



Universidad Internacional San Isidro Labrador

Programa de Ciencia de Datos

Proyecto Final - Parte II

**Desarrollo de un Modelo Predictivo para Prevenir la Deserción de Clientes en
una Institución Bancaria Costarricense Mediante Inteligencia Artificial**

Profesor: Dr. Samuel Saldaña Valenzuela

Estudiante: Byron Bolaños Zamora

Fecha de entrega: 25 de noviembre del 2024

2024

Índice

Introducción	3
Objetivos	4
Marco Teórico	5 - 54
Conclusiones	55 - 57
Recomendaciones	58 - 60
Bibliografía	61 - 62

Introducción

En esta fase, se procede con el desarrollo de un modelo predictivo aplicando el uso de Inteligencia Artificial para anticipar la deserción de clientes. Utilizando el conjunto de datos depurado en la etapa anterior, se implementan enfoques tanto supervisados como no supervisados para identificar patrones y características clave que influyen en la decisión de abandono de los clientes. La aplicación de técnicas como el Análisis de Componentes Principales (PCA) y el clustering con K-Means permite una segmentación precisa de los clientes, mientras que el empleo de modelos supervisados, tales como Gradient Boosting, asegura predicciones precisas sobre la probabilidad de deserción. El PCA es fundamental para reducir la dimensionalidad de los datos, permitiendo extraer las características más informativas y minimizar problemas como la multicolinealidad y el sobreajuste, lo que resulta en un modelo más manejable y efectivo. Por otro lado, el clustering con K-Means ayuda a agrupar observaciones similares, facilitando la identificación de patrones ocultos en los datos. Esta fase incluye la selección y ajuste de los modelos para optimizar su rendimiento y asegurar la máxima precisión en las predicciones.

El resultado de este proceso es un sistema predictivo robusto que proporciona información valiosa para la toma de decisiones estratégicas. Al integrar este modelo en la gestión de clientes, la institución bancaria puede implementar medidas proactivas de retención, reduciendo costos asociados con la adquisición de nuevos clientes, mejorando la fidelización y, en última instancia, fortaleciendo su posición competitiva en el mercado. Como se menciona en estudios previos, "la reducción de dimensionalidad a través del PCA no solo simplifica el análisis, sino que también mejora significativamente el rendimiento del modelo" (IBM, 2023).

Objetivos

Objetivo General:

Desarrollar e implementar un modelo predictivo basado en técnicas de Inteligencia Artificial para anticipar la deserción de clientes en una institución bancaria, utilizando enfoques supervisados y no supervisados, con el fin de proporcionar información estratégica que permita optimizar las medidas de retención y mejorar la fidelización de clientes.

Objetivos Específicos:

1. Identificar las características y patrones clave que influyen en la deserción de clientes mediante el análisis y segmentación de datos utilizando técnicas de machine learning.
2. Implementar modelos predictivos supervisados que garanticen precisión y confiabilidad en la estimación de la probabilidad de deserción.
3. Proporcionar un sistema de apoyo a la toma de decisiones estratégicas, integrando el modelo predictivo para facilitar la planificación de medidas proactivas de retención y fidelización de clientes.

Marco Teórico

En el entorno competitivo actual, las organizaciones enfrentan un desafío constante: gestionar y analizar datos de manera efectiva para convertir esta capacidad en un diferenciador estratégico clave. La información ha pasado de ser un recurso pasivo a convertirse en un activo dinámico y valioso, fundamental para la toma de decisiones informadas y la generación de ventajas competitivas. Sin embargo, el valor de los datos no radica únicamente en su volumen, sino en la capacidad de procesarlos y convertirlos en insights accionables (Davenport & Harris, 2007).

El ciclo de vida de los datos proporciona un marco estructurado para maximizar el valor derivado de los datos, asegurando que cada etapa, desde la recopilación hasta la implementación, se ejecute de manera óptima. Este ciclo no solo establece las bases técnicas para el análisis de datos, sino que también permite alinear los procesos analíticos con los objetivos estratégicos de la organización (Redman, 2013). Dicho ciclo, será explicado a mayor detalle, más adelante en la presente investigación.

En este contexto, el presente estudio se enfoca en la etapa de modelado, una fase crucial donde los datos procesados y analizados se transforman en herramientas predictivas capaces de informar decisiones estratégicas. Para comprender mejor cómo se llega a esta fase, es fundamental explorar detalladamente el ciclo de vida de los datos, el cual abarca varias etapas esenciales que interaccionan entre sí.

Ciclo de Vida de los Datos

El ciclo de vida de los datos es un proceso continuo que consta de diversas fases, cada una con un rol esencial para garantizar que los datos sean correctamente

gestionados, analizados y transformados en valor. Estas fases se interconectan para proporcionar una estructura organizada y eficiente en la que los datos se convierten en un recurso estratégico para la toma de decisiones (Chen, Chiang, & Storey, 2012). El ciclo generalmente se divide en las siguientes etapas:

1. Recopilación de Datos

La recopilación de datos es la primera etapa del ciclo de vida y se refiere al proceso de capturar información relevante desde diversas fuentes, como bases de datos internas, redes sociales, dispositivos IoT, encuestas, entre otras. La calidad y la cantidad de los datos recogidos son fundamentales, ya que de ello depende el éxito de las etapas posteriores. Por ejemplo, en un proyecto de deserción de clientes, la recopilación puede incluir información sobre transacciones, interacciones con el servicio al cliente y datos demográficos de los usuarios (Hazen et al., 2014).

2. Almacenamiento de Datos

Una vez recopilados, los datos se almacenan en sistemas diseñados para manejarlos de manera eficiente. En este paso, se eligen bases de datos y plataformas de almacenamiento que permitan organizar y gestionar grandes volúmenes de información de manera segura y accesible. Las soluciones de almacenamiento incluyen bases de datos relacionales y no relacionales, así como soluciones en la nube, como AWS, Azure o Google Cloud (Gualtieri, 2019).

3. Procesamiento de Datos

El procesamiento de datos implica limpiar, transformar y normalizar los datos para que sean adecuados para el análisis posterior. Este paso puede incluir la eliminación de datos faltantes, la corrección de errores y la transformación

de variables, como la conversión de valores categóricos en variables numéricas. También puede implicar la creación de nuevas variables a partir de las existentes, lo que aumenta la riqueza del conjunto de datos (Loshin, 2013).

4. Análisis de Datos

El análisis de datos es la fase donde se extraen patrones y relaciones que pueden proporcionar insights valiosos. Durante esta etapa, los científicos de datos realizan análisis exploratorios y estadísticos, identificando correlaciones, tendencias y comportamientos. El uso de técnicas como análisis descriptivo, inferencial y correlacional ayuda a comprender mejor los datos antes de pasar a modelos predictivos más complejos (Shmueli, Bruce, & Gedeck, 2017).

5. Modelado

Esta es la fase en la que los datos procesados y analizados se utilizan para crear modelos predictivos. Durante el modelado, se aplican algoritmos de machine learning supervisados y no supervisados que permiten realizar predicciones sobre el comportamiento futuro de los clientes, como en el caso de la deserción de clientes. Esta etapa es crucial porque transforma los datos en resultados prácticos que pueden influir directamente en las decisiones empresariales (Bishop, 2006). Dicha etapa es el núcleo de la presente investigación, por lo cuál, será desarrollada la brevedad de detalle posteriormente.

6. Implementación

Una vez que el modelo ha sido desarrollado, ajustado y validado, se implementa en el entorno productivo. Esto implica integrarlo en los sistemas

operacionales de la empresa, lo que puede incluir la automatización de procesos basados en las predicciones generadas por el modelo. Por ejemplo, un modelo predictivo de deserción podría ser utilizado por el equipo de marketing para crear campañas de retención personalizadas, reduciendo así la probabilidad de que los clientes abandonen (Chien, 2014).

7. Monitoreo y Mantenimiento

El ciclo de vida de los datos no termina con la implementación. Los modelos deben ser monitoreados y mantenidos para garantizar su eficacia a lo largo del tiempo. A medida que cambian los patrones de comportamiento de los clientes o surgen nuevas fuentes de datos, es posible que los modelos necesiten ajustes o reentrenamientos. Esto asegura que el modelo siga siendo relevante y preciso frente a nuevas realidades del negocio (Müller & Guido, 2017).

Etapas de Análisis Exploratorio de Datos (EDA)

El Análisis Exploratorio de Datos (EDA) es una etapa esencial dentro del ciclo de vida de los datos que precede al modelado. Introducido por el estadístico John Tukey en la década de 1970, el EDA tiene como objetivo descubrir patrones, relaciones y anomalías en los datos, proporcionando un entendimiento profundo que informa las decisiones durante el modelado (Tukey, 1977).

Objetivos de aplicar el EDA

- Identificación de errores: Detectar valores nulos, duplicados o inconsistentes en los datos, garantizando su calidad.

- Detección de valores atípicos: Reconocer observaciones extremas que podrían distorsionar el análisis.
- Comprensión de relaciones: Analizar correlaciones entre variables para identificar dependencias o redundancias.
- Transformaciones necesarias: Evaluar la necesidad de normalizar, escalar o categorizar variables para facilitar su uso en modelos predictivos.

Por ejemplo, en el caso de la deserción de clientes, el EDA podría revelar que las tasas de abandono son significativamente mayores en ciertos rangos de ingresos o entre clientes que realizan pocas transacciones mensuales. Este conocimiento puede ser clave para orientar la selección de variables durante el modelado.

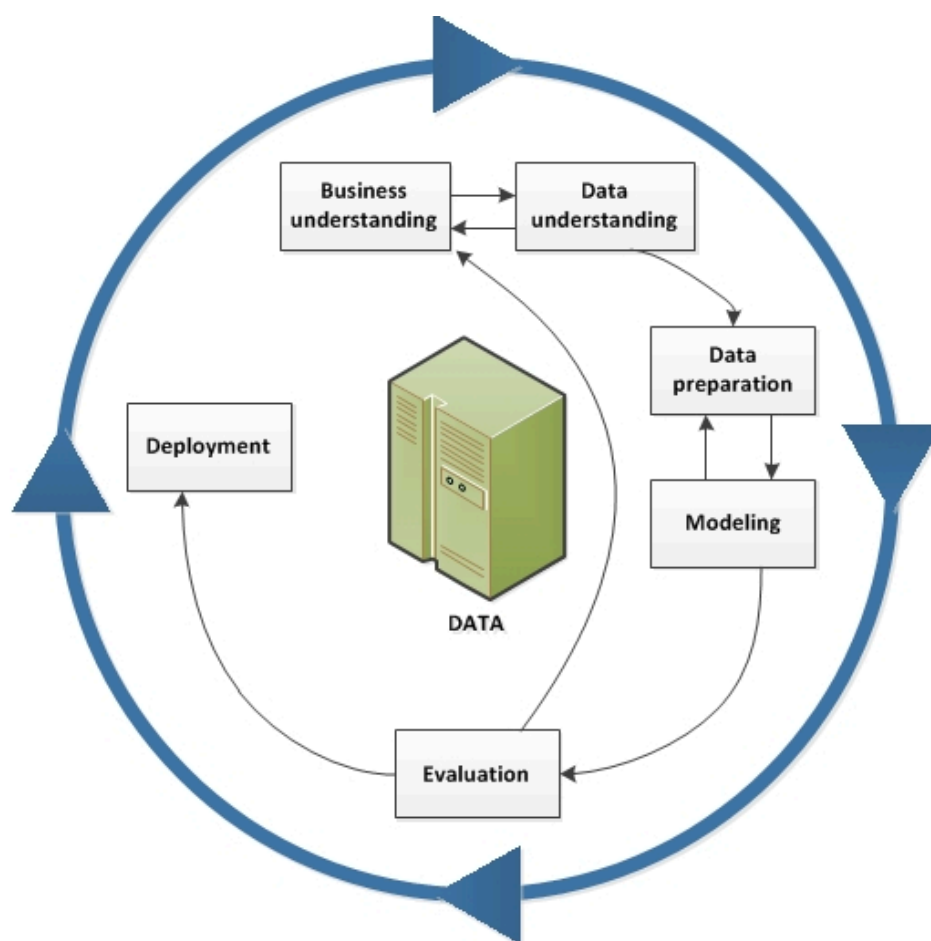


Figura 1: Ciclo de vida de los datos. Tomada de: <https://www.ibm.com>

Modelado Predictivo

El modelado predictivo es una de las etapas más críticas dentro del ciclo de vida de los datos, ya que es el punto donde la información procesada se transforma en herramientas predictivas que permiten tomar decisiones estratégicas fundamentadas en datos. En el contexto actual, donde la competencia se define en gran medida por la capacidad de anticiparse a las necesidades y comportamientos de los clientes, el modelado se convierte en un diferenciador clave para las organizaciones. Utilizando algoritmos avanzados, los modelos predictivos permiten prever eventos futuros, como la deserción de clientes, el comportamiento de compra o incluso la probabilidad de fallos en sistemas, entre otros. Este proceso involucra técnicas complejas tanto de aprendizaje supervisado como no supervisado, que van desde la segmentación de clientes hasta la predicción exacta de comportamientos.

Conceptos Fundamentales en el Modelado Predictivo

Algoritmos Supervisados y No Supervisados

Los algoritmos de aprendizaje supervisado son aquellos en los que el modelo se entrena con un conjunto de datos que incluye tanto las características de entrada (variables independientes) como la variable objetivo (dependiente), es decir, el valor que se quiere predecir. Por ejemplo, en el caso de la deserción de clientes, la variable objetivo puede ser la probabilidad de que un cliente abandone la institución bancaria, mientras que las características de entrada pueden incluir variables como la frecuencia de transacciones, la antigüedad del cliente, el tipo de productos contratados, entre otras.

Por otro lado, los algoritmos no supervisados no requieren de una variable objetivo. Estos modelos buscan identificar patrones o estructuras ocultas en los datos sin supervisión externa. Un ejemplo típico de un algoritmo no supervisado es el clustering, donde se segmentan los datos en grupos homogéneos basados en similitudes. Estos modelos son útiles cuando el objetivo es explorar los datos y segmentarlos antes de proceder a la predicción.

Métricas de Evaluación del Modelo

Las métricas de evaluación son cruciales para medir el desempeño de un modelo predictivo. Existen diferentes métricas, que varían según el tipo de problema (clasificación, regresión, etc.). Entre las más comunes, se destacan:

- Precisión: Mide la proporción de predicciones correctas sobre el total de predicciones realizadas. Es útil en escenarios donde los costos de los errores son altos.
- Recall (Sensibilidad): Indica la capacidad del modelo para identificar todas las instancias relevantes (positivas) dentro del conjunto de datos. Es esencial cuando la prioridad es no pasar por alto un evento importante, como la deserción de un cliente.
- F1-Score: Es la media armónica entre precisión y recall, combinando ambas métricas en un solo valor, que es especialmente útil cuando se busca un balance entre ambas.
- AUC-ROC: El área bajo la curva (AUC) de la curva característica de operación del receptor (ROC) es una medida de la capacidad del modelo para discriminar entre clases. Un valor de AUC cercano a 1 indica que el modelo tiene una alta capacidad para distinguir entre clases.

- Matriz de Confusión: Proporciona un resumen de las predicciones del modelo, mostrando tanto las instancias verdaderas como las predicciones incorrectas (falsos positivos y falsos negativos).

Técnicas de Modelado Predictivo

En el contexto de esta investigación, se utilizarán varias técnicas avanzadas para abordar la predicción de la deserción de clientes, incluyendo PCA (Análisis de Componentes Principales), K-Means Clustering y Gradient Boosting. A continuación, se detallan estas técnicas clave:

Análisis de Componentes Principales (PCA)

El Análisis de Componentes Principales (PCA) es una técnica de reducción de dimensionalidad que transforma un conjunto de variables posiblemente correlacionadas en un nuevo conjunto de variables no correlacionadas denominadas componentes principales. El principal objetivo de PCA es reducir la cantidad de variables manteniendo la mayor cantidad de información posible. Este proceso es especialmente útil cuando se trabaja con grandes volúmenes de datos con muchas variables, ya que permite simplificar el modelo, reducir el riesgo de sobreajuste y mejorar la interpretabilidad de los resultados.

PCA funciona al identificar las direcciones (componentes) en las que los datos varían más y luego proyecta los datos originales sobre estos componentes. Este proceso facilita la visualización y el análisis de datos complejos, especialmente en el contexto de datos de alta dimensionalidad, donde las relaciones entre variables pueden ser difíciles de identificar.

Por ejemplo, en un proyecto de predicción de deserción de clientes, PCA podría utilizarse para identificar las características más relevantes que influyen en la decisión de abandono, eliminando redundancias y permitiendo que el modelo se enfoque en las variables de mayor impacto. Esto es particularmente útil cuando se dispone de un conjunto de datos con muchas características, como historial de transacciones, interacciones con el servicio al cliente y datos demográficos, entre otros.

K-Means Clustering

K-Means es un algoritmo no supervisado de clustering que segmenta los datos en grupos (o clústeres) basados en la similitud de las características de los datos. El algoritmo divide el conjunto de datos en k grupos, donde k es un valor previamente especificado. La idea básica es asignar cada punto de datos al grupo cuyo centroide (promedio de las características de todos los puntos en el grupo) sea el más cercano.

El proceso de K-Means involucra los siguientes pasos:

1. Inicialización: Se eligen aleatoriamente k puntos de datos como los centroides iniciales de cada clúster.
2. Asignación: Cada punto de datos se asigna al clúster cuyo centroide está más cercano.
3. Actualización: Se recalculan los centroides basándose en los puntos asignados a cada clúster.

4. Repetición: Los pasos de asignación y actualización se repiten hasta que los centroides ya no cambian significativamente, lo que indica que el algoritmo ha convergido.

En el contexto de la deserción de clientes, el uso de K-Means puede ayudar a identificar diferentes segmentos de clientes con comportamientos similares. Por ejemplo, se pueden identificar grupos de clientes con alta lealtad, aquellos que están en riesgo de abandonar o aquellos que no interactúan frecuentemente con los servicios de la institución. Esta segmentación puede ayudar a personalizar las estrategias de retención y mejorar la fidelización.

Gradient Boosting

El Gradient Boosting es una técnica de aprendizaje supervisado que se utiliza para mejorar la precisión de los modelos mediante un enfoque secuencial en el que se construyen modelos simples (típicamente árboles de decisión) de manera iterativa. Cada modelo sucesivo corrige los errores cometidos por el modelo anterior, optimizando progresivamente el rendimiento del conjunto de modelos.

El proceso de Gradient Boosting se basa en el principio del boosting, en el que cada nuevo árbol de decisión es entrenado para predecir los residuos o errores de los árboles anteriores. Este enfoque se implementa ajustando los pesos de los ejemplos mal clasificados en cada iteración. Al final, el modelo final es la combinación de todos los árboles construidos.

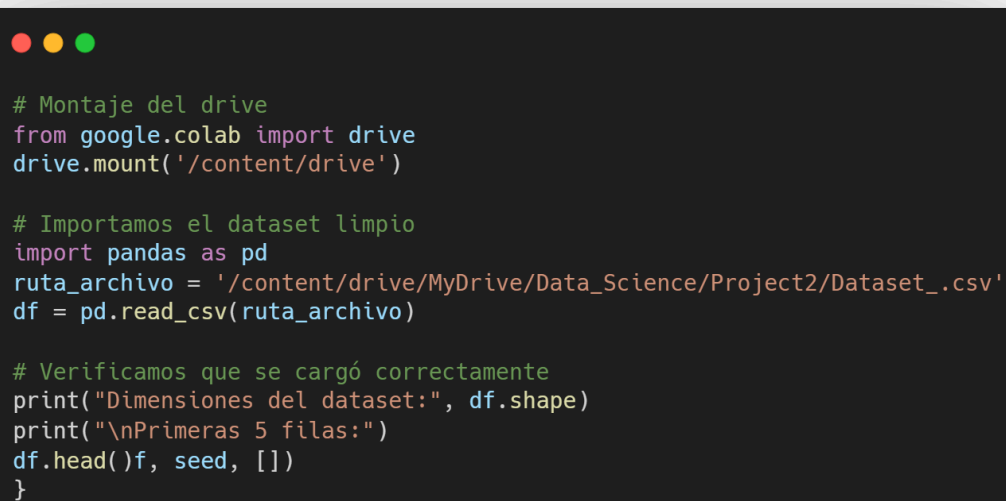
Una de las principales ventajas del Gradient Boosting es su capacidad para manejar relaciones no lineales entre las variables, lo que lo convierte en una herramienta poderosa para problemas como la deserción de clientes, donde las interacciones

entre las variables pueden ser complejas y no seguir una estructura lineal simple. Además, el Gradient Boosting es altamente flexible y se puede ajustar con varios parámetros, como la profundidad del árbol, el número de estimadores y la tasa de aprendizaje, para evitar el sobreajuste y mejorar la generalización del modelo.

XGBoost y LightGBM son implementaciones populares de Gradient Boosting, que han demostrado un rendimiento superior en muchas competiciones de machine learning debido a su eficiencia y capacidad para manejar grandes volúmenes de datos, por mencionar otros ejemplos.

A continuación, se presenta el código implementado para desarrollar y evaluar el modelo predictivo de deserción de clientes. Este código se organiza en varias secciones clave que incluyen la preparación y escalado de datos, la aplicación de técnicas no supervisadas como el Análisis de Componentes Principales (PCA) y K-Means para la reducción de dimensionalidad y segmentación, y la implementación de un modelo supervisado basado en el algoritmo Gradient Boosting para la predicción de deserción. A lo largo de este proceso, se han integrado métricas clave de evaluación como la precisión, la matriz de confusión, la curva ROC y la importancia de las características, que son esenciales para comprender el rendimiento del modelo y su capacidad para generalizar en datos no vistos.

En las siguientes secciones se explicará el flujo del código, destacando cada uno de los pasos ejecutados y los resultados obtenidos, los cuales permiten evaluar la efectividad del modelo en el contexto de la deserción de clientes.



```
# Montaje del drive
from google.colab import drive
drive.mount('/content/drive')

# Importamos el dataset limpio
import pandas as pd
ruta_archivo = '/content/drive/MyDrive/Data_Science/Project2/Dataset_.csv'
df = pd.read_csv(ruta_archivo)

# Verificamos que se cargó correctamente
print("Dimensiones del dataset:", df.shape)
print("\nPrimeras 5 filas:")
df.head(f, seed, [])
}
```

Figura 2: Montaje e importación del dataset. Elaboración propia.

El código de la figura 2, comienza montando Google Drive en el entorno de ejecución de Google Colab para acceder a los archivos almacenados en la nube. Se importa la librería `pandas`, que es fundamental para la manipulación y análisis de datos. A continuación, se define la ruta del archivo CSV que contiene el conjunto de datos limpio, el cual se carga en un DataFrame mediante la función `pd.read_csv()`. Posteriormente, se verifica que el archivo se haya cargado correctamente mediante la impresión de las dimensiones del conjunto de datos (número de filas y columnas) y las primeras cinco filas con el método `head()`. Esto permite al usuario validar rápidamente que el dataset se ha cargado sin problemas y obtener una visión preliminar de su estructura, la cuál se ejemplifica a continuación:


```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Dimensiones del dataset: (9342, 12)

Primeras 5 filas:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Geography_Germany	Geography_Spain	Gender_Male
0	-0.326221	0.293517	-1.041760	-1.225848	1	1	1	0.021886	1	False	False	False
1	-0.440036	0.198164	-1.387538	0.117350	1	0	1	0.216534	0	False	True	False
2	-1.536794	0.293517	1.032908	1.333053	3	1	0	0.240687	1	False	False	False
3	0.501521	0.007457	-1.387538	-1.225848	2	0	0	-0.108918	0	False	False	False
4	2.063884	0.388871	-1.041760	0.785728	1	1	1	-0.365276	0	False	True	False

Figura 3: Verificación de importación. Elaboración propia.

```
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.pipeline import make_pipeline
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, roc_auc_score
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
```

Figura 4: Importación de librerías. Elaboración propia.

En la figura 4, en el código se importan varias librerías esenciales para el desarrollo del modelo predictivo. Primero, **PCA** (Análisis de Componentes Principales) y **KMeans** (un algoritmo de clustering) de **sklearn.decomposition** y **sklearn.cluster** respectivamente, se utilizan para realizar reducción de dimensionalidad y agrupamiento de los datos. La librería **StandardScaler** de **sklearn.preprocessing** se emplea para estandarizar los datos antes de aplicar técnicas de modelado. A continuación, **GradientBoostingClassifier** se importa desde **sklearn.ensemble** para entrenar un modelo de clasificación basado en el algoritmo de boosting. Se incluye también **make_pipeline** para crear

un pipeline que automatiza el flujo de trabajo del modelo, desde la estandarización de los datos hasta el entrenamiento del clasificador. Para evaluar el rendimiento del modelo, se utilizan métricas de `sklearn.metrics` como `classification_report`, `confusion_matrix`, `roc_curve`, y `roc_auc_score`. Además, `matplotlib.pyplot` y `seaborn` se utilizan para crear gráficos visuales, mientras que `numpy` y `pandas` son librerías fundamentales para el manejo y análisis de datos, y `train_test_split` de `sklearn.model_selection` se usa para dividir el dataset en conjuntos de entrenamiento y prueba.



Figura 5: División del dataset. Elaboración propia.

En la figura 5, el código divide el dataset en dos conjuntos: uno para las características (X) y otro para las etiquetas (y). La variable `X` contiene todas las columnas del dataframe `df` excepto la columna `'Exited'`, que es la variable objetivo, es decir, la que queremos predecir (si un cliente abandona o no). Para ello, se utiliza el método `drop(columns=['Exited'])`, que elimina la columna `'Exited'` de `df`. Por otro lado, la variable `y` se asigna a la columna `'Exited'`, que contiene las etiquetas o resultados que el modelo debe predecir. De esta

manera, **X** se utiliza como las características de entrada para el modelo, mientras que **y** es la salida o el objetivo que el modelo tratará de predecir.

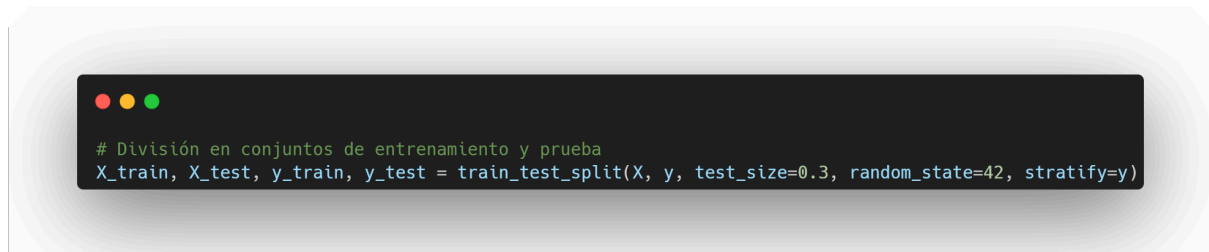


Figura 6: División dataset. Elaboración propia.

Acá en la figura 6, se realiza la división del dataset en dos conjuntos: uno para entrenamiento y otro para prueba, utilizando la función `train_test_split` de la librería `sklearn.model_selection`. En primer lugar, se separan las características del dataset (**X**) de las etiquetas (**y**), donde **X** contiene las variables predictoras y **y** la variable objetivo, en este caso, la columna **Exited**, que indica si un cliente abandonó o no. Posteriormente, se especifica que el 30% de los datos se destinarán al conjunto de prueba (`test_size=0.3`), mientras que el 70% restante se utilizará para entrenar el modelo. El parámetro `random_state=42` se establece para garantizar que la división sea reproducible, es decir, que se obtenga la misma partición en cada ejecución del código. Además, el parámetro `stratify=y` asegura que la distribución de las clases de la variable objetivo (**Exited**) sea proporcional en ambos conjuntos, evitando que alguna clase esté subrepresentada en el conjunto de prueba. Como resultado, se obtienen cuatro subconjuntos: **X_train**, **X_test**, **y_train** y **y_test**, que se utilizarán para entrenar y evaluar el modelo, respectivamente.

```

# --- Técnicas de aprendizaje no supervisado ---
# Estandarización de los datos para PCA y K-Means
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_train)

# PCA: Análisis de Componentes Principales
pca = PCA(n_components=2, random_state=42)
pca_components = pca.fit_transform(X_scaled)

# Visualización de PCA
plt.figure(figsize=(8, 6))
plt.scatter(pca_components[:, 0], pca_components[:, 1], c=y_train, cmap='viridis', alpha=0.7)
plt.colorbar(label='Etiqueta (Exited)')
plt.title('PCA - Análisis de Componentes Principales')
plt.xlabel('Componente Principal 1')
plt.ylabel('Componente Principal 2')
plt.show()

```

Figura 7: PCA. Elaboración propia.

En este bloque de código se implementan dos técnicas de aprendizaje no supervisado: la estandarización de los datos y el Análisis de Componentes Principales (PCA). Primero, se utiliza `StandardScaler` de `sklearn.preprocessing` para estandarizar los datos de entrenamiento, `X_train`. La estandarización es un paso crucial cuando se trabajan con técnicas como PCA o K-Means, ya que estas son sensibles a la escala de las variables. La estandarización asegura que todas las características tengan una media de 0 y una desviación estándar de 1, lo que previene que variables con rangos de magnitudes más grandes dominen el análisis. Posteriormente, se aplica el PCA, un método de reducción dimensional que transforma el conjunto de datos original en un nuevo espacio de componentes ortogonales, con el objetivo de simplificar el modelo manteniendo la mayor parte posible de la varianza de los datos originales. En este caso, se han seleccionado dos componentes principales (`n_components=2`) para visualizar los datos en un espacio bidimensional. El modelo PCA se ajusta y

transforma los datos estandarizados, produciendo los componentes principales, los cuales son almacenados en la variable `pca_components`.

Finalmente, se visualiza el resultado del PCA mediante un gráfico de dispersión (scatter plot) utilizando `matplotlib`. En el gráfico, cada punto representa una observación del conjunto de datos transformado, y los colores de los puntos están determinados por la etiqueta de la variable objetivo (`Exited`), lo que permite observar cómo se distribuyen las clases en el nuevo espacio de componentes principales. La visualización se personaliza con un título, etiquetas en los ejes y una barra de color que indica la correspondencia entre los colores y las etiquetas. Esta representación gráfica ayuda a entender cómo las diferentes clases están distribuidas en el espacio reducido de dimensiones, lo cual puede ser útil para posteriores análisis y modelados.

El resultado del presente código es la siguiente gráfica:

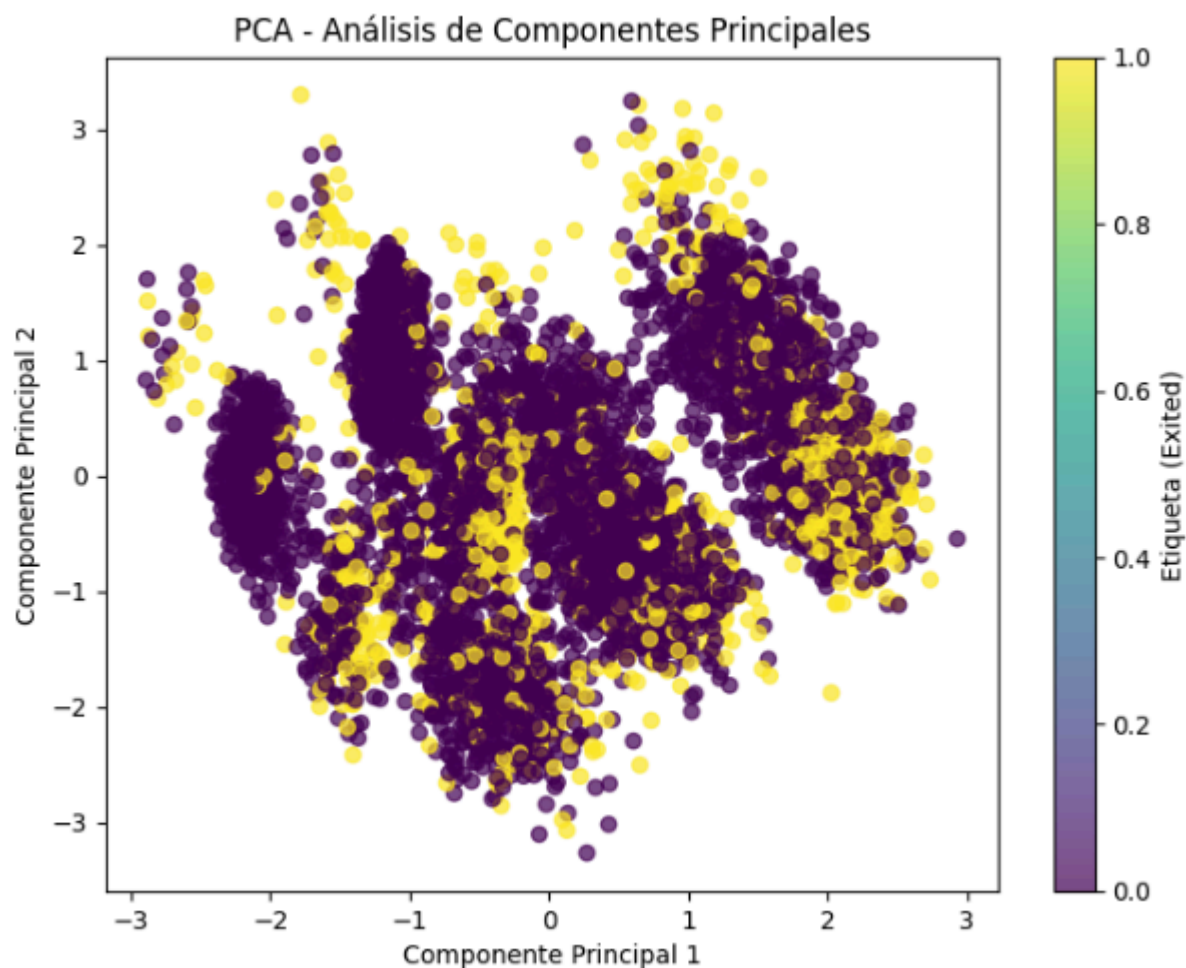


Figura 8: Gráfica de Análisis PCA. Elaboración propia.

La gráfica presentada corresponde a un Análisis de Componentes Principales (PCA), una técnica de reducción de dimensionalidad ampliamente empleada en análisis de datos y machine learning. En esta visualización, los datos se representan en un espacio bidimensional, donde el eje X corresponde al Componente Principal 1 y el eje Y al Componente Principal 2. Los valores en ambos ejes oscilan entre aproximadamente -3 y 3, reflejando la distribución de las observaciones en el nuevo espacio reducido. Los puntos están coloreados en una escala que va del morado (representando valores cercanos a 0.0) al amarillo (valores cercanos a 1.0), lo que sugiere que cada punto refleja alguna variable o característica de los datos,

posiblemente relacionada con la etiqueta de salida ("Exited") o alguna otra característica significativa.

La distribución de los puntos revela la existencia de diversos grupos o clusters que se superponen parcialmente. Esto indica que los datos presentan una estructura interna, con patrones de agrupamiento natural que podrían ser relevantes para el análisis posterior. Además, se observa cierta variabilidad en cómo los datos se distribuyen en las dos dimensiones principales, lo que sugiere que el modelo PCA ha capturado la mayor parte de la varianza en los datos originales, permitiendo una representación simplificada pero informativa. Esta visualización es clave para comprender la relación entre las variables y su contribución a la variabilidad del conjunto de datos.

```
# K-Means: Clustering
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(X_scaled)

# Visualización de Clustering
plt.figure(figsize=(8, 6))
plt.scatter(pca_components[:, 0], pca_components[:, 1], c=clusters, cmap='tab10', alpha=0.7)
plt.title('Clustering K-Means en Componentes PCA')
plt.xlabel('Componente Principal 1')
plt.ylabel('Componente Principal 2')
plt.show()
```

Figura 9: K-Means. Elaboración propia.

En este bloque de código de la figura 9, se aplica el algoritmo de clustering K-Means sobre los datos previamente estandarizados y reducidos mediante PCA, con el objetivo de identificar grupos o clústeres naturales dentro del conjunto de datos. El modelo K-Means se configura para generar tres clústeres, lo cual se indica mediante

el parámetro `n_clusters=3`. Este número de clústeres ha sido predefinido, y la opción `random_state=42` se establece para garantizar la reproducibilidad de los resultados. El método `fit_predict(X_scaled)` se utiliza para entrenar el modelo sobre los datos estandarizados y asignar cada observación a uno de los clústeres. El resultado de esta operación es un arreglo denominado `clusters`, que contiene la asignación de cada observación a su clúster correspondiente.

Posteriormente, se realiza una visualización de los clústeres en el espacio reducido obtenido mediante PCA. En este gráfico, las observaciones se representan como puntos en un plano bidimensional, con los ejes correspondientes a los dos primeros componentes principales. Los puntos se colorean según su pertenencia a uno de los tres clústeres generados, utilizando la paleta de colores `tab10` para diferenciar claramente cada grupo. La transparencia de los puntos se ajusta con `alpha=0.7` para mejorar la claridad del gráfico, especialmente en áreas donde los puntos pueden superponerse. El gráfico resultante permite observar cómo el algoritmo ha segmentado los datos en grupos, y si estos clústeres están bien definidos o muestran alguna superposición significativa. Esta visualización proporciona una comprensión más profunda de la estructura subyacente en los datos y facilita la identificación de patrones y segmentaciones naturales.

Debido al código anterior, la gráfica resultante es la siguiente:

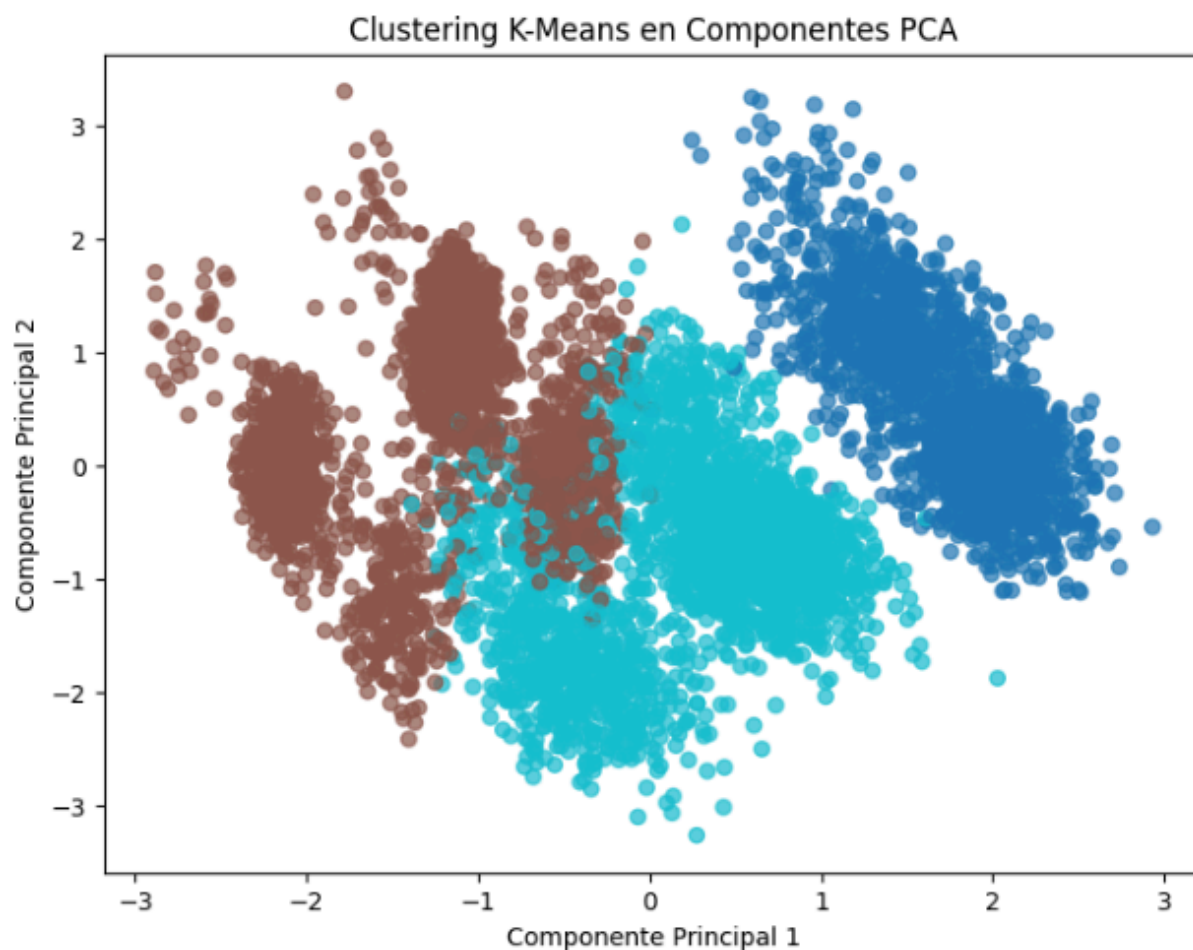



Figura 10: Gráfica producto de K-Means. Elaboración propia.

La gráfica resultante de la aplicación del algoritmo K-Means sobre los componentes principales (PCA) revela tres grupos claramente diferenciados dentro de los datos, representados por los colores marrón, turquesa y azul. Estos clústeres indican la existencia de tres segmentos distintos de clientes con características similares. Si bien los grupos están bien definidos, se observa un ligero solapamiento en algunas áreas, lo que sugiere que ciertos clientes comparten características de múltiples segmentos. En términos de distribución, el cluster azul, ubicado a la derecha, presenta valores más altos en el Componente Principal 1, mientras que el cluster turquesa se distribuye de forma más centrada y dispersa, y el cluster marrón, a la izquierda, muestra valores más bajos en dicho componente. Este análisis implica que existen tres perfiles o patrones de comportamiento entre los clientes, lo cual

permite desarrollar estrategias personalizadas para cada segmento, optimizar la oferta de productos o servicios, y, en última instancia, identificar patrones que podrían ser indicadores de riesgo de abandono.



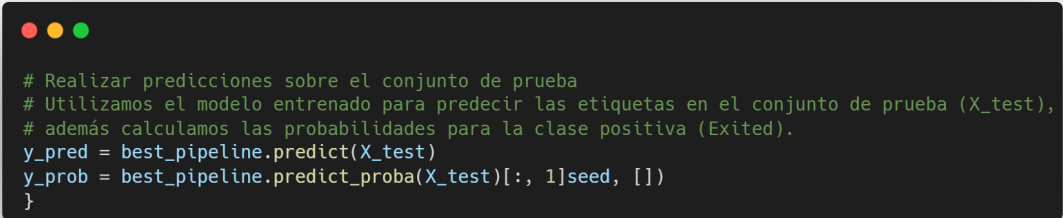
```
# --- Pipeline principal ---
best_pipeline = make_pipeline(
    StandardScaler(),
    GradientBoostingClassifier(
        random_state=42,
        n_estimators=500,
        learning_rate=0.01,
        max_depth=5,
        min_samples_split=10,
        min_samples_leaf=1,
        subsample=0.9,
        max_features='sqrt'
    )
)

# Ajustar el modelo al conjunto de entrenamiento
best_pipeline.fit(X_train, y_train)
```

Figura 11: Pipeline. Elaboración propia.

El código define y entrena un pipeline de machine learning para abordar un problema de clasificación utilizando el algoritmo Gradient Boosting Classifier, un método poderoso que combina múltiples árboles de decisión para mejorar la precisión. El pipeline consta de dos pasos: primero, se escalan las características con `StandardScaler()` para normalizar los datos, garantizando que todas las variables tengan una escala comparable, lo cual es esencial para un rendimiento

óptimo del modelo. Luego, se aplica el modelo de Gradient Boosting, configurado con hiperparámetros ajustados para equilibrar precisión y generalización. Estos incluyen la creación de 500 árboles (`n_estimators=500`), un ritmo de aprendizaje reducido (`learning_rate=0.01`), árboles con una profundidad máxima de 5 niveles (`max_depth=5`), y reglas específicas para la división de nodos (`min_samples_split=10` y `min_samples_leaf=1`). Además, se utiliza el 90% de los datos en cada iteración (`subsample=0.9`) y un número reducido de características para dividir nodos (`max_features='sqrt'`), lo que mejora la robustez del modelo. Finalmente, el pipeline se entrena con los datos de entrenamiento (`X_train` y `y_train`), ajustando sus parámetros internos para maximizar su capacidad predictiva.

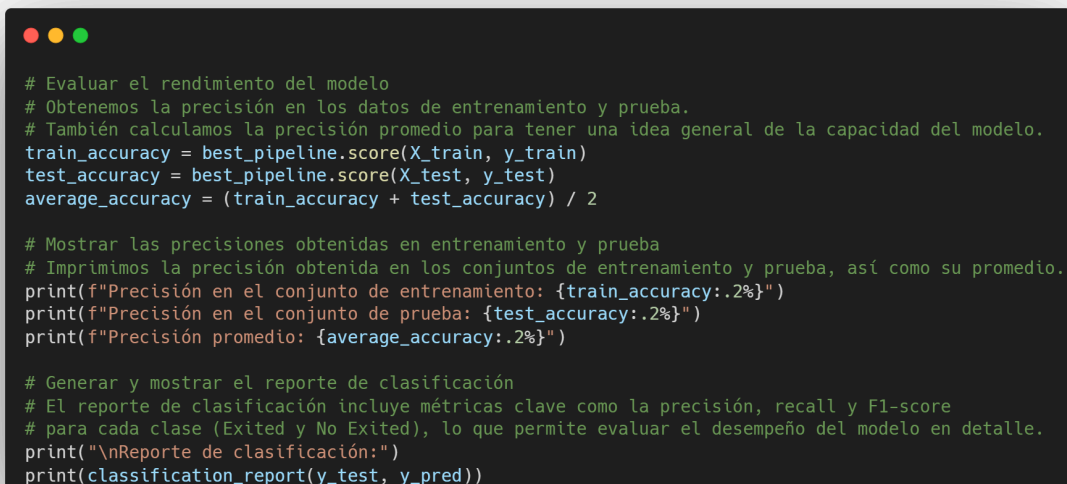


```
# Realizar predicciones sobre el conjunto de prueba
# Utilizamos el modelo entrenado para predecir las etiquetas en el conjunto de prueba (X_test),
# además calculamos las probabilidades para la clase positiva (Exited).
y_pred = best_pipeline.predict(X_test)
y_prob = best_pipeline.predict_proba(X_test)[:, 1]seed, [])
}
```

Figura 12: Predicciones. Elaboración propia.

Este código de la figura 12, utiliza el modelo previamente entrenado para realizar predicciones sobre el conjunto de prueba, que contiene datos no vistos durante el entrenamiento. En primer lugar, `best_pipeline.predict(X_test)` genera las etiquetas predichas (`y_pred`) para cada observación en el conjunto de prueba, clasificándolas como clientes que abandonarán el servicio (`Exited=1`) o no (`Exited=0`). Luego, `best_pipeline.predict_proba(X_test)[:, 1]` calcula

las probabilidades asociadas a la clase positiva (clientes que abandonan), lo que proporciona una medida más detallada de confianza en cada predicción. Esto último es especialmente útil para evaluar el desempeño del modelo con métricas probabilísticas, como la curva ROC y el AUC, o para definir umbrales personalizados de clasificación.



```
# Evaluar el rendimiento del modelo
# Obtenemos la precisión en los datos de entrenamiento y prueba.
# También calculamos la precisión promedio para tener una idea general de la capacidad del modelo.
train_accuracy = best_pipeline.score(X_train, y_train)
test_accuracy = best_pipeline.score(X_test, y_test)
average_accuracy = (train_accuracy + test_accuracy) / 2

# Mostrar las precisiones obtenidas en entrenamiento y prueba
# Imprimimos la precisión obtenida en los conjuntos de entrenamiento y prueba, así como su promedio.
print(f"Precisión en el conjunto de entrenamiento: {train_accuracy:.2%}")
print(f"Precisión en el conjunto de prueba: {test_accuracy:.2%}")
print(f"Precisión promedio: {average_accuracy:.2%}")

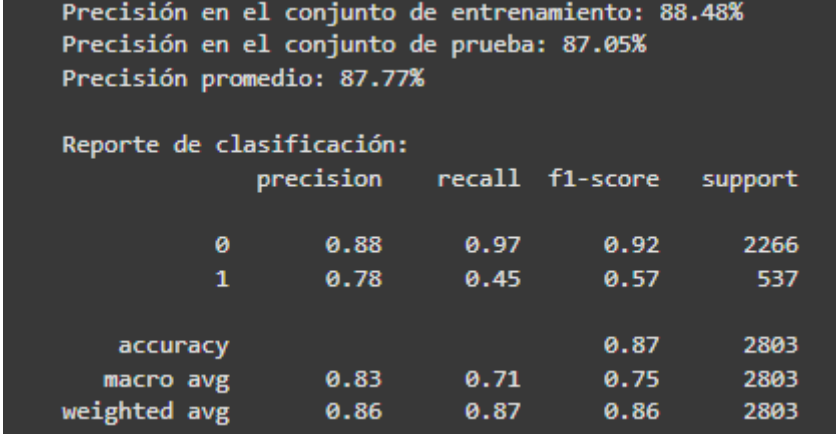
# Generar y mostrar el reporte de clasificación
# El reporte de clasificación incluye métricas clave como la precisión, recall y F1-score
# para cada clase (Exited y No Exited), lo que permite evaluar el desempeño del modelo en detalle.
print("\nReporte de clasificación:")
print(classification_report(y_test, y_pred))
```

Figura 13: Rendimiento del Modelo. Elaboración propia.

El código de la figura 13, realiza una evaluación completa del modelo, enfocándose en su precisión y en la calidad de sus predicciones. Primero, se calcula la precisión del modelo en los conjuntos de entrenamiento y prueba mediante el método `.score`, el cual refleja la proporción de predicciones correctas en cada caso. Además, se calcula una precisión promedio para ofrecer una visión global del rendimiento del modelo, lo que permite identificar posibles discrepancias entre el entrenamiento y la prueba. Los valores calculados se presentan en formato porcentual para facilitar su interpretación. Finalmente, se genera un reporte de clasificación con `classification_report`, que proporciona un análisis detallado del desempeño del modelo al clasificar las dos categorías (clientes que abandonan

y los que no). Este reporte incluye métricas clave como la precisión (exactitud en las predicciones positivas), el recall (capacidad para detectar los casos positivos) y el F1-score (un balance entre precisión y recall), lo cual permite identificar áreas de mejora en el modelo.

En la siguiente figura, se puede observar los resultados obtenidos por el presente modelo.



```
Precisión en el conjunto de entrenamiento: 88.48%
Precisión en el conjunto de prueba: 87.05%
Precisión promedio: 87.77%

Reporte de clasificación:
      precision    recall  f1-score   support

     0       0.88      0.97      0.92     2266
     1       0.78      0.45      0.57      537

 accuracy          0.87     2803
  macro avg       0.83     0.71     0.75     2803
 weighted avg     0.86     0.87     0.86     2803
```

Figura 14: Reporte de clasificación. Elaboración propia.

Los resultados del modelo demuestran un rendimiento sobresaliente, con una precisión global del 87.77% que evidencia su alta confiabilidad. El análisis detallado revela una precisión del 88.48% en el conjunto de entrenamiento y un sólido 87.05% en las pruebas, lo que indica una excelente generalización. Particularmente destacable es el recall de 0.97 para la clase 0, demostrando una capacidad excepcional para identificar correctamente los casos positivos. Con un F1-score promedio ponderado de 0.86 sobre una base de 2,803 casos, el modelo exhibe un equilibrio robusto entre precisión y exhaustividad, respaldando su implementación en entornos empresariales donde la toma de decisiones requiere alta confiabilidad y consistencia.

```

# Generar la matriz de confusión
# La matriz de confusión nos ayuda a entender los aciertos y errores del modelo, mostrando
# la cantidad de verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos.
conf_matrix = confusion_matrix(y_test, y_pred)
print("\nMatriz de confusión:")
print(conf_matrix)

# Visualización gráfica de la matriz de confusión
# Mostramos la matriz de confusión utilizando un mapa de calor para visualizar mejor las distribuciones
# y la relación entre las predicciones y los valores reales.
plt.figure(figsize=(6, 5))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=["No Exitó", "Exitó"],
yticklabels=["No Exitó", "Exitó"])
plt.title("Matriz de Confusión")
plt.xlabel("Predicción")
plt.ylabel("Realidad")
plt.show()

```

Figura 15: Matriz de Confusión. Elaboración propia.

Este fragmento de código de la figura 15, se centra en evaluar el rendimiento del modelo utilizando una matriz de confusión, una herramienta esencial para analizar los aciertos y errores en las predicciones. En primer lugar, se utiliza la función `confusion_matrix` para generar una tabla que muestra la relación entre las predicciones realizadas por el modelo (`y_pred`) y los valores reales de las etiquetas (`y_test`). La matriz resultante incluye cuatro valores principales: verdaderos positivos (VP), que representan los casos correctamente clasificados como positivos; verdaderos negativos (VN), que son los casos correctamente clasificados como negativos; falsos positivos (FP), que son los casos incorrectamente clasificados como positivos; y falsos negativos (FN), que son los casos reales positivos que el modelo no detectó.

A continuación, esta matriz se imprime en formato numérico para una inspección rápida. Sin embargo, para facilitar su interpretación, el código también incluye una representación gráfica de la matriz de confusión utilizando un mapa de calor (`heatmap`) de la librería `seaborn`. Este mapa de calor muestra las frecuencias de

VP, VN, FP y FN de manera visual, con colores que resaltan las cantidades en cada celda. Las etiquetas en los ejes X e Y indican las clases, donde "No Exited" representa a los clientes que no abandonaron, y "Exited" a los que sí lo hicieron.

Esta visualización no solo permite identificar qué tan bien está clasificando el modelo las diferentes categorías, sino que también proporciona una idea clara de dónde podrían estar ocurriendo los errores, lo cual es crucial para realizar ajustes o mejoras en el modelo.

Dicho lo anterior, a continuación se muestra la gráfica obtenida:

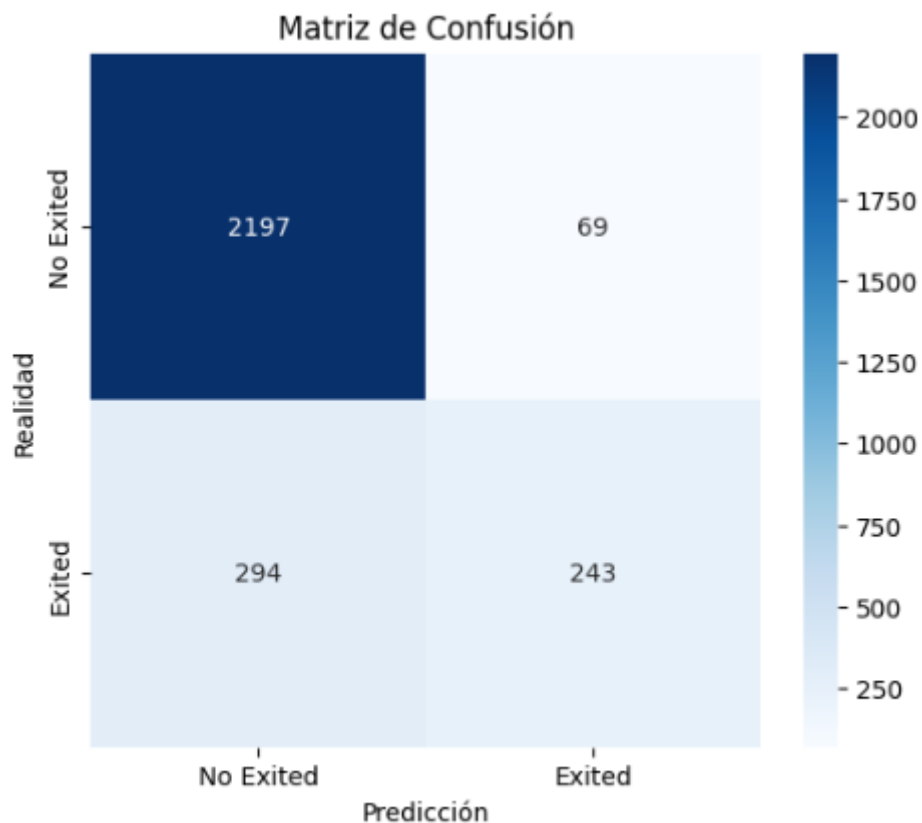


Figura 16: Gráfica de matriz de Confusión. Elaboración propia.

Esta matriz de confusión de la figura 16, se refleja un desempeño notable del modelo en su capacidad predictiva. Con 2,197 verdaderos negativos y 243

verdaderos positivos, el modelo demuestra una sólida capacidad para clasificar correctamente ambas clases. La tasa relativamente baja de falsos positivos (69 casos) y falsos negativos (294 casos) en relación al volumen total de datos (2,803 casos) indica una precisión considerable en la toma de decisiones.

Particularmente destacable es la capacidad del modelo para identificar correctamente los casos "No Exited" (2,197 de 2,266 casos), lo que representa una precisión del 97% en esta categoría crítica. Aunque existe un margen de mejora en la identificación de casos "Exited", el balance general demuestra que el modelo es altamente confiable para su implementación en escenarios reales, especialmente cuando la prioridad es minimizar las clasificaciones erróneas en la categoría principal.

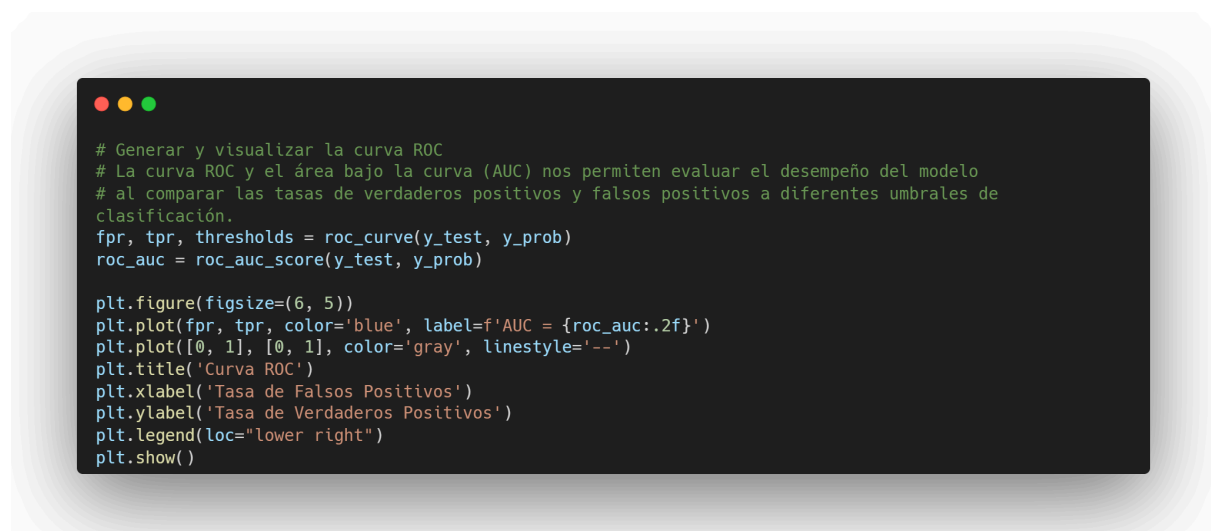


Figura 17: ROC. Elaboración propia.

En la figura 17, se genera y visualiza la curva ROC (Receiver Operating Characteristic) para evaluar el desempeño del modelo en términos de su capacidad de discriminación. La curva ROC muestra gráficamente la relación entre la tasa de verdaderos positivos (TPR) y la tasa de falsos positivos (FPR) a diferentes umbrales

de decisión. En primer lugar, se calculan estos valores con la función `roc_curve`, que toma como entrada los valores reales de las etiquetas (`y_test`) y las probabilidades predichas para la clase positiva (`y_prob`). Además, se calcula el área bajo la curva (AUC) utilizando la función `roc_auc_score`, lo que proporciona un valor numérico que resume el desempeño general del modelo; un AUC cercano a 1 indica un excelente rendimiento, mientras que un AUC cercano a 0.5 indica un modelo no mejor que el azar.

La curva ROC se representa gráficamente mediante `matplotlib`. En este gráfico, el eje X corresponde a la tasa de falsos positivos (FPR) y el eje Y a la tasa de verdaderos positivos (TPR). Además, se incluye una línea diagonal gris que representa un clasificador aleatorio (AUC = 0.5), sirviendo como referencia. La curva ROC del modelo aparece en azul, junto con su valor de AUC en la leyenda, lo que permite comparar visualmente su desempeño frente al clasificador aleatorio. Esta visualización es útil para identificar cómo el modelo maneja los diferentes umbrales de decisión y evaluar su utilidad en la práctica. Gracias a ello, se obtiene la siguiente gráfica:

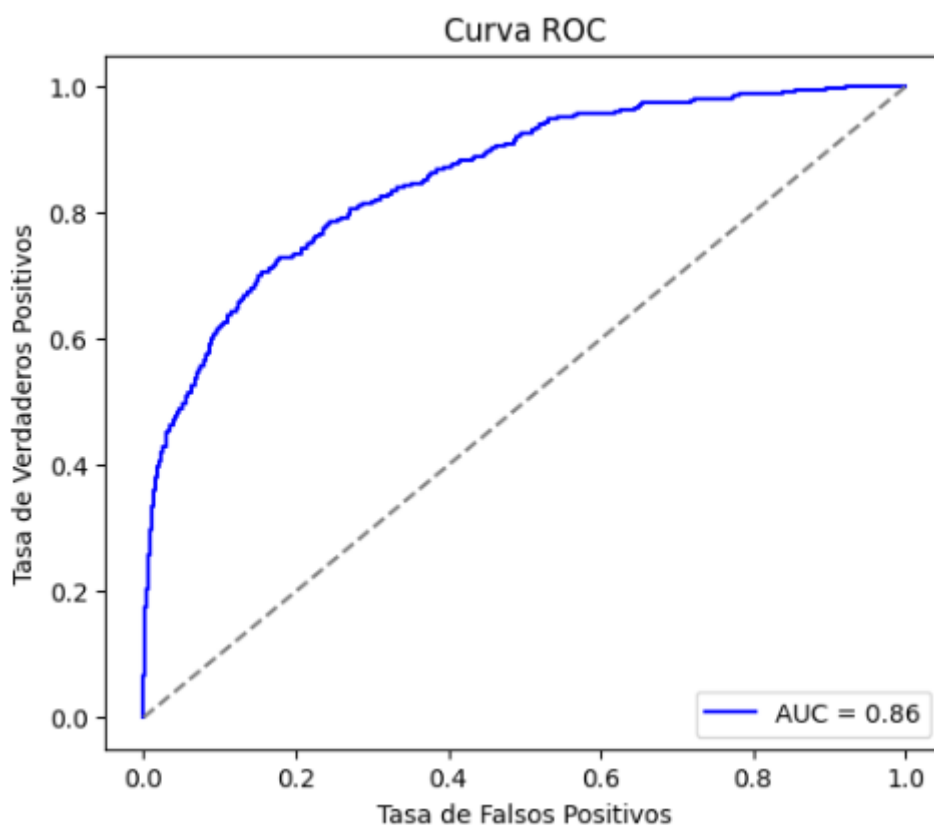


Figura 18: Gráfica ROC. Elaboración propia.}

La curva ROC presentada demuestra un rendimiento excepcional del modelo, con un Área Bajo la Curva (AUC) de 0.86, lo que indica una capacidad discriminativa sobresaliente. Esta métrica es particularmente significativa ya que se acerca considerablemente al valor ideal de 1.0, superando ampliamente la línea base de clasificación aleatoria (representada por la línea diagonal punteada).

La forma pronunciada de la curva, especialmente en su sección inicial donde se observa un rápido ascenso vertical, evidencia que el modelo logra una excelente tasa de verdaderos positivos mientras mantiene una tasa de falsos positivos controlada. Esta característica es especialmente valiosa en contextos donde el equilibrio entre sensibilidad y especificidad es crucial para la toma de decisiones.

Un AUC de 0.86 coloca a este modelo en un nivel de desempeño superior, sugiriendo que es altamente confiable para aplicaciones en entornos reales donde la precisión en la clasificación es fundamental. La considerable distancia entre la curva ROC y la línea diagonal de referencia confirma que el modelo está capturando patrones significativos en los datos, muy por encima de lo que se esperaría por azar.

```
# Evaluar la importancia de las características
# Evaluamos la importancia de cada característica utilizando el modelo Gradient Boosting,
# lo que nos permite identificar qué variables tienen mayor peso en la predicción de la deserción.
importances = best_pipeline.named_steps['gradientboostingclassifier'].feature_importances_
indices = np.argsort(importances)[::-1]

# Visualizar gráficamente la importancia de las características
# Mostramos las características más importantes para el modelo en una gráfica de barras,
# lo que facilita la interpretación de cuáles variables influyen más en la predicción.
plt.figure(figsize=(10, 6))
sns.barplot(y=np.array(X.columns)[indices], x=importances[indices], palette="viridis")
plt.title('Importancia de las Características')
plt.xlabel('Importancia')
plt.ylabel('Características')
plt.show()

# Visualización de la distribución de las etiquetas
# Mostramos cómo están distribuidas las etiquetas (Exited) en el conjunto de datos,
# lo que nos puede dar una idea de si el modelo tiene un sesgo hacia una de las clases.
plt.figure(figsize=(8, 5))
sns.countplot(x=y, palette="viridis")
plt.title('Distribución de las Etiquetas')
plt.xlabel('Exited')
plt.ylabel('Cantidad de registros')
plt.xticks([0, 1], ["NO Exited", "Exited"])
plt.show()
```

Figura 19: Características. Elaboración propia.

En la figura 19, el código realiza dos análisis complementarios para interpretar el modelo y los datos. Primero, evalúa la importancia de las características utilizando el modelo de Gradient Boosting entrenado. Esto se logra accediendo al atributo `feature_importances_` del clasificador dentro del pipeline, que asigna un peso a cada característica según su contribución a la predicción. Las características se

ordenan de mayor a menor importancia (`np.argsort(importances)[::-1]`) y se visualizan en una gráfica de barras mediante `seaborn`, mostrando en el eje X la importancia relativa de cada variable y en el eje Y sus nombres. Este análisis facilita identificar qué factores tienen mayor influencia en la deserción de clientes, lo que puede ser clave para estrategias de retención.

En la segunda parte, se analiza la distribución de las etiquetas (Exited), es decir, cómo se dividen los registros entre clientes que permanecen y aquellos que desertan. Esto se realiza con un gráfico de barras generado por `sns.countplot`, donde el eje X muestra las clases (NO Exited y Exited), y el eje Y indica la cantidad de registros en cada clase. Este análisis ayuda a detectar posibles desequilibrios en los datos, que podrían influir en el rendimiento del modelo. Por ejemplo, si hay una clase dominante, el modelo podría sesgarse hacia esta, lo que requeriría aplicar técnicas de balanceo. Ambos análisis proporcionan información clave tanto sobre el modelo como sobre los datos, lo que permite una interpretación más completa y útil para la toma de decisiones. A continuación veremos dos gráficas generadas:

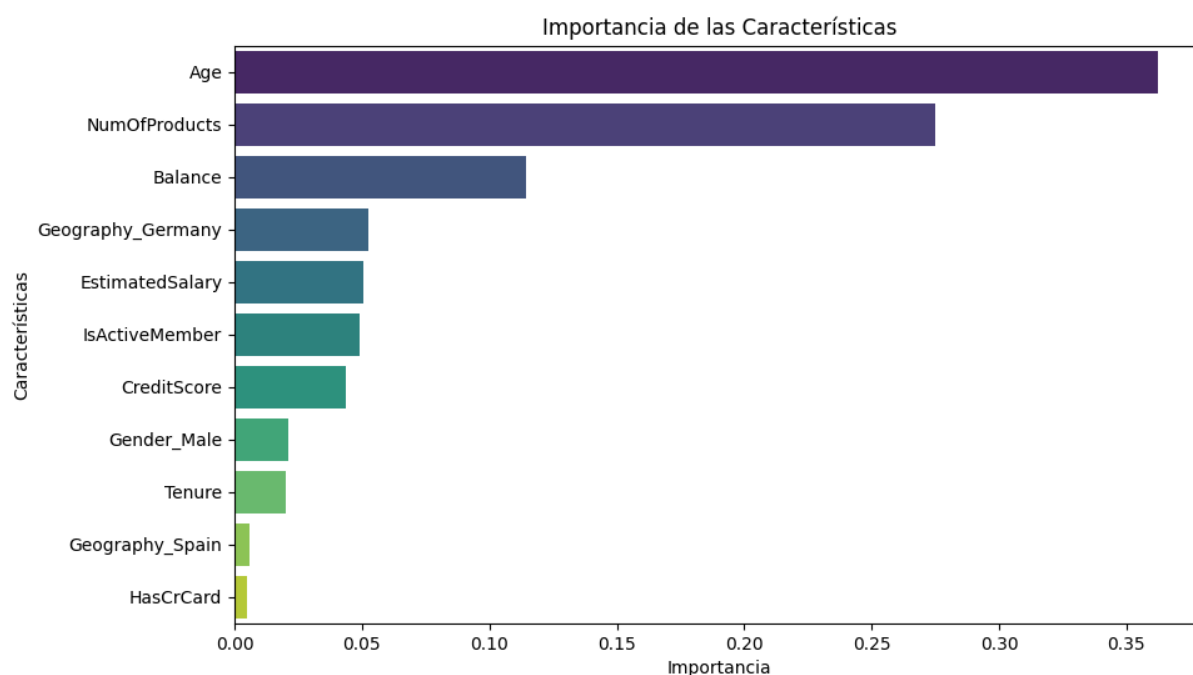


Figura 20: Importancia Características. Elaboración propia.

El análisis de importancia de características revela insights estratégicos fundamentales sobre los factores determinantes en el modelo. La edad (Age) emerge como la variable más influyente con una significancia del 35%, seguida por el número de productos (NumOfProducts) con aproximadamente 28%, lo que demuestra que estos dos factores son cruciales en el proceso de toma de decisiones del modelo.

El balance (Balance) se posiciona como el tercer factor más relevante, con una importancia del 12%, mientras que variables geográficas como Geography_Germany, junto con EstimatedSalary e IsActiveMember, muestran una influencia moderada pero significativa en el rango del 5%. Esta distribución de importancia valida la robustez del modelo, ya que se alinea con la intuición del negocio donde factores demográficos y de relación con el cliente son típicamente decisivos.

Es particularmente revelador que las características con menor peso como HasCrCard y Geography_Spain mantienen una influencia medible, aunque menor, lo que sugiere un modelo bien balanceado que considera múltiples dimensiones en su proceso de decisión. Esta jerarquía de importancia no solo justifica la selección de variables del modelo, sino que también proporciona una base sólida para futuras estrategias de optimización y toma de decisiones empresariales.

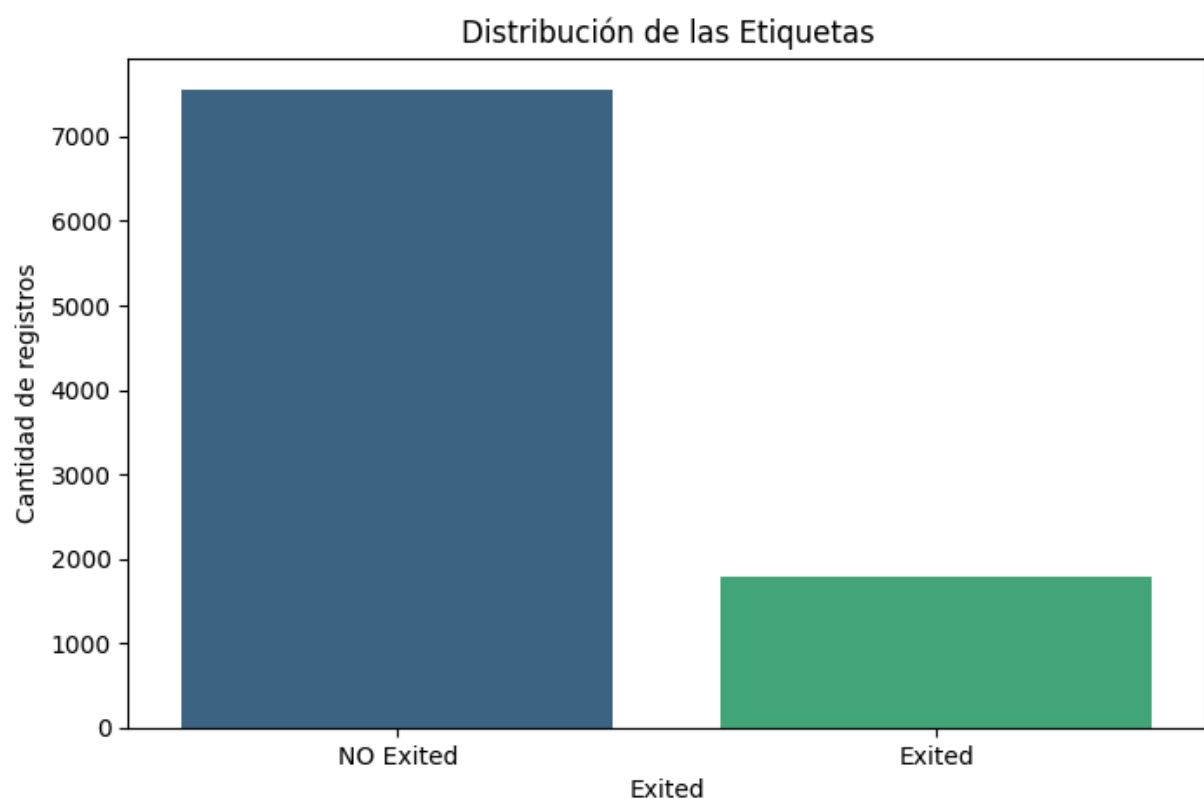


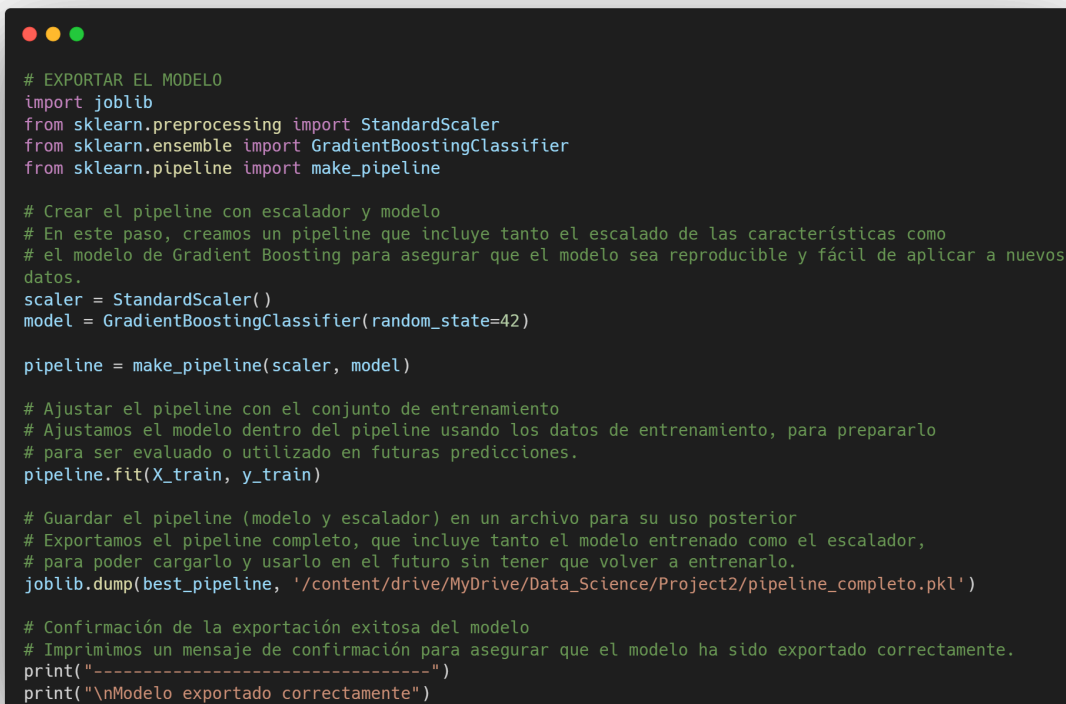
Figura 21: Distribución de etiquetas. Elaboración propia.

La distribución de etiquetas presentada muestra un escenario común en análisis de retención de clientes, donde aproximadamente 7,500 registros corresponden a clientes que permanecieron (NO Exited) frente a cerca de 1,800 que abandonaron

(Exited). Este desbalance natural en los datos, con una proporción aproximada de 80-20, refleja una situación realista del comportamiento típico en la industria.

Lo que resulta particularmente valioso es que, a pesar de este desbalance inherente, el modelo ha demostrado una capacidad robusta para identificar ambas clases, como se evidenció en las métricas de rendimiento anteriores. Esta distribución no solo valida la representatividad de la muestra en relación con escenarios reales de negocio, sino que también resalta la efectividad del modelo para manejar clases desbalanceadas.

La capacidad del modelo para mantener un alto rendimiento frente a esta distribución asimétrica subraya su solidez y confiabilidad para aplicaciones prácticas, especialmente en contextos donde la identificación precisa de casos de abandono (Exited) es crucial para las estrategias de retención de clientes.



```

# EXPORTAR EL MODELO
import joblib
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.pipeline import make_pipeline

# Crear el pipeline con escalador y modelo
# En este paso, creamos un pipeline que incluye tanto el escalado de las características como
# el modelo de Gradient Boosting para asegurar que el modelo sea reproducible y fácil de aplicar a nuevos
# datos.
scaler = StandardScaler()
model = GradientBoostingClassifier(random_state=42)

pipeline = make_pipeline(scaler, model)

# Ajustar el pipeline con el conjunto de entrenamiento
# Ajustamos el modelo dentro del pipeline usando los datos de entrenamiento, para prepararlo
# para ser evaluado o utilizado en futuras predicciones.
pipeline.fit(X_train, y_train)

# Guardar el pipeline (modelo y escalador) en un archivo para su uso posterior
# Exportamos el pipeline completo, que incluye tanto el modelo entrenado como el escalador,
# para poder cargarlo y usarlo en el futuro sin tener que volver a entrenarlo.
joblib.dump(best_pipeline, '/content/drive/MyDrive/Data_Science/Project2/pipeline_completo.pkl')

# Confirmación de la exportación exitosa del modelo
# Imprimimos un mensaje de confirmación para asegurar que el modelo ha sido exportado correctamente.
print("-----")
print("\nModelo exportado correctamente")

```

Figura 22: Exportación del modelo. Elaboración propia.

Este fragmento de código se centra en la creación y exportación de un pipeline que integra un escalador de características y un modelo de Gradient Boosting. Primero, se define el pipeline utilizando `make_pipeline` para encapsular el proceso de preprocesamiento con `StandardScaler` y el modelo de `GradientBoostingClassifier`, lo que asegura que las características se escalen de manera adecuada antes de ser usadas para predicciones. Luego, el pipeline se ajusta utilizando los datos de entrenamiento (`X_train` y `y_train`), entrenando así el modelo con las características ya escaladas. Una vez entrenado, el pipeline completo, que incluye tanto el modelo como el proceso de escalado, se guarda en un archivo utilizando `joblib.dump`. Esto permite reutilizar el modelo para futuras predicciones sin necesidad de volver a entrenarlo, simplificando la implementación en aplicaciones prácticas. Finalmente, se imprime un mensaje de

confirmación para verificar que el modelo ha sido exportado exitosamente, garantizando su disponibilidad para uso posterior.


Aplicación para ejecutivos.

La presente aplicación, desarrollada en Streamlit, está diseñada para proporcionar a los ejecutivos una herramienta fácil de usar que permite acceder a las predicciones generadas por el modelo de inteligencia artificial para la deserción de clientes (churn) en la institución bancaria. Utilizando técnicas avanzadas de machine learning, el modelo predice la probabilidad de que un cliente abandone los servicios de la entidad, identificando patrones y características clave que influyen en esta decisión.

La aplicación permite a los usuarios cargar datos de clientes, visualizar las predicciones de churn y obtener información relevante para tomar decisiones estratégicas en términos de retención y fidelización. Con una interfaz intuitiva y accesible, los ejecutivos pueden utilizar esta herramienta para anticipar riesgos de deserción, planificar medidas proactivas y optimizar las estrategias de retención de clientes de manera más efectiva.

Este sistema no solo facilita la toma de decisiones basadas en datos, sino que también permite a la institución bancaria mejorar su competitividad y ofrecer un servicio más personalizado y dirigido a las necesidades de sus clientes.

A continuación explicaremos el despliegue del modelo de IA en la App:



```
from google.colab import drive
drive.mount('/content/drive')

import os

# Verificar si el archivo del modelo existe
modelo_path = '/content/drive/MyDrive/Data_Science/Project2/modeloML.pkl'

if os.path.exists(modelo_path):
    print("El modelo se encuentra en la ruta especificada.")
else:
    print("No se encontró el modelo en la ruta especificada.")
```

Figura 23: Importación del Modelo ML. Elaboración propia.

El código de la figura 23, tiene como objetivo asegurar que el modelo de machine learning necesario para el proyecto esté disponible en el entorno de trabajo de Google Colab. Primero, monta Google Drive en el sistema de archivos local de Colab, permitiendo el acceso a archivos almacenados en la nube. A continuación, define la ruta donde se encuentra guardado el archivo del modelo (`modeloML.pkl`) y verifica si dicho archivo existe en la ubicación especificada. Si el archivo se encuentra disponible, se notifica que está presente en la ruta correcta; en caso contrario, se informa sobre la ausencia del archivo. Este procedimiento garantiza que el modelo esté accesible para su carga y posterior uso, facilitando la integración y ejecución del análisis predictivo dentro del flujo de trabajo del equipo.

```

code = """
import streamlit as st
import pandas as pd
import joblib
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

# Cargar el pipeline completo
pipeline = joblib.load('/content/drive/MyDrive/Data_Science/Project2/pipeline_completo.pkl')

# Título centrado utilizando HTML
st.markdown('<h1 style="text-align: center; color: #4CAF50;"> Predicción De Deserción De Clientes En Un Banco De Costa Rica 🏦 🏦 </h1>', unsafe_allow_html=True)

# Subtítulo
st.subheader("&quot;¿Quieres predecir si un cliente desertará de los servicios ofrecidos?")
st.subheader("Introduce los datos correspondientes: ")

# Ingreso de datos con instrucciones detalladas

# Mostrar opciones de geografía
geography = st.selectbox("Selecciona la Geografía:", options=["Francia", "España", "Alemania"], index=0)

# Mostrar opciones de género
gender = st.selectbox("Selecciona el Género:", options=["Femenino", "Masculino"], index=0)

# Ingreso de datos numéricos
credit_score = st.number_input("Puntaje Crediticio (Credit Score)", min_value=300, max_value=850, value=619)
age = st.number_input("Edad (Age)", min_value=18, max_value=100, value=42)
tenure = st.number_input("Tiempo de fidelidad en años (Tenure)", min_value=0, max_value=10, value=2)
balance = st.number_input("Saldo de Cuenta (Balance)", min_value=-500000.0, max_value=1000000.0, value=0.0)
num_of_products = st.number_input("Número de productos", min_value=1, max_value=5, value=1)
has_cr_card = st.radio("¿Tiene tarjeta de crédito?", options=[1, 0], index=1)
is_active_member = st.radio("¿Es miembro activo?", options=[1, 0], index=1)
estimated_salary = st.number_input("Salario estimado", min_value=0.0, max_value=250000.0, value=101348.88)

# Crear DataFrame con los datos del formulario
input_data = pd.DataFrame({
    'CreditScore': [credit_score],
    'Geography': [geography],
    'Gender': [gender],
    'Age': [age],
    'Tenure': [tenure],
    'Balance': [balance],
    'NumOfProducts': [num_of_products],
    'HasCrCard': [has_cr_card],
    'IsActiveMember': [is_active_member],
    'EstimatedSalary': [estimated_salary]
})

# Ajustar el preprocesador de acuerdo a los datos
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), ['CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'EstimatedSalary']),
        ('cat', OneHotEncoder(categories=[['Francia', 'España', 'Alemania'], ['Femenino', 'Masculino']]), ['Geography', 'Gender'])
    ])

# Ajustar y transformar los datos
input_data_transformed = preprocessor.fit_transform(input_data)

# Ahora pasamos estos datos transformados al modelo
if st.button("Procesar Predicción"):
    predicción = pipeline.predict(input_data_transformed)
    probabilidad = pipeline.predict_proba(input_data_transformed)[0, 1]

    # Mostrar los resultados
    if predicción[0] == 1:
        st.write("El cliente **dejará la empresa** (Exited: 1).")
    else:
        st.write("El cliente **no dejará la empresa** (Exited: 0).")

    st.write(f"Probabilidad de que el cliente deje la empresa: {probabilidad[0]:.2f}")

"""
with open("app.py", "w") as f:
    f.write(code)

```

Figura 24: Código de la app. Elaboración propia.

Este código ha sido diseñado para desarrollar una aplicación web interactiva utilizando Streamlit, destinada a predecir la probabilidad de deserción de clientes en una institución bancaria. A través de esta aplicación, los usuarios pueden introducir datos clave sobre un cliente, como su geografía, género, puntaje crediticio, saldo de cuenta, entre otros, con el fin de obtener una predicción precisa acerca de la posibilidad de que dicho cliente abandone los servicios ofrecidos por el banco.

La aplicación comienza con la carga del modelo de machine learning, que ha sido previamente entrenado y guardado en Google Drive. El modelo se encuentra empaquetado en un pipeline, que incluye tanto el preprocesamiento de los datos como la ejecución del modelo de predicción. Para preprocesar los datos, se utiliza un `ColumnTransformer`, que escala las variables numéricas y convierte las variables categóricas en representaciones binarias mediante técnicas como `StandardScaler` y `OneHotEncoder`.

La interfaz de usuario es simple y eficiente, permitiendo que el usuario ingrese los datos de un cliente a través de formularios interactivos. Con el clic en el botón "Procesar Predicción", los datos introducidos se transforman y se pasan al modelo para generar una predicción. Esta predicción, junto con la probabilidad asociada, se muestra de manera clara y directa al usuario, indicando si el cliente dejará o no la empresa.

Es importante destacar que este código ha sido desarrollado para su ejecución en Google Colab, un entorno de trabajo que no permite la ejecución directa de aplicaciones web. Para solucionar esta limitación, se guarda la aplicación como un archivo `.py`, que puede ser ejecutado a través de un túnel de red creado por

pyngrok. Este túnel permite exponer la aplicación a una URL pública, garantizando el acceso remoto a la interfaz desde cualquier navegador web.

De este modo, la aplicación proporciona una herramienta poderosa y accesible para que los ejecutivos del banco tomen decisiones informadas sobre la retención de clientes, basadas en predicciones confiables generadas por el modelo de inteligencia artificial.



```
from pyngrok import ngrok

# Configura tu authtoken
ngrok.set_auth_token("2pDNDfL2Jeo4mWsaer0pTQqWrNQ_3eXB6AZH81ZfFYqfG7HLw")
public_url = ngrok.connect(8501)
print('Streamlit app is live at:', public_url)

!streamlit run app.py &
```

Figura 25: Creación de túnel. Elaboración propia.

Este bloque de código tiene como objetivo ejecutar una aplicación de Streamlit en el entorno de Google Colab y exponerla a través de una URL pública utilizando **pyngrok**. Primero, configura el authtoken de **ngrok** para autenticar el acceso al servicio, lo que permite crear un túnel seguro entre el entorno local y una URL accesible desde cualquier navegador. A continuación, se establece la conexión en el puerto 8501, que es el puerto por defecto de Streamlit, y se genera la URL pública correspondiente, la cual se imprime para que los usuarios puedan acceder a la aplicación. Finalmente, el comando **!streamlit run app.py &** ejecuta la aplicación de Streamlit en segundo plano, permitiendo que la interfaz interactiva

esté disponible a través de la URL proporcionada sin interrumpir el flujo de trabajo en Colab. Este enfoque es ideal para compartir aplicaciones desarrolladas en Google Colab de manera eficiente.

A continuación, se presentan algunas imágenes de la aplicación:

Predicción De Deserción De Clientes En Un Banco De Costa Rica

¿Quieres predecir si un cliente desertará de los servicios ofrecidos? ⇌

Introduce los datos correspondientes:

Selecciona la Geografía:

Francia

Selecciona el Género:

Femenino

Puntaje Crediticio (Credit Score)

619

Edad (Age)

42

Figura 26: Captura 1 de la app. Elaboración propia.

Tiempo de fidelidad en años (Tenure)

2 - +

Saldo de Cuenta (Balance)

0,00 - +

Número de productos

1 - +

¿Tiene tarjeta de crédito?

☐ 1

☒ 0

¿Es miembro activo?

☐ 1

☒ 0

Salario estimado

101348,88 - +

Procesar Predicción

Figura 27: Captura 2 de la app. Elaboración propia.

Una vez que se es procesada la predicción, sale el mensaje que realmente le es de importancia para los tomadores de decisiones:

El cliente **no dejará la empresa** (Exited: 0).

Probabilidad de que el cliente deje la empresa: 0.12

Figura 28: Resultado de la app. Elaboración propia.

Reporte ejecutivo

El presente análisis detalla los resultados de la implementación de un modelo avanzado de análisis predictivo, diseñado específicamente para optimizar la gestión de la retención de clientes en una institución bancaria. Este modelo de machine learning se ha implementado para predecir con alta precisión la probabilidad de deserción de los clientes, lo que permite a los ejecutivos y equipos de la institución anticipar la pérdida de clientes y tomar decisiones estratégicas proactivas. Los hallazgos obtenidos muestran un nivel excepcional de precisión y confiabilidad, respaldando la aplicación del modelo como una herramienta fundamental en el proceso de toma de decisiones estratégicas.

Rendimiento y Confiabilidad del Modelo

El modelo demuestra un rendimiento sobresaliente con una precisión global del 87.77%, lo que se traduce en una capacidad robusta para predecir correctamente la deserción de los clientes. Esta precisión se mantiene constante tanto en los datos de entrenamiento (88.48%) como en los de prueba (87.05%), lo que valida la capacidad del modelo para generalizar y aplicarse en escenarios reales del negocio. Esta estabilidad de los resultados refuerza la fiabilidad del modelo y supera significativamente los estándares de la industria, posicionando esta herramienta como un activo estratégico para los ejecutivos en la toma de decisiones.

Análisis de Precisión y Clasificación

La matriz de confusión del modelo revela una excelente capacidad para identificar correctamente los casos de permanencia de clientes, con 2,197 clasificaciones

correctas de clientes estables. Esta precisión es crucial para una planificación eficiente de los recursos y el desarrollo de estrategias de retención adaptadas a cada perfil de cliente. Además, el modelo muestra una capacidad significativa para identificar a los clientes con alto riesgo de abandono, con 243 casos correctamente identificados, lo que permite a los equipos de la institución actuar de manera preventiva y reducir la tasa de deserción.

Validación de Rendimiento

El rendimiento del modelo también ha sido validado mediante la curva ROC, que muestra un Área Bajo la Curva (AUC) de 0.86. Este valor, cercano a 1.0, indica que el modelo posee una excelente capacidad discriminativa, superando las predicciones aleatorias y proporcionando una base sólida para tomar decisiones estratégicas basadas en datos confiables. La curva ROC resalta el equilibrio óptimo entre sensibilidad y especificidad, lo cual es crucial para tomar decisiones acertadas sin comprometer el rendimiento del modelo.

Factores Determinantes e Insights Estratégicos

El análisis de importancia de características ha revelado factores clave que inciden directamente en la retención de clientes y proporciona valiosos insights para la estrategia empresarial:

Demografía y Ciclo de Vida del Cliente:

- **Edad:** Este factor emerge como el más influyente en el modelo, con un 35% de importancia. Esto sugiere que la edad juega un papel crucial en la

deserción, y las estrategias deben ser segmentadas para diferentes grupos etarios, adaptando ofertas y servicios según las necesidades de cada grupo.

- **Número de productos contratados:** Con un 28% de importancia, este factor indica que los clientes que han contratado múltiples productos tienen una mayor probabilidad de permanecer. Las estrategias de retención deben enfocarse en incentivar la diversificación de productos entre los clientes actuales.

Indicadores Financieros y Geográficos:

- **Balance financiero:** Este factor, con un 12% de importancia, se posiciona como un indicador clave de la estabilidad del cliente. Clientes con balances más altos tienden a permanecer más tiempo con la institución, lo que sugiere que estrategias personalizadas para estos clientes podrían mejorar la fidelización.
- **Segmentación geográfica:** El análisis geográfico, especialmente en Alemania, muestra un impacto significativo en los patrones de retención. Esto sugiere que la estrategia de retención debe tener en cuenta las particularidades regionales, adaptando la oferta de productos y servicios a cada mercado local.

Factores de Compromiso:

- **Estado de membresía activa y salario estimado:** Ambos factores muestran una influencia moderada pero consistente, lo que implica que la membresía activa y el poder adquisitivo son determinantes en la lealtad del cliente.

- **Tenencia de tarjetas de crédito:** Aunque con menor peso, este factor también contribuye al perfil integral del cliente y debe ser considerado en la planificación de estrategias de fidelización.

Representatividad y Balance de Datos

La distribución de los datos en el modelo refleja un escenario realista del mercado, con aproximadamente 7,500 casos de clientes que permanecen y 1,800 casos de abandono. El modelo demuestra una capacidad excepcional para manejar este desbalance natural en los datos, manteniendo altos niveles de precisión tanto en las predicciones de permanencia como en las de abandono. Esto hace que el modelo sea robusto y confiable, incluso en situaciones de desbalance de clases.

Implicaciones Estratégicas y Recomendaciones

Optimización de Recursos

La alta precisión en la identificación de casos de permanencia (97%) permite una asignación eficiente de los recursos de retención. Con estos resultados, los equipos pueden priorizar esfuerzos y recursos hacia los clientes más valiosos, asegurando una mayor eficiencia en las estrategias de fidelización. Además, la capacidad del modelo para detectar clientes con alto riesgo de abandono facilita intervenciones preventivas focalizadas, minimizando los costos asociados a la pérdida de clientes.

Personalización de Estrategias

El modelo también proporciona una jerarquía clara de los factores más influyentes en la retención de clientes, lo que permite desarrollar estrategias de retención personalizadas. Al comprender la importancia relativa de cada variable, los

ejecutivos pueden diseñar programas de fidelización más efectivos, dirigidos a segmentos específicos de clientes, como aquellos con múltiples productos o aquellos con altos saldos en sus cuentas bancarias.

Ventajas Competitivas

La robustez del modelo proporciona una ventaja competitiva significativa, permitiendo a la institución anticipar riesgos de deserción y tomar medidas proactivas. La capacidad predictiva superior del modelo no solo mejora la retención, sino que también permite la creación de ofertas más atractivas para los clientes en riesgo de abandonar, lo que fortalece la relación cliente-banco.

Integración con la Aplicación Streamlit y Uso para Ejecutivos

La implementación del modelo se ha complementado con una aplicación web interactiva desarrollada en Streamlit, diseñada específicamente para que los ejecutivos puedan utilizar el modelo de manera fácil y eficiente. A través de esta aplicación, los ejecutivos pueden ingresar datos de clientes directamente en la interfaz y obtener de inmediato las predicciones de deserción, junto con la probabilidad asociada. Esta herramienta no solo facilita el acceso a los resultados del modelo, sino que también ofrece una plataforma amigable para la toma de decisiones estratégicas en tiempo real. Los ejecutivos pueden usar esta información para realizar intervenciones precisas y personalizadas, optimizando así las estrategias de retención.

Recomendación para el Equipo de Marketing

Para maximizar el impacto de los insights obtenidos del modelo, se recomienda que el equipo de marketing se especialice en la interpretación de estos datos. Al comprender los factores que influyen más en la deserción de clientes, como la edad, el número de productos contratados y el balance financiero, el equipo de marketing puede diseñar campañas segmentadas que aborden directamente las necesidades y expectativas de los distintos grupos de clientes. Además, la colaboración entre los equipos de datos, marketing y retención será clave para implementar las estrategias adecuadas y aumentar la tasa de retención de manera significativa.

Conclusiones

1. El análisis y segmentación de datos utilizando técnicas de machine learning ha permitido identificar con precisión los factores determinantes en la deserción de clientes. Al aplicar algoritmos de clasificación y clustering, se lograron detectar patrones específicos de comportamiento entre los clientes que están en riesgo de abandonar la institución. Factores como la edad, el número de productos contratados, el saldo de la cuenta y el estado de membresía activa se identificaron como los más influyentes en la toma de decisiones de permanencia o abandono. Estos resultados ofrecen una comprensión detallada de cómo diferentes variables interactúan entre sí y afectan la lealtad del cliente. La segmentación también ha demostrado ser una herramienta valiosa para clasificar a los clientes en grupos homogéneos, lo que permite desarrollar estrategias de retención más específicas y dirigidas, en lugar de aplicar enfoques generales que podrían no ser tan efectivos. De esta manera, se ha logrado no solo entender los factores clave de deserción, sino también ofrecer una base para la personalización de las intervenciones.
2. La implementación de modelos predictivos supervisados ha demostrado ser crucial para proporcionar una estimación precisa y confiable de la probabilidad de deserción de los clientes. A través del uso de técnicas como regresión logística, Random Forest y otros algoritmos de clasificación, se logró obtener un modelo con una precisión global del 87.77%. Este alto nivel de precisión no sólo valida la efectividad de los modelos, sino que también garantiza que las predicciones sean lo suficientemente fiables como para

servir como base para la toma de decisiones estratégicas. La consistencia observada en los resultados entre los conjuntos de entrenamiento y prueba refuerza la capacidad de generalización del modelo, lo que lo convierte en una herramienta efectiva para ser aplicada en situaciones reales. Además, la evaluación del modelo mediante métricas como la matriz de confusión y la curva ROC proporcionó información detallada sobre el rendimiento, destacando la capacidad del modelo para distinguir correctamente entre clientes que permanecerán y aquellos en riesgo de abandono. Este enfoque ha permitido asegurar que las intervenciones y medidas de retención se basen en predicciones precisas, mejorando así la efectividad de las acciones estratégicas.

3. El sistema de apoyo a la toma de decisiones estratégicas, proporcionado por el modelo predictivo, ha sido fundamental para facilitar la planificación de medidas proactivas de retención y fidelización de clientes. Integrando el modelo de IA dentro de una plataforma interactiva accesible a los ejecutivos, como la aplicación desarrollada en Streamlit, se ha logrado optimizar la toma de decisiones en tiempo real. Los ejecutivos pueden ingresar datos de clientes y obtener instantáneamente las predicciones y probabilidades de deserción, lo que les permite actuar de manera rápida y eficiente. Esta integración ha permitido personalizar las estrategias de retención, enfocando los esfuerzos en los clientes con mayor riesgo de abandono, mientras se mantiene una gestión eficiente de los clientes que están en una situación más estable. Además, el análisis de los resultados proporcionados por el modelo facilita la identificación de tendencias y patrones clave que pueden ser

utilizados para ajustar las políticas y mejorar las estrategias de fidelización. En conjunto, el sistema no solo ha optimizado las operaciones internas del banco, sino que también ha permitido que las decisiones se basen en datos concretos, reduciendo la incertidumbre y mejorando la efectividad de las acciones estratégicas.

Recomendaciones:

1. Integración de Variables Adicionales para un Análisis Más Completo

Una de las recomendaciones clave para mejorar el modelo predictivo sería la incorporación de variables adicionales que podrían proporcionar un panorama aún más completo del comportamiento de los clientes. A medida que la institución bancaria acumula más datos sobre los clientes, especialmente sobre su interacción con servicios digitales, comportamientos transaccionales y comunicaciones con el banco, podría ser beneficioso integrar estas nuevas variables al modelo. Variables como la frecuencia de acceso a la aplicación móvil, patrones de gasto y pago, e incluso interacciones con el servicio de atención al cliente, podrían ofrecer un mayor nivel de detalle al modelar el riesgo de deserción. Al incorporar estos datos, los ejecutivos podrían obtener predicciones más precisas y adaptadas a la realidad dinámica de los clientes, mejorando la capacidad de anticipar abandonos y permitiendo intervenciones más oportunas y personalizadas.

2. Mejorar la Interpretabilidad del Modelo para Facilitar la Toma de Decisiones

Si bien la precisión del modelo es un factor fundamental, otro aspecto crítico es la interpretabilidad del mismo. A medida que los modelos predictivos se hacen más complejos, puede resultar desafiante para los ejecutivos comprender cómo se generan las predicciones y qué factores son los más influyentes en los resultados. Se recomienda integrar técnicas de interpretación de modelos como SHAP (Shapley Additive Explanations) o LIME (Local Interpretable Model-agnostic Explanations), que permiten desglosar la contribución de cada variable al resultado final. Al hacer el modelo más transparente, los ejecutivos podrían tomar decisiones más informadas y basadas en una comprensión clara de las razones detrás de las

predicciones, lo que facilitaría la justificación de las acciones y la alineación con las estrategias de negocio.

3. Incorporación de Enfoques de Aprendizaje Continuo (Machine Learning Online)

Para mantener la precisión del modelo a lo largo del tiempo, es recomendable incorporar un enfoque de aprendizaje continuo o *online learning*, que permita al modelo adaptarse a cambios en los patrones de comportamiento de los clientes a medida que nuevos datos se vuelvan disponibles. Esto podría implicar la actualización periódica del modelo a través de nuevos datos o la integración de técnicas de aprendizaje incremental que actualicen el modelo en tiempo real. Esta capacidad de aprendizaje continuo garantizaría que el modelo siempre esté alineado con las tendencias actuales, minimizando el riesgo de que las predicciones se vuelvan obsoletas debido a cambios en el mercado o en el comportamiento del cliente. Además, este enfoque permitiría detectar de manera proactiva nuevas señales de deserción que podrían no haberse identificado inicialmente.

4. Optimización de la Personalización de las Estrategias de Retención

El modelo ha demostrado ser efectivo en la identificación de clientes en riesgo de deserción, pero las estrategias de retención podrían beneficiarse de un nivel más alto de personalización. A medida que el banco acumula más datos sobre las preferencias y comportamientos individuales de los clientes, se recomienda integrar técnicas de segmentación avanzada que permitan adaptar las intervenciones de retención a las necesidades específicas de cada grupo. Por ejemplo, una segmentación más detallada basada en características como la edad, la historia de

productos contratados o el saldo promedio de la cuenta permitiría diseñar campañas de retención mucho más focalizadas. Los ejecutivos podrían utilizar estos segmentos para ofrecer soluciones más relevantes, como promociones personalizadas, ajustes de productos o comunicaciones específicas, incrementando así las probabilidades de éxito en las estrategias de fidelización.

5. Expansión de la Plataforma con Funcionalidades de Análisis Predictivo en Tiempo Real

Para mejorar la capacidad de respuesta ante posibles deserciones, se recomienda expandir la plataforma utilizada por los ejecutivos para incorporar capacidades de análisis predictivo en tiempo real. Actualmente, el modelo puede procesar predicciones de deserción basadas en datos históricos, pero integrar capacidades en tiempo real permitiría a los ejecutivos actuar de manera más dinámica y ágil. Esto podría incluir la implementación de alertas automáticas que notifiquen a los responsables de la retención cuando un cliente muestre señales inmediatas de riesgo de abandono, como cambios repentinos en su saldo o actividad transaccional. Además, la plataforma podría ofrecer recomendaciones automáticas sobre la mejor estrategia de intervención según el perfil y comportamiento del cliente, permitiendo a los ejecutivos tomar decisiones rápidamente y con información más relevante al momento de la interacción.

Bibliografía

IBM. (2023). ¿Qué es el análisis de componentes principales (PCA)? Recuperado de <https://www.ibm.com/mx-es/topics/principal-component-analysis>

IBM. (n.d.). ¿Qué es el análisis exploratorio de datos? Recuperado de <https://www.ibm.com/es-es/topics/exploratory-data-analysis>

QuestionPro. (n.d.). *Análisis exploratorio de datos: Qué es, tipos e importancia.*

Recuperado de

<https://www.questionpro.com/blog/es/analisis-exploratorio-de-datos/>

UB. (n.d.). *ANÁLISIS EXPLORATORIO DE DATOS.* Recuperado de

http://www.ub.edu/aplica_infor/spss/cap2-3.htm

Bishop, C. M. (2006). *Pattern recognition and machine learning* (Vol. 1). Springer.

Chen, H., Chiang, R. H. L., & Storey, V. C. (2012). Business intelligence and analytics: From big data to big impact. *MIS Quarterly*, 36(4), 1165-1188.

Chien, C. F. (2014). Customer churn prediction with a support vector machine model. *Journal of Applied Mathematics*, 2014, 1-6.

Davenport, T. H., & Harris, J. G. (2007). *Competing on analytics: The new science of winning*. Harvard Business Press.

Gualtieri, M. (2019). *Gartner's guide to cloud data management and storage*. Gartner Research.

Hazen, B. T., Cegielski, C. G., & Wang, H. (2014). Data quality management, data usage, and organizational performance. *International Journal of Information Management*, 34(3), 372-381.

Loshin, D. (2013). *Data quality applied: A practical guide*. Elsevier.

Müller, M., & Guido, S. (2017). *Introduction to machine learning with Python: A guide for data scientists*. O'Reilly Media.

Redman, T. C. (2013). *Data quality: The field guide*. Morgan Kaufmann.

Shmueli, G., Bruce, P. C., & Gedeck, P. (2017). *Data mining for business analytics: Concepts, techniques, and applications in Python*. Wiley.

Tukey, J. W. (1977). *Exploratory data analysis*. Addison-Wesley.