Desafío de final de curso - Machine Learning

Instituto Humai
Alumno: Felipe Aguirre
DNI: 41702703
Email: lipe.aguirre@gmail.com
Fecha: 13/10/2021

Diccionario

ML_Desafio_final:
Notebook de tratamiento de los datos de entrenamiento, junto con el entrenamiento del modelo.
https://colab.research.google.com/drive/1g0f8-GGEskRxnKyEvifM2bvb1CRjKBk8?usp=sharing

Validación ML_Desafio_Final:
Notebook de tratamiento de los datos de validación.
https://colab.research.google.com/drive/11HgR_Zs2rCU752pe19ZKnVPUpizEyXDA?usp=sharing

Resultados.csv:
datos de validación con su probabilidad predicha.

Análisis Exploratorio

El primer paso fue importar pandas para la lectura del dataset. La primera aproximación fue a través del método “head”, el cual reveló que se trataba de una base de datos bastante grande. También, que muchas de sus columnas tenían nombres y valores que no significaban nada para mí, incluso al leer la descripción provista por la página de la empresa. Supuse que esto dificultaría la identificación a simple vista de la variable causante de Target Leakage, por lo que no podría eliminarla a priori.

Además, observe que algunos nombres de columnas y sus datos tenían caracteres con coma o con espacios, lo cual puede traer confusiones. Por eso, los reemplace por “_”. Con un print de las columnas pude constatar que no hayan otros caracteres que puedan generar dificultades.

La función “corr” me permitió observar la relación entre los datos, tanto para comprender más de ellos como para pensar en el futuro feature engineering. Luego, utilice la función “value_counts” para revelar el contenido de algunas de las variables. Con la función “unique” comparé variables que me resultaban similares para observar en que se diferenciaban.

Intente utilizar los métodos exploratorios complejos vistos en clase, específicamente “sweetviz” y “ProfileReport”, pero no logré aplicar ninguno.

Feature Engineering

Imputación de datos faltantes

Con el método “isnull” y “sum” observé que sólo dos variables numéricas tenían una gran cantidad de datos faltantes: ASP y ASP_(converted). La gran diferencia entre la

media y la mediana de ambas variables sugiere la presencia de datos extremos. Por eso, elegí imputar los datos faltantes con su mediana. Para eso utilice la función SimpleImputer. Realice un print del valor máximo, mínimo, la media, la mediana y el desvío estándar antes y después de la imputación para evaluar que no hayan diferencias grandes y asegurarme que la imputación fue adecuada. Luego, realice un "dropna" para descartar las otras pocas filas con datos faltantes. A continuación, utilice la misma función (cambiando previamente el valor de np.nan) para descartar los datos con valor "NaN".

Para las fechas, más adelante en el código, escribí un for-loop que itera por la columna de fechas con datos faltantes. Si el valor es "NaN", lo imputa por la fecha anterior (siempre y cuando tampoco sea un "NaN", en cuyo caso accede a la fecha anterior).

Tratamiento de la variable target

La variable target original, "Stage", contaba con cuatro valores aproximadamente. De ellos, los valores de interés son "Closed_Won" y "Closed_Lost", haciendo referencia al resultado de la oportunidad. Por eso, elimine los registros con otros valores y luego transforme los valores de interés a 1 y 0 respectivamente. A continuación evalúe que la frecuencia de ambos valores de la variable sea relativamente similar, y así fue (57% de casos Won y 43% de casos Lost).

Transformación de variables a dummys

A través del value counts observé que en muchas variables más del 93% de los valores eran "None". Para evitar el fenómeno de overfitting (por generar muchas columnas con pocas observaciones), decidí transformarlas en dummys en función de si tienen o no un valor asignado (distinto a None). Estas variables fueron Brand, Product Type, Size, Product Category B y Price.

Eliminación de variables

Elimine aquellas columnas que, a mi criterio, no tenían valor predictivo. Estas fueron:

- ID: porque es un número para identificar la observación, y ya contamos con el index.
- Account_Name: porque son 908 variables categóricas que en principio no aportan nada, ya contamos con el index.
- Opportunity_Name: porque son 4139 variables categóricas que en principio no aportan nada, ya contamos con el index.
- Opportunity_ID: porque son 4139 variables categóricas que en principio no aportan nada, ya contamos con el index.
- Last_Activity: por tener un solo valor.
- Submitted_for_Approval: por tener un solo valor.
- ASP_(converted)_Currency: por tener un solo valor.
- Actual_Delivery_Date: por tener un solo valor y causar Target Leakage.

- Prod_Category_A: por tener un solo valor.
- Delivery_Year: porque son años que ya pasaron, no sirven al momento de llevar el modelo al mundo actual.

Binning de variables categóricas

Realicé un print de value counts de las variables categóricas para identificar visualmente cuales requerían una agrupación de los valores menos frecuentes en una categoría “Otro”. El objetivo fue evitar el overfitting por generar múltiples categorías con pocas observaciones. Mi criterio implicó ubicar a partir de qué índice caían bruscamente la cantidad de observaciones a valores bajos. Esto dependia entonces de cada variable. Así, creé un diccionario con el nombre de la variable como key y el índice del value counts hasta el cual me interesaba conservar sus valores originales como value. A partir de dicho índice, el valor original de las observaciones se cambiará a “Otro”.

Variables con fechas

Las variables con fechas implicaron un desafío. Primero debí pensar si debían conservarse o no, ya que una fecha pasada no tendría sentido en un árbol de decisión implementado en la actualidad. Al final, concluí que sí tenía sentido calcular ciertas variables a partir de ellas. Mis ideas fueron:

1. Amplitud_fecha_entrega = Planned_Delivery_End_Date - Planned_Delivery_Start_Date.
   Es decir, la amplitud en días de la fecha en la cual se entrega el producto. A mayor amplitud, se espera una menor probabilidad de éxito.

2. Amplitud_cuenta_oportunidad = Opportunity_Created_Date - Account_Created_Date.
   Es decir, tiempo entre crearse la cuenta y crearse la oportunidad. A mayor amplitud, se esperaría una menor probabilidad dado que su creación implicó más recursos temporales, puede haber pasado la fecha para la cual se necesitaba el producto, el cliente estaba poco convencido en relación a que producto comprar, etc.

3. Amplitud_oportunidad_modificacion = Last_Modified_Date - Opportunity_Created_Date.
   Es decir, tiempo de definición de detalles de la oportunidad, esfuerzo, burocracia, recursos utilizados, etc. A mayor amplitud, se espera una menor probabilidad de éxito.

4. Amplitud_oportunidad_max_entrega = Planned_Delivery_End_Date - Opportunity_Created_Date.

Es decir, el tiempo entre la creación de la oportunidad y la fecha máxima de entrega. A mayor amplitud se espera una menor probabilidad de éxito dado que el cliente puede guiarse por el peor de los casos (fecha máxima de entrega) para evaluar si le sirve o no el producto (no compraría un producto que podría llegarle después de cuando lo necesita). También, puede concluir que simplemente podría tardar demasiado en llegar.

5. Estación del año.
   Importante para una empresa de aires acondicionados. La estación se calculó por país (Billing Country, ya que Territory tenía demasiados datos faltantes) y su hemisferio. La estación se calculó para cada fecha de interés.

6. Semana del mes.
   Se espera que la probabilidad sea mayor durante la primera semana, cuando muchos trabajadores cobran su dinero.

7. Mes de cada variable.
   Calculé el mes en el que se dio cada evento temporal de interés.

Por otro lado, transforme la variable Mes y Mes de cada variable en categóricas. De tener un formato int, la distancia entre el mes 1 y el mes 12 sería mucho mayor que entre el mes 1 y el mes 2, lo cual en términos de temperatura no es cierto, además de otras diferencias que podrían no existir (como comportamiento de compra). Sin embargo, también deje otra variable Month_num como una variable numérica del mes por si tenía algún valor.

Eliminación de Outliers

La detección y eliminación del 10% de los valores más extremos se realizó con Isolation Forest en cada modelo antes de entrenarlo.

Modelos

Antes de probar distintos modelos, realice una copia del data frame en df_boost para que las modificaciones en el data frame con el que voy trabajando no se arrastren a los siguientes modelos.

Regresión Logística

Decidí comenzar con un modelo de Regresión Logística para poder comparar su performance con la de modelos más complejos.

Para las variables categóricas realice OneHotEncoding. Para las numéricas me asegure de que tengan formato int. Realice el train test split, dejando como variable “y” a Stage y como variables “X” al resto de las features. El test size fue del 25%. Entrené al

modelo con un max_iter de 10000 para que el límite de iteraciones no sea un limitante (aunque idealmente debería haber buscado su valor óptimo con Random Search).

El Accuracy reportado en el set de test fue de 64%, es decir que clasificó bien a ese porcentaje de observaciones, y no difirió del set de train. El accuracy fue bajo pero el poder de generalización fue bueno.

Simple XGBoost

El fenómeno de Target Leakage sucede cuando el modelo es entrenado con información/una variable con la cual no contaría realmente al momento de predecir la variable objetivo. En este caso, el modelo tiene un rendimiento extremadamente alto pero poco realista. Por ejemplo, un modelo podría ser entrenado para predecir salarios anuales con múltiples variables, una de ellas siendo el salario mensual. Tendría un rendimiento perfecto, pero en verdad no contaríamos con el salario mensual al utilizar el modelo en un entorno real. Hasta el momento existe una variable que causa este problema en mi Data Frame.

En un principio no tenía planeado entrenar un modelo de XGBoost “simple” (sin búsqueda de hiperparametros), pero al intentar plotear las importancias de un modelo realizado a través de un pipeline, no aparecían los nombres originales de las variables y por lo tanto no podía identificar la variable que producía Target Leakage. Así, antes de comenzar con los modelos complejos y su ajuste de hiperparametros con pipeline y cross validation, primero entrené este modelo simple para poder plotear su importancia e identificar la variable causante de Target Leakage.

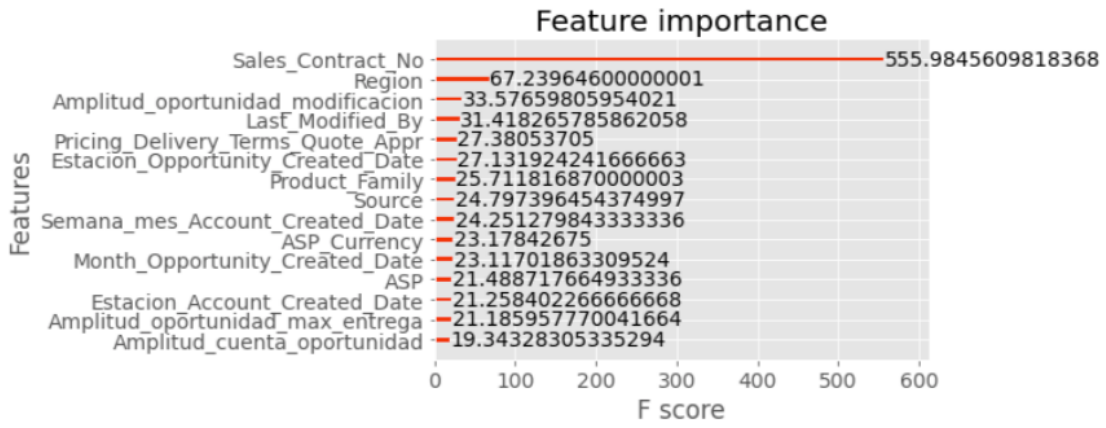
Las variables numéricas fueron escaladas con SimpleScaler, de modo que los valores originales fueron reemplazados por sus desvíos estándar. Esto se realizó para prevenir que las diferencias de magnitud entre ellas sesguen al modelo.

Los valores de las variables categóricas fueron reemplazados por valores ordinales con Ordinal Encoder. Los valores nuevos serán imputados con un valor extremo. Así, si tienen algo en común, quedan todos agrupados y aislados.

El modelo fue entrenado con n_jobs=3 para acelerar el training, “logloss” como métrica de evaluación, y label encoder desactivado.

No voy a profundizar en los resultados obtenidos, pero fueron significativamente mejores que los alcanzados con los posteriores modelos que no contaban con la variable causante de Target Leakage.

Una vez entrenado, utilice la función plot_importance para plotear la importancia de cada feature. Se puede observar en el gráfico que la variable “Sales_Contract_No” tiene una importancia desproporcionadamente mayor al resto. Según la descripción de variables, es el número de contrato asignado a la oportunidad. Teniendo todo esto en cuenta, concluí que esta es la variable responsable del Target Leakage y por eso la eliminé de los posteriores modelos.



Sales_Contract_No posee una importancia anormalmente alta.

Búsqueda de Hiperparametros con XGBoost

XGBoost Classifier es una muy buena herramienta de clasificación, por lo tanto valía la pena realizar una optimización de hiperparametros.

Realice el mismo tratamiento de las variables que en el modelo anterior, agregando una transformación polinómica de grado dos sobre las variables numéricas para que cualquier efecto que tengan se vea amplificado. Este paso lo realicé con PolynomialFeatures(2).

Para la optimización de hiperparametros utilice RandomizedSearchCV. Esta función permite evaluar múltiples combinaciones aleatorias de los valores que se le pase. Además, utiliza una parte del data frame para entrenar al modelo y el resto para validarlo, proceso que repite más de una vez variando la porción del data frame que utiliza para cada paso, y finalmente promedia los resultados. En este caso, el número de particiones para entrenamiento-validación fue solo de tres para acelerar el entrenamiento.

Los valores pasados al pipeline fueron un rango de valores para cada parámetro de XGBoost. El rango de valores fue obtenido desde lo recomendado por foros de internet y desde las notebooks de las clases de Humai. El número de combinaciones a evaluar fue de 100, mayor al valor por default, con el objetivo de maximizar la calidad del modelo.

Antes de comentar los resultados del modelo, es necesario definir las métricas de evaluación:

1. Accuracy: porcentaje de datos clasificados correctamente por el modelo.
2. Precisión: porcentaje de correctos positivos (observaciones correctamente predichas como “Won”).
3. Recall: porcentaje de datos con etiqueta “Won” que efectivamente fueron identificados como tal. Es decir, cuánto del universo total de oportunidades cerradas en “Won” fue capaz de predecir el modelo.
4. F1 Score: medida armónica entre precisión y recall.

5. AUC: área bajo la curva ROC. Es decir, la capacidad del modelo de distinguir entre clases en función del punto de corte (a partir de qué probabilidad se concluye que una observación pertenece a una u otra categoría). Cuanto mayor sea el área bajo la curva, medido en porcentaje, mayor será la capacidad del modelo de generar verdaderos positivos y menor será su generación indeseada de falsos positivos.

Una vez entrenado, las métricas de evaluación arrojaron los siguientes resultados:

6. Accuracy: 80.5% (ampliamente superior al obtenido con la Regresión Logística).
7. Precisión: 81%.
8. Recall: 87%.
9. F1 Score: 84%.
10. AUC: 79%.

Búsqueda del mejor modelo

En este paso, el objetivo fue probar todas las combinaciones de hiperparametros de cada modelo predictivo de Boosting: XGBoost, CatBoost y LightGBM. Así, se esperaba encontrar el mejor modelo posible. Una vez más utilice RandomSearchCV pasandole los mismos rangos de hiperparametros a evaluar pero esta vez probandolo en los tres modelos.

El mejor modelo encontrado fue CatBoost Classifier. Las métricas de evaluación arrojaron los siguientes resultados:

1. Accuracy: 81%
2. Precisión: 83%
3. Recall: 86%
4. F1 Score: 85%
5. AUC: 80%

Como puede apreciarse, este modelo obtuvo mejores resultados que el XGBoost incluso antes de la optimización de sus hiperparametros.

CatBoost Optimizado

Dado que CatBoost obtuvo los mejores resultados de entre todos los modelos de boosting, decidí realizar una optimización de sus hiperparametros.

Utilice RandomSearchCV con los mismos valores que antes, pero esta vez estreché más los rangos de búsqueda en función de los resultados obtenidos con best_params y volvía a entrenarlo. Este proceso lo realice tres veces.

Los hiperparametros con mejor rendimiento fueron los siguientes:

1. border_count: 51. Determina el número de splits para las variables numéricas.

2. Colsample_by_level: 0.99. Determina el ratio de la muestra de las columnas a la hora de construir el árbol.

3. Learning_rate: 0.12. Determina la velocidad de aprendizaje del modelo.

4. Max_depths: 9. Determina la profundidad máxima de cada árbol.

5. N_estimators: 199. Determina el número total de árboles.

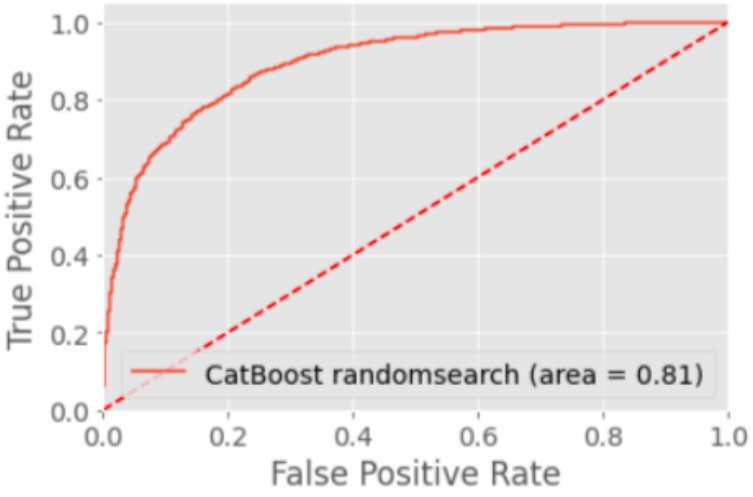
6. Reg_lambda: 0.36. Determina el peso del regularizador lambda, el cual previene el overfitting.

7. Subsample: 0.92. Determina el ratio de la muestra de observaciones.

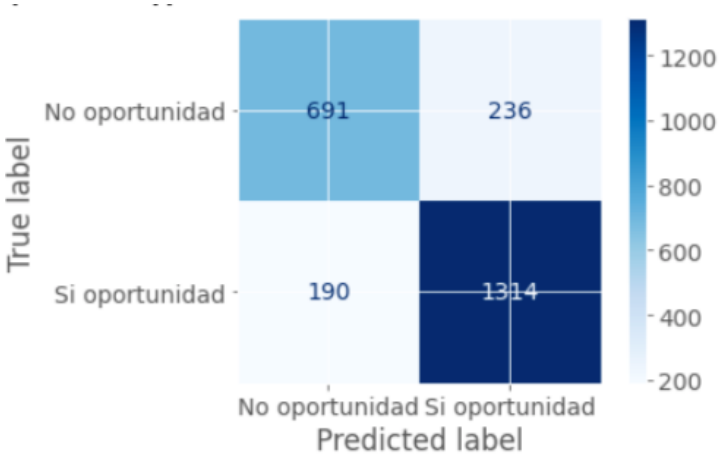
8. Thread_count: 4. Determina la velocidad del entrenamiento.

Las métricas de evaluación arrojaron los siguientes resultados:

1. Accuracy: 82%
2. Precisión: 85%
3. Recall: 87%
4. F1 Score: 86%
5. AUC: 81%



Curva ROC y área bajo la curva graficados.



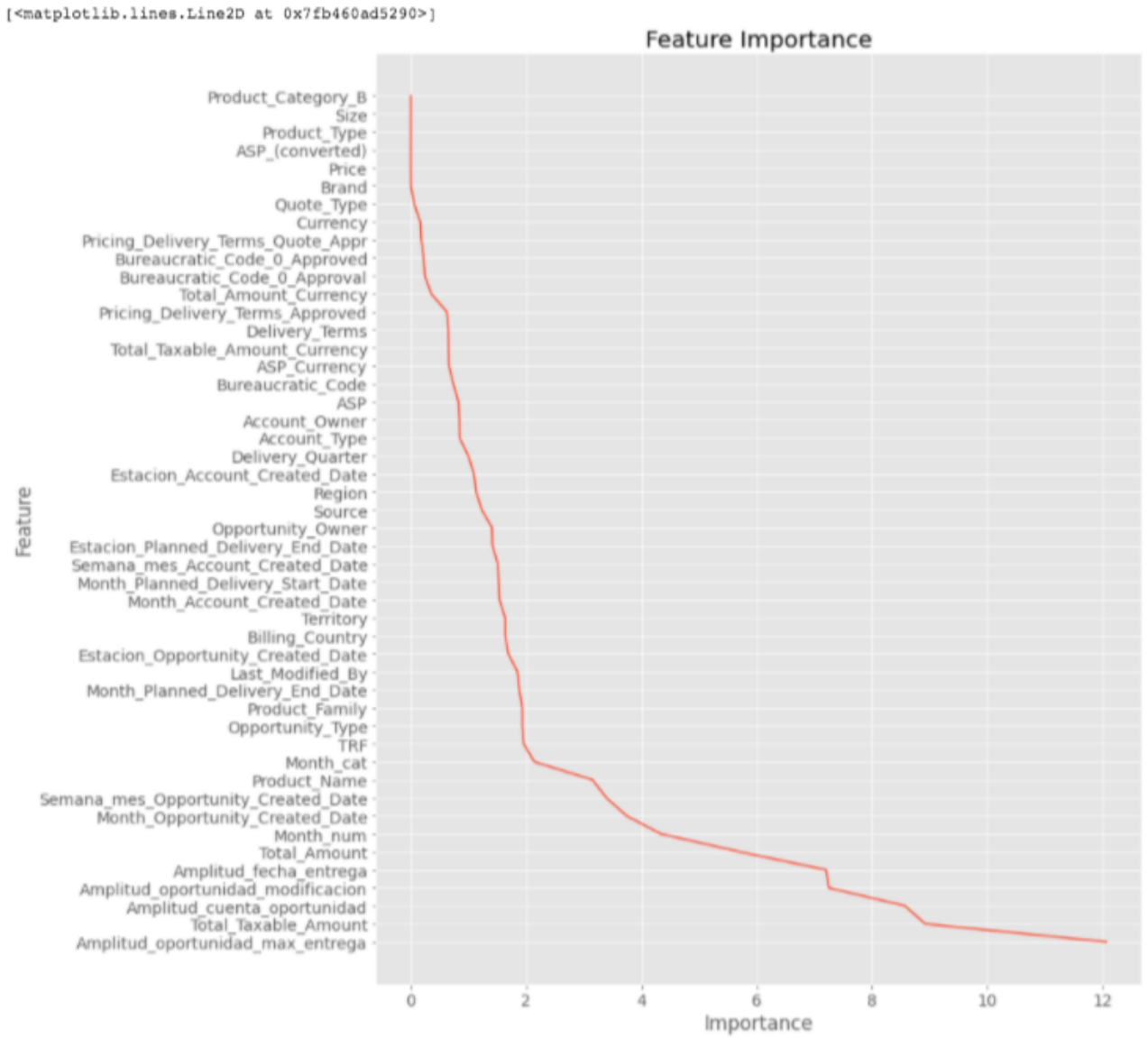
Matriz de Confusión.

Gracias a la Matriz de Confusión se puede observar cómo, de haberse utilizado en el mundo real, el modelo hubiera perdido solo 190 oportunidades (es decir, casos en los que el modelo predijo “Lost” pero el resultado real fue “Won”), pero hubiera dedicado los recursos a 1314 oportunidades correctamente predichas como “Won”. Por último, hubiera dedicado recursos en vano a 236 casos.

Se puede apreciar que este modelo obtuvo los mejores resultados en las métricas de evaluación.

Feature Importance

Feature Importance refiere a la importancia de cada variable en el modelo a la hora de predecir un resultado. Un mayor puntaje (“importance” en el gráfico) significa que el valor que tome la variable determina fuertemente la probabilidad predicha, y un puntaje bajo significa que la variable no tiene un gran efecto sobre la probabilidad.



Plot de Feature Importance

La imagen superior representa un gráfico de la importancia de las 48 variables que constituyen al modelo. Las primeras cuatro variables tienen una importancia nula. Esto quiere decir que da igual el valor que tomen, no afecta en nada la probabilidad de éxito.

Tomemos la variable Size, por ejemplo. Si el producto tiene un tamaño asignado o no, no tiene ningún efecto sobre la probabilidad de éxito de la oportunidad comercial. Lo mismo sucede con el precio. Debemos recordar que ambas variables, junto con las otras dos que constituyen las cuatro variables con nula importancia, poseían aproximadamente un 94% de valores nulos. Esto implica que su nula importancia se debe posiblemente más a la falta de datos que a una desconexión real con la probabilidad de éxito. De contarse con todos los datos, su importancia podría ser mucho mayor.

En el extremo opuesto tenemos a la variable “Amplitud_oportunidad_max_entrega”. Su influencia sobre la probabilidad es por lo menos doce veces mayor que las 21

variables de menor importancia, y es un 33% más influyente que la segunda variable más importante. Recordemos que refiere a la amplitud en días entre el día de la creación de la oportunidad comercial y la fecha máxima de entrega. Esta es, por lejos, la variable más importante al predecir la probabilidad de éxito de una oportunidad comercial.

Las 10 variables más importantes son:

	Feature	Importance
1	Amplitud_oportunidad_max_entrega	12.0632
2	Total_Taxable_Amount	8.92281
3	Amplitud_cuenta_oportunidad	8.57204
4	Amplitud_oportunidad_modificacion	7.253
5	Amplitud_fecha_entrega	7.20282
6	Total_Amount	5.71043
7	Month_num	4.33615
8	Month_Opportunity_Created_Date	3.74427
9	Semana_mes_Opportunity_Created_Date	3.39449
10	Product_Name	3.15286

Top 10 variables más importantes.

La mayoría de las variables ya fueron descritas, sin embargo cabe definir al resto:

- Total_Taxable_Amount: total de impuestos asociados a la compra.
- Total_amount: monto/precio total.
- Month_num: mes del límite inferior del rango previsto para la fecha de entrega.
- Month_Opportunity_Created_Date: mes en el cual se creó la oportunidad.
- Semana_Mes_Opportunity_Created_Date: semana del mes en la cual se creó la oportunidad.
- Product_name: nombre del producto.

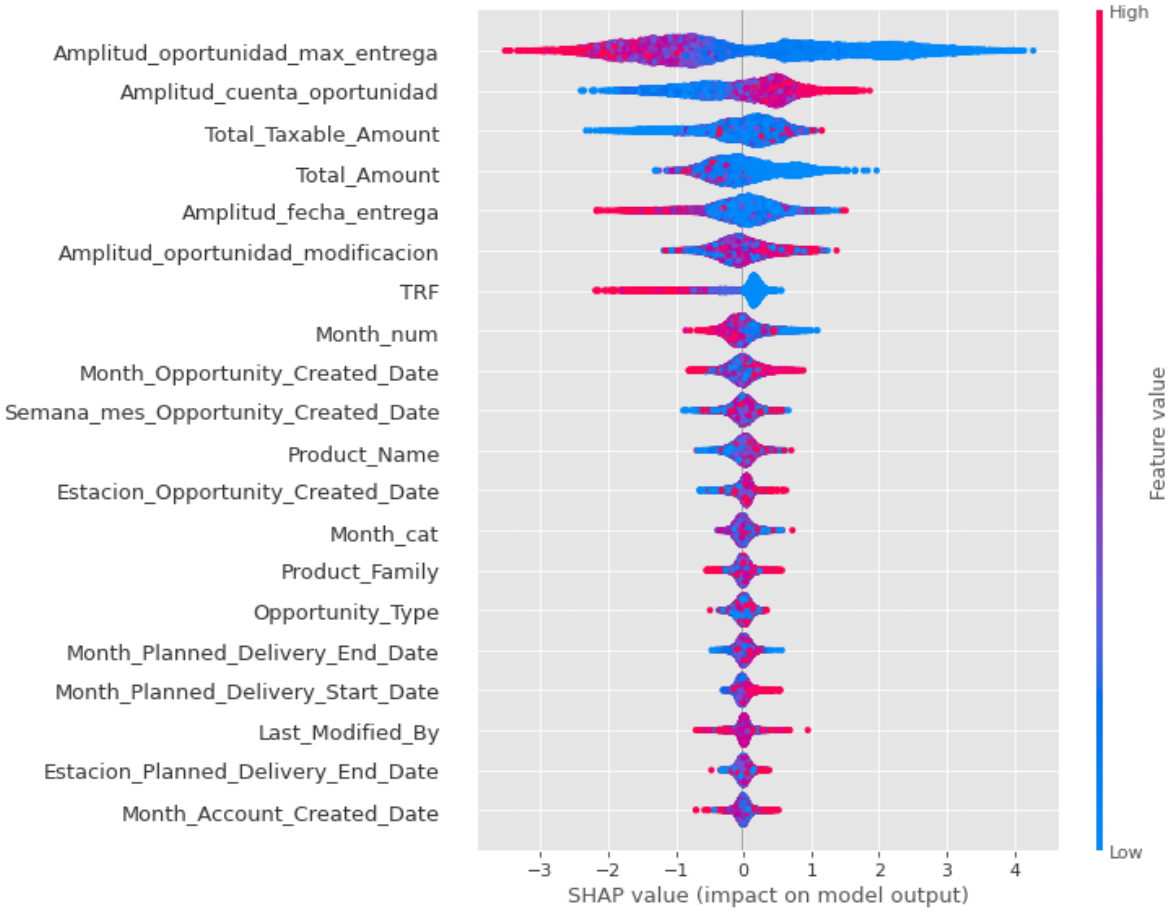
Shap Values

Ya hemos definido las variables con la mayor influencia sobre la probabilidad de éxito de la oportunidad, pero aún no he aclarado la dirección de esa influencia ni el rol del valor que toma la variable sobre dicha dirección. Por ejemplo, si una variable aumenta la probabilidad de éxito al aumentar su valor, o, por el contrario, la disminuye. Para responder esta incógnita utilicé la función ShapValues.

Los valores SHAP explican las predicciones de un modelo a partir de computar la contribución de cada variable individual a la predicción. Para esto, se compara la

probabilidad predicha cuando esa variable toma un cierto valor, con la probabilidad predicha cuando esa misma variable toma un valor base.

El siguiente gráfico representa el top 20 variables con mayores valores SHAP. El color de la variable representa su valor (rojo significa un valor alto, azul un valor bajo). Debemos tener en cuenta que algunas de las variables son en verdad categóricas (transformadas a datos ordinales con Ordinal Encoder), por lo que su color no representa más o menos de esa variable sino tan solo la etiqueta numérica asignada a cada valor categórico. Para estas variables, la interpretación del gráfico se reduce solo a su impacto en la dirección de la probabilidad, y no debemos tener en cuenta su color. Esas variables son: Product_name, Semana_mes_Opportunity_Created_Date, Estacion_Opportunity_Created_Date, Month_cat, Product_Family, Opportunity_Type, Last_Modified_By, Estacion_Planned_Delivery_End_Date.
El eje X representa la dirección de la influencia. A la izquierda del cero, la variable mueve la probabilidad de éxito a valores más bajos. A la derecha del cero, aumenta la probabilidad.



Valores Shap de todas las variables.

A partir del gráfico pueden extraerse las siguientes conclusiones de las cinco primeras variables:

1. Amplitud_oportunidad_max_entrega:
    - Cuando la variable toma valores bajos, la probabilidad de éxito aumenta. Lo contrario ocurre cuando toma valores altos, aunque el efecto es algo menos potente.
    - Esto puede deberse a que el cliente es menos propenso a comprar cuando la fecha máxima estimada de entrega es muy lejana.
    - Más allá del color, su amplitud en el eje X sugiere que es una variable muy influyente tanto para aumentar como para disminuir la probabilidad de éxito.

2. Amplitud_cuenta_oportunidad:
    - Cuando toma valores bajos, la probabilidad disminuye. Lo contrario ocurre cuando toma valores altos, aunque el efecto es algo menos potente.
    - Esto podría deberse a varias razones. Por ejemplo, el cliente ya ha realizado otras compras, ya lleva un largo tiempo siendo cliente.
    - Es interesante que los resultados sean contrarios a la hipótesis por la cual cree la variable en primer lugar.

3. Total_Taxable_Amount
    - La homogeneidad del color sugiere que los valores son en promedio bajos, y por lo tanto no puede interpretarse la influencia del valor.
    - Si podemos observar que su influencia aplica sobre todo en la disminución de la probabilidad de éxito.

4. Total_Amount
    - Ocurre lo mismo que con la anterior variable. Por el contrario, su influencia aplica sobre todo en el aumento de la probabilidad.

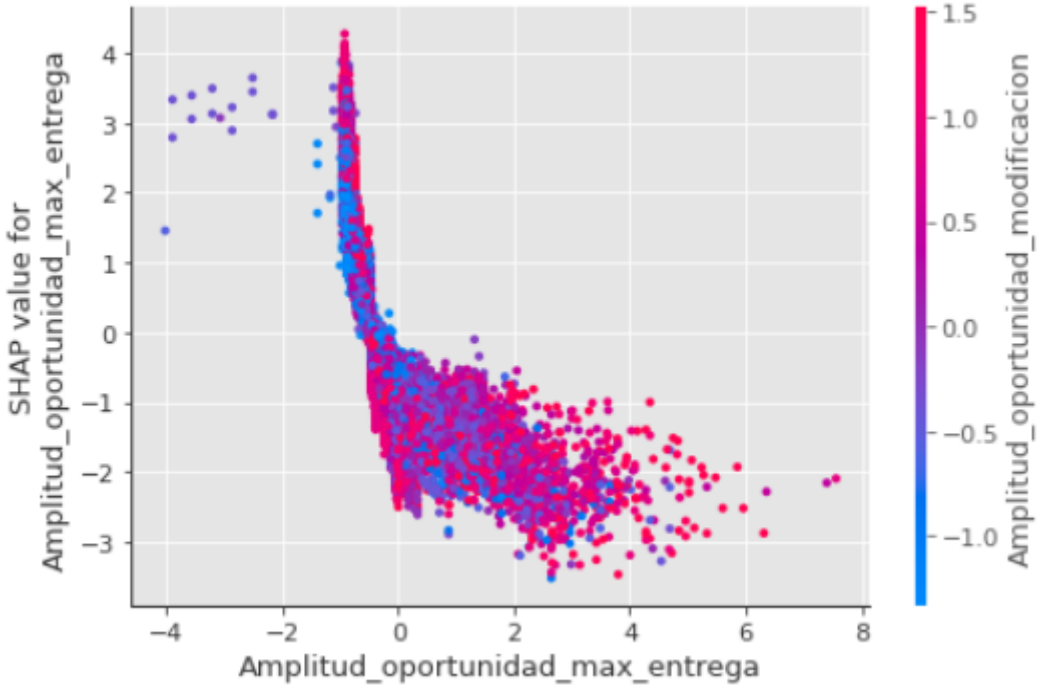
5. Amplitud_Fecha_Entrega
    - Cuando toma valores altos, la probabilidad de éxito disminuye. Lo contrario ocurre cuando toma valores bajos.
    - Esto puede deberse a que el cliente está menos dispuesto a comprar cuando es muy amplio el rango de fechas dentro de las cuales su producto puede ser entregado.

SHAP Value: Amplitud_oportunidad_max_entrega

El siguiente gráfico revela la relación entre el valor de Amplitud_oportunidad_max_entrega (Eje X) y su influencia en la probabilidad de éxito medida con SHAP value (Eje Y). Puede observarse que la variable es extremadamente influyente en la probabilidad de éxito cuando toma valores muy bajos, y es cada vez menos influyente conforme toma valores más altos, aunque la relación parece no ser del todo lineal.

Esto implica que el cliente es muy propenso a comprar cuando la fecha máxima de entrega es muy cercana. De todas las variables del modelo, una fecha máxima de entrega cercana a la fecha de creación de la oportunidad es el principal determinante de la compra del producto, concretamente aumentando su probabilidad.

Este gráfico también demuestra la relación con otra variable; Amplitud_oportunidad_modificacion. Si el color es rojo, el valor de esta variable es alto, y lo contrario ocurre si es azul. Puede observarse todos los casos para los cuales Amplitud_oportunidad_max_entrega fue muy influyente en la probabilidad de éxito, Amplitud_oportunidad_modificacion tomo valores bajos. Por el contrario, todas las veces que Amplitud_oportunidad_max_entrega fue poco influyente, Amplitud_oportunidad_modificacion tomo valores altos. Sin embargo, lo más probable es que esta relación trate de una correlación y no de una relación causal entre sí.



Shap Value de amplitud_oportunidad_max_entrega y su relación con amplitud_oportunidad_modificacion.

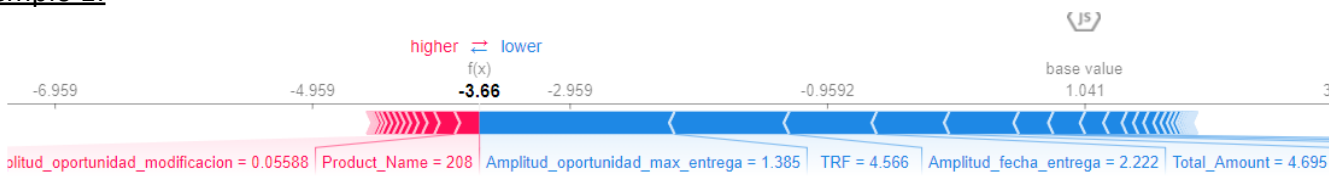
SHAP Values: predicciones individuales.

Para comprender mejor la influencia de ciertas variables sobre las predicciones, a continuación se demuestra cómo se llegó a la predicción individual de algunas observaciones seleccionadas aleatoriamente. Por razones de formato del gráfico, no todas las variables podrán incluirse en el esquema, pero pueden apreciarse las más relevantes. Se recomienda verlas en la Notebook.

La predicción base se representa en el valor “base value” (debemos tener en cuenta que la probabilidad no se encuentra ajustada entre 0 y 1). Las variables en rojo aumentan la probabilidad de éxito, mientras que las variables en azul la disminuyen. El

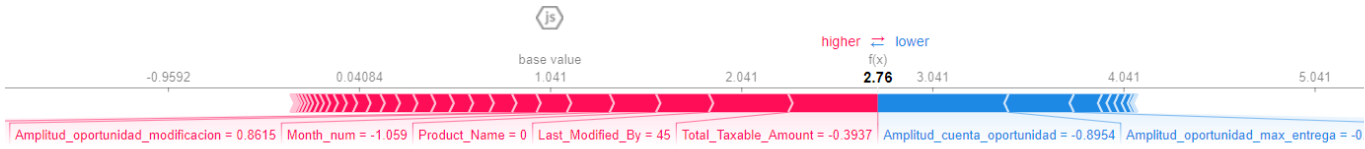
tamaño que ocupa la variable en el gráfico es proporcional a su influencia en esta observación.

Ejemplo 1:



- Esta observación se predijo como “Lost” o fracaso en el cierre de la oportunidad. Puede observarse cómo las variables azules ocupan más espacio y mueven la predicción hacia la izquierda del base value.
- Para esta observación, la variable más influyente fue Amplitud_oportunidad_max_entrega. Dentro de las variables que movían la predicción hacia la izquierda, la más influyente fue un tipo particular de Product_Name (el 208). Sin embargo, fueron mayores las influencias del lado azul.

Ejemplo 2:



- Esta observación se predijo como “Win” o éxito en el cierre de la oportunidad. Puede observarse cómo las variables rojas ocupan más espacio y mueven la predicción hacia la derecha del base value.
- Para esta observación, la variable más influyente fue Total_Taxable_Amount. Dentro de las variables que movían la predicción hacia la izquierda, la más influyente fue Amplitud_cuenta_oportunidad. Sin embargo, fueron más las influencias del lado rojo.

Predicción de los datos de evaluación

La base de datos de validación, es decir, aquellas observaciones a predecir, recibieron un tratamiento muy similar a los datos de entrenamiento. Esto se debe a dos razones; por un lado requieren de la misma limpieza (transformación de las columnas, eliminación de columnas sin valor predictivo, etc.), y por el otro requieren de la misma ingeniería de datos (creación de variables con fechas, binning, escalado, etc.). Por cuestiones de practicidad, esto se realizó en una Notebook distinta a la de entrenamiento del modelo, nombrada “Validación ML_Desafio_Final”.

La principal diferencia implicó no eliminar la columna ID_Oportunidad. Esto se debió a que la columna que la consigna sugiere como ID de la oportunidad, columna “ID”, contenía valores repetidos. Esto puede deberse a que más de una oportunidad se agrupa en un ID, por ejemplo si es realizada por la misma persona. Ante la duda decidí agregar ambas columnas a la tabla de datos final.

Una vez realizado el tratamiento de las observaciones a predecir, guardé el Data Frame en un excel y lo importé a la Notebook con el modelo entrenado. Allí utilice la función “predict_proba” para obtener la probabilidad de que la oportunidad correspondiente a cada observación sea exitosa o se cierre en “Won”. Finalmente, guarde los resultados en un excel.

Conclusiones

Frío Frío es una empresa B2B (“Business To Business”), por lo que resulta esencial optimizar los esfuerzos de sus representantes comerciales. De tener que predecir cuales negocios se cerrarán exitosamente y cuáles no lo harán, se puede esperar un accuracy, o capacidad de decidir correctamente, de aproximadamente el 50%. El juicio de un experto seguramente sea algo mejor, pero implica considerables gastos económicos y especialmente temporales, ya que analizar los datos de una sola oportunidad puede tomar mucho tiempo.

Este modelo ofrece un accuracy del 81% y un recall del 87%. Esta última métrica implica que casi 9 de cada 10 oportunidades que serán exitosas, serán reconocidas por el modelo y por lo tanto aprovechadas. A la vez, la precisión es del 85% lo que implica que sólo un 15% de las veces que se dediquen recursos a una oportunidad será en vano. Por último y tal vez la principal ventaja, este modelo permite analizar miles de oportunidades sin costo alguno y en unos pocos minutos.

Si fuese consultor de la empresa, recomienda enfocar todos los recursos en mejorar la variable que, por mucho, en mayor medida determina la probabilidad de éxito: disminuir el tiempo entre la creación de la oportunidad y la fecha máxima de entrega.

Comentarios

Calidad de los datos:

- Algunas de las variables que aparecen en el diccionario de variables provisto por la empresa no figuran en la base de datos (Ej: Total_power).
- Algunas variables que figuran en la base de datos no figuran en el diccionario (Ej: TRF).
- Muchas de las variables contienen solo un valor.
- Muchas variables contienen gran porcentaje de datos faltantes.
- ID_Oportunidad se repite, por lo que no es útil para identificar cada oportunidad individual.

Propuestas de mejora:

De haber tenido más tiempo, hubiera mejorado los siguientes aspectos:

1. Emprolijar la Notebook. Realmente es el principal punto a mejorar. Celdas que se repiten, celdas que no cumplen ninguna función, celdas poco eficientes, etc.
2. Mejorar el análisis exploratorio del comienzo.
3. Mejorar el Feature Engineering:
    - El binning de categorías podría haberse evitado dado que para el modelo final utilicé Ordinal Encoder, por lo que no se hubieran generado múltiples columnas con muy pocas observaciones con valor. Lo mismo para las variables que transforme en dummys por tener pocos valores distintos a “None”.
    - No eliminar Account_name: al pensarlo mejor, si podría tener valor predictivo.
    - Crear variable “Client_history”: calcular si el cliente ya hubo realizado otra compra anteriormente (en función de la cantidad de veces que aparece el valor en Account_name), y cuantas compras realizó.

4. CatBoost Optimizado:
    - Seleccionar mejor el rango de los siguientes parámetros: N_estimators y Thread_count.
    - Al borrar las columnas con importancia nula o cercana a nula, el rendimiento del modelo empeoraba. Nunca pude comprender la razón.

Mensaje final

El nombre “desafío” no podría haber sido más apropiado. Antes de comenzar el curso, jamás había escuchado sobre una regresión lineal. Jamás había trabajado con una base de datos. Mi formación es en psicología, por lo que todo esto es nuevo para mi.
Dicho esto, fue muy satisfactorio afrontar y concluir el desafío. Todo el proceso de prueba, error y aprendizaje fue recompensante. Los temas vistos en la cursada y la posibilidad de aplicarlos a un problema del mundo real despertaron mi interés y mi dedicación. Los profesores me impresionaron amables, dedicados y con una genuina preocupación por nuestro aprendizaje. Espero con ansias el próximo módulo.