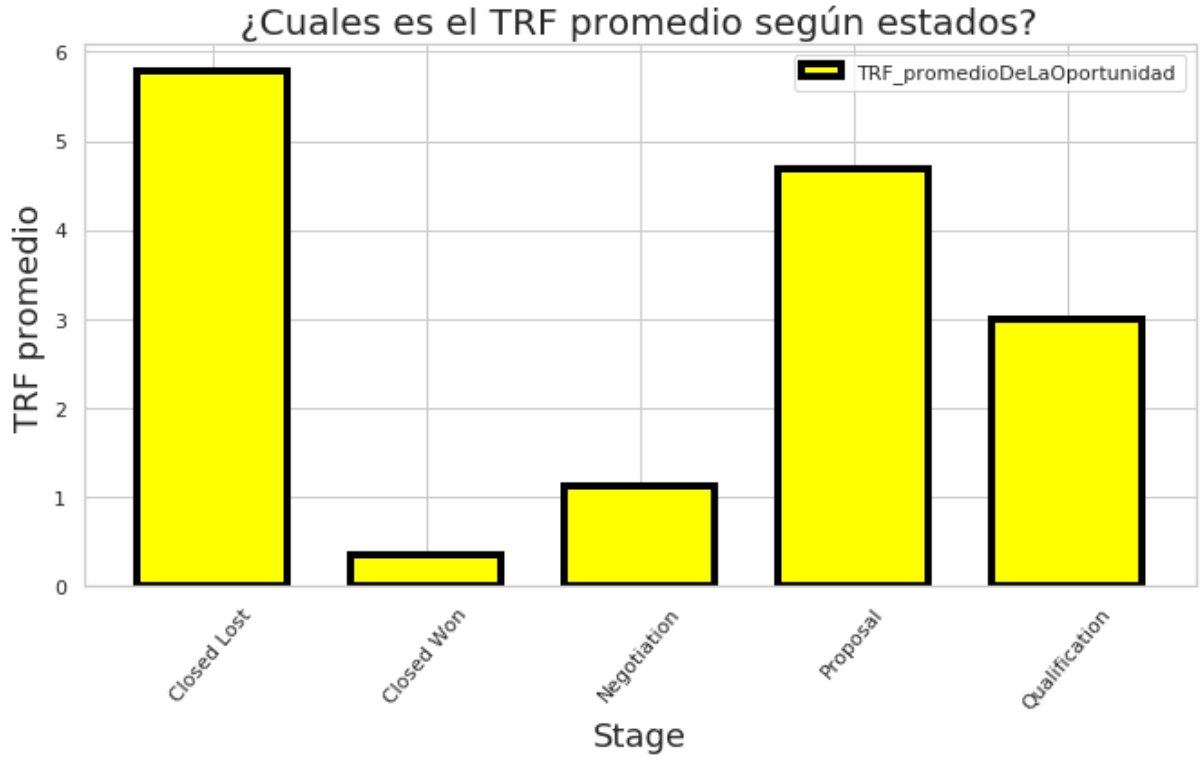

Figura 2

7º Total Taxable Amount

El agregado de esta columna solamente con la conversión de sus valores a una misma moneda dio una mejora en la predicción con un score de 0.47307. Sin embargo, sabíamos que en el set de entrenamiento tenía varios valores vacíos que podían ser rellenados con la ayuda de los datos de Total Amount, esperando aunque sea una leve mejora. Y así fue; con los datos rellenados se consiguió un score de 0.47164. La mejora no fue grande, pero llegado a este punto del TP y la competencia, habiendo probado muchos otros features que no funcionaron (serán

mencionados en la sección 3.2) y habiendo puntajes parecidos entre los grupos de Kaggle, esas pequeñas mejoras pueden hacerte avanzar algunos puestos.

8º Rango de tiempo

Aprovechar las fechas para la predicción fue un desafío un poco más grande de lo esperado. Usar simplemente los años o meses no daba buenos resultados. Mas aun, en el caso de Planned_Delivery_Start_Date y Planned_Delivery_End_Date, no veiamos que tuviera sentido usarlas de esa forma simple ya que estas columnas tienen diferentes valores para los diferentes productos de una oportunidad, y quedarnos solo con uno para la oportunidad parecía arbitrario. Hasta que se nos ocurrió como sacarles provecho calculando el tiempo de entrega en días. Se logra haciendo la diferencia entre la minima y maxima fecha de entrega prevista (el Planned_Delivery_Start_Date mínimo de todos los productos de una oportunidad y el Planned_Delivery_End_Date máximo). El score obtenido fue de 0.46730.

9º Cantidad de productos

Finalmente, la cantidad de productos por oportunidad fue el último feature que presentó mejoras con la ayuda del Random Forest. A pesar de su simpleza, y quizás obvia necesidad de entrar en el modelo, se nos había pasado por alto. El resultado final con Random Forest quedó en: 0.46713.

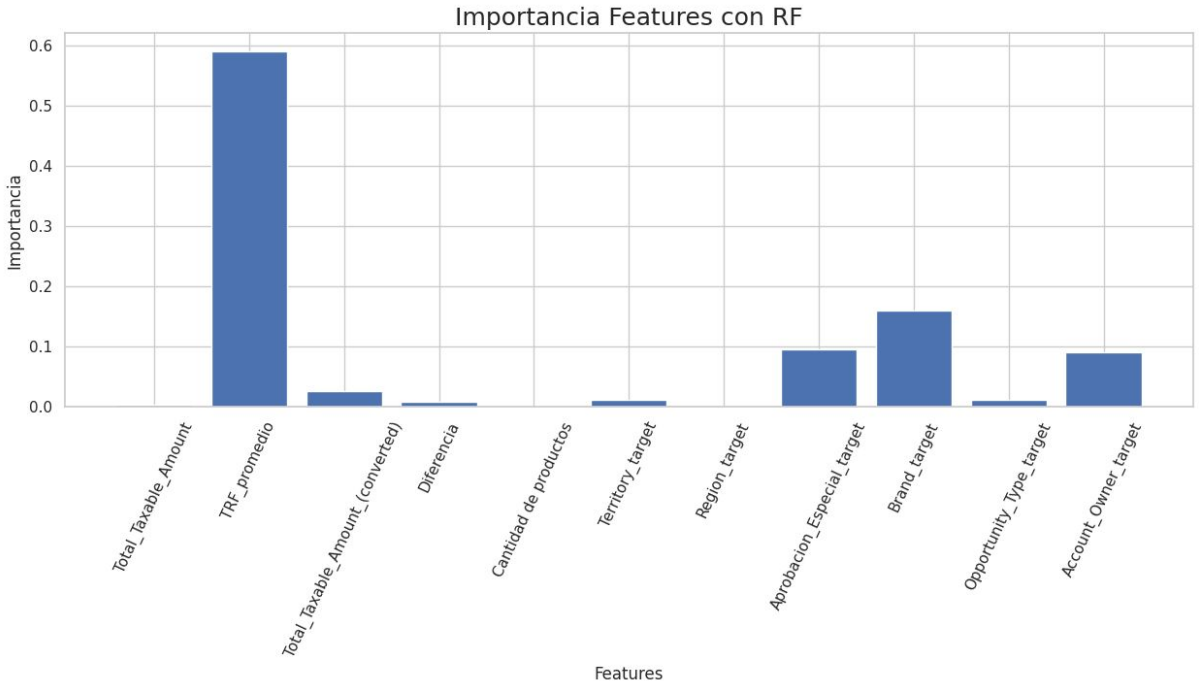

Figura 3

10º CatBoost

Luego de haber progresado todo lo que podíamos con Random Forest, pasamos a probar otros algoritmos (están explicados en la sección 4). Muchos de ellos presentaron puntajes más bajos. Otro como XGBoost, levemente mejor. Pero hubo uno que mejoró considerablemente el score y terminó convirtiéndose en el algoritmo que más puntaje nos dió en la competencia: CatBoost. Usando exactamente los mismo features que Random Forest, el puntaje en Kaggle que nos dio fue de 0.44248.

11ºAccount Name

Luego, de descubrir que CatBoost era el que mejor nos funcionaba, procedimos a probar los features que previamente habíamos descartado. En casi todos los casos los features que no presentaron mejora con Random Forest tampoco lo hicieron con otros algoritmos. Todos excepto uno: Account_Name. Con Random Forest la utilización de esta columna generaba overfitting. Pero cuando lo probamos con CatBoost, si bien el GridSearchCV seguía generando un puntaje mucho mejor que el de Kaggle, la agregación de este nuevo feature generó un resultado bastante mejor en la competencia, dejándonos con el mejor score que conseguimos: 0.41373.

Resumen de los resultados con Random Forest

Feature agregado	Puntaje en Kaggle
Territory	0.68564
+Region	0.67906
+Aprobacion Especial	0.65521
+Brand	0.56756
+Opportunity Type	0.55441
+Account_Owner	0.55237
+TRF promedio	0.48445
+Total Taxable Amount	0.47307
+TTA rellenados*	0.47164
+Rango de entrega	0.46730
+Cantidad de productos	0.46713

** TTA rellenados: Los campos vacíos de Total Taxable Amount fueron rellenados con la suma correspondiente hecha con la columna Total Amount.*

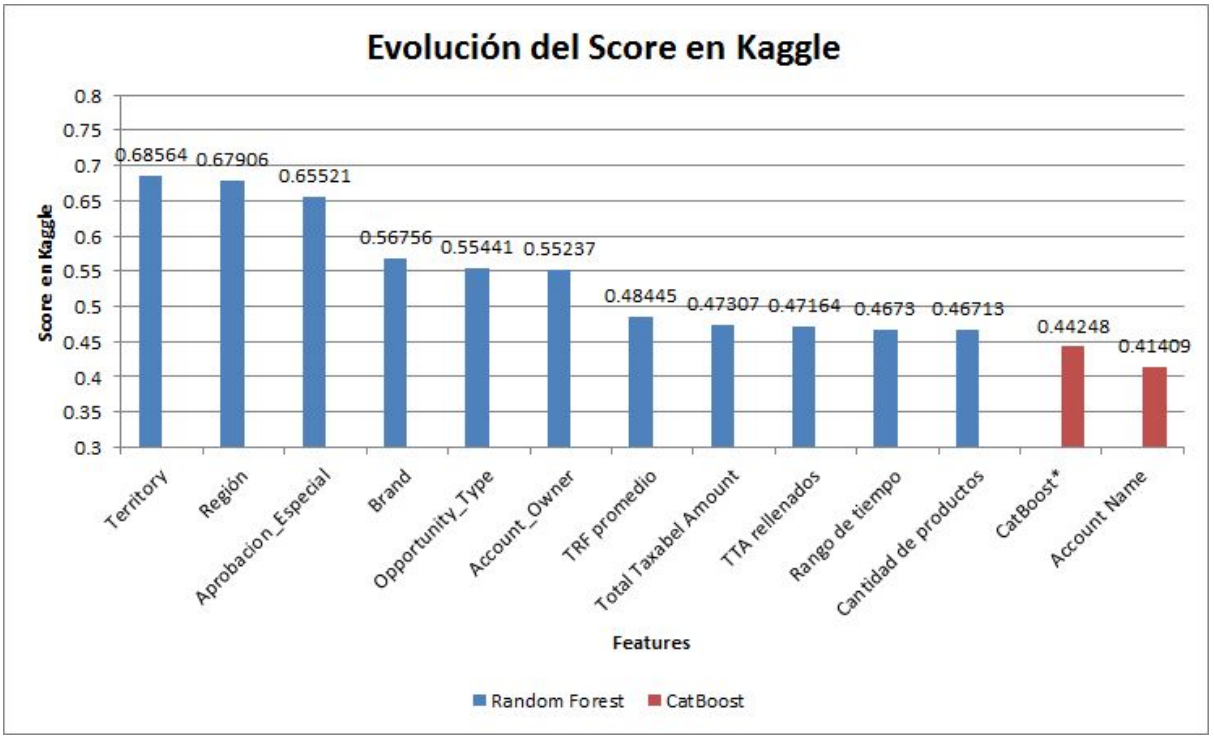

Figura 4

** CatBoost no es un feature. Manteniendo los mismos features hasta el paso anterior, cambiamos el algoritmo Random Forest por CatBoost.*

3.2. Features que no mejoraron el score

- **Código burocrático (si necesita u obtuvo)**

Del mismo modo que se creó el feature Aprobacion_Especial, creamos el feature Codigo_Burocratico a partir de las columnas Bureaucratic_Code_0_Approval y Bureaucratic_Code_0_Approved. A diferencia de Bureaucratic_Code que posee el código burocrático, este nuevo feature indica la necesidad y obtención del código con las categorías "NecesitaObtuvo", "NecesitaNoObtuvo" y "NoNecesitaNoObtuvo".

- **Total Amount promedio y Precio promedio**

Así como el TRF_promedio fue fundamental para lograr una buena predicción, pensamos que hacer lo mismo con Total Amount podría dar buen resultado. Previamente tuvimos el cuidado de pasar todos los montos a una misma moneda. También se calculó el precio promedio pero con la columna ASP_(converted).

- **Price**

Price es una columna que tiene demasiados valores nulos, por lo que era esperable que no funcione. Aun así, lo probamos habiendo nuevamente convertido todos los valores a una misma moneda.

- **Tiempo de oportunidad**

Indica el tiempo entre la creación de la oportunidad y la última modificación. El proceso de creación fue muy similar al de "Rango de tiempo". Se hizo con la diferencia en días entre Opportunity_Created_Date y Last_Modified_Date.

- **Binding**

Este feature es el tipo de presupuesto (valor booleano) el cual se creó con la columna Quote_Type.

- **Hemisferio**

Una de las apuestas fuertes era este feature que indicaba si la oportunidad se encontraba en el hemisferio sur o no. Dado que la empresa es de venta e instalación de equipos de aire acondicionado, pensamos que la ubicación por arriba o debajo del ecuador podría tener una relación con la probabilidad de éxito o fracaso de la oportunidad, debido a las diferentes estaciones. Sin embargo no mostró mejoras en el resultado, posiblemente a que muy pocos países del set de entrenamiento corresponden al hemisferio sur.

- **Fechas**

Con las columnas de fechas (Account_Created_Date, Opportunity_Created_Date, Quote_Expiry_Date, y Last_Modified_Date) creamos dos features por cada una, uno con el año y otro con el mes. Sin embargo ninguna mejoró el resultado. Creemos que se debe a que los datos a predecir corresponden a fechas recientes (2019, 2020) mientras que el set de entrenamiento corresponde a oportunidades anteriores, por lo que no hay coincidencia en ellas ni ningún patrón por el mes del año.

- **Familia y nombre del producto**

Dado que esta columna hace referencia a una característica del producto, y las oportunidades pueden tener muchos productos, buscamos una alternativa a elegir una sola familia o nombre representativo de la oportunidad. Se nos ocurrió hacer algo parecido al método de One Hot Encoder, creando nuevas columnas por cada familia/nombre existente, indicando con un 0 o 1 donde corresponda, para luego sumar. Esto nos deja por cada oportunidad, la cantidad total de productos con determinada familia/nombre. Pero el resultado no fue el esperado. Tal vez debido a que se crean demasiados nuevos features.

- **Otros**

En cuanto a las columnas del set de entrenamiento que también probamos sin mayor elaboración que la de codificar los datos categóricos con Target Mean Encoder, fueron: Source, Delivery_Terms, Account_Type, Opportunity_Owner, Bureaucratic_Code, Product_Type, Size, Product_Category_B, Last_Modified_By,

Billing_Country y Account_Name. Algunos de ellas entendiblemente no mejoraron el resultado debido a su gran cantidad de valores nulos, como Source, Product_Type, Size, Product_Category_B. Un caso particular y llamativo fue el de la columna Account_Name que generaba overfitting con Random Forest.

4. Algoritmos utilizados

Con el set de entrenamiento con los features que nos dieron el mejor puntaje, procedimos a probar diferentes algoritmos. Algunos de los algoritmos que probamos fueron Random Forest, XGBoost, LigthGBM, CatBoost, LSTM, Partial Least Squares Regression, Bayesian Ridge. Para todos ellos utilizamos A continuación explicaremos los algoritmos con los que tuvimos mejor resultado con nuestro set.

4.1. Random Forest

Para encontrar los hiper-parámetros utilizamos Grid Search Cross Validation. Aunque tuvimos problemas en un principio. Utilizar demasiados hiperparametros con ciertos valores arrojaba muy mal resultado.

Probamos varios como por ejemplo: n_estimators, max_depth, min_samples_split, min_samples_leaf, max_features. Pero finalmente los que arrojaron mejor resultado fueron n_estimators (que es la cantidad de árboles a construir) y min_samples_split (que es la cantidad mínima de datos requeridos para splitear un nodo interno).

El mejor resultado lo conseguimos con n_estimators=80 y min_samples_split=1000

4.2. XGBoost

Utilizamos el XGBoostCXGBClassifier de la librería xgboost.sklearn, y utilizamos la funcion *predict_proba* para obtener la probabilidad de la clase.

En primer lugar, probamos el algoritmo con valores de parámetros que suelen ser más comunes para tener una primera aproximación:

- learning_rate =0.1
- n_estimators=140
- max_depth=5
- min_child_weight=3
- gamma=0
- subsample=0.8
- colsample_bytree=0.8

Con estos valores logramos un puntaje de **0.47043** en Kaggle. Luego pasamos a buscar los hiper parámetros que funcionan mejor. Comenzamos por fijar un valor de learning rate , en este caso elegimos 0,1 e intentamos determinar la cantidad de árboles óptima para este learning rate. XGBoost cuenta con una función cv que realiza cross validation en cada iteración con la cual podemos obtener el número óptimo de árboles a usar para el learning rate elegido.

Una vez que obtuvimos la cantidad de árboles, pasamos a buscar los mejores parámetros específicos de los árboles (max_depth, min_child_weight, subsample, gamma, colsample_bytree) para lo cual utilizamos grid search.

Y por último pasamos a buscar los mejores valores de los parámetros de regularización alpha y lambda, también utilizando grid search, aunque obtuvimos que los valores por defecto (0 para ambos parámetros) serían los óptimos. Luego de nuestra búsqueda obtuvimos los siguientes valores como óptimos:

- learning_rate=0.1
- n_estimators=193
- max_depth=5
- min_child_weight=5
- gamma=0
- subsample=0.6
- colsample_bytree=0.75

Con estos nuevos valores obtenidos, procedimos a entrenar nuevamente y realizar una nueva predicción y subirla a Kaggle. Obtuvimos un puntaje de **0.46380**, el cual es una mejora sobre el puntaje que obtuvimos con los valores que no son óptimos.

4.3. CatBoost

CatBoost es otro algoritmo basado en árboles que se diferencia de los dos anteriores en el hecho de que puede utilizarse con variables categóricas. Utilizamos los mismos features que se describen en la tabla 3.1 pero sin realizarles encoding ni ningún tipo de procesamiento a las variables categóricas. Como primera aproximación, utilizamos Catboost con sus hiper parámetros por default, solo probamos ajustar un poco la cantidad de iteraciones y realizamos la primera predicción obteniendo un puntaje en Kaggle de **0.44668**. Al ver que con Catboost obtuvimos un mejor puntaje que el resto de los algoritmos con los hiper parámetros ya mejorados, nos preguntamos si los features que dejamos afuera y que en los otros algoritmos no funcionaron bien, generarían algún cambio con Catboost. Procedimos entonces a sumar a nuestros features, los que no funcionaron con anterioridad, y al agregar Account Name mejoró la predicción a Kaggle, obteniendo un score de **0.42537.** Vimos entonces, que CatBoost trata a las variables categóricas de una mejor forma que nuestra codificación, y seguimos entonces con este algoritmo pero ajustando sus hiper parámetros para ver si podemos mejorar

nuestra última predicción. Utilizamos Grid Search y encontramos que los parámetros óptimos son: iterations=100, depth=5, learning_rate= 0.05, y con estos hiper parámetros obtuvimos el mejor score en Kaggle que es de **0.41373**.

4.4. LSTM

Las redes neuronales LSTM son capaces de recordar un dato relevante en la secuencia y de preservarlo por varios instantes de tiempo.También tienen la capacidad de eliminar o añadir información que se considere relevante para el procesamiento de la secuencia.

Capas LSTM y Densas

Estas capas fueron las que nos dieron más trabajo, ya que hay que definir la cantidad de unidades muy cuidadosamente, e irlas ajustando según mejore o empeore los resultados del modelo a la hora de entrenarlo. Fuimos agregando y sacando capas densas, según los resultados que nos daba el modelo.También agregamos la capa Dropout (capa de abandono) la cual establece aleatoriamente las unidades de entrada en 0 con una frecuencia de rate (que definimos nosotros) en cada paso durante el tiempo de entrenamiento, lo que ayuda a evitar el sobreajuste.

Para la compilación del modelo utilizamos las siguientes especificaciones:

- **Loss:** Utilizamos ”binary crossentropy”, es la función de pérdida predeterminada que se usa para problemas de clasificación binaria. La función requiere que la capa de salida esté configurada con un solo nodo y una activación “sigmoidea” para predecir la probabilidad de la clase.
  Por lo que como última capa tenemos una densa de salida de un solo nodo y de activación sigmoidea.
- **Optimizer:** Utilizamos adam, optimizador que implementa el algoritmo Adam.
- **Metrics:** Usamos “acc” que esta métrica es para problemas de clasificación binaria.

Como conclusión, con los features que veníamos trabajando, detallados en la sección correspondiente, con este algoritmo nos dió 0.74535, empeorando el puntaje.
Pero con los features “Territory”, “TRF” y “Total Taxable Amount” obtuvo un puntaje de 0.56663.

Como mencionamos anteriormente, otros de los algoritmos que probamos fueron Bayesian Ridge y Partial Least Squares Regression, con los cuales en primer instancia obtuvimos puntajes bastante lejanos a las primeras aproximaciones del resto de los algoritmos.

4.5. Bayesian Ridge

Los hiper-parámetros estudiados con Grid Search Cross Validation fueron:
n_iter: [10, 50, 100, 300, 500, 1000]
compute_score: [True, False]
normalize: [True, False]
copy_X: [True,False]

Logrando la mejor estimación con:
compute_score=True, copy_X=True, n_iter=10, normalize=False, tol=0.001, verbose=False

El mejor score obtenido con GridSearchCV fue: 0.55282
Y su puntaje en Kaggle fue: 0.62931

4.6. Partial Least Squares Regression

Los hiper-parámetros estudiados con Grid Search Cross Validation fueron:
scale: [True, False]
copy: [True, False]
max_iter: [50, 100, 300, 500, 1000, 1500]

Logrando la mejor estimación con:
copy=True, max_iter=50, scale=False

El mejor score obtenido con GridSearchCV fue: 0.68464
Y su puntaje en Kaggle fue: 0.67921

Resumen de los algoritmos

Algoritmo	Hiper-parámetros	Mejor Score en Kaggle
CatBoost	iterations=100 depth=5 learning_rate=0.05	0.41373
XGBoost	learning_rate=0.1 n_estimators=193 max_depth=5 min_child_weight=5 gamma=0 subsample=0.6 colsample_bytree=0.75	0.46380
Random Forest	n_estimators=80 min_samples_split=1000	0.46713
LSTM	Units = 124 dropout=0.25 recurrent_dropout=0.2	0.56663
Bayesian Ridge	compute_score=True copy_X=True n_iter=10 normalize=False tol=0.001 verbose=False	0.62931
Partial Least Squares Regression	max_iter=50 copy=True scale=False	0.67921

5. Conclusiones

Durante este trabajo práctico aprendimos que la etapa de feature engineering es la más complicada y la que requiere de más imaginación. También pensamos que es la más importante ya que el agregado o quita de un feature puede subir o bajar mucho el score de una predicción, y si bien este suele variar dependiendo el algoritmo elegido, por lo general los features que benefician la predicción con un algoritmo, también lo hace con los demás.

Por otro lado la etapa de aplicación de algoritmos, no es tan complicada pero si requiere de bastante tiempo la búsqueda de los mejores hiper-parámetros. El método de Grid Search Cross Validation fue el primero que probamos porque en la teoría nos parecía el mejor. Pero con varios hiper-parámetros y varios valores por cada uno de ellos, el total de combinaciones que se hacen puede ser enorme. Además para nuestra sorpresa, con cierta combinación de valores daba un score muy bajo, cuando con un conjunto de valores inferiores daba mejor. Encontrar los valores justos para los que el método funcione bien también requería bastante tiempo, por lo que este método no es tan "automático" como pensábamos.

En particular, al grupo nos agradó mucho los algoritmos basados en árboles de decisión. Nos resultaron muy útiles en ambas etapas descriptas anteriormente. Corroboramos que son muy útiles para la selección de features al ser rápidos, no requerir normalización de datos y funcionar bien para la mayoría de los set de datos. Aunque también corroboramos que pueden generar overfitting en ciertas situaciones. A nosotros nos dio con una sola columna: Account_Name.

Por último, no queríamos dejar de señalar la importancia del preprocesamiento y limpieza de datos al comienzo de este tipo de trabajos. Particularmente en este set de entrenamiento no había demasiado trabajo por hacer en esta etapa, ya que había columnas enteras con valores nulos fáciles de eliminar, otras con demasiados valores nulos que no se podían rellenar, y la mayoría con datos completos. Aún así completar los datos que se puedan como por ejemplo el caso de Total Taxable Amount, y más aún, convertir sus valores a una misma moneda para que el entrenamiento tenga sentido, ayuda a la predicción. También es importante saber que se está haciendo, comprendiendo bien al set de entrenamiento. Por ejemplo, entrenar rápidamente sin saber que los registros son productos y no necesariamente oportunidades diferentes, sería un error grave. Como también lo sería entrenar con el problema de target leakage.