# Proyecto Final

## Data Science

Martín Viera
Matías Franco

# Introducción

**Las tarjetas de crédito son una buena fuente de ingresos para los bancos debido a los diferentes tipos de comisiones que cobran.**

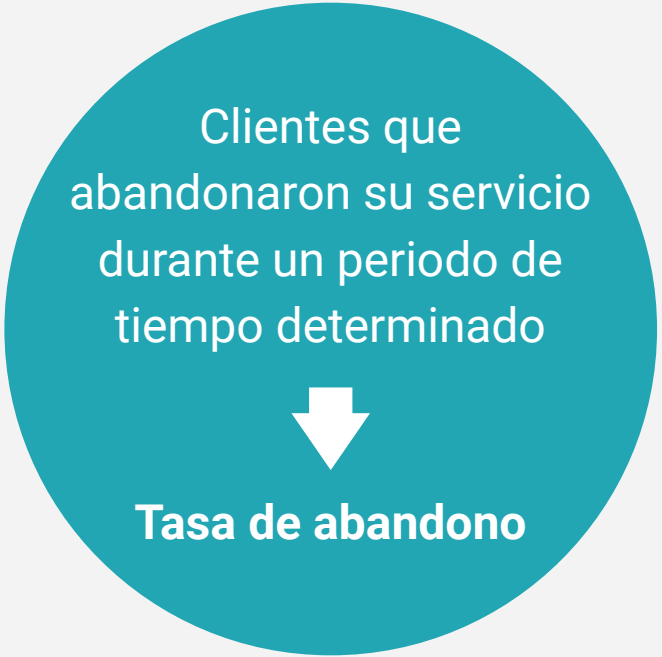

Algunas comisiones se cobran a todos los usuarios, independientemente de su uso, mientras que otras se cobran en determinadas circunstancias.

Que los clientes abandonen los servicios de las tarjetas de crédito supondría una pérdida para el banco, por lo que éste quiere analizar los datos de los clientes e **identificar a los que abandonan los servicios** de las tarjetas de crédito y las razones para hacerlo, de modo que el banco pueda mejorar en esas áreas.

# Modelo de churn

## Un modelo de churn es una representación matemática de cómo la deserción de clientes puede afectar un negocio.

Los cálculos del churn se construyen a partir de los datos existentes. Un **modelo predictivo** de abandono extrapola estos datos para mostrar las **tasas de abandono** de los clientes potenciales en el futuro. El churn (también conocido como deserción de clientes) es un problema para las empresas de suscripción. Unas altas tasas de retención son vitales para su supervivencia.



**La creación de un modelo de predicción de bajas le ayudará a realizar cambios proactivos en sus esfuerzos de retención que reduzcan las tasas de bajas.**

# Objetivo: Explorar y visualizar el conjunto de datos

Construir un modelo de clasificación para predecir si el cliente se dará de baja o no.

Optimizar el modelo utilizando las técnicas adecuadas.

Generar un conjunto de ideas y recomendaciones para ayudar al banco.

# Problema

¿Tienen los ingresos algún efecto sobre el abandono?

¿Tiene el género alguna relación con el abandono?

¿Cuáles son los signos de deserción?

# Miembros

**Martín Viera**

mviera@gmail.com

**Matías Franco**

mfranco@clarin.com

# Dataset

https://www.kaggle.com/code/xavier14/predicting-churn-with-tree-based-models

# Variables

### CLIENTNUM

Número de cliente. Identificador único del cliente titular de la cuenta.

### Attrition_Flag

Variable de evento interno (actividad del cliente) - si la cuenta está cerrada, entonces "cliente dado de baja", si no, "cliente existente"

## Customer_Age

Edad en años

## Gender

Género del titular de la cuenta

## Dependent_count

Número de personas a cargo

## Nivel_de_educación

Cualificación educativa del titular de la cuenta - Graduado, Bachillerato, Desconocido, Sin estudios, Universidad (se refiere a un estudiante universitario), Postgrado, Doctorado.

## Estado_marital

Estado civil del titular de la cuenta

## Income_Category

Categoría de ingresos anuales del titular de la cuenta

## Card_Category

Tipo de tarjeta

## Months_on_book

Periodo de relación con el banco

## Total_Relationship_Count

Nº total de productos que tiene el cliente

## Months_Inactive_12_mon

Nº de meses inactivos en los últimos 12 meses

## Contacts_Count_12_mon

Número de contactos entre el cliente y el banco en los últimos 12 meses

## Credit_Limit

Límite de crédito de la tarjeta de crédito

## Total_Revolving_Bal

El saldo que se arrastra de un mes a otro es el saldo rotativo

## Avg_Open_To_Buy

Cantidad que queda en la tarjeta de crédito para usar (Media de los últimos 12 meses)

## Total_Trans_Amt

Importe total de las transacciones (últimos 12 meses)

## Total_Trans_Ct

Recuento total de transacciones (últimos 12 meses)

## Total_Ct_Chng_Q4_Q1

Relación entre el recuento total de transacciones en el cuarto trimestre y el recuento total de transacciones en el primer trimestre

## Total_Amt_Chng_Q4_Q1

Relación entre el importe total de las transacciones en el cuarto trimestre y el importe total de las transacciones en el primer trimestre

# Avg_Utilization_Ratio

Representa la parte del crédito disponible que ha gastado el cliente

# Análisis del dataset

Hay un total de **21 columnas** y **10.127 observaciones en el conjunto de datos**

Podemos ver que **Education_Level** y **Marital_Status** tienen menos de 10.127 valores no nulos, es decir, las columnas tienen valores perdidos.

La edad sólo tiene 45 valores únicos, es decir, la mayoría de los clientes tienen una **edad similar**

El valor medio de la columna Edad del cliente es de aproximadamente 46 y la mediana también es de 46. Esto muestra que **la mayoría de los clientes son menores de 46 años**.

Hay un desequilibrio en el conjunto de datos.

El **93%** de los clientes tienen tarjeta azul.

La categoría de ingresos tiene un **valor abc para el 10% de los registros**, que cambiaremos por Desconocido.

# Procesamiento de datos previo a la EDA

**Tratamiento de los valores perdidos en el nivel de educación y el estado civil.**

Nota:

El tratamiento de los valores perdidos debe hacerse después de dividir los datos en conjuntos de entrenamiento, validación y prueba. Sin embargo, en este caso, el tratamiento es genérico, ya que estamos rellenando los datos con Desconocidos. Por lo tanto, el tratamiento puede realizarse en el conjunto de datos.

Una estrategia similar es aplicable para tratar el valor de la columna Categoría de ingresos abc.

# Análisis exploratorio de datos

## Análisis univariante

**Resumen de características numéricas:**

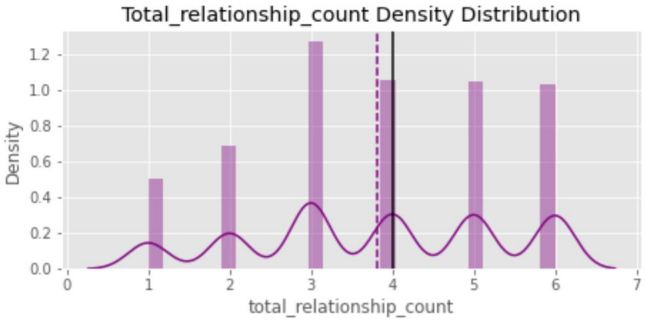
El primer paso del análisis univariante es **comprobar la distribución/extensión de los datos**.

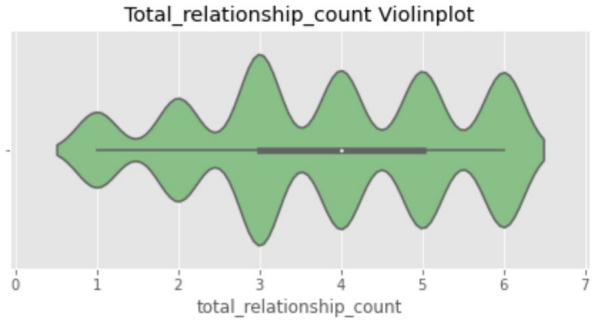
Para ello se utilizan principalmente **histogramas y gráficos de caja**. Además, trazaremos cada característica numérica en un gráfico de violín y en un gráfico de distribución de densidad acumulada.

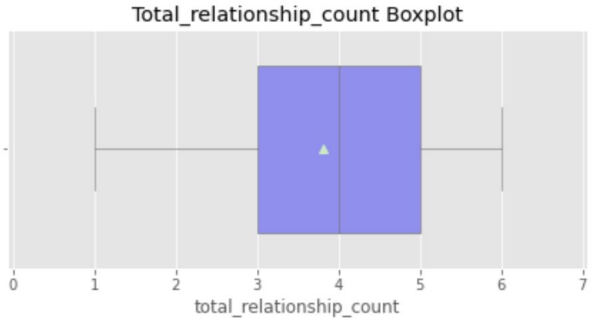
Para estos 4 tipos de gráficos, estamos construyendo abajo la función summary() para trazar cada uno de los atributos numéricos. Además, mostraremos el resumen de 5 puntos de las características.

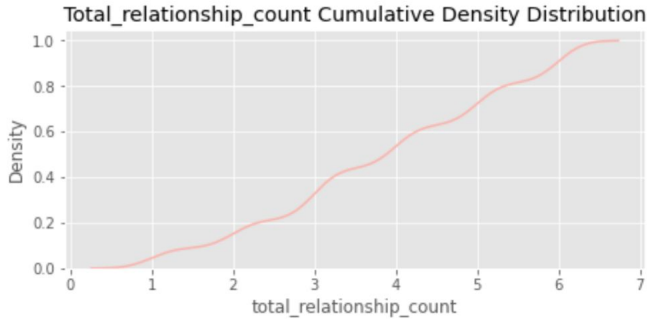
# Total_Relationship_Count

Nº total de productos que tiene el cliente



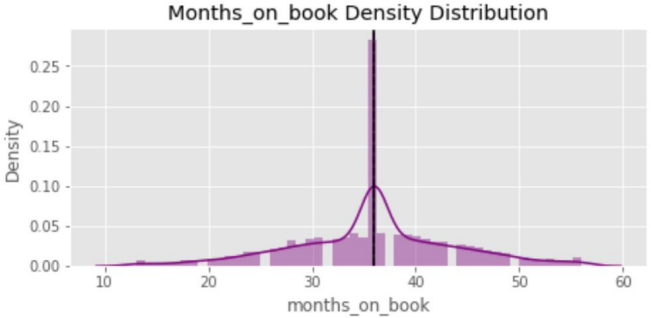



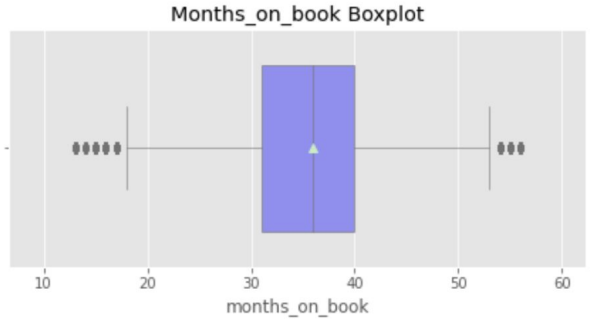

**La mayoría de los clientes tienen 4 o más relaciones con el banco**

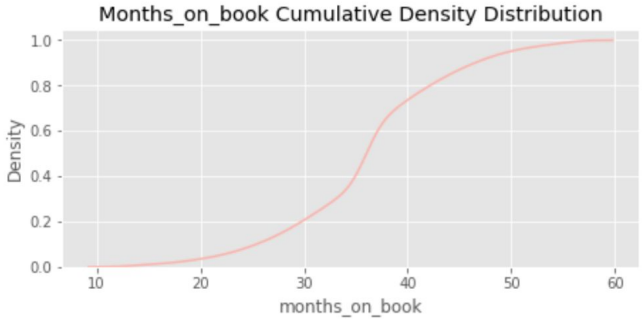
# Months_on_book

## Periodo de relación con el banco







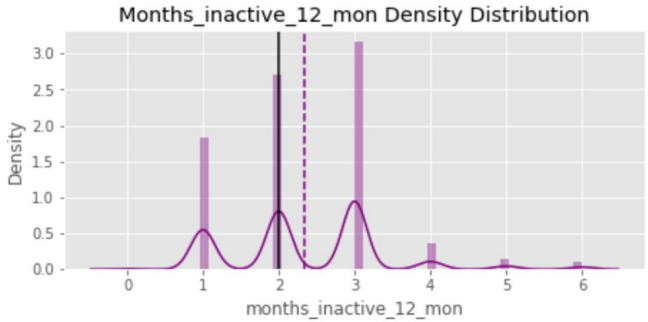



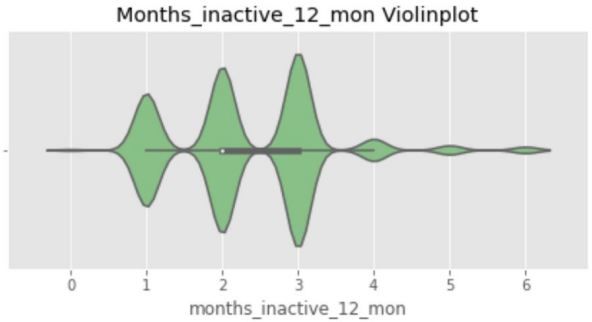

**La mayoría de los clientes están en los libros durante 3 años.**

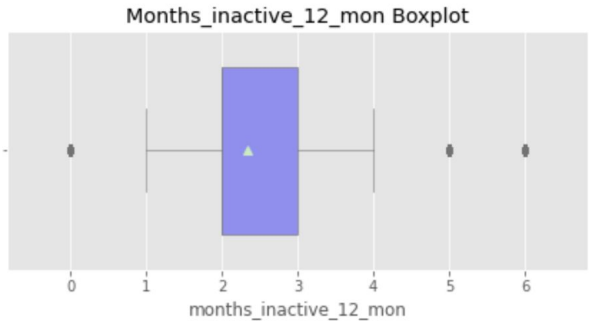
**Hay valores atípicos tanto en el extremo inferior como en el superior.**

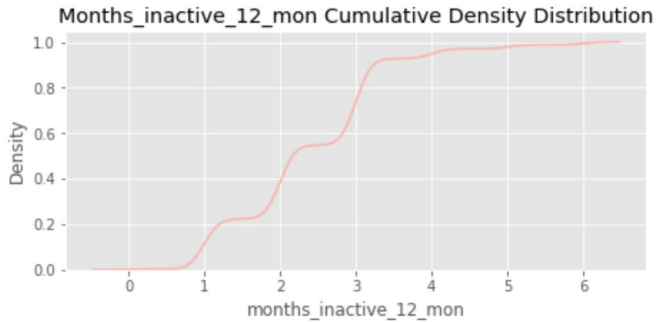
# Months_Inactive_12_mon

Nº de meses inactivos en los últimos 12 meses



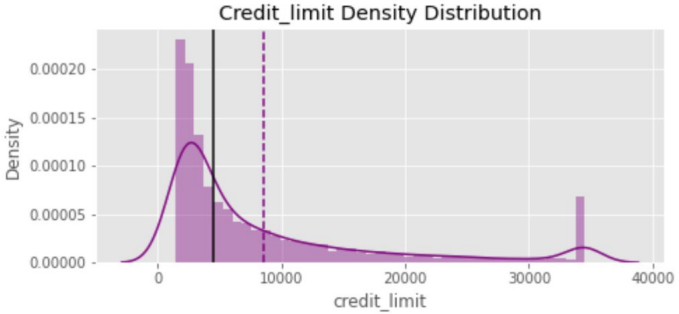



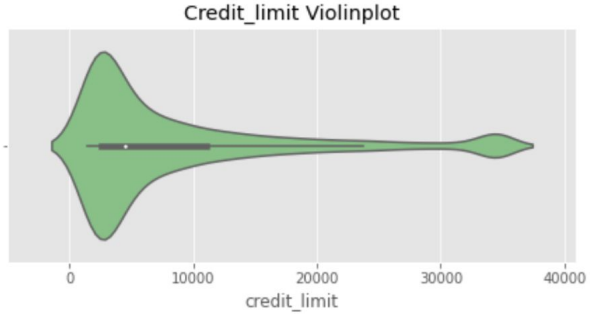



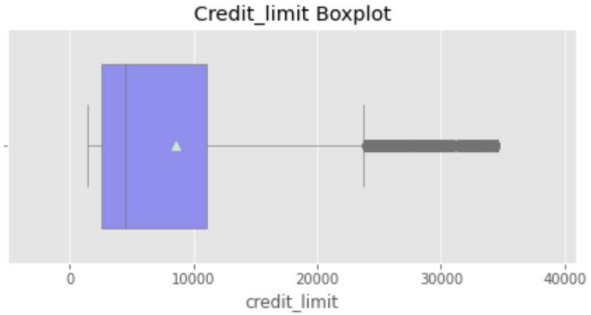

**Los clientes que están inactivos durante 5 o más meses son los que deben preocupar.**

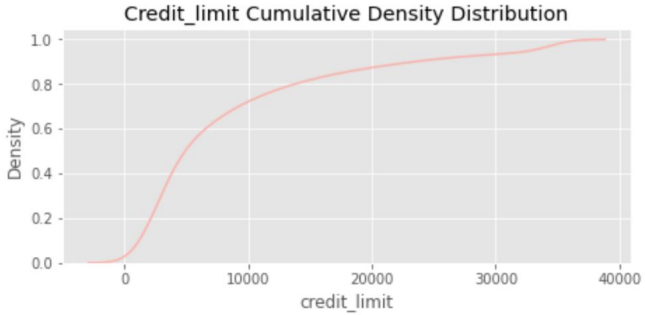
# Credit_limit

## Límite de crédito de la tarjeta de crédito



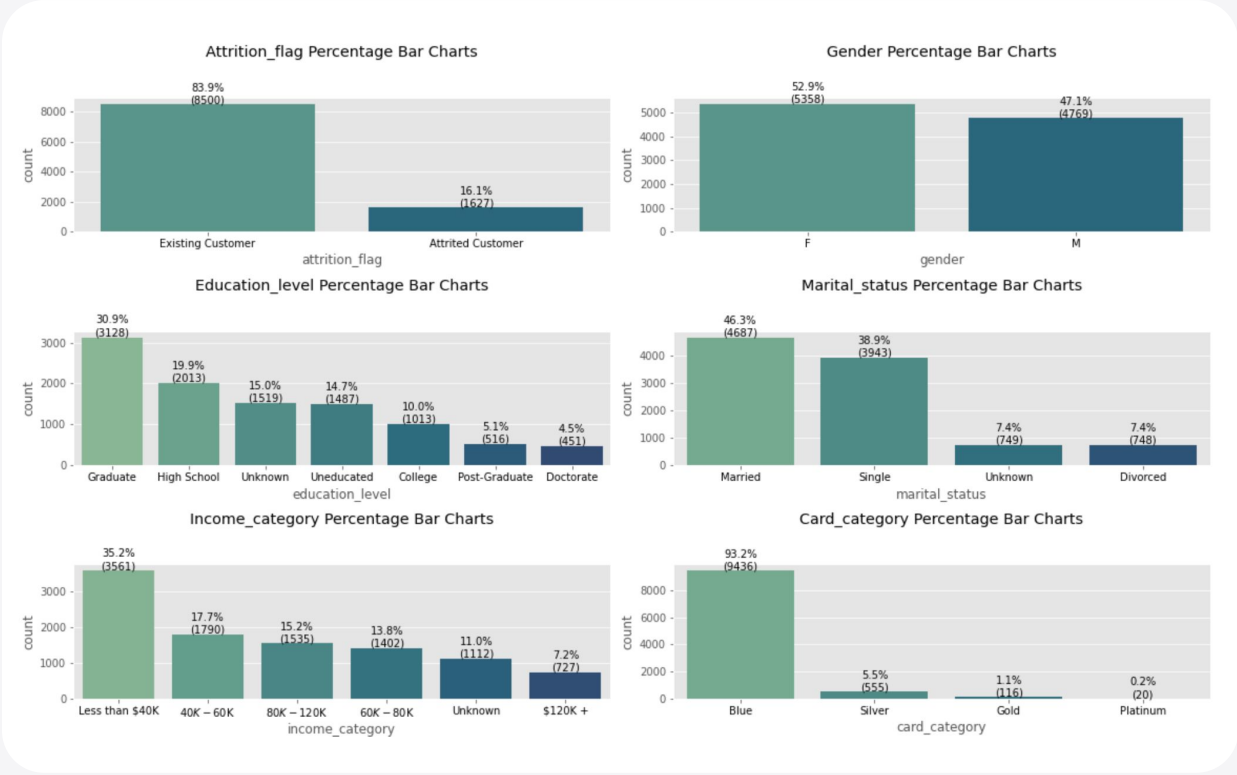




**Hay valores atípicos de gama alta en el Límite de Crédito. Esto puede deberse a que los clientes son de gama alta.**

# Porcentaje en el gráfico de barras para características categóricas

**Para las variables categóricas, es mejor analizarlas en porcentaje del total en gráficos de barras.**

Tomamos una columna de categorías como entrada y trazamos un gráfico de barras con porcentajes en la parte superior de cada barra.

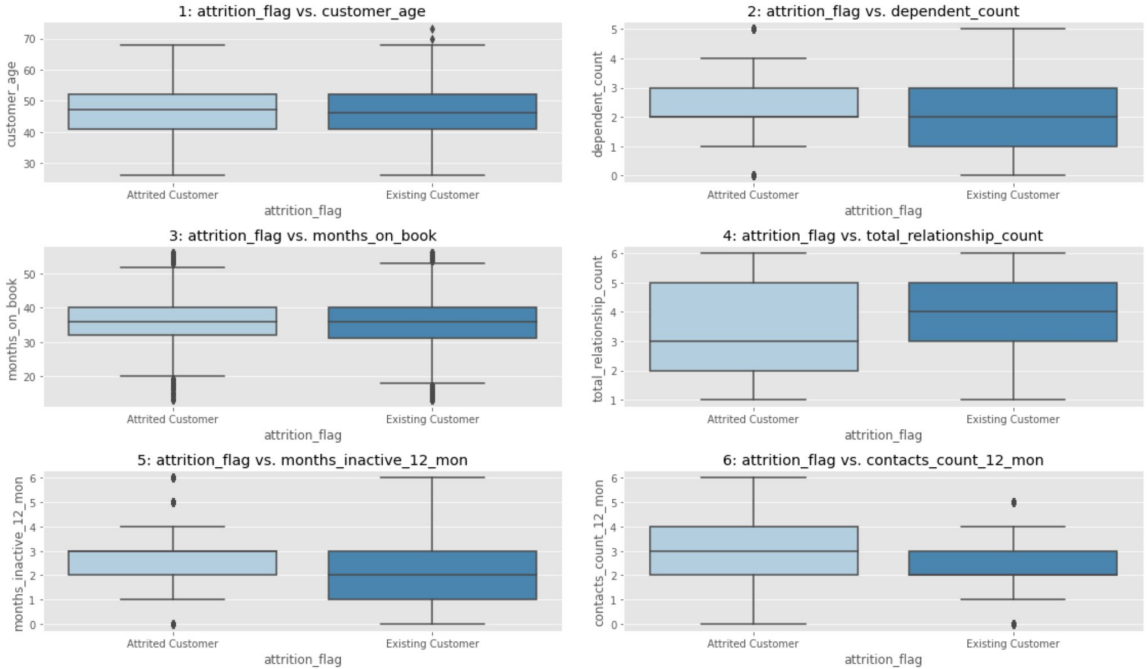


# Porcentaje en el gráfico de barras para características categóricas

- **Gran desequilibrio en los datos**, ya que la proporción de clientes existentes frente a los abandonados es de 84:16
- Los datos se distribuyen **casi por igual** entre hombres y mujeres
- El **31%** de los clientes son **graduados**
- El **85%** de los clientes son **solteros o casados**, de los cuales el **46,7%** están **casados**.
- El **35%** de los clientes **gana menos de 40.000 dólares** y el **36% gana 60.000 dólares o más**
- El **93%** de los clientes tiene **tarjeta azul**

# Análisis exploratorio de datos

## Análisis bivariante




**El objetivo del análisis bivariado es encontrar las interdependencias entre las características.**

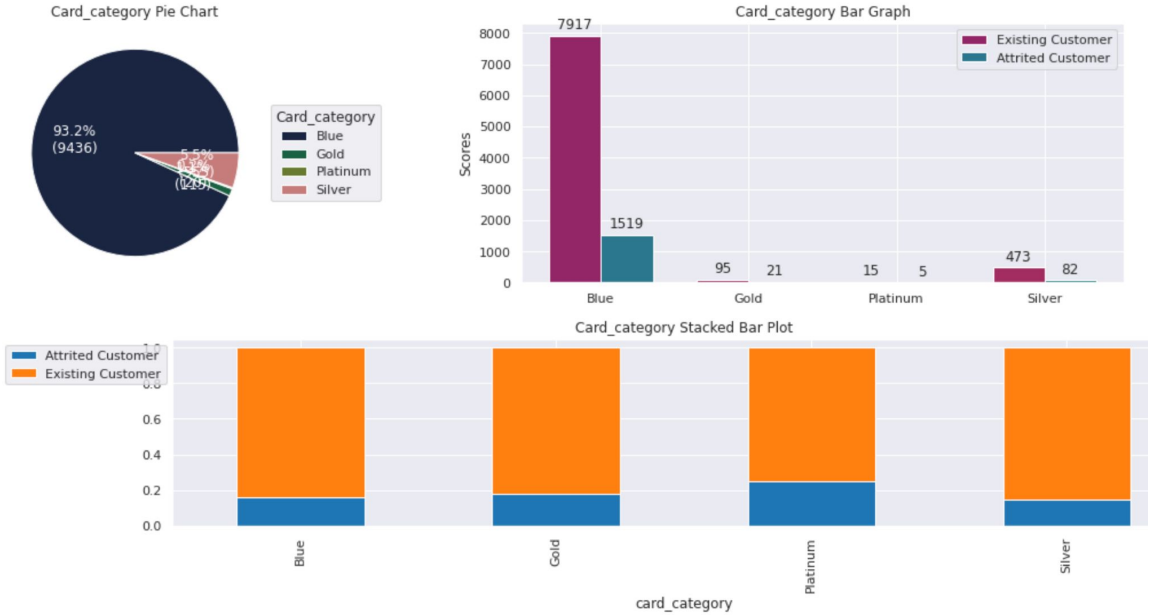
# Las clientes que se retiran tienen:

- Menor cantidad de transacciones totales
- Menor número de transacciones totales
- Menor índice de utilización
- Menor número de transacciones entre el cuarto y el primer trimestre
- Mayor número de veces de contacto con o por el banco

# Variables categóricas

**Creamos una función que devuelve un gráfico circular y un gráfico de barras para las variables categóricas.**

El titular de la tarjeta platino parece tener tendencia a la deserción, sin embargo, como sólo hay 20 puntos de datos para los titulares de la tarjeta platino, esta observación estaría sesgada.



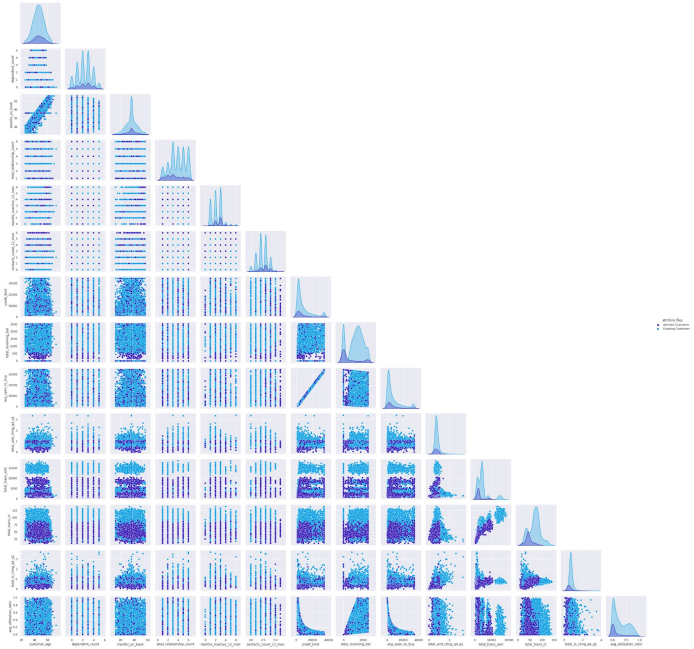
# Análisis exploratorio de datos

## Análisis multivariado

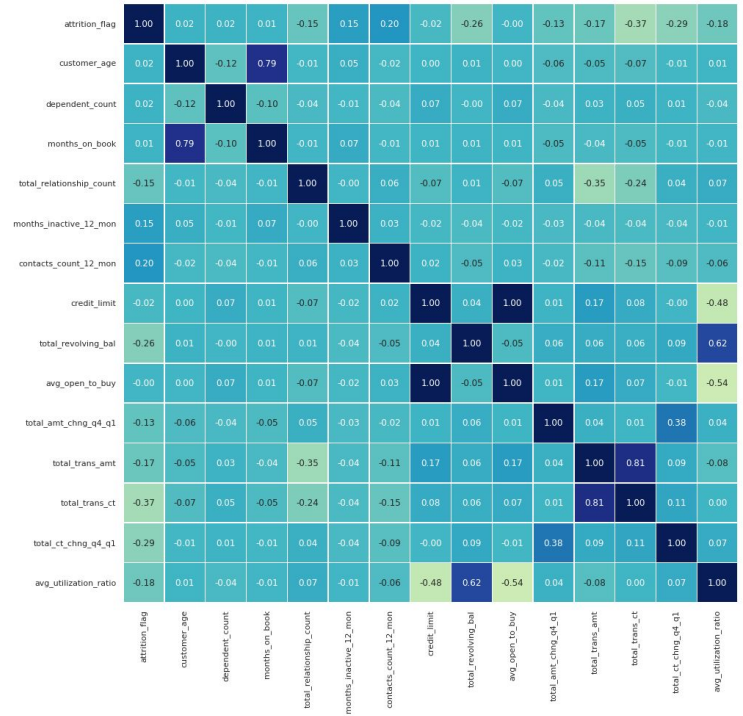
**Pairplot de todas las columnas numéricas disponibles, matizadas por Préstamo Personal.**

Analizamos las correlaciones entre las características numéricas del conjunto de datos.

Se han formado grupos con respecto a la deserción para las variables importe total renovado, cambio del importe total del cuarto al primer trimestre, importe total de las transacciones, recuento total de las transacciones, cambio del recuento total de las transacciones del cuarto al primer trimestre.



# Mapa de calor




**Generamos un mapa de calor para comprender las correlaciones entre las variables independientes y dependientes.**

También hay una fuerte correlación entre algunas columnas, que podemos comprobar en el mapa de correlación.

# Correlaciones entre las características numéricas del conjunto de datos

- El **límite de crédito** y la **apertura media de compra** tienen una colinealidad del 100%.
- **Meses en cartera** y **Edad del cliente** tienen una correlación bastante fuerte
- Parece que el **índice de utilización media** y el **saldo total rotativo** también están un poco correlacionados
- La **Bandera de Desgaste** no tiene una correlación muy fuerte con ninguna de las variables numéricas
- La **pérdida de clientes** no parece estar correlacionada con la edad de los clientes, el recuento de dependientes, los meses en cartera, la apertura a la compra y el límite de crédito, por lo que los eliminaremos del conjunto de datos.

# Procesamiento de datos

**Pasos de preprocesamiento:**

- Dividir los datos en conjuntos dependientes y objetivo
- Dividir los datos en conjuntos de entrenamiento, prueba y validación
- Estandarización de los nombres de las características
- Eliminación de las columnas innecesarias (número de cliente, edad del cliente, recuento de dependientes, meses en cartera, abierto a la compra, límite de crédito)
- Tratamiento de valores perdidos/incorrectos
- Codificación
- Tratamiento de la escala/los valores atípicos

**En primer lugar, trabajaremos en la construcción de modelos individualmente después del preprocesamiento de datos, y más tarde construiremos un pipeline de ML para ejecutar el proceso de preprocesamiento y construcción de modelos de principio a fin.**

# Procesamiento de datos

**El preprocesamiento de datos es una de las partes más importantes del trabajo antes de empezar a entrenar el modelo con el conjunto de datos.**

Tenemos que imputar los valores que faltan, arreglar cualquier valor ilógico de los datos en las columnas, convertir las columnas de categorías en numéricas (ya sean ordinales o binarias utilizando la codificación de un punto), escalar los datos para tratar la asimetría de la distribución y los valores atípicos, antes de alimentar los datos a un modelo.

Estamos utilizando las clases de transformación pre-disponibles y las clases personalizadas que hemos creado para ajustar primero los datos de entrenamiento y luego transformar el conjunto de datos de entrenamiento, validación y prueba. Esta es la práctica lógica estándar para mantener la influencia de los datos de prueba y validación en el conjunto de datos de entrenamiento para prevenir/evitar la fuga de datos mientras se entrena o valida el modelo.

# Ya tenemos todo listo para construir, entrenar y validar el modelo

# Consideraciones sobre la construcción del modelo

## Criterio de evaluación del modelo

**El modelo puede hacer predicciones erróneas como:**

- Predecir que un cliente se va a dar de baja y el cliente no se da de baja - Pérdida de recursos
- Predecir que un cliente no se va a dar de baja y el cliente se da de baja - Pérdida de la oportunidad de cambiar al cliente

**¿Qué caso es más importante?:**

- Predecir que un cliente no se va a dar de baja, pero realmente se da de baja, supondría una pérdida para el banco, ya que si se hubiera predicho correctamente, el equipo de marketing/ventas podría haber contactado con el cliente para retenerlo. Esto supondría pérdidas. Por tanto, hay que minimizar los falsos negativos.

# Consideraciones sobre la construcción del modelo

## Criterio de evaluación del modelo

**¿Cómo reducir esta pérdida, es decir, cómo reducir los falsos negativos?:**

- La empresa quiere maximizar el Recall, ya que cuanto mayor sea el Recall, menores serán las posibilidades de falsos negativos.

**Empezamos por construir diferentes modelos utilizando KFold y cross_val_score y afinamos el mejor modelo utilizando RandomizedSearchCV.**

# Consideraciones sobre la construcción del modelo

**La validación cruzada K-Folds estratificada proporciona índices de conjuntos de datos para dividirlos en conjuntos de entrenamiento / validación.** Divide el conjunto de datos en k pliegues consecutivos (sin barajar por defecto) manteniendo la distribución de ambas clases en cada pliegue igual que la variable objetivo.

Cada pliegue se utiliza una vez como validación mientras que los k - 1 pliegues restantes forman el conjunto de entrenamiento. Funciones de evaluación del modelo - Puntuación y matriz de confusión. Creamos algunas funciones para puntuar los modelos, mostrar la matriz de confusión.

# Construcción de modelos

Hacemos 9 modelos:

**Regresión Logística**
**Bagging**
**Random Forest**
**Gradient Boosting**
**Ada Boosting**
**Extreme Gradient Boosting**
**Decision Tree**
**KNN**
**Light Gradient Boosting**

## Construir y entrenar modelos

Construimos los siguientes 8 modelos:

- Bagging
- Clasificación Random Forest
- Máquina Gradient Boosting
- Boosting adaptativo
- eXtreme Gradient Boosting
- Clasificación de árboles de decisión (árboles de clasificación y regresión - CART)
- Máquina de refuerzo de gradiente ligero
- Regresión logística
- KNN

# Construcción de modelos

Light GBM es un marco de refuerzo de gradiente rápido, distribuido y de alto rendimiento basado en el algoritmo de árbol de decisión, que se utiliza para la clasificación y muchas otras tareas de aprendizaje automático.

Dado que se basa en algoritmos de árboles de decisión, divide el árbol en función de las hojas con el mejor ajuste, mientras que otros algoritmos de refuerzo dividen el árbol en función de la profundidad o el nivel, en lugar de en función de las hojas.

Por lo tanto, al crecer en la misma hoja en Light GBM, el algoritmo por hojas puede reducir más pérdidas que el algoritmo por niveles y, por lo tanto, da como resultado una precisión mucho mayor que rara vez puede ser alcanzada por cualquiera de los algoritmos de refuerzo existentes.

**Creamos un diagrama elaborado por los creadores del GBM ligero para explicar claramente la diferencia.**

# Diagrama

| | Model | Cross_Val_Score_Train | Train_Accuracy | Test_Accuracy | Train_Recall | Test_Recall | Train_Precision | Test_Precision | Train_F1 | Test_F1 | Train_ROC_AUC | Test_ROC_AUC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | Light GBM | 0.844267 | 0.999342 | 0.976308 | 0.998975 | 0.929231 | 0.996933 | 0.923547 | 0.997953 | 0.926380 | 0.999988 | 0.993571 |
| 2 | GBM | 0.817620 | 0.969712 | 0.966930 | 0.873975 | 0.876923 | 0.933260 | 0.913462 | 0.902646 | 0.894819 | 0.992689 | 0.989508 |
| 4 | Xgboost | 0.809426 | 0.966255 | 0.968904 | 0.849385 | 0.880000 | 0.934611 | 0.922581 | 0.889962 | 0.900787 | 0.992539 | 0.990775 |
| 3 | Adaboost | 0.799137 | 0.956379 | 0.958045 | 0.830943 | 0.864615 | 0.890231 | 0.872671 | 0.859565 | 0.868624 | 0.987073 | 0.985719 |
| 0 | Bagging | 0.785862 | 0.996049 | 0.955084 | 0.980533 | 0.846154 | 0.994802 | 0.870253 | 0.987616 | 0.858034 | 0.999899 | 0.974716 |
| 1 | Random forest | 0.770440 | 1.000000 | 0.954590 | 1.000000 | 0.803077 | 1.000000 | 0.903114 | 1.000000 | 0.850163 | 1.000000 | 0.989199 |
| 5 | dtree | 0.754113 | 1.000000 | 0.929911 | 1.000000 | 0.821538 | 1.000000 | 0.760684 | 1.000000 | 0.789941 | 1.000000 | 0.886078 |
| 7 | Logistic Regression | 0.562487 | 0.901235 | 0.904245 | 0.563525 | 0.600000 | 0.759669 | 0.752896 | 0.647059 | 0.667808 | 0.920375 | 0.934017 |
| 8 | KNN | 0.502051 | 0.922798 | 0.902270 | 0.597336 | 0.501538 | 0.884674 | 0.819095 | 0.713150 | 0.622137 | 0.968172 | 0.925656 |

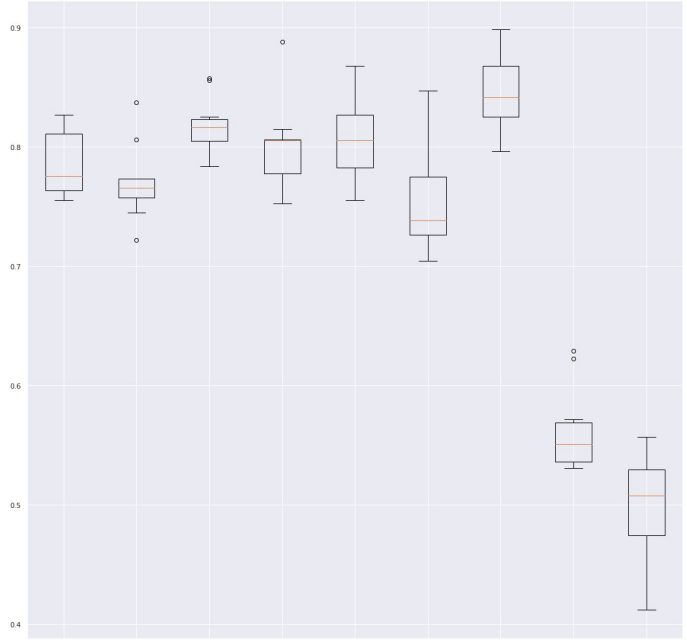
El mejor modelo con respecto a la puntuación de la validación cruzada y la recuperación de la prueba es Light GBM.
Los siguientes mejores modelos son XGBoost, GBM y AdaBoost respectivamente.

# Trazado de la comparación de los resultados de la validación cruzada

**Trazamos los resultados de la validación cruzada de los 7 modelos en un gráfico de cajas, para comprobar qué modelos son potencialmente buenos.**

Parece que Light GBM, XGBoost, GBM son los modelos con buen potencial.

Ada Boost también tiene buena pinta con la puntuación de rendimiento más alta de los extremos.



# Sobremuestreo de datos de entrenamiento usando SMOTE

Nuestro conjunto de datos tiene un gran desequilibrio en las etiquetas de las variables objetivo. Para hacer frente a este tipo de conjuntos de datos, tenemos algunos trucos, que llamamos Clasificación desequilibrada.

La clasificación desequilibrada consiste en desarrollar modelos predictivos en conjuntos de datos de clasificación que presentan un grave desequilibrio de clases.

El reto de trabajar con conjuntos de datos desequilibrados es que la mayoría de las técnicas de aprendizaje automático ignorarán, y a su vez tendrán un rendimiento pobre en la clase minoritaria.

Aunque normalmente es el rendimiento en la clase minoritaria lo más importante, que es el caso de nuestro estudio aquí.

# Sobremuestreo de datos de entrenamiento usando SMOTE

Una forma de abordar los conjuntos de datos desequilibrados es sobremuestrear la clase minoritaria.

El enfoque más sencillo consiste en duplicar los ejemplos de la clase minoritaria, aunque estos ejemplos no añaden ninguna información nueva al modelo.

En cambio, se pueden sintetizar nuevos ejemplos a partir de los existentes.

Este es un tipo de aumento de datos para la clase minoritaria y se denomina Técnica de Sobremuestreo Sintético de Minorías, o SMOTE para abreviar.

# Construir modelos con datos sobremuestreados

## Construir y entrenar modelos

Construimos y entrenamos los mismos 7 modelos que antes.

Sin embargo, utilizamos los datos de entrenamiento sobremuestreados para entrenar los modelos.

Mediante el Análisis de Componentes Principales (ACP), reducimos la dimensionalidad del conjunto de datos lo más "pequeño" posible sin perder ninguna información.

| | Model | Cross_Val_Score_Train | Train_Accuracy | Test_Accuracy | Train_Recall | Test_Recall | Train_Precision | Test_Precision | Tra |
|---|---|---|---|---|---|---|---|---|---|
| 6 | Light GBM | 0.844267 | 0.999342 | 0.976308 | 0.998975 | 0.929231 | 0.996933 | 0.923547 | 0.9 |
| 15 | Light GBM UpSampling | 0.977642 | 0.998137 | 0.967917 | 0.999216 | 0.920245 | 0.997065 | 0.884956 | 0.9 |
| 11 | GBM UpSampling | 0.967837 | 0.971269 | 0.957552 | 0.974701 | 0.914110 | 0.968056 | 0.837079 | 0.9 |
| 12 | Adaboost UpSampling | 0.956857 | 0.952442 | 0.942251 | 0.959012 | 0.911043 | 0.946574 | 0.771429 | 0.9 |
| 13 | Xgboost UpSampling | 0.965092 | 0.967739 | 0.951629 | 0.971367 | 0.907975 | 0.964369 | 0.813187 | 0.9 |
| 17 | KNN UpSampling | 0.989604 | 0.942146 | 0.857848 | 0.997647 | 0.892638 | 0.897970 | 0.534926 | 0.9 |
| 4 | Xgboost | 0.809426 | 0.966255 | 0.968904 | 0.849385 | 0.880000 | 0.934611 | 0.922581 | 0.8 |
| 2 | GBM | 0.817620 | 0.969712 | 0.966930 | 0.873975 | 0.876923 | 0.933260 | 0.913462 | 0.9 |
| 3 | Adaboost | 0.799137 | 0.956379 | 0.958045 | 0.830943 | 0.864615 | 0.890231 | 0.872671 | 0.8 |
| 10 | Random forest UpSampling | 0.979801 | 1.000000 | 0.951629 | 1.000000 | 0.861963 | 1.000000 | 0.841317 | 1.0 |
| 9 | Bagging UpSampling | 0.959015 | 0.997352 | 0.941757 | 0.997058 | 0.846626 | 0.997645 | 0.802326 | 0.9 |
| 0 | Bagging | 0.785862 | 0.996049 | 0.955084 | 0.980533 | 0.846154 | 0.994802 | 0.870253 | 0.9 |
| 5 | dtree | 0.754113 | 1.000000 | 0.929911 | 1.000000 | 0.821538 | 1.000000 | 0.760684 | 1.0 |
| 14 | dtree UpSampling | 0.946265 | 1.000000 | 0.923001 | 1.000000 | 0.819018 | 1.000000 | 0.733516 | 1.0 |
| 1 | Random forest | 0.770440 | 1.000000 | 0.954590 | 1.000000 | 0.803077 | 1.000000 | 0.903114 | 1.0 |
| 16 | Logistic Regression UpSampling | 0.896057 | 0.895470 | 0.875123 | 0.898215 | 0.751534 | 0.893310 | 0.587530 | 0.8 |
| 7 | Logistic Regression | 0.562487 | 0.901235 | 0.904245 | 0.563525 | 0.600000 | 0.759669 | 0.752896 | 0.6 |
| 8 | KNN | 0.502051 | 0.922798 | 0.902270 | 0.597336 | 0.501538 | 0.884674 | 0.819095 | 0.7 |

# PCA Reducing Dimensions

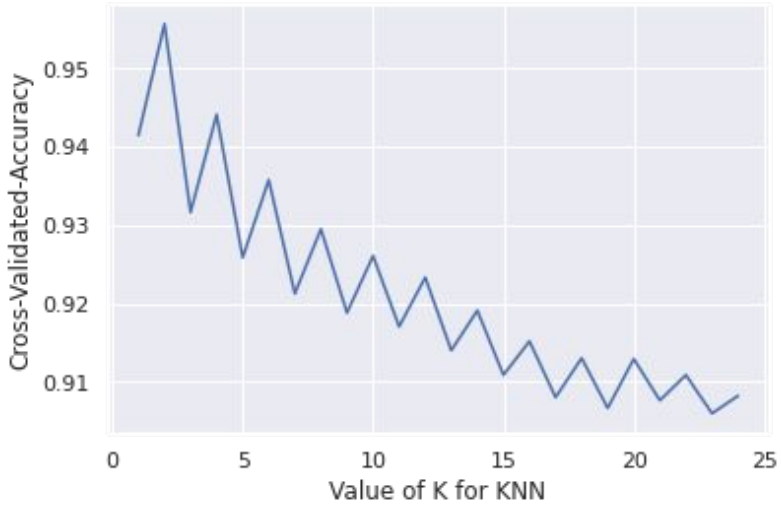

Shape of X before PCA: (10198, 27)

Shape of X after PCA: (10198, 16)

0.9258691720063885

# KNN

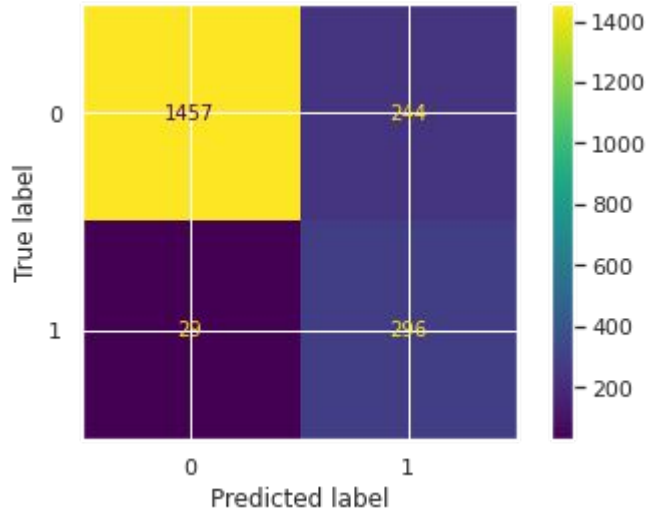
Text(0, 0.5, 'Cross-Validated-Accuracy')




tuned hyperparameter K: {'n_neighbors': 2}

tuned parameter (best score): 0.9556789624583887

# Decision Tree




[[1457 244]

[ 29 296]]

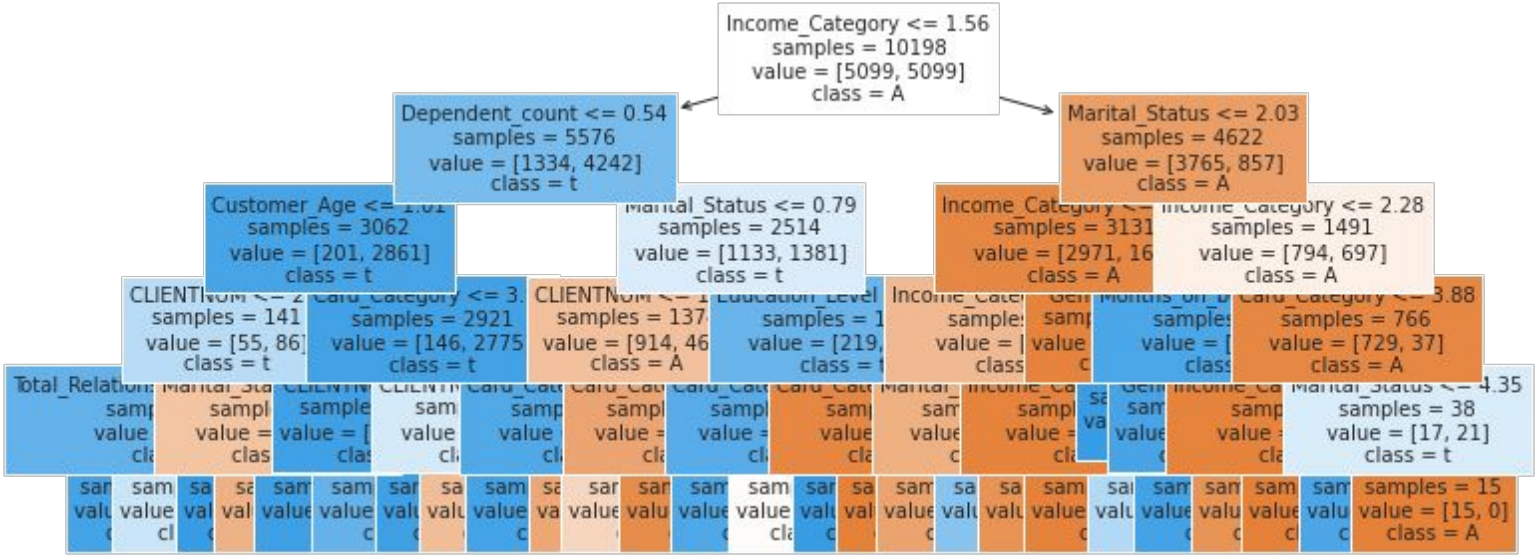
[[1457 244]

[ 29 296]]

We have identified succesfully 86.53% of all Attrited customers

# Confution Matrix

Profundidad del árbol: 5  Número de nodos terminales: 29



# Construir modelos con datos sobremuestreados

**Los 4 mejores modelos con respecto a la recuperación de la validación y la puntuación de la validación cruzada, son los siguientes:**

- GBM ligero entrenado con datos sobremuestreados/sobremuestreados
- GBM entrenado con datos sobremuestreados/sobremuestreados
- AdaBoost entrenado con datos sobre/muestreados
- XGBoost entrenado con datos sobre/muestreados

# Submuestreo de los datos del tren mediante el submuestreo aleatorio

**El submuestreo es otra forma de tratar el desequilibrio en el conjunto de datos.**

El submuestreo aleatorio consiste en seleccionar al azar ejemplos de la clase mayoritaria y eliminarlos del conjunto de datos de entrenamiento hasta crear un conjunto de datos equilibrado.

# Construir modelos con datos no muestreados

## Construir y entrenar modelos

Volvemos a construir los mismos 7 modelos que antes y entrenamos con el conjunto de datos submuestreados, y utilizamos el conjunto de datos de validación para puntuar los modelos.

| | Model | Cross_Val_Score_Train | Train_Accuracy | Test_Accuracy | Train_Recall | Test_Re |
|---|---|---|---|---|---|---|
| 21 | Adaboost DownSampling | 0.925174 | 0.947746 | 0.936328 | 0.952869 | 0.96 |
| 22 | Xgboost DownSampling | 0.946686 | 0.968238 | 0.939783 | 0.977459 | 0.96 |
| 20 | GBM DownSampling | 0.951799 | 0.967725 | 0.938796 | 0.979508 | 0.95 |
| 24 | Light GBM DownSampling | 0.950789 | 1.000000 | 0.940770 | 1.000000 | 0.95 |
| 19 | Random forest DownSampling | 0.935388 | 1.000000 | 0.928430 | 1.000000 | 0.93 |
| 18 | Bagging DownSampling | 0.920029 | 0.994365 | 0.924482 | 0.990779 | 0.93 |
| 6 | Light GBM | 0.844267 | 0.999342 | 0.976308 | 0.998975 | 0.92 |
| 15 | Light GBM UpSampling | 0.977642 | 0.998137 | 0.967917 | 0.999216 | 0.92 |
| 11 | GBM UpSampling | 0.967837 | 0.971269 | 0.957552 | 0.974701 | 0.97 |
| 12 | Adaboost UpSampling | 0.956857 | 0.952442 | 0.942251 | 0.959012 | 0.97 |
| 13 | Xgboost UpSampling | 0.965092 | 0.967739 | 0.951629 | 0.971367 | 0.90 |
| 17 | KNN UpSampling | 0.989604 | 0.942146 | 0.857848 | 0.997647 | 0.89 |
| 23 | dtree DownSampling | 0.896423 | 1.000000 | 0.891412 | 1.000000 | 0.88 |
| 4 | Xgboost | 0.809426 | 0.966255 | 0.968904 | 0.849385 | 0.88 |
| 26 | KNN | 0.849348 | 0.887807 | 0.857354 | 0.886270 | 0.87 |

# Construir modelos con datos no muestreados

**Los 4 mejores modelos son:**

- XGBoost entrenado con datos submuestreados
- AdaBoost entrenado con datos submuestreados
- GBM ligero entrenado con datos submuestreados
- Random forest entrenado con datos submuestreados

Ahora intentaremos afinar estos 4 modelos utilizando la búsqueda aleatoria CV.

# Elección de modelos para el ajuste

**XGBoost** con muestreo descendente tiene la mejor validación del 96,3%, junto con una puntuación de validación cruzada del 95% en el entrenamiento, y un AUC de 0,99, lo que significa que tiene grandes posibilidades de funcionar muy bien en un conjunto de datos no visto. Hay un poco de sobreajuste, que espero que se resuelva afinando.

**AdaBoost** generaliza muy bien el modelo, no se ajusta en exceso ni tiene ningún sesgo, el AUC es de 0,985 y la puntuación de validación cruzada en el entrenamiento es del 93%, el recuerdo en el conjunto de validación es el mismo que el de XGBoost (96,3%). Espero mejorar el modelo (~94% en el conjunto de validación) mediante el ajuste.

# Elección de modelos para el ajuste

**Light GBM** funciona muy bien en todos los aspectos, pero hay un ligero problema de sobreajuste, que espero resolver mediante el ajuste. La precisión en la validación es del 94%, con una puntuación de validación cruzada en el tren del 95%, la recuperación en la validación ~96%, el AUC es de 0,99. Parece un modelo muy prometedor.

El **GBM** no está sobreajustado, y tampoco sufre de sesgo o varianza. La recuperación en la validación es ~96%, la precisión en la validación ~94%, el AUC es ~0,99, la puntuación de la validación cruzada en el entrenamiento es ~95%. Esta sería mi mejor opción porque ninguna de las puntuaciones de entrenamiento es del 100%, lo que significa que no está tratando de explicar cada aspecto de los datos de entrenamiento sobreajustándolos.

# Ajuste del modelo mediante RandomizedSearchCV

Normalmente, un hiperparámetro tiene un efecto conocido en un modelo en sentido general, pero no está claro cómo establecer mejor un hiperparámetro para un conjunto de datos determinado. Además, muchos modelos de aprendizaje automático tienen una serie de hiperparámetros y pueden interactuar de forma no lineal.

Por ello, a menudo es necesario buscar un conjunto de hiperparámetros que den como resultado el mejor rendimiento de un modelo en un conjunto de datos.

Esto se denomina optimización de hiperparámetros, ajuste de hiperparámetros o búsqueda de hiperparámetros.

Un procedimiento de optimización implica definir un espacio de búsqueda. Éste puede considerarse geométricamente como un volumen de n dimensiones, en el que cada hiperparámetro representa una dimensión diferente y la escala de la dimensión son los valores que puede tomar el hiperparámetro, como los de valor real, los de valor entero o los categóricos.

# Ajuste del modelo mediante RandomizedSearchCV

**Espacio de búsqueda**: Volumen en el que se busca, donde cada dimensión representa un hiperparámetro y cada punto representa una configuración del modelo. Un punto en el espacio de búsqueda es un vector con un valor específico para cada valor del hiperparámetro.

El objetivo del procedimiento de optimización es encontrar un vector que dé como resultado el mejor rendimiento del modelo tras el aprendizaje, como la máxima precisión o el mínimo error.

Se pueden utilizar diferentes algoritmos de optimización, aunque dos de los métodos más sencillos y comunes son la búsqueda aleatoria y la búsqueda en cuadrícula.

**Búsqueda aleatoria**: Definir un espacio de búsqueda como un dominio acotado de valores de hiperparámetros y muestrear aleatoriamente puntos en ese dominio.

**Búsqueda en cuadrícula**: Definir un espacio de búsqueda como una cuadrícula de valores de hiperparámetros y evaluar cada posición de la cuadrícula.

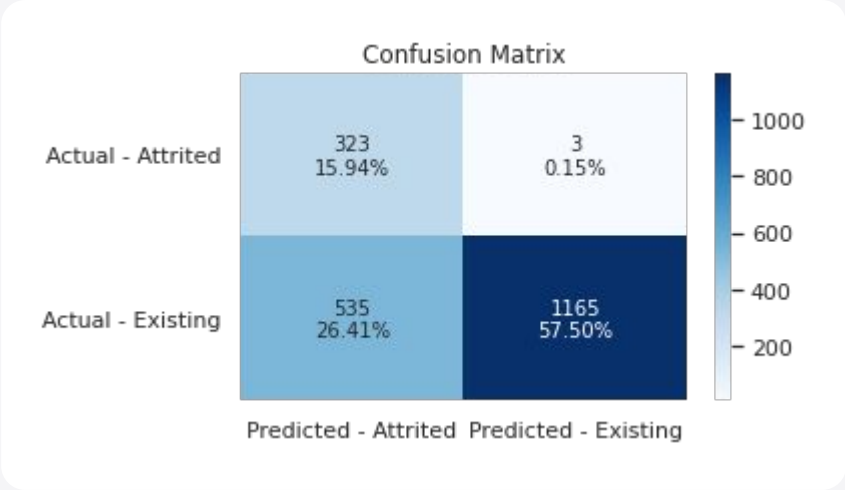
# Ajuste de XGBOOST con datos de muestreo descendente

**Encontrar el mejor parámetro para una alta recuperación utilizando la búsqueda aleatoria con validación cruzada.**

## Confusion Matrix AdaBoost

**Ajuste de AdaBoost con datos de muestreo descendente.**

Encontrar el mejor parámetro para una alta recuperación utilizando la búsqueda aleatoria con validación cruzada.

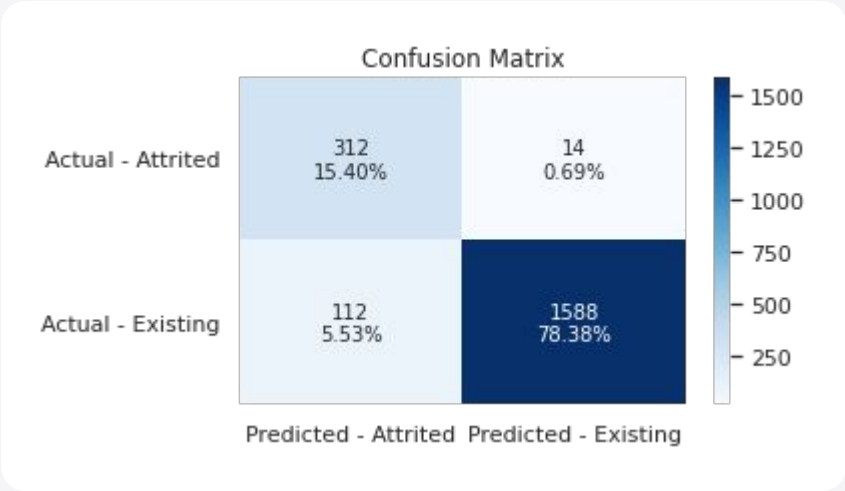


# Ajuste de XGBOOST con datos de muestreo descendente

## Confusion Matrix GBM ligero

**Ajuste de GBM ligero con datos muestreados.**

Encontrar el mejor parámetro para una alta recuperación utilizando la búsqueda aleatoria con validación cruzada.

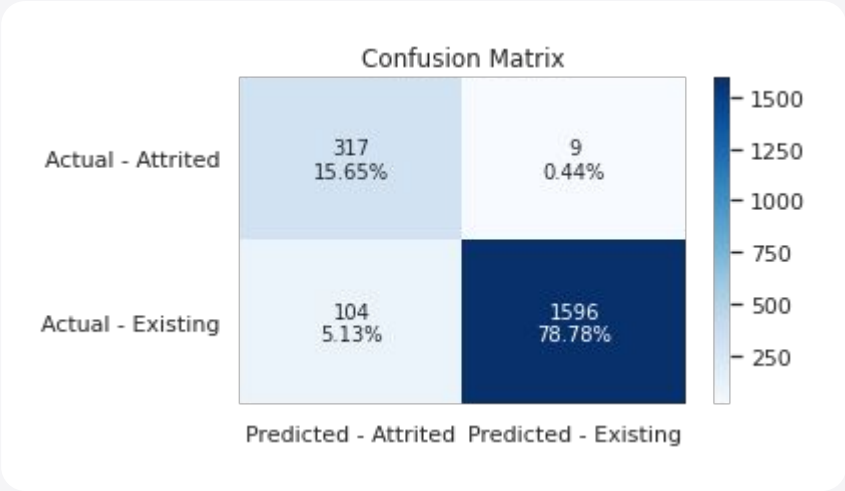


# Ajuste de XGBOOST con datos de muestreo descendente

## Confusion Matrix GBM

**Ajuste de GBM con datos de muestreo descendente.**

Encontrar el mejor parámetro para una alta recuperación utilizando la búsqueda aleatoria con validación cruzada.



# Selección del modelo final

El modelo **XGBoost** con ajuste de hiperparámetros y entrenado con un conjunto de datos submuestreados, tiene la mejor recuperación en el conjunto de validación de ~99%, pero la precisión es inferior a la precisión a nivel humano (es decir, clasificar a todos como clientes no expulsados).

Por lo tanto, no seleccionamos este modelo como modelo final.

El modelo de validación de ~97% es proporcionado por el **GBM** con ajuste de hiperparámetros entrenado con un conjunto de datos submuestreados, tiene una exactitud de validación de ~94%, y una precisión de ~74%, AUC de validación ~99%, media de validación cruzada de 96%.

Además, el modelo no sufre de sesgo, ni de varianza.

**Estamos seleccionando el modelo GBM Tuned with Down Sampling como nuestro modelo final.**

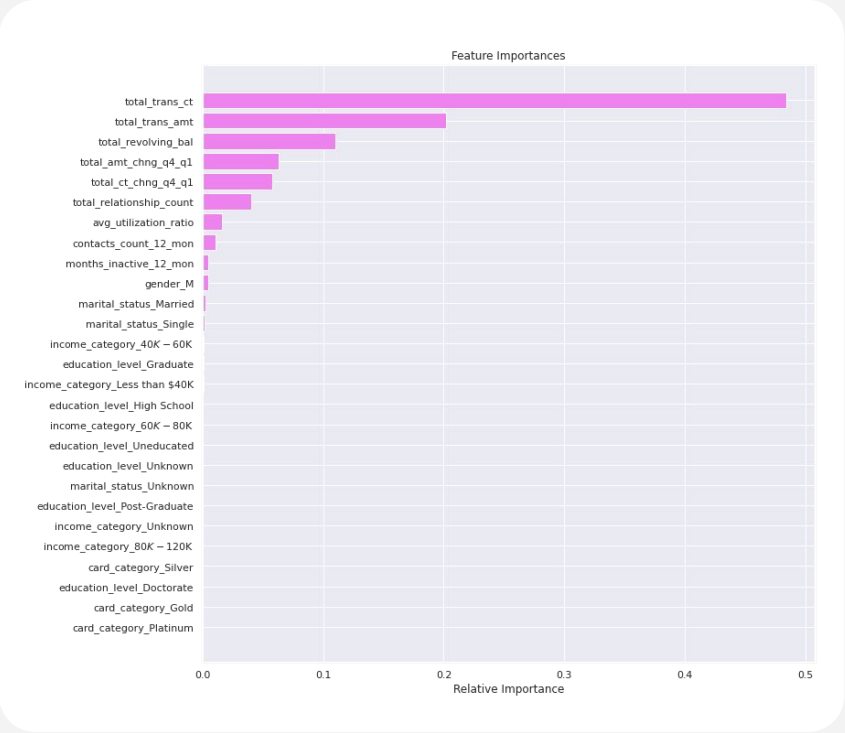
# Importancia de las características

Comprobar los datos de prueba en el GBM ajustado y entrenado con datos desmuestreados.

**Resultados de la prueba:**

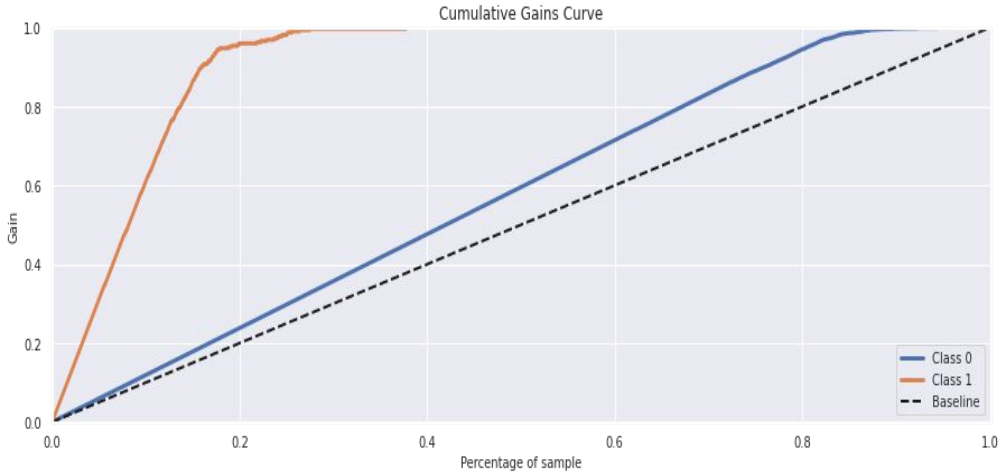
Comprobamos el rendimiento del modelo en el conjunto de datos de prueba (no vistos).

El rendimiento del modelo con los datos de prueba es casi similar al rendimiento en el conjunto de datos de validación.



# Gráfico de elevación/ganancia acumulada

El gráfico de elevación/ganancia acumulada es importante para entender cómo se comportaría un modelo en un sistema de producción con datos no vistos.




**El gráfico muestra que si clasificamos a los clientes en orden descendente de probabilidad de deserción (clase 1) y nos dirigimos al 30% de la población, es muy probable que encontremos al 100% de las personas que realmente se desertarían.**

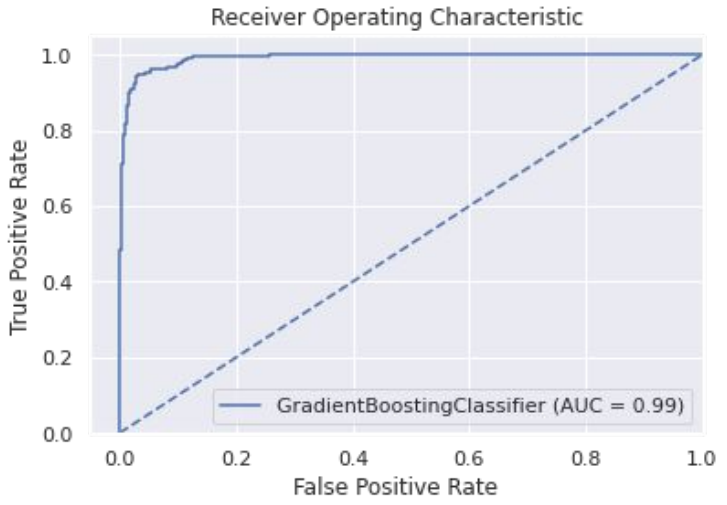
# Curva ROC-AUC

La característica ROC AUC es importante para entender lo bueno que es el modelo.

Si el modelo es realmente bueno en la identificación de las clases, el área bajo la curva es realmente alta, cercana a 1.

Si el modelo no puede distinguir bien las clases, el área bajo la curva es muy baja, cercana a 0,5.

**Nuestro modelo parece ser realmente bueno, ya que el AUC es casi 1.**



# Producción del modelo

Ahora que hemos finalizado nuestro modelo, **construiremos una canalización del modelo** para agilizar todos los pasos de la construcción del mismo. Empezaremos con el conjunto de datos inicial y seguiremos con los pasos de construcción del pipeline.

El pipeline de aprendizaje automático (ML), en teoría, representa diferentes pasos que incluyen la transformación de datos y la predicción a través de los cuales pasan los datos. **El resultado del pipeline es el modelo entrenado que puede ser utilizado para hacer las predicciones.**

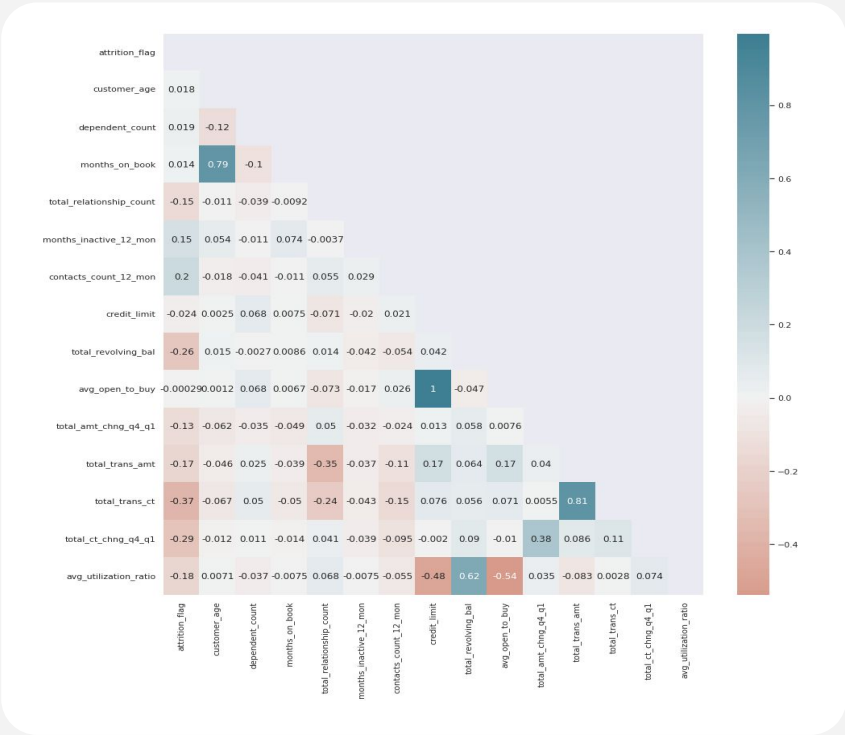
Sklearn.pipeline es una implementación en Python del pipeline de ML. En lugar de pasar por los pasos de ajuste del modelo y transformación de datos para los conjuntos de datos de entrenamiento y de prueba por separado, podemos utilizar Sklearn.pipeline para automatizar estos pasos.

Aquí está el diagrama que representa la tubería para la formación de nuestro modelo de aprendizaje automático basado en el aprendizaje supervisado, y luego usar los datos de prueba para predecir las etiquetas.

# Producción del modelo

Las características más importantes para entender el churn de las tarjetas de crédito de los clientes, son:

- Recuento total de transacciones
- Importe total de las transacciones
- Saldo rotativo total
- Cambio del importe total del cuarto al primer trimestre
- Cambio del recuento total del cuarto al primer trimestre
- Recuento total de relaciones



**Todas estas características están negativamente correlacionadas con el indicador de desgaste, lo que significa que cuanto más bajos sean los valores de estas características, mayores serán las posibilidades de que un cliente se desgaste.**

**El banco debería conectar con el cliente más a menudo para aumentar la conexión, y proporcionarle varias ofertas y planes para aumentar las relaciones del cliente con el banco.**

**El banco debería ofrecer planes de devolución de dinero en efectivo en las tarjetas de crédito, lo que podría animar a los clientes a utilizar la tarjeta de crédito más a menudo.**

**El banco también debería ofrecer un aumento del límite de crédito a los clientes que utilizan regularmente la tarjeta de crédito. Esto debería aumentar los gastos/transacciones de la tarjeta de crédito.**

**Otra oferta que se puede ofrecer a los clientes es un EMI al 0% de interés para animarles a comprar productos de mayor coste con la tarjeta de crédito y convertir el gasto en EMI, de modo que el importe total de las transacciones y el número de transacciones aumenten. El saldo también giraría muy bien.**

**Además de los tipos de tarjetas disponibles, el banco puede introducir tarjetas de crédito específicas para compras en línea (con ofertas de devolución de dinero en efectivo) o para pedidos de comida en línea. De este modo, la tarjeta se utilizará con más frecuencia.**

**Con nuestro modelo, podemos predecir qué clientes son propensos a la deserción, y según la probabilidad prevista, se puede contactar con al menos el 20-30% de los clientes más importantes para discutir ofertas de tarjetas de crédito, aumento del límite de crédito, etc., para intentar retener a esos clientes.**

# ¡Gracias!