Relevamiento, estudio de variables y aplicación de técnicas de modelado con machine learning, en las ventas al público en cafeterías de New York a lo largo del mes de abril de 2019

IBM Coffee Shop Sample Data

Proyecto final de Data Science de CoderHouse

Néstor Alfredo Fuenzalida Troyano

23 de octubre de 2023

TABLA DE CONTENIDO

1. INTRODUCCIÓN

1.1 PROBLEMA COMERCIAL

La tarea principal es, por medio del entrenamiento de un modelo de machine learning, intentar determinar el tipo de producto que el cliente desea ordenar. Para esto primero se plantea un formateo las variables proporcionadas y una serie de visualizaciones que suministren información para entender mejor el dataset.

1.2 CONTEXTO ANALÍTICO

Por medio de una descarga con la API de Kaggle se proporciona un archivo CSV que contiene detalles comerciales (ficticias), de ventas entre abril y marzo del 2019 de una cadena comercial de bebidas de cafetería, con tres ubicaciones en la ciudad de Nueva York, EEUU. Entre estos se podemos ver las siguientes variables: transaction_id, transaction_date, transaction_time, sales_outlet_id, staff_id, customer_id, instore_yn, order, line_item_id, product_id, quantity, line_item_amount, unit_price y promo_item_yn.

1.3 AUDIENCIA DEL PROYECTO

El proyecto está destinado a todas aquellas personas que estén interesadas en este tipo de mercados, como así también a todos aquellos estudiantes y particulares que deseen ver la implementación de técnicas de ciencia de datos y modelos de machine learning a ejemplos reales de aplicación.

2. OBJETIVOS DEL MODELO

El objetivo del proyecto poder determinar un modelo que, por medio de la interacción y relación entre las variables del conjunto de datos, pueda predecir con el mejor grado de exactitud el tipo de producto que se ordena en cada transacción. Para eso, en principio vamos a intentar responder preguntas tales como la canasta de productos más importantes, si existe un patrón de estacionalidad en las ventas, y en que horarios se reporta la mayor cantidad de transacciones.

3. ANÁLISIS EXPLORATORIO DE LOS DATOS (EDA)

Como se describe en la portada, el presente proyecto se basa en un conjunto de datos ficticio de ventas de una cadena de cafeterías en Nueva York entre abril y marzo de 2019. Estos datos provienen de IBM como un estudio de caso y están disponibles en Kaggle a través del siguiente link:

"https://community.ibm.com/community/user/businessanalytics/blogs/steven-macko/2019/07/12/beanie-coffee-1113"

3.1 BREVE DESCRIPCIÓN DE LOS DATOS

El dataset está compuesto de 14 variables con 49.894 instancias cada una. A continuación, una breve descripción de las variables del dataset:

- transaction_id: Número de la transacción comercia realizada.
- transaction_date: Fecha en la que se realizó la transacción comercial.
- transaction_time: Hora en la que se realizó la transacción comercial.
- sales_outlet_id: Identificación del punto comercial en el que se realiza la transacción.
- staff_id: Identificación del personal que realizó la transacción.

- customer_id: Identificación del cliente involucrado en la transacción.
- instore_yn: Variable que describe si la compra fue en el local comercial de la cafetería o no.
- order: Variable que corresponde al número de orden del pedido.
- line_item_id: Identificación de la línea en la que se realiza la compra del producto.
- product_id: Número que identifica el artículo o combo individual que se compró en la transacción.
- quantity: Cantidad de artículos o combos comprados en la transacción.
- line_item_amount: Precio total. Corresponde al producto del precio unitario por la cantidad.
- unit_price: Precio unitario del artículo o combo adquirido.
- promo_item_yn: Variable que describe si el artículo o combo entran bajo alguna promoción.

Podemos dividir las 14 variables en dos grupos:

I. Variables Cualitativas Nominales
1.transaction_id
2.sales_outlet_id
3.staff_id
4.customer_id
5.instore_yn
6.order
7.line_item_id
8.product_id
9.promo_item_yn

II. Variables Cuantitativas Continuas y Discretas
1.quantity
2.line_item_amount
3.unit_price
4.transaction_date
5.transaction_time

3.2 VALORES NULOS

En el dataset original no se encontraron valores nulos. Lo que si se detecto es la existencia de caracteres del tipo espacio " ", en la variable "instore_yn". Este resultaba como 3 valor único de dicha variable, que solo podía contener los valores "yes" y "no". Dado que la cantidad de instancias que presentaban este error eran 294 de 49.894, una cantidad realmente pequeña, se optó por eliminarlas del dataset.

3.3 VALORES DUPLICADOS

El dataset original no presentaba valores duplicados.

3.4 OPERACIONES CON LA VARIABLE FECHA

Originalmente las variables "transaction_date" y "transaction_time" (fecha y hora en términos más comunes), estaban el formato "object". Para una aplicación más sencilla de técnicas de transformación y operación de decidió transformarlas al formato "datetime64[ns]" con el método "to_datetime" de la librería "pandas".

3.5 IDENTIFICACIÓN DE OUTLIERS

En las variables cuantitativas del dataset original "quantity"(cantidad), "line_item_amount"(El valor total por producto) y "unit_price" (el precio unitario), no se encuentran outliers. El resto de las variables tampoco lo presentan.

3.6 CORRELACIÓN ENTRE VARIABLES

Utilizando el índice de correlación de Pearson entre las variables se generó un heatmap, del que se resaltan las relaciones cuyo índice ronda el aproximado de “0,7”:

- Staff_id y Sales_outlet_id: ic = 0,7
- Unit_price y Order: ic = 0,76
- Unit_price y Line_item_amount: ic = 0,67
- product_id y Line_item_id: ic = 0,60

Destacamos la correlación entre "product_id" y "line_item_id", por ser la primera nuestra variable objetivo en el proyecto.

A modo de resumen se agrega la siguiente tabla que contiene la totalidad de las variables, la cantidad de valores nulos y no nulos, el tipo de dato que la representa y un breve ejemplo de la misma:

Variable	Tipo de Variable	Valores Nulos	Valores No Nulos	Ejemplo de la Variable
transaction_id	int64	0	49600	7
transaction_date	datetime64[ns]	0	49600	2019-04-01 00:00:00
transaction_time	datetime64[ns]	0	49600	2023-10-22 12:04:43
sales_outlet_id	int64	0	49600	3
staff_id	int64	0	49600	12
customer_id	int64	0	49600	558
instore_yn	object	0	49600	N
order	int64	0	49600	1
line_item_id	int64	0	49600	1
product_id	int64	0	49600	52
quantity	int64	0	49600	1
line_item_amount	float64	0	49600	2.5
unit_price	float64	0	49600	2.5
promo_item_yn	object	0	49600	N

4. CÁLCULO DE LA VARIABLE OBJETIVO MEDIANTE UN ALGORITMO DE CLASIFICACIÓN

En esta sección se entrenó un modelo de machine learning para poder predecir el tipo de producto que corresponde a las diversas transacciones del dataset.

Para esto, en primer lugar, se utilizó un método de codificación para transformar las variables categóricas en numéricas. Luego se creó un solo dataset con las variables anteriores más las variables cuantitativas, para después someterlo una transformación de escalado para adecuar a todas las variables para el proceso de entrenamiento del modelo.

Finalmente, se utilizó un modelo de feature selection para reducir la dimensionalidad, se seleccionó y entrenó un algoritmo de clasificación, y finalmente se calcularon las métricas correspondientes para evaluar su desempeño.

4.1 DATA ENCODING

En el dataset se contaba con 9 variables cualitativas "nominales", de las cuales 7 ya estaban en formato numérico y dos tenían el formato no numérico "object". Estas eran "instore_yn" y "promo_item_yn" que contaban con dos posibles valores: "Y" y "N". Entonces optó por transformar el valor "Y" en el valor numérico "1", y el "N", en el valor numérico "0". Luego también se transformaron al formato “int64”

4.2 OPERACIONES CON LA VARIABLE FECHA

Luego de haber transformado las variables “transaction_date” y “transaction_time” al formato “datetime64[ns]”, se desglosaron en nuevas variables.

- transaction_date
  - year
  - month
  - day
- transaction_time
  - hour
  - minute
  - second

Luego se borraron “transaction_date” y “transaction_time” y se llevaron las restantes al formato “int64”. Entonces podemos decir que transformamos 2 variables de tipo fecha en 6 de tipo cualitativas numéricas.

4.3 TRANSFORMACIONES DE ESCALADO

En este punto se extrajo la variable objetivo del dataset formando dos conjuntos de datos “x” e “y”. Luego se sometió al escalado por medio de la librería “sklearn”, mediante la clase “MinMaxScaler”.

4.4 REDUCCIÓN DE LA DIMENSIONALIDAD

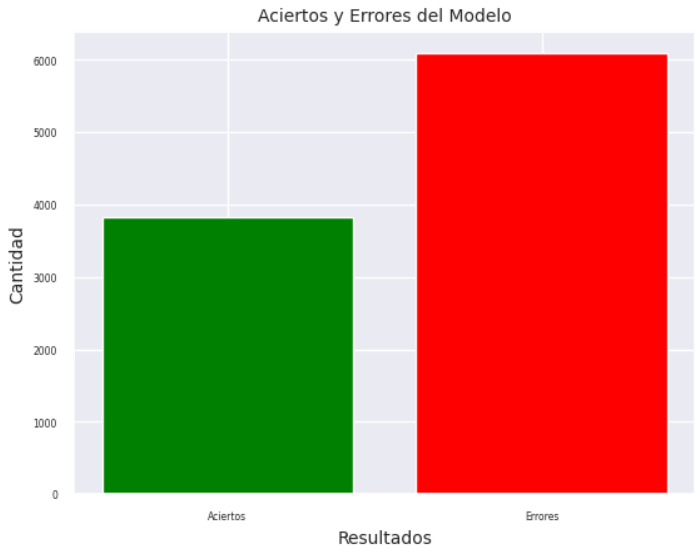
Para eliminar las variables que disminuyen el rendimiento de nuestro modelo de aprendizaje automático se optó por utilizar “SequentialFeatureSelector” con las siguientes características:

- el algoritmo "Random Forest" para el estimador, ya que buscamos un modelo de clasificación.
- El valor de 5 variables, dado que buscamos que se ejecute en un tiempo razonable.
- El método "Forward selection".
- verbose = 2
- La métrica "accuracy".
- "3" divisiones de entrenamiento para la validación cruzada utilizando k-fold.
- Y aplicamos "n_jobs = -1" es decir, se utilizarán todos los núcleos disponibles.

Cabe destacar que se intentó mejorar la métrica anterior ajustando los hiperparámetros mediante “GridSearchCV”, lo que no resulto positivo (bajó al 30%) principalmente en función de los recursos computacionales poseídos, invertidos y los resultados obtenidos. Nos quedamos con la primera iteración con una accuracy aproximada del 37%.

4.5 SELECCIÓN Y ENTRENAMIENTO DEL ALGORITMO DE CLASIFICACIÓN

En esta sección, partiendo de los resultados de la anterior, se optó por utilizar el algoritmo "Random Forest" dado que los clasificadores de bosques aleatorios son robustos y pueden manejar múltiples clases y son una buena opción cuando hay una gran cantidad de clases y datos. Además, se utilizó la métrica “precisión”. El resultado obtenido es un puntaje de accuracy de aproximadamente 0.386, lo que significa que el modelo de clasificación Random Forest fue capaz de predecir correctamente alrededor del 38.6% de las instancias en el conjunto de prueba. Esto sigue siendo un valor bajo de aciertos.



Finalmente se decidió comparar estos resultados con el dataset de 17 variables completo. De esto último se obtuvo una baja de del puntaje de la métrica del orden del 1,32% (en comparación), lo que puso de manifiesto el ahorro de recursos a la hora de utilizar la técnica de reducción de la dimensionalidad.

5 SEGUNDA RONDA DE FEATURE ENGINEERING

En esta ronda no solo se intentó aumentar el número de variables a utilizar, sino que también se buscó balancear las clases y reducir el número de instancias para mejorar las métricas anteriores.

5.1 APLICACIÓN DE LA LIBRERÍA "IMBALANCED-LEARN"

El procesamiento del dataset original, frente a los recursos computacionales poseídos, hace que los tiempos de operación sean realmente largos. Por esta razón, y sumado al hecho del balanceo de clases de la variable objetivo para el mejor entrenamiento del modelo, hacen que sea imperativo disminuir de una forma balanceada el número de instancias. Haciendo uso de la librería del título realizamos dos pasos:

- utilizo del algoritmo SMOTE (Generación sintética de muestras), que genera muestras sintéticas basadas en las muestras existentes de la clase minoritaria. Esto es con el fin de aumentar la cantidad de instancias de estas últimas, ya que existían valores de frecuencia de 50 en estas, comparadas con las clases mayoritarias que tenían alrededor de 1000. Esto es para evitar el sobreajuste y a mantener la diversidad.

- Utilizo de la técnica de submuestreo estratégico para reducir la cantidad de instancias del conjunto de datos.

Este proceso hizo que la cantidad de instancias aumentaran de 49.894 a 82.480, para finalmente disminuirlas a 10.005 totalmente balanceadas (115 valores por clase), en términos de la variable objetivo.

5.2 SEGUNDA RONDA DE REDUCCIÓN DE LA DIMENSIONALIDAD

Con los cambios realizados en la sección anterior se aplicó nuevamente una la técnica de reducción de dimensionalidad. En este caso, en vez de obtener 5 variables se intentó obtener 9. Nuevamente utilizamos “SequentialFeatureSelector”, pero con las siguientes características:

- el algoritmo "Random Forest" para el estimador.
- El valor de 9 variables.
- El método "Forward selection".
- verbose = 2
- La métrica "accuracy".
- "3" divisiones de entrenamiento para la validación cruzada utilizando k-fold.
- Y aplicamos "n_jobs = -1".

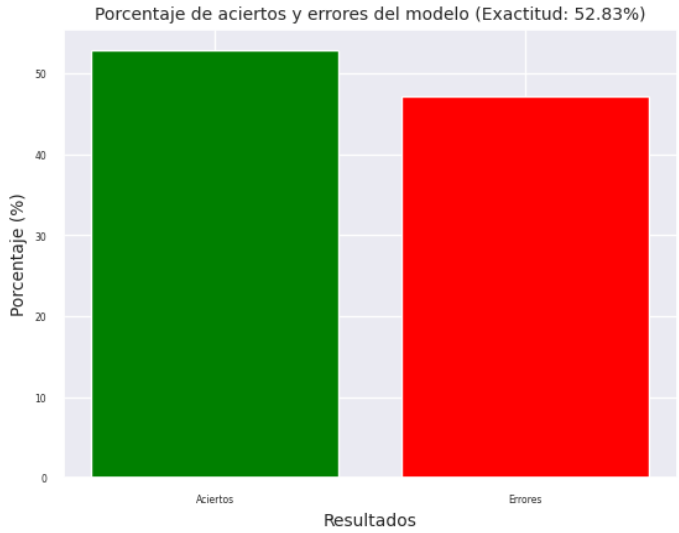
El resultado de esta nueva iteración fue el aumento de la métrica accuracy de 36,6% a 53,2%. Las nuevas variables son: “transaction_id”, “sales_outlet_id”, “staff_id”, “customer_id”, “line_item_id”, “line_item_amount”, “unit_price”, “minute” y “second”. Cabe destacar que posteriormente también se realizó una iteración con “GridSearchCV”, pero no otorgó resultados significativos (del orden de un 0,38%).

5.3 RENTRENAMIENTO DEL ALGORITMO DE CLASIFICACIÓN

El uso de los resultados anteriores en una nueva implementación del algoritmo “Random Forest” dio como resultado las siguientes métricas:

- Accuracy: 53%
- Precision: 0.54%
- Recall: 0.53%
- F1 Score: 0.53%

Como se observó anteriormente es un aumento significativo del puntaje inicial (de 45,31%). Es importante también, destacar que solo se esta trabajando con una fracción de las instancias originales.



6 INGENIERÍA DE ATRIBUTOS Y SELECCIÓN DE VARIABLES

En esta sección se crearon nuevas variables sintéticas para mejorar el desempeño del modelo. Además, se probaron varios tipos de algoritmos de clasificación para, en base a sus respectivas métricas, seleccionar el mejor. Finalmente se aplicó la técnica de reducción de la dimensionalidad PCA "Análisis de Componentes Principales", para encontrar las variables que más contribuyen a su mejoramiento.

6.1 CREACIÓN DE NUEVAS VARIABLES SINTÉTICAS

Con el motivo de capturar mejor la relación existente entre el tipo de producto y su precio unitario se creó una nueva variable. A partir de "unit_price" (precio unitario de cada artículo), creamos la variable "promedio_relativo". Esta representa la diferencia porcentual del precio unitario de cada producto en relación con el promedio total de la variable.

Producto de la creación de la variable surgieron 460 valores nulos que se procedió a eliminar dado que estos no se pueden entrenar por el modelo, y reemplazarlos por otro valor distorsionaría la su relación con el precio unitario.

6.2 PRUEBA DE DISTINTOS MODELOS

Con el fin de obtener un mejor equilibrio entre sesgo-varianza se probaron varios modelos en los cuatro tipos de métricas anteriores para determinar el que mejor se ajusta a la predicción de la variable objetivo. Los resultados de la prueba fueron los siguientes:

`Resultados del modelo 'Regresión Logística':`

- `Accuracy: 0.07208237986270023`
- `Precisión: 0.06297398985173319`
- `Recall: 0.07208237986270023`
- `F1 Score: 0.05654124861313248`

`Resultados del modelo 'Random Forest':`

- `Accuracy: 0.6287185354691075`
- `Precisión: 0.6317731588193634`
- `Recall: 0.6287185354691075`
- `F1 Score: 0.6265107334963944`

Resultados del modelo 'Gradient Boosting':

- Accuracy: 0.5949656750572082
- Precisión: 0.5975609027494508
- Recall: 0.5949656750572082
- F1 Score: 0.5937399858629356

Resultados del modelo 'SVM':

- Accuracy: 0.13272311212814644
- Precisión: 0.13656162153603943
- Recall: 0.13272311212814644
- F1 Score: 0.12129329566562455

En el resultado anterior vemos que:

- El modelo de Regresión Logística muestra un bajo rendimiento en todas las métricas. La precisión, recall y F1 Score son significativamente bajos, lo que sugiere que el modelo tiene dificultades para clasificar correctamente las muestras. Esto podría indicar un alto sesgo (bias) en el modelo, es decir, el modelo no es lo suficientemente complejo para capturar la relación entre las características de entrada y la variable objetivo.
- El modelo Random Forest muestra un rendimiento considerablemente mejor en todas las métricas en comparación con la Regresión Logística. Tiene una precisión, recall y F1 Score más altos, lo que sugiere que es capaz de clasificar las muestras con mayor precisión. Esto podría indicar un mejor equilibrio entre sesgo y varianza en comparación con la Regresión Logística.
- El modelo Gradient Boosting también muestra un rendimiento sólido en todas las métricas, similar al Random Forest. Tiene una precisión, recall y F1 Score altos, lo que indica que es capaz de clasificar las muestras de manera efectiva. En términos de sesgo-varianza, podría estar en un nivel similar al Random Forest.
- El modelo SVM muestra un rendimiento más bajo en comparación con los otros dos modelos (Random Forest y Gradient Boosting). Tiene una precisión, recall y F1 Score significativamente más bajos. Esto podría sugerir un alto sesgo en el modelo o que el modelo no es adecuado para este problema específico.

En conclusión, en términos del balance "bias-variance tradeoff" parece ser que los algoritmos Random Forest y Gradient Boosting muestran un mejor equilibrio entre sesgo y varianza, con un rendimiento más sólido en todas las métricas, mientras que la SVM parece tener un sesgo más alto en este contexto debido a su bajo rendimiento. Por este motivo vamos a seguir utilizando "Random Forest". Además queda de manifiesto que el agregado de la variable sintética “promedio_relativo” ha tenido un impacto positivo, resultando en un aumento de la métrica accuracy de aproximadamente un 53% a 63%.

6.3 ANÁLISIS DE COMPONENTES PRINCIPALES (PCA)

Para el conjunto de datos resultante de la sección anterior se realizó la técnica de reducción de la dimensionalidad PCA. Luego, el resultado obtenido se utilizó para entrenar un algoritmo del tipo “Random Forest”. Las métricas resultantes fueron:

- Accuracy: 0.02745995423340961
- Precisión: 0.013861080715059022
- Recall: 0.02745995423340961
- F1 Score: 0.010713624690560135

Los resultados indican que nuestro modelo de Random Forest con PCA y 5 componentes principales no funciona bien en nuestro conjunto de datos. Tiene dificultades para realizar predicciones precisas y tiende a clasificar incorrectamente la mayoría de las muestras.

Esto podría deberse a la reducción drástica de la dimensionalidad, que puede haber causado la pérdida de información crucial para la clasificación. Podríamos considerar ajustar el número de componentes principales o explorar otras técnicas de reducción de dimensionalidad para mejorar el rendimiento del modelo en futuros experimentos.

7 CROSSVALIDATION E HYPERTUNING

En esta sección se intentó nuevamente perfeccionar el valor de las métricas arrojadas por el modelo. Para tal fin, esta vez se aplicaron dos herramientas

- El método de validación cruzada
- Técnicas de mejoramiento de hiperparámetros

Se destaca que se vuelve a utilizar el algoritmo de clasificación "Random Forest" dado que con este se consiguió el mejor conjunto de métricas para nuestro problema de Machine Learning.

7.1 MÉTODO "STRATIFIED K-FOLD CROSS VALIDATION"

En nuestro caso particular la estrategia seleccionada para Cross Validation fue "Stratified K-Fold Cross Validation". La razón principal para elegir esta técnica es que asegura que las proporciones de las 87 clases de "product_id" sean aproximadamente las mismas en cada fold. Dadas las múltiples clases y fue importante que el modelo se entrenara y evalúe de manera equitativa en todas ellas, dado que la estratificación ayuda a evitar desequilibrios en la distribución de clases en los conjuntos de entrenamiento y prueba. Utilizando un número de 10 folds los resultados fueron:

- `Accuracy promedio en 10 folds: 0.6005405681585939`
- `Precisión promedio en 10 folds: 0.6020988005032241`
- `Recall promedio en 10 folds: 0.6005405681585939`
- `F1-score promedio en 10 folds: 0.5973633486983798`

Podemos ver que mediante el uso de Stratified K-Fold Cross Validation, obtuvimos resultados ligeramente inferiores en términos de métricas promedio en comparación con el modelo utilizado en la sección "6.2", sin validación cruzada. Esto se debe a que la validación cruzada realiza un proceso de partición de datos en múltiples conjuntos de entrenamiento y prueba, lo que puede llevar a una ligera reducción en el rendimiento promedio en comparación con entrenar y evaluar el modelo en todo el conjunto de datos.

7.2 HYPERTUNING PARAMETERS

Para la optimización de hiper parámetros en nuestro modelo se utilizaron los conjuntos de datos de la sección 6.2, sin la aplicación de la técnica “Stratified K-Fold Cross Validation” porque esta disminuía el puntaje de las métricas.

En dicha sección, implementamos un modelo de clasificación utilizando Random Forest sin ajuste de hiperparámetros, lo que nos proporcionó resultados iniciales decentes en términos de métricas de rendimiento. Los resultados originales del modelo 'Random Forest' fueron los siguientes:

- Accuracy: 0.6293
- Precisión: 0.6345
- Recall: 0.6293
- F1 Score: 0.6297

En la presente sección, considerando los métodos de optimización de hiperparámetros, se entendió que aún había margen para mejorar el rendimiento de nuestro modelo. Por lo tanto, aplicamos Randomized Search para encontrar la combinación óptima de hiperparámetros para nuestro modelo. Después de la búsqueda aleatorizada, obtuvimos los siguientes resultados de los hiperparámetros:

- n_estimators': 300
- 'min_samples_split': 2
- 'min_samples_leaf': 2
- 'max_depth': None

Luego, los aplicamos en el modelo "best_rf_model", de esta forma las métricas fueron:

- Accuracy en el conjunto de prueba: 0.6138
- Precisión en el conjunto de prueba: 0.6211
- Recall en el conjunto de prueba: 0.6138
- F1-score en el conjunto de prueba: 0.6122

Al comparar estos resultados con nuestros resultados originales, observamos que el modelo ajustado con Randomized Search muestra una ligera disminución en algunas métricas, como el accuracy, la precisión y el recall, en comparación con el modelo inicial. Sin embargo, la diferencia en el rendimiento es mínima, lo que indica que nuestro modelo original ya estaba bastante bien ajustado.

Es importante destacar que, aunque no logramos una mejora significativa en el rendimiento, la ventaja de aplicar Randomized Search radica en que hemos explorado sistemáticamente un rango amplio de hiperparámetros en busca de mejoras, lo que podría ser beneficioso en casos más complejos, y constituye un modelo más sólido. Además, hemos obtenido una configuración de hiperparámetros más óptima para nuestro modelo que puede ser útil en situaciones futuras.

8. CONCLUSIÓN FINAL

En resumen, el proyecto logró desarrollar un modelo de machine learning basado en "Random Forest" para predecir el tipo de producto en las transacciones comerciales. Sin embargo, a pesar de los esfuerzos de ingeniería de características y optimización, las métricas de rendimiento del modelo (accuracy, precisión, recall, F1-score) siguieron siendo moderadas. Esto podría deberse a 3 factores fundamentalmente: la complejidad del problema, la falta de datos más descriptivos y además que se cuenta con recursos computacionales escasos para el procesamiento y búsqueda de los anteriores.

En futuros trabajos, se podría explorar la posibilidad de obtener datos adicionales o aplicar técnicas más avanzadas de modelado, como redes neuronales, para mejorar el rendimiento del modelo. Además, se recomienda evaluar la posibilidad de crear más variables sintéticas, y repetir los procesos de validación cruzada y la búsqueda de hiperparámetros, dado que son pasos esenciales para garantizar que el modelo sea lo más robusto posible en un contexto de machine learning.