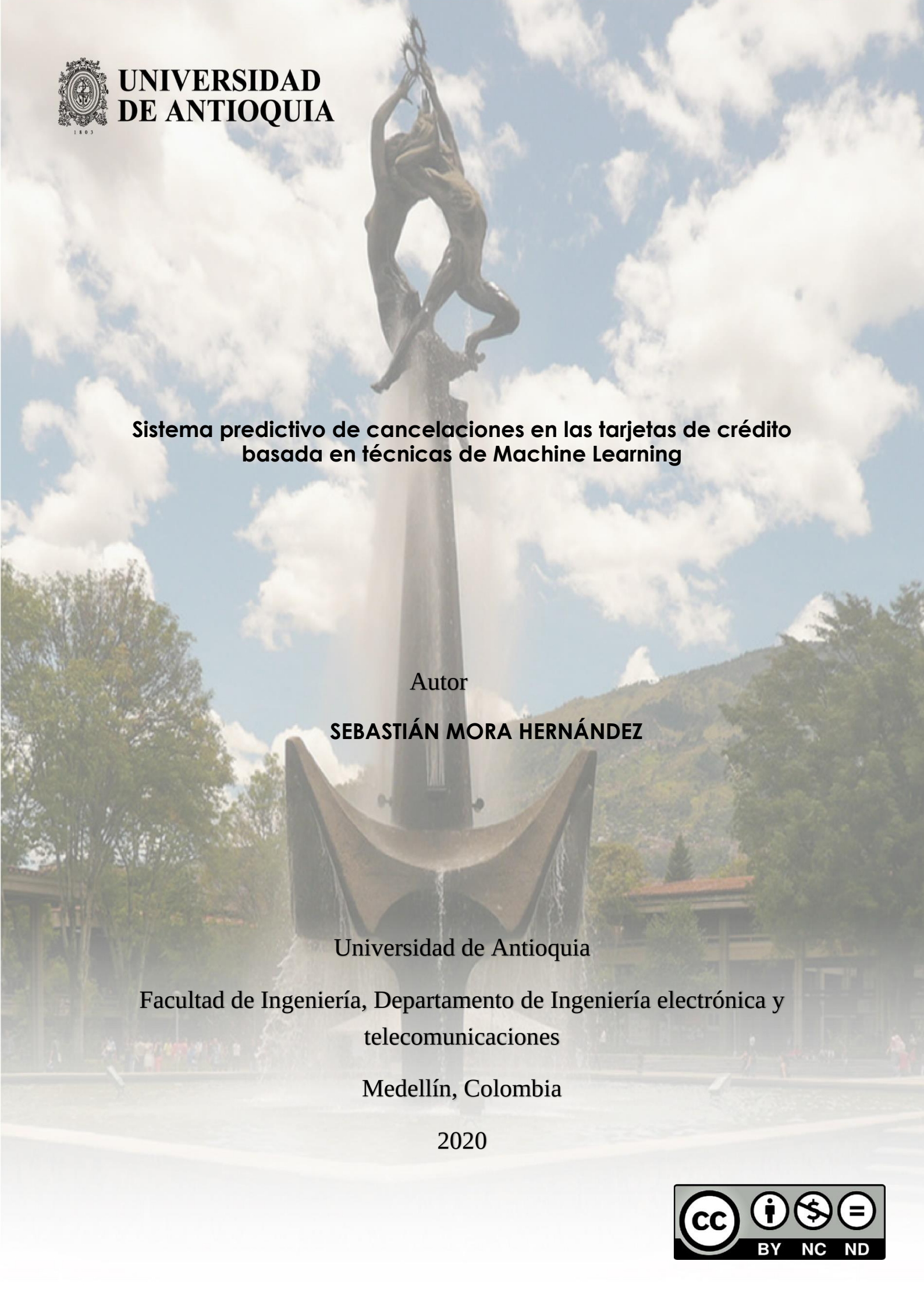


UNIVERSIDAD DE ANTIOQUIA

Sistema predictivo de cancelaciones en las tarjetas de crédito basada en técnicas de Machine Learning

Autor

SEBASTIÁN MORA HERNÁNDEZ

Universidad de Antioquia

Facultad de Ingeniería, Departamento de Ingeniería electrónica y telecomunicaciones

Medellín, Colombia

2020

Sistema predictivo de cancelaciones en las tarjetas de crédito basada en técnicas de *Machine Learning*

SEBASTIÁN MORA HERNÁNDEZ

Informe de práctica como requisito para optar al título de:
Ingeniero electrónico.

Asesores

CLAUDIA VICTORIA ISAZA NARVÁEZ
Profesora Facultad de Ingeniería

DILLAN ALEXIS MUÑETÓN
Analista III, Trans y Soporte Inteligencia de Negocios

Universidad de Antioquia
Facultad de Ingeniería. Departamento de ingeniería electrónica y de telecomunicaciones
Medellín, Colombia
2020.

Resumen

El proyecto consistió en la creación de un sistema predictivo de las cancelaciones de las tarjetas de crédito de los clientes, en donde se probaron 15 modelos con diferentes algoritmos y diferentes configuraciones, el primer modelo es de tipo estadístico y se obtuvo con Regresión logística, también se implementaron dos algoritmos de *Machine Learning* los cuales fueron arboles de decisión y *Random Forest*. Dado que se trabajó con una base de datos con clases desbalanceadas, se generaron diferentes modelos usando técnicas de balanceo de datos obteniendo mejores resultados respecto a los modelos sin utilizar técnicas de balanceo de datos. Las técnicas usadas fueron el método de *SMOTE*, método de sobre muestreo el cual crea muestras sintéticas de los datos haciendo que la clase minoritaria tenga el mismo número de muestras que la clase mayoritaria. La segunda técnica fue el método de *NEARMISS*, método de submuestreo que iguala la cantidad de muestras de la clase mayoritaria a la minoritaria. Los resultados obtenidos para el algoritmo de regresión logística fueron de un 79% de predicción en la cancelación de las tarjetas de crédito. Para el algoritmo de *Random Forest* se obtuvo un porcentaje de predicción del 84% y para el algoritmo de árboles de decisión se obtuvo un rendimiento del 74%. Una vez que se encontró el modelo que tenía mejor rendimiento, se procedió a integrar del modelo predictivo en un *Dashboard* de *Power BI* en donde se pudo observar el desempeño del modelo y, además, el poder aprovechar todas las funcionalidades de *Power BI* para que el cliente final pueda observar los resultados del modelo predictivo y análisis descriptivos de *Python* en *Power BI*.

Introducción

Una de las principales actividades que hacen los integrantes de las células del *Full Stack* de inteligencia de negocios de Bancolombia, es la elaboración de pruebas de concepto o PoC. Las pruebas de concepto son una posible implementación de un concepto o una idea, llevada a cabo de manera resumida, en donde se busca verificar que todas ellas sean viables para su implementación, buscando algún beneficio en el negocio del banco. Estas pruebas de concepto son llevadas a cabo frecuentemente y son orientadas a varias tecnologías de Microsoft como la suite de *SQL Server* y *Microsoft Power BI*.

En el área de inteligencia de negocio, se tuvo la necesidad de analizar la información comercial relacionada con las tarjetas de crédito de los clientes durante un periodo determinado. Si bien, el área ya contaba con un modelo analítico que permitía predecir estas cancelaciones, lo que no contaba este modelo era poder identificar cuáles son los posibles factores que influyen en la cancelación de las tarjetas de

crédito, el poder identificar estos factores es de vital importancia para que el banco implemente medidas para disminuir el número de tarjetas de crédito canceladas en un futuro cercano. Este modelo analítico era un modelo con una gran cantidad de datos que el banco recopila por medio de sus modelos analíticos y que son de difícil acceso por lo que se consideró utilizar otras alternativas que arrojen resultados similares al modelo en cuestión basándose en la selección de características más relevantes [1]. Para solucionar el inconveniente de la creación del nuevo modelo predictivo, surgió la propuesta de utilizar *Power Bi* como herramienta de visualización de datos junto con *Python*, en el caso de *Power BI*, este permite la integración de *Python*, se puede ejecutar scripts directamente en *Power BI* Desktop e importar los datos resultantes en un modelo de datos, es decir, se hace el análisis con *Python* y se visualiza los resultados por medio de un *Dashboard* de *Power BI*. Esta solución benefició a las personas que trabajan en el área de inteligencia de negocios, ya que pudieron acceder a los resultados de este modelo predictivo de una manera más amigable y práctica y, además, se tomaron una serie de decisiones que beneficiaron al negocio con base a la información que arrojó este modelo.

Para la realización de este modelo predictivo, se hizo uso de las fases de la metodología SEMMA con el fin de realizar procesos de selección, exploración y modelamiento de grandes cantidades de datos, utilizando etapas de la metodología como lo es la etapa de extracción. La etapa de extracción permite obtener un conjunto de datos sobre la cual se llevó a cabo el análisis. En la etapa de exploración se realizó un análisis de los datos haciendo uso de herramientas de visualización como *Power Bi* y de diferentes técnicas estadísticas y algoritmos de *Machine Learning*. En la etapa de modificación se transformaron ciertos datos que fueron ingresados al modelo. En la etapa del modelado se encontró el algoritmo que mejor porcentaje de predicción dio respecto a otros. De esta manera se aplicaron técnicas de *Machine Learning* en el sector financiero para predecir posibles cancelaciones de las tarjetas de crédito de los clientes [2].

Objetivo General

Construir un sistema basado en técnicas de *Machine Learning*, que permita predecir posibles cancelaciones de las tarjetas de crédito, basadas en datos históricos.

Objetivos Específicos

- Plantear un sistema que permita identificar cuáles son las posibles variables más significativas que influyen en la cancelación de la tarjeta de crédito.

- Crear una base de datos con datos significativos, que permitan realizar un análisis predictivo.

- Analizar algoritmos de *Machine Learning* que permitan proponer un predictor de posibles cancelaciones de las tarjetas de crédito por medio de datos históricos.

- Integrar y visualizar los resultados del análisis predictivo dentro de un reporte de *Power BI*.

- Verificar el desempeño del sistema de predicción propuesto usando nuevos datos suministrados por el banco.

Marco Teórico

Power BI es una colección de servicios de software, aplicaciones y conectores que funcionan conjuntamente para convertir orígenes de datos sin relación entre sí en información coherente, interactiva y atractiva visualmente. Sus datos pueden ser una hoja de cálculo de Excel o una colección de almacenes de datos híbridos locales y basados en la nube. *Power BI* permite conectarse con facilidad a los orígenes de datos, visualizar y descubrir qué es importante y compartirlo con cualquiera o con todos los usuarios que desee [3].

Power BI Desktop es una aplicación gratuita que se puede instalar en el equipo local y que permite conectarse a los datos, transformarlos y visualizarlos. Con *Power BI* Desktop, puede conectarse a varios orígenes de datos diferentes y combinarlos (lo que se suele denominar modelado) en un modelo de datos. Este modelo de datos permite compilar objetos visuales y colecciones de objetos visuales que se pueden compartir como informes con otras personas de dentro de la organización. La mayoría de los usuarios que trabajan en proyectos de inteligencia empresarial usan *Power BI* Desktop para crear informes y luego usan el servicio *Power BI* para compartir los informes con otros.

Algoritmos de Regresión: Se usan para estimar las relaciones entre variables dependientes y una o más variables independientes (o predictoras). Más específicamente, el análisis de regresión ayuda a entender cómo el valor de la variable dependiente varía al cambiar el valor de una de las variables independientes, manteniendo el valor de las otras variables independientes fijas. Uno de los principales algoritmos de regresión es la ***Regresión Logística.*** La regresión logística es un método estadístico para predecir clases binarias. Se usa en algunos

casos para calcular la probabilidad de que ocurra un evento, además se usa en predicciones binarias [4].

Machine Learning es una rama de la inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan sobre creación de modelos y aprendizajes automáticos a partir de la extracción de características de los datos.

- *Random Forest*: Un bosque aleatorio es un conjunto de árboles de decisión aleatorios clasificadores, que hace predicciones combinando las predicciones de los árboles individuales. Se puede usar un bosque aleatorio para hacer predicciones sobre atributos categóricos (Clasificación) o numéricos (Regresión). Los bosques aleatorios son uno de los modelos predictivos de mejor rendimiento. Cada árbol es construido usando el siguiente algoritmo:

- Sea N el número de casos de prueba, M es el número de variables en el clasificador.

- Sea m el número de variables de entrada a ser usado para determinar la decisión en un nodo dado; m debe ser mucho menor que M.

- Elegir un conjunto de entrenamiento para este árbol y usar el resto de los casos de prueba para estimar el error.

- Para cada nodo del árbol, elegir aleatoriamente m variables en las cuales basar la decisión. Calcular la mejor partición del conjunto de entrenamiento a partir de las m variables.

Para la predicción de un nuevo caso, este es empujado hacia abajo por el árbol. Luego se le asigna la etiqueta del nodo terminal. Este proceso es iterado por todos los árboles en el ensamblado, y la etiqueta que obtenga la mayor cantidad de incidencias es reportada como la predicción [5]. En *Random Forest* se hace el entrenamiento de un sistema clasificador, pero al tiempo se hace una selección de las variables más representativas para diferenciar entre las clases esperadas.

Arboles de decisión: Un árbol de decisión es un modelo de predicción basado en diagramas. Dado un conjunto de datos se fabrican diagramas de construcciones lógicas, muy similares a los sistemas de predicción basados en reglas, que sirven para representar y categorizar una serie de condiciones que ocurren de forma sucesiva, para la resolución de un problema. Los árboles de decisión están formados por nodos y su lectura se realiza de arriba hacia abajo con las siguientes características:

- Primer nodo o nodo raíz en donde se produce la primera división en función de la variable más importante.

- Nodos internos o intermedios: tras la primera división encontramos estos nodos, que vuelven a dividir el conjunto de datos en función de las variables.

- Nodos terminales u hojas: se ubican en la parte inferior del esquema y su función es indicar la clasificación definitiva.

- la profundidad de un árbol, que viene determinada por el número máximo de nodos de una rama.

Consideraciones sobre la herramienta

Cuando se trabajan con análisis que requieren algoritmos de *Machine Learning*, se debe emplear un lenguaje de programación que facilite la creación del código y permita utilizar librerías que encapsulen muchas funcionalidades referentes al *Machine Learning*, por lo tanto se decidió utilizar *Python* como el lenguaje de programación que servirá para la creación del modelo predictivo gracias a que este lenguaje tiene un sin número de librerías que soportan las funcionalidades requeridas para el proyecto. A continuación, se describirá las librerías usadas.

- *NumPy*.
- *Pandas*.
- *Matplotlib*.
- *Seaborn*.
- *Sklearn*.

Se trabajó con la versión de *Python* 3.7. y se hizo la programación en el entorno de trabajo *Jupyter Notebook*.

Descripción de la base de datos

La base de datos que se usó en este proyecto contiene información sobre factores demográficos, pagos predeterminados, datos crediticios, historial de pagos y estados de cuenta y cancelaciones de las tarjetas de crédito de los clientes desde abril del 2005 hasta septiembre del 2005. A continuación, se describe la base de datos en cuestión.

- ID: Identificación de cada cliente.
- LIMIT_BAL: Monto de crédito aprobado.
- SEX: Genero (1=Masculino, 2=Femenino).
- EDUCATION: (1=Secundaria, 2=Universitario, 3=Bachillerato, 4=Otros, 5=Desconocidos, 6=Desconocidos).

- MARRIAGE: Estado civil (1=Casado, 2=Soltero, 3=Otros).
- AGE: Edad.
- PAY_ (0 – 6): Estado de reembolso desde Septiembre (0) hasta abril del 2005 (6) (-1=Pago a tiempo, 1=Retraso en el pago por 1 mes, 2= Retraso en el pago por 2 meses, ... ,9= Retraso en el pago por 9 meses) en dólares.
- BILL_AMT (0 – 6): Cantidad de dinero en la cuenta desde septiembre (0) hasta abril (6) del 2005 en dólares.
- PAY_AMT (0 – 6): Cantidad de dinero que se pagó de la deuda desde septiembre (0) hasta abril (6) del 2005 en dólares.
- Default payment next month: Cancelación de la tarjeta de crédito (1=Si, 0=No).

La base de datos consta 30.000 registros de los cuales 23.364 son clientes que pertenecen a la clase 0, clientes que cancelaron su tarjeta de crédito y 6636 son clientes que pertenecen a la clase 1, clientes que cancelaron su tarjeta de crédito al siguiente mes como lo muestran las siguientes imágenes.

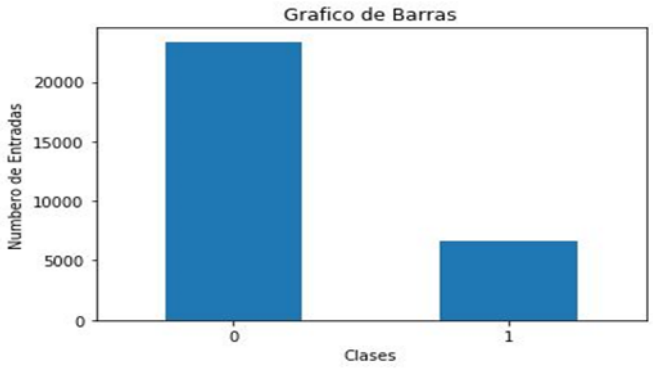


Figura 1. Distribución del número de registros de las clases.

Se trata de una base de datos con clases desbalanceadas donde la clase mayoritaria ocupa el 78% de los datos totales y la clase minoritaria ocupa el 22% del total de los datos. Este dato es relevante al momento de ingresar las características a los algoritmos ya que la clase minoritaria en este caso es la clase para analizar, por lo tanto, se deberá utilizar alguna técnica de balanceo de datos para cada una de las dos clases ya que esto afecta sustancialmente el posterior porcentaje de predicción de los algoritmos.

La base de datos consta de 25 características, es decir, 25 columnas que aportan información tanto demográfica como transaccional de las tarjetas de crédito. Si bien la base de datos tiene mucha información que a simple vista parece útil, en muchos de los casos, estas características no aportan un valor sustancial a la hora de abordar un problema que requiera de algoritmos de *Machine Learning*, por lo tanto, es

necesario hacer una depuración de las características con el objetivo de encontrar cuales son las más importantes para ser utilizadas como insumos de los algoritmos.

Selección de características

La Selección de Características es el proceso de seleccionar un subconjunto de características pertinentes (variables, predictores) de un conjunto de datos para su uso en la construcción de modelos analíticos. Este proceso de selección trae consigo ciertos beneficios como el de mejorar el rendimiento de predicción de los predictores. Básicamente esta selección de características se usa cuando la base de datos contiene información irrelevante, pudiendo ser removidos sin incurrir en alguna perdida de información relevante. Las características adicionales actúan como ruido en los datos de insumos para alimentar los modelos de *Machine Learning*. Si estas características no son removidas, probablemente los algoritmos tengan bajos porcentajes de predicción, haciendo que los modelos tarden más tiempo en entrenarse y la asignación de recursos innecesarios para entrenar los algoritmos ya que el volumen de datos seria mayor. Existen muchas maneras de verificar si una característica es importante o no, este proceso implica la realización de análisis estadísticos como histogramas, diagramas de cajas y bigotes, diagrama de violín, entre otros.

- **Análisis Descriptivo**

Como primer ejercicio se planteó analizar el comportamiento de las variables demográficas respecto a la cancelación o no de la tarjeta de crédito, de acá se desprende que dicha información no aporta el valor suficiente para concluir que las variables influyen en la cancelación de las tarjetas de crédito. En la información que suministra el diagrama de barras de ambos sexos, se evidencia que la cantidad de hombres que cancelan las tarjetas de crédito es similar a la información respectiva de las mujeres, por lo que la característica de sexo del cliente no aporta valor para ser insumo del algoritmo. De igual manera se hace el análisis para la característica de educación y estado civil. Ambas características carecen de información influyente.

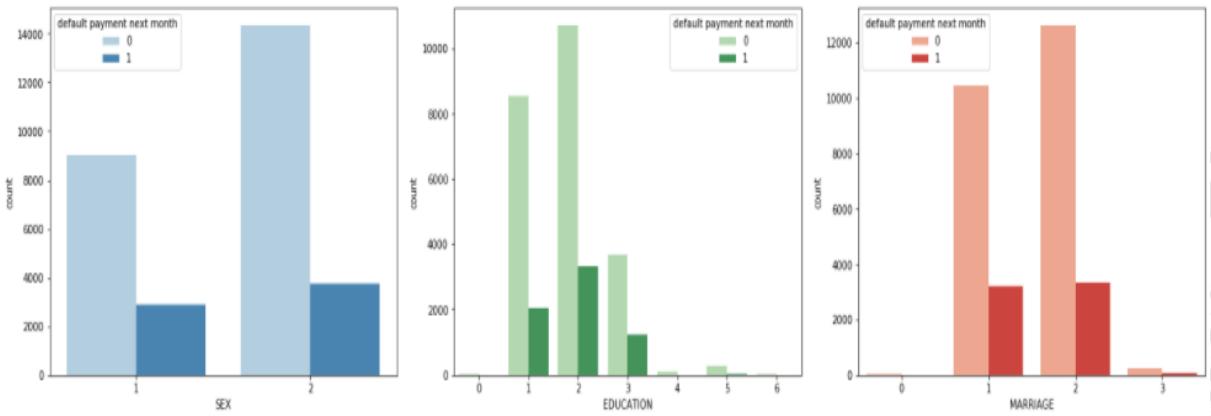


Figura 2. Diagrama de barras del número de cancelaciones por sexo, educación y estado civil.

Adicional al diagrama de barras, también se hizo el ejercicio de representar gráficamente la variable cupo tarjeta, por medio de un histograma, el cual nos da información sobre las frecuencias de los valores representados. En este caso el histograma nos arroja una información que puede resultar útil en aras de conocer cómo se comportan todas las variables. De este histograma se puede deducir que la cancelación de la tarjeta de crédito tiene una relación respecto al cupo de la tarjeta, es decir, entre menos cupo se tenga en la tarjeta de crédito, la frecuencia de las cancelaciones es mayor por lo que se puede concluir que el mayor porcentaje de personas que cancelan su tarjeta de crédito, corresponden a bajos cupos en la misma.

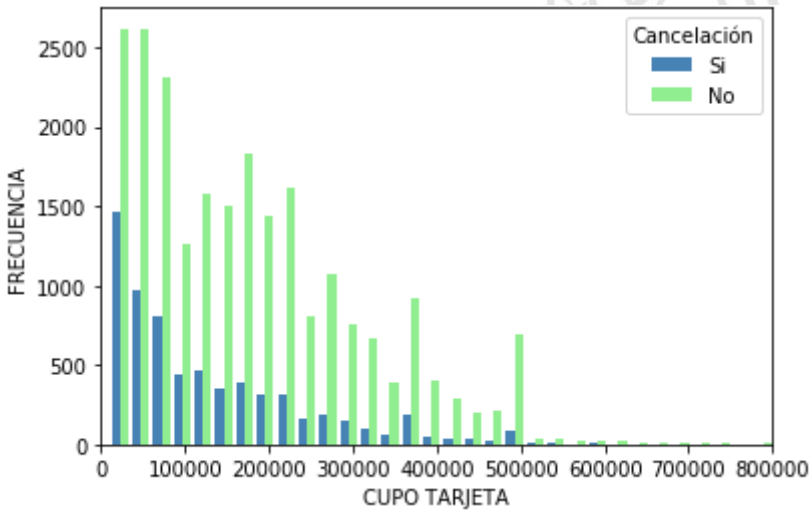


Figura 3. Histograma del cupo de las tarjetas de crédito.

Otro de los métodos comúnmente utilizados al momento de escoger las principales características son los diagramas de cajas y bigotes. Un diagrama de cajas y bigotes es una manera conveniente de mostrar visualmente grupos de datos numéricos a través de sus cuartiles. En este caso se observa que las variables demográficas nuevamente arrojan datos irrelevantes, presentando la misma distribución y simetría de los datos por lo que se concluye que no hay diferencia entre las variables examinadas. Para los valores transaccionales se puede inferir que tienen un comportamiento similar en cuanto a la distribución, simetría y mediana del conjunto de datos, por lo que se hace difícil el análisis para determinar de forma gráfica, cuál de estas características son las que más influyen en la cancelación o no de las tarjetas de crédito, por lo que se plantea otro tipo de solución utilizando algoritmos de eliminación de características.

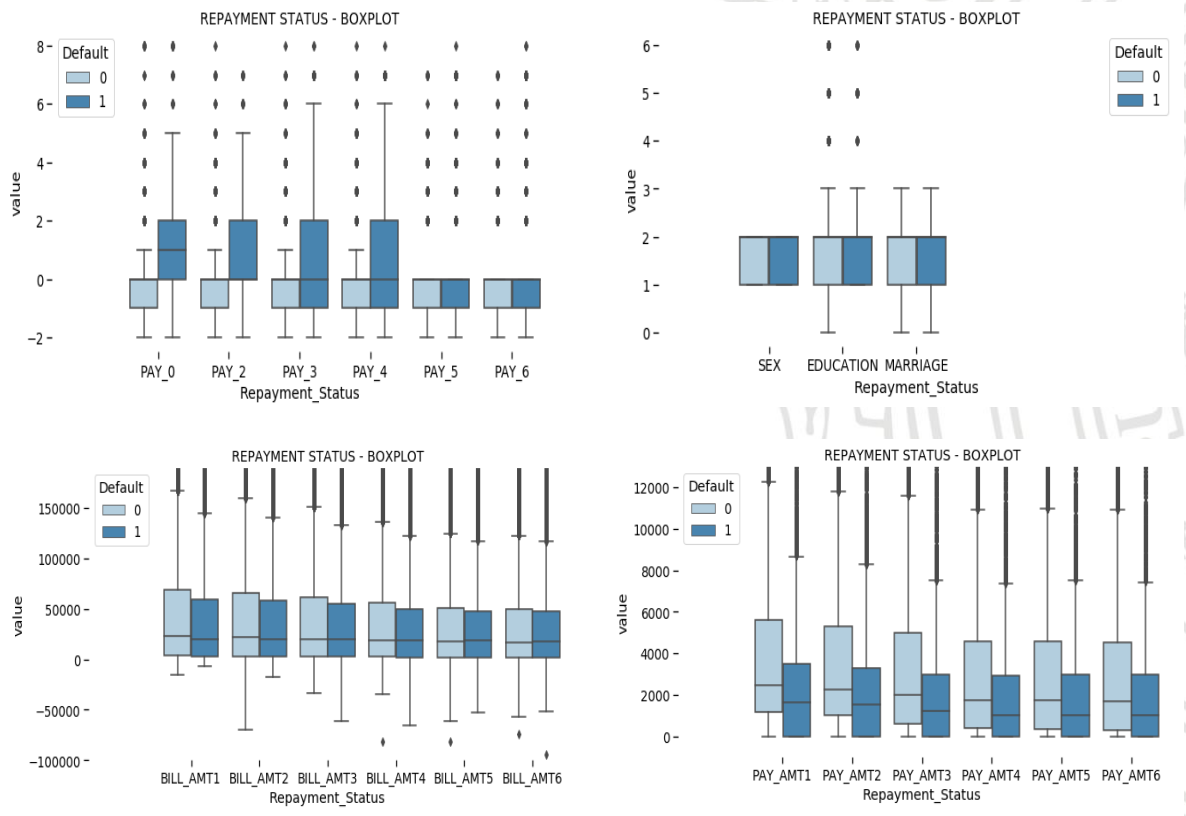


Figura 4. Diagramas de cajas y bigotes.

En este caso, se utilizó un algoritmo de eliminación recursiva de características (RFE). Este algoritmo lo que hace es que elimina características, se le asigna un valor de un peso a cada una de ellas y se entrena el algoritmo con los datos de entrenamiento. El algoritmo hace una categorización por prioridad de estos pesos, luego hace un

listado de las características más importantes y por último estas son utilizadas para calcular la precisión del modelo. Se le ingresó al algoritmo las 25 características y como resultado final se obtuvo un ranking de las más importantes.

```
Feature Ranking: [ 6 13  7  5  8  1  4  2 16 12 19  1  3  1 18 10 17  1  1 11  9 14 15]
Standardized Model Score with selected features is: 0.812467 (0.000000)

Most important features (RFE): ['PAY_0' 'BILL_AMT1' 'BILL_AMT3' 'PAY_AMT1' 'PAY_AMT2']
```

Figura 5. Mejores características.

- **Modelo predictivo**

Para la realización del modelo predictivo, se escogieron 3 algoritmos comúnmente utilizados para resolver este tipo de problemas que implican predicciones binarias sobre una gran cantidad de datos, para este caso se trabajó sobre las posibles cancelaciones de las tarjetas de crédito de los clientes. A continuación, se hará énfasis en cada uno de los modelos planteados. Los algoritmos fueron los siguientes:

- Regresión logística
- Arboles de decisión
- *Random Forest*

La idea principal de la búsqueda del modelo por medio de estos algoritmos es el poder encontrar el modelo que mejor se acomoda a los datos que se tienen, es decir, encontrar el modelo que tenga el porcentaje de predicción mayor respecto a las demás. A continuación, se hará una descripción de cada uno de ellos.

- **Regresión logística**

Para la implementación del modelo utilizando el algoritmo de regresión logística, primero se crea un *DataFrame* con las características de entrada y otro *DataFrame* con el valor esperado, es decir el valor que nos indica si la tarjeta fue cancelada o no y luego se separa el conjunto de datos de entrenamiento y de prueba. En esta etapa se plantea el ejercicio de búsqueda del modelo que mejor porcentaje de rendimiento arroje, por lo tanto, se planteó realizar unas modificaciones a los datos de entrada. En total se hizo 3 pruebas las cuales fueron ingresarle al algoritmo las 25

características, luego se probó normalizando las 25 características y por último se probó con las mejores 5 características normalizadas [6].

```
Default = df['default payment next month'] # Default será la columna del valor esperado, es decir, si se cancela o no la tar
Features = df.drop(['ID','default payment next month'],axis=1) # Features será la base de datos son las columnas 'ID','defau
X = Features
y = Default
# Original dataset
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, stratify=y, random_state=42)
stdX = (Features - Features.mean()) / (Features.std())
# Dataset with standardized features
Xstd_train, Xstd_test, ystd_train, ystd_test = train_test_split(X,y, test_size=0.2, stratify=y,random_state=42)
# Dataset with five most important features
Ximp = stdX[['PAY_0', 'BILL_AMT1', 'BILL_AMT3', 'PAY_AMT1', 'PAY_AMT2']]
X_tr, X_t, y_tr, y_t = train_test_split(Ximp,y, test_size=0.2, stratify=y, random_state=42)
```

Figura 6. Separación del dataset en datos de entrenamiento y prueba.

Con los datos de entrenamiento y prueba definidos, el siguiente paso fue la creación del modelo predictivo utilizando el algoritmo de regresión logística. Se definen los atributos necesarios del modelo y se procedió a correr el modelo el cual arrojó los siguientes resultados.

	precision	recall	f1-score	support
0	0.82	0.97	0.89	4673
1	0.70	0.24	0.36	1327
accuracy			0.81	6000
macro avg	0.76	0.61	0.62	6000
weighted avg	0.79	0.81	0.77	6000

Modelo con las 25 características

	precision	recall	f1-score	support
0	0.82	0.97	0.89	4673
1	0.70	0.27	0.39	1327
accuracy			0.81	6000
macro avg	0.76	0.62	0.64	6000
weighted avg	0.80	0.81	0.78	6000

Modelo con las 25 características normalizadas

	precision	recall	f1-score	support
0	0.82	0.97	0.89	4673
1	0.70	0.27	0.39	1327
accuracy			0.81	6000
macro avg	0.76	0.62	0.64	6000
weighted avg	0.80	0.81	0.78	6000

Modelo con las 5 mejores características

Tabla 1. Resultados de los modelos utilizando datos de prueba.

Como se pudo observar en los 3 resultados, el modelo responde muy bien frente a las predicciones de la clase mayoritaria pero el rendimiento que tuvo frente a las predicciones de la clase minoritaria es muy bajo. Dado que lo que se busca es que el modelo tenga un alto grado de generalización para ambas clases y que este a su vez tenga un buen rendimiento frente a la clase de interés que en este caso es la clase minoritaria, es decir, las personas que si cancelaron las tarjetas de crédito. La métrica de ***Precision*** nos da información sobre el número de elementos identificados correctamente como positivo de un total de elementos identificados como positivos. La métrica de ***Accuracy*** nos da información sobre el porcentaje total de elementos clasificados correctamente y ***Recall*** es el número de elementos identificados correctamente como positivos del total de positivos verdaderos. Al observar las métricas anteriormente mencionadas, se observó que el ***Recall*** en los 3 modelos era muy bajo, por lo que estos modelos clasificaban adecuadamente la clase mayoritaria pero no lo hacían tan bien para la clase minoritaria. Por lo que el

planteamiento de estos modelos no está acorde con la finalidad esperada el cual es predecir las posibles cancelaciones de las tarjetas de crédito. Se obtuvo un porcentaje de acierto utilizando *5-fold cross validation* del 96% con los datos de entrenamiento y un 73% con datos de prueba, por lo que se planteó balancear la base de datos para intentar solucionar este tema.

Balance de clases

En los problemas de clasificación en donde tenemos que etiquetar diferentes clases, se suele encontrar que el conjunto de datos de entrenamiento cuenta con alguna de las clases es una clase "minoritaria" es decir, de la cual se tienen pocos ejemplos. Esto provoca un desbalance en las clases que afecta el entrenamiento de los algoritmos. En este caso es evidente que, para un banco, el número de personas que cancelan la tarjeta de crédito es mucho menor que el número de personas que continúan con su tarjeta, por lo que resulta sumamente útil realizar análisis con la clase minoritaria para que el banco tome medidas y evite dicha cancelación.

Por lo general este desbalance de clases afecta a los algoritmos en su proceso de generalización de la información y perjudicando a las clases minoritarias, es decir, si al algoritmo se le entrena con 990 muestras de la clase 1 y solo 10 muestras para la clase 2, lo más probable es que el algoritmo no logre diferenciar entre una clase u otra. Para solucionar este inconveniente, se planteó utilizar 2 técnicas de balanceamiento de datos las cuales fueron *Smote* y *NearMiss*:

- ***SMOTE***

es una técnica de sobre muestreo de minorías sintéticas, es uno de los métodos de sobremuestreo más utilizados para resolver el problema del desequilibrio. Su objetivo es equilibrar la distribución de clases aumentando aleatoriamente los ejemplos de clases minoritarias al replicarlos. *SMOTE* sintetiza nuevas instancias minoritarias entre instancias minoritarias existentes. Genera los registros de entrenamiento virtual por interpolación lineal para la clase minoritaria. Estos registros de entrenamiento sintéticos se generan seleccionando aleatoriamente uno o más de los k vecinos más cercanos para cada ejemplo en la clase minoritaria. Después del proceso de sobremuestreo, los datos se reconstruyen y se pueden aplicar varios modelos de clasificación para los datos procesados [8].

Para la creación del modelo con el algoritmo de *Smote*, se creó una nueva partición para los datos de entrenamiento y los de prueba y se ajustaron al algoritmo en cuestión. Luego se crea las funciones para el modelo y para la visualización de los resultados por medio de la matriz de confusión.

```
from imblearn.over_sampling import SMOTE
sm = SMOTE()
X_train_res, y_train_res = sm.fit_sample(X_tr, y_tr)
X_test_res, y_test_res = sm.fit_sample(X_t, y_t)

#Funcion para Dataset normalizado con las 5 mejores caracteristicas ( balanceado con Smoth)
def run_model_RLS(X_train_res, X_test_res, y_train_res, y_test_res):
    clf_base = LogisticRegression(C=1.0,penalty='l2',random_state=1,solver="newton-cg")
    clf_base.fit(X_train_res,y_train_res)
    return clf_base

#definimos funciona para mostrar los resultados
def mostrar_resultados_RLS(y_test_res, pred_RLS):
    conf_matrix = confusion_matrix(y_test_res, pred_RLS)
    plt.figure(figsize=(12, 12))
    sns.heatmap(conf_matrix, annot=True, fmt="d");
    plt.title("Matriz de Confusión Dataset original")
    plt.ylabel('Clase verdadera')
    plt.xlabel('Clase Predecida')
    plt.show()
    print (classification_report(y_test_res, pred_RLS))

model_RLS = run_model_RLS(X_train_res, X_test_res, y_train_res, y_test_res)
pred_RLS = model_RLS.predict(X_test_res)
mostrar_resultados_RLS(y_test_res, pred_RLS)
```

Figura 7. Modelo balanceado con Smote.

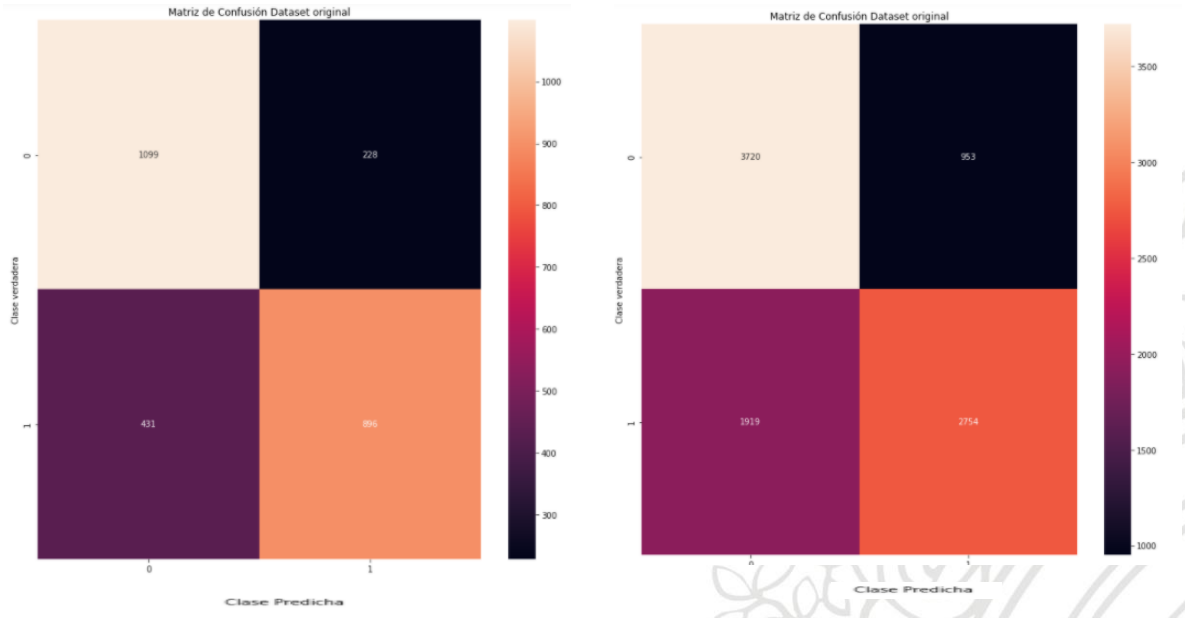
- ***NEARMISS***

Es una técnica de submuestreo. Su objetivo es equilibrar la distribución de clases eliminando aleatoriamente los ejemplos de clases mayoritarias. Cuando las instancias de dos clases diferentes están muy cerca una de la otra, eliminamos las instancias de la clase mayoritaria para aumentar los espacios entre las dos clases. Esto ayuda en el proceso de clasificación. Para evitar el problema de la pérdida de información en la mayoría de las técnicas de muestreo insuficiente, se utilizan ampliamente los métodos del vecino cercano [8].

Para la creación del modelo con el algoritmo de *NearMiss*, se creó una nueva partición para los datos de entrenamiento y los de prueba y se ajustaron al algoritmo en cuestión. Luego se crea las funciones para el modelo y para la visualización de los resultados por medio de la matriz de confusión.

```
# apply near miss
from imblearn.under_sampling import NearMiss
nr = NearMiss()

X_train_miss, y_train_miss = nr.fit_sample(X_tr, y_tr)
X_test_miss, y_test_miss = nr.fit_sample(X_t, y_t)

#Funcion para Dataset normalizado con las 5 mejores caracteristicas ( balanceado con Near miss)
def run_model_RLN(X_train_miss, X_test_miss, y_train_miss, y_test_miss):
    clf_base = LogisticRegression(C=1.0,penalty='l2',random_state=1,solver="newton-cg")
    clf_base.fit(X_train_miss,y_train_miss)
    return clf_base

#definimos funciona para mostrar los resultados
def mostrar_resultados_RLN(y_test_miss, pred_RLN):
    conf_matrix = confusion_matrix(y_test_miss, pred_RLN)
    plt.figure(figsize=(12, 12))
    sns.heatmap(conf_matrix, annot=True, fmt="d");
    plt.title("Matriz de Confusión Dataset original")
    plt.ylabel('Clase verdadera')
    plt.xlabel('Clase Predecida')
    plt.show()
    print (classification_report(y_test_miss, pred_RLN))

model_RLN= run_model_RLN(X_train_miss, X_test_miss, y_train_miss, y_test_miss)
pred_RLN = model_RLN.predict(X_test_miss)
mostrar_resultados_RLN(y_test_miss, pred_RLN)
```

Figura 8. Modelo balanceado con NearMiss



	precision	recall	f1-score	support
0	0.72	0.83	0.77	1327
1	0.80	0.68	0.73	1327
accuracy			0.75	2654
macro avg	0.76	0.75	0.75	2654
weighted avg	0.76	0.75	0.75	2654

	precision	recall	f1-score	support
0	0.64	0.76	0.70	4673
1	0.71	0.58	0.64	4673
accuracy			0.67	9346
macro avg	0.68	0.67	0.67	9346
weighted avg	0.68	0.67	0.67	9346

Figura 9. Resultados de los modelos con balance de clases NearMiss y Smote respectivamente utilizando datos de prueba.

En la figura 9 se presenta el resultado del modelo propuesto por medio de una **matriz de confusión**. Una matriz de confusión es una herramienta que permite la visualización del desempeño de un algoritmo que se emplea en aprendizaje supervisado. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real. Uno de los beneficios de las matrices de confusión es que facilitan ver si el sistema está confundiendo dos clases.

Utilizando los algoritmos de balanceamiento de datos, para el caso del *NearMiss* se observó que el submuestreo de los datos, bajándole muestras de la clase mayoritaria a la clase minoritaria, el rendimiento del modelo creció sustancialmente, pasó de reconocer bien la clase mayoritaria y de no reconocer a la clase minoritaria, a reconocer ambas clases con porcentajes de rendimiento similares. Se obtuvo un porcentaje de acierto utilizando un *5-fold cross validation* del 98.1% con los datos de entrenamiento y un 80.4% con datos de prueba. Ocurre de manera similar para el caso del algoritmo de *Smote*, pasó de reconocer bien la clase minoritaria y de no

reconocer a la clase minoritaria, a reconocer ambas clases con porcentajes de rendimiento un poco menores que el modelo creado con el algoritmo de *NearMiss*. Se obtuvo un porcentaje de acierto utilizando un *5-fold cross validation* del 97.4% con los datos de entrenamiento y un 78% con datos de prueba.

Random Forest

Para la implementación del modelo utilizando el algoritmo de *Random Forest*, se hace un proceso similar al visto con el algoritmo de Regresión Logística, primero se crea un DataFrame con las características de entrada y otro DataFrame con el valor esperado, es decir el valor que nos indica si la tarjeta fue cancelada o no y luego se separa el conjunto de datos de entrenamiento y de prueba. En esta etapa se plantea de igual manera, realizar la búsqueda del modelo que mejor porcentaje de rendimiento arroje, igualmente se hizo 3 pruebas las cuales fueron ingresarle al algoritmo las 25 características, luego se probó normalizando las 25 características y por último se probó con las mejores 5 características.

	precision	recall	f1-score	support
0	0.84	0.94	0.89	4673
1	0.62	0.36	0.46	1327
accuracy			0.81	6000
macro avg	0.73	0.65	0.67	6000
weighted avg	0.79	0.81	0.79	6000

Modelo con las 25 características

	precision	recall	f1-score	support
0	0.84	0.94	0.89	4673
1	0.64	0.36	0.46	1327
accuracy			0.81	6000
macro avg	0.74	0.65	0.67	6000
weighted avg	0.79	0.81	0.79	6000

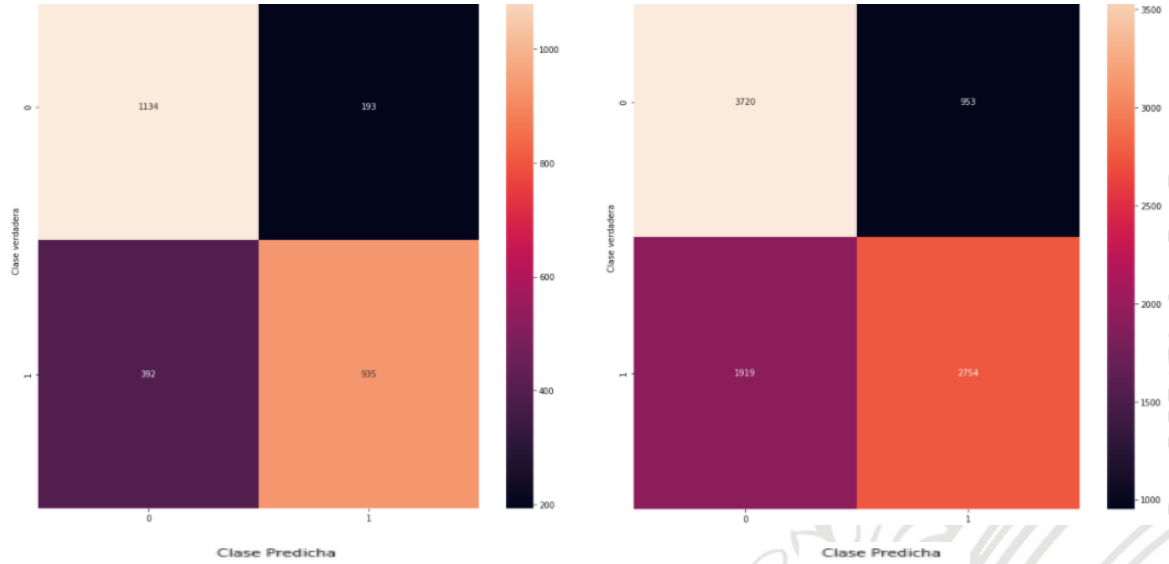
Modelo con las 25 características normalizadas

	precision	recall	f1-score	support
0	0.84	0.93	0.88	4673
1	0.59	0.36	0.45	1327
accuracy			0.80	6000
macro avg	0.71	0.64	0.66	6000
weighted avg	0.78	0.80	0.78	6000

Modelo con las 5 mejores características

Tabla 2. Resultados de los modelos utilizando datos de prueba.

Nuevamente en los 3 resultados del modelo responden muy bien frente a las predicciones de la clase mayoritaria pero el rendimiento que tuvo frente a las predicciones de las clases minoritarias sigue siendo muy pobres, si bien la ***Precision*** el ***Recall*** y el ***F1-Score*** mejoraron en una pequeña medida, los modelos planteados siguen sin obtener un porcentaje de predicción que sea acorde a lo esperado, por lo que nuevamente se planteó un balanceo de las clases para intentar mejorar este tema.



	precision	recall	f1-score	support
0	0.74	0.85	0.79	1327
1	0.83	0.70	0.76	1327
accuracy			0.78	2654
macro avg	0.79	0.78	0.78	2654
weighted avg	0.79	0.78	0.78	2654

	precision	recall	f1-score	support
0	0.66	0.80	0.72	4673
1	0.74	0.59	0.66	4673
accuracy			0.69	9346
macro avg	0.70	0.69	0.69	9346
weighted avg	0.70	0.69	0.69	9346

Figura 10. Resultados de los modelos con balanceamiento de datos NearMiss y Smote respectivamente utilizando datos de prueba.

Utilizando los algoritmos de balance de clases, para el caso del *NearMiss* se observó que el submuestreo de los datos, bajándole muestras de la clase mayoritaria a la clase minoritaria, el rendimiento del modelo es mucho mayor al algoritmo de Regresión logística, este pasó de reconocer bien la clase mayoritaria y de no reconocer a la clase minoritaria, a reconocer ambas clases con porcentajes de rendimiento mucho mayores al algoritmo anterior. Se obtuvo un porcentaje de acierto utilizando un *5-fold cross validation* del 98.3% con los datos de entrenamiento y un 83.7% con datos de prueba. Ocurre de manera similar para el caso del algoritmo de *Smote*, pasó de reconocer bien la clase minoritaria y de no reconocer a la clase minoritaria, a reconocer ambas clases con porcentajes de rendimiento un poco menores que el modelo creado con el algoritmo de *NearMiss*. Se obtuvo un porcentaje de acierto utilizando un *5-fold cross validation* del 97.5% con los datos de entrenamiento y un 80.1% con datos de prueba.

Arboles de decisión

Para la implementación del modelo utilizando el algoritmo de Arboles de decisión, se hace un proceso similar al visto con los anteriores algoritmos, primero se crea un DataFrame con las características de entrada y otro DataFrame con el valor esperado, luego se separa el conjunto de datos de entrenamiento y de prueba. En esta etapa se plantea de igual manera, se realiza la búsqueda del modelo que mejor porcentaje de rendimiento arroje, igualmente se hizo 3 pruebas las cuales fueron ingresarle al algoritmo las 25 características, luego se probó normalizando las 25 características y por último se probó con las mejores 5 características.

	precision	recall	f1-score	support
0	0.83	0.81	0.82	4673
1	0.37	0.40	0.38	1327
accuracy			0.72	6000
macro avg	0.60	0.60	0.60	6000
weighted avg	0.72	0.72	0.72	6000

Modelo con las 25 características

	precision	recall	f1-score	support
0	0.83	0.81	0.82	4673
1	0.38	0.41	0.39	1327
accuracy			0.72	6000
macro avg	0.60	0.61	0.61	6000
weighted avg	0.73	0.72	0.72	6000

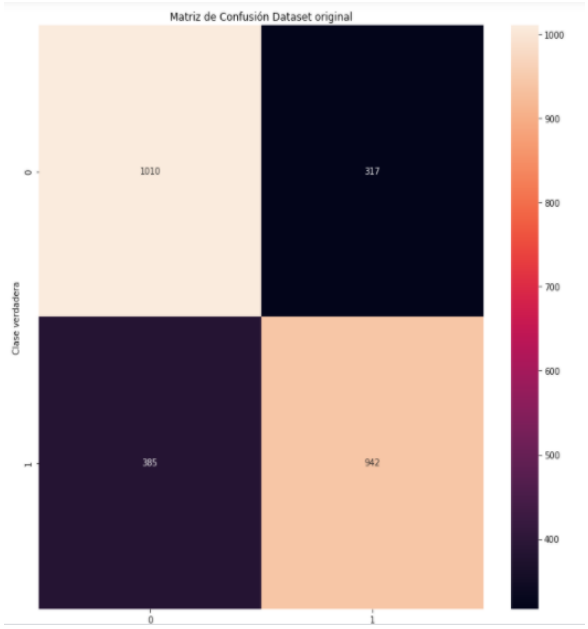
Modelo con las 25 características normalizadas

	precision	recall	f1-score	support
0	0.82	0.84	0.83	4673
1	0.39	0.37	0.38	1327
accuracy			0.73	6000
macro avg	0.61	0.60	0.60	6000
weighted avg	0.73	0.73	0.73	6000

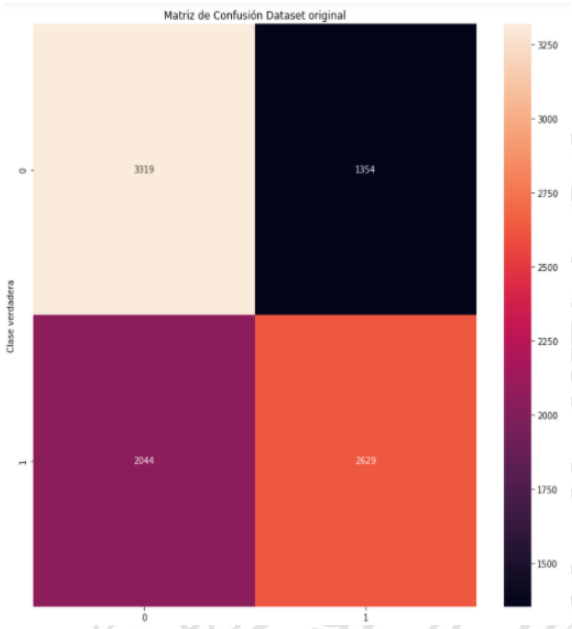
Modelo con las 5 mejores características

Tabla 3. Resultados de los modelos utilizando datos de prueba.

Los 3 resultados del modelo responden muy bien frente a las predicciones de la clase mayoritaria pero el rendimiento que tuvo frente a las predicciones de las clases minoritarias fue el menor rendimiento entre los algoritmos , si bien la ***Precision*** el ***Recall*** y el ***F1-Score*** mejoraron mucho más respecto a los anteriores modelos, nuevamente siguen sin obtener un porcentaje de predicción que sea acorde a lo esperado, por lo que se planteó un balance de las clases para intentar mejorar este problema. Se obtuvo un porcentaje de acierto utilizando un *5-fold cross validation* del 95% con los datos de entrenamiento y un 76% con datos de prueba.



	precision	recall	f1-score	support
0	0.72	0.76	0.74	1327
1	0.75	0.71	0.73	1327
accuracy			0.74	2654
macro avg	0.74	0.74	0.74	2654
weighted avg	0.74	0.74	0.74	2654



	precision	recall	f1-score	support
0	0.62	0.71	0.66	4673
1	0.66	0.56	0.61	4673
accuracy			0.64	9346
macro avg	0.64	0.64	0.63	9346
weighted avg	0.64	0.64	0.63	9346

Figura 11. Resultados de los modelos utilizando datos de prueba.

Utilizando los algoritmos de balance de clases, para el caso del *NearMiss* se observó que el rendimiento del modelo es mucho mayor respecto al modelo con el algoritmo de *Smote*.

Resultados y análisis

Para la representación de los resultados se usa la llamada curva ROC. Una curva ROC (curva de característica operativa del recepto) es un gráfico que muestra el rendimiento de un modelo de clasificación en todos los umbrales de clasificación. Esta curva representa dos parámetros.

- Tasa de verdaderos positivos
- Tasa de falsos positivos

Tasa de verdaderos positivos (TPR) y se define de la siguiente manera.

$$TPR = \frac{VP}{VP + FN}$$

Donde $VP = verdaderos\ positivos\ y\ FN = falsos\ negativos.$

Tasa de falsos positivos (FPR) se define de la siguiente manera.

$$FPR = \frac{FP}{FP + VN}$$

Donde $FP = Falsos\ positivos\ y\ VN = Verdaderos\ negativos.$

Una curva ROC representa TPR frente a FPR en diferentes umbrales de clasificación.

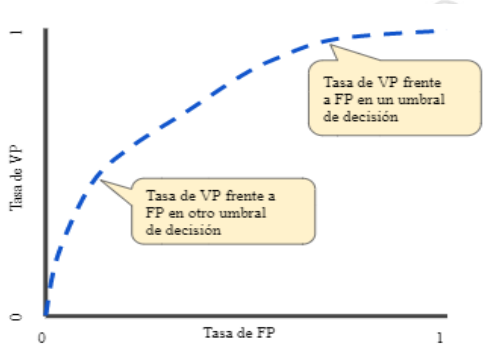


Figura 12. Representación de una ROC.

- **Regresión logística**

De los 5 modelos que se probaron, el que mejor dio resultado mostrando un rendimiento superior respecto a los demás modelos de este algoritmo con un 79% utilizando datos de prueba fue el modelo con la técnica de submuestreo *NearMiss*.

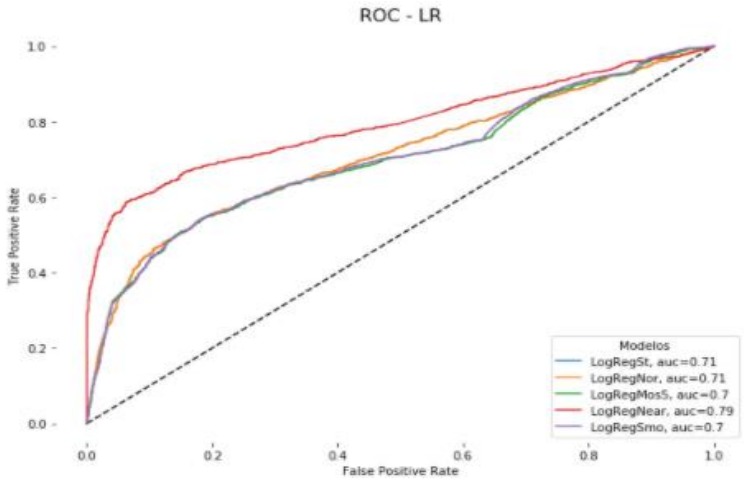


Figura 13. ROC del modelo de Regresión logística utilizando datos de prueba.

- ***Random Forest***

De los 5 modelos que se probaron, el que mejor dio resultado mostrando un rendimiento superior respecto a los demás modelos de este algoritmo con un 84% utilizando datos de prueba fue el modelo con la técnica de submuestreo *NearMiss*.

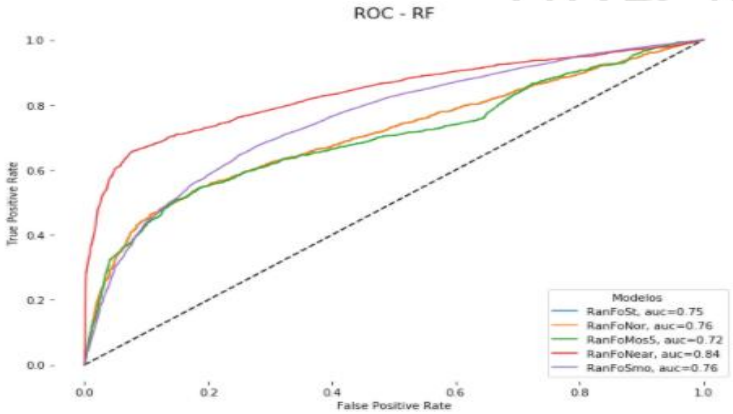


Figura 14. ROC del modelo de Random Forest utilizando datos de prueba.

- **Arboles de decisión**

De los 5 modelos que se probaron, el que mejor dio resultado mostrando un rendimiento superior respecto a los demás modelos de este algoritmo con un 74% utilizando datos de prueba fue el modelo con la técnica de submuestreo *NearMiss*.

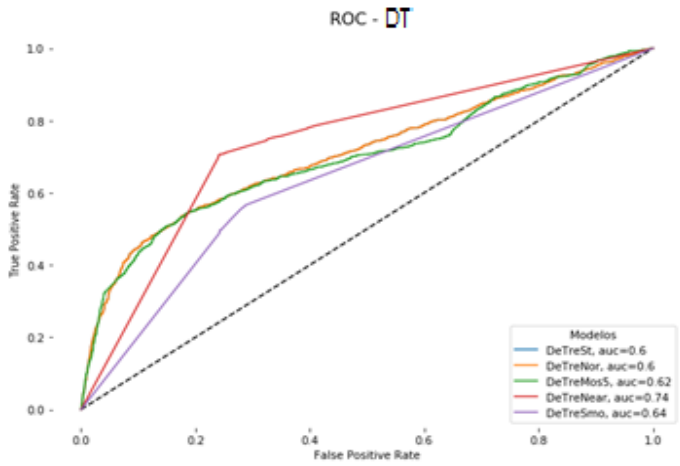


Figura 15. ROC del modelo de Arboles de decisión utilizando datos de prueba.

Tablas comparativas

A continuación, se mostrará detalladamente las métricas y los resultados de los modelos propuestos utilizando datos de prueba.

REGRESIÓN LOGÍSTICA	Precision	*Recall*	*F1-Score*	*Accuracy*	*AUC*
Características	**0.76**	**0.605**	**0.625**	**0.81**	**0.71**
Características normalizadas	**0.76**	**0.62**	**0.64**	**0.81**	**0.71**
Mejores 5 características	**0.76**	**0.62**	**0.63**	**0.81**	**0.7**
NearMiss	**0.76**	**0.755**	**0.75**	**0.75**	**0.79**
Smote	**0.67**	**0.67**	**0.67**	**0.67**	**0.7**

Tabla 4. Métricas y desempeño Regresión logística utilizando datos de prueba.

RANDOM FOREST	*Precision*	*Recall*	*F1-Score*	*Accuracy*	*AUC*
Características	**0.73**	**0.65**	**0.675**	**0.81**	**0.75**
Características normalizadas	**0.74**	**0.65**	**0.675**	**0.81**	**0.76**
Mejores 5 características	**0.715**	**0.645**	**0.665**	**0.8**	**0.72**
NearMiss	**0.785**	**0.775**	**0.78**	**0.78**	**0.84**
smote	**0.7**	**0.695**	**0.69**	**0.69**	**0.76**

Tabla 5. Métricas y desempeño Random Forest utilizando datos de prueba.

ARBOLES DE DECISIÓN	*Precision*	*Recall*	*F1-Score*	*Accuracy*	*AUC*
Características	**0.6**	**0.605**	**0.6**	**0.72**	**0.6**
Características normalizadas	**0.605**	**0.61**	**0.605**	**0.72**	**0.61**
Mejores 5 características	**0.605**	**0.605**	**0.605**	**0.73**	**0.62**
NearMiss	**0.735**	**0.735**	**0.735**	**0.74**	**0.74**
Smote	**0.64**	**0.635**	**0.635**	**0.64**	**0.64**

Tabla 6. Métricas y desempeño Arboles de Decisión utilizando datos de prueba.

Durante el transcurso de este experimento, se trabajó con 15 modelos de los cuales 9 se analizaron con datos desbalanceados y a 6 se les aplicó unas técnicas de balanceo con el fin de mejorar el rendimiento de los modelos planteados. De los modelos con datos desbalanceados se observó que la gran mayoría de ellos predecían de forma correcta la clase mayoritaria, pero a la hora de predecir la clase minoritaria, no lo hacían correctamente. Al algoritmo se le proporcionaba una gran cantidad de datos de la clase mayoritaria pero no se le hacía lo propio a la clase minoritaria, por esta razón el algoritmo por medio de la etapa de entrenamiento aprendió a diferenciar la clase mayoritaria pero no se hizo lo propio con la clase minoritaria. El algoritmo no logró esa generalización ya que no se le suministró los suficientes datos para que el aprendiera por medio de las características de entrada, que datos pertenecían a la clase 1 y que datos pertenecían a la clase 2. Durante la etapa de entrenamiento de los modelos con clases desbalanceadas, se observó que los porcentajes de acierto eran superiores al 94%, porcentaje que corresponde en su mayoría a los aciertos de las clases mayoritarias, pero los resultados en la etapa de prueba no fueron los adecuados ya que el algoritmo no logró la generalización requerida para predecir la clase minoritaria, obteniendo como porcentajes de acierto no mayores al 78%. Caso contrario pasa cuando se les aplican a los modelos una técnica de desbalanceo, al algoritmo se le administra una igual cantidad de muestras para ambas clases, con esto en la etapa de entrenamiento, el algoritmo alcanza una mayor generalización y puede predecir de una manera más acertada la cancelación de las tarjetas de crédito. Este cambio se vio reflejado en los porcentajes de acierto en las etapas de entrenamiento y prueba. En la etapa de entrenamiento se evidenció una mejoría respecto a los modelos con clases desbalanceadas superando el 97% de acierto. En la etapa de prueba también se evidenció una mejoría en donde los porcentajes de acierto ascendieron hasta un máximo del 84%.

visualización del modelo en *Power BI*

Una vez se tuvo el modelo que presentaba un mejor rendimiento, se procedió a representar los resultados del análisis predictivo en un reporte de *Power BI*. Esto se hizo con ayuda de un objeto visual de *Python*, estos objetos se manejan mediante programación y se ejecutan por la conexión del motor de *Python* a través de *Power BI*. *Power BI* es una herramienta de análisis de datos empresariales que permite conectarse a diferentes tipos de datos fuentes, hacer un modelamiento de los datos y visualizar la información de forma fácil y dinámica. Esta herramienta permite la manipulación de filtros para segmentar la información, utiliza múltiples graficas para la visualización de datos y crea reportes dinámicos en donde el usuario final puede deslizarse por el reporte manipulando las funcionalidades del programa para obtener

los resultados más relevantes de los análisis con el fin de tomar decisiones respecto a algún tema de negocio.

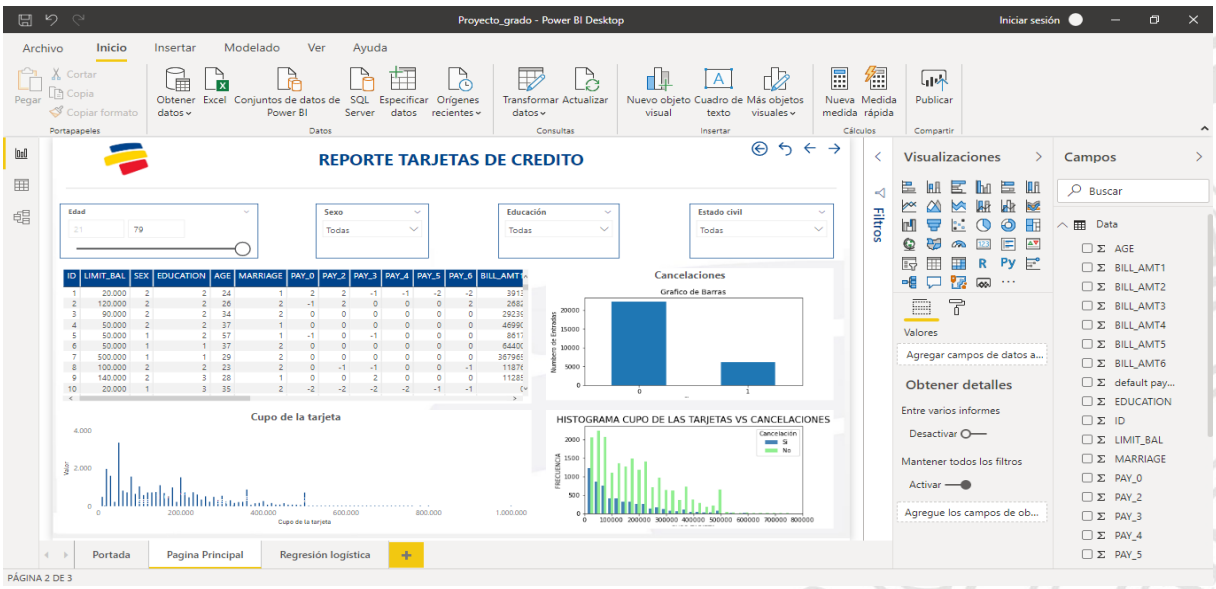


Figura 16. Reporte de resultados en Power BI.

Una vez configurado el *Power BI* para que permita el análisis y los objetos visuales en *Python*, se procede a la creación de las vistas del tablero que va a contener información sobre el análisis predictivo que se implementó.

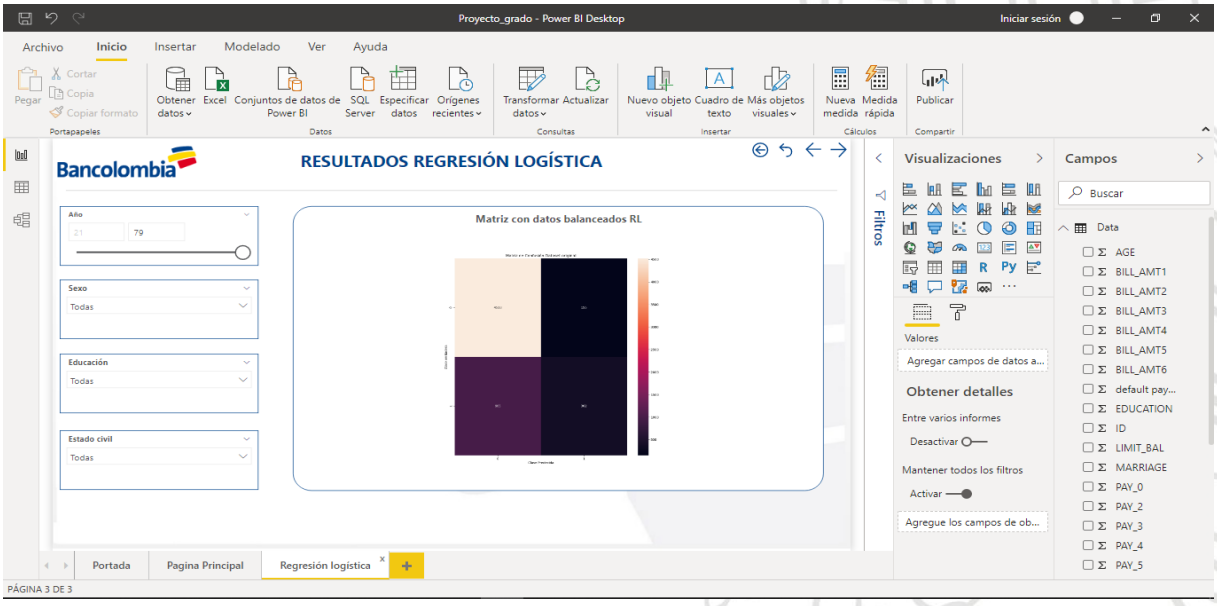


Figura 17. Visualización del análisis predictivo en Power BI.

Conclusiones

Dada las características de los datos y la cantidad de información que tuvo la base de datos, se buscó una solución que involucró algoritmos que permitieron la manipulación y el procesamiento de una gran cantidad de datos, creando modelos predictivos a partir de datos históricos. Por tanto, se consideró el uso de algoritmos de *Machine Learning* porque se trató de un problema complejo que implicó analizar una gran cantidad de datos con muchas variables, en donde el algoritmo buscó patrones naturales y relaciones entre datos para predecir las posibles cancelaciones en las tarjetas de crédito de los clientes.

En este informe se presentó una investigación sobre las posibles cancelaciones en las tarjetas de crédito de los clientes, el cual es un problema muy importante que las entidades bancarias deben tener presente y que debe ser solucionado mediante análisis de datos. Se propuso un sistema mediante la utilización de distintos algoritmos como la **REGRESIÓN LOGÍSTICA**, ***RANDOM FOREST*** y **ARBOLES DE DECISIÓN**. Este sistema arrojó los mejores resultados cuando se trabajó con técnicas de balanceo como lo fueron ***SMOTE*** y ***NEARMISS*****.** Si bien la utilización de estas técnicas arrojó los mejores resultados, con la técnica de ***NEARMISS*** para el algoritmo de ***RANDOM FOREST*****,** fue la combinación que presentó el mayor porcentaje con un 84% de predicción, seguido de la combinación entre ***NEARMISS*** con el algoritmo de **REGRESIÓN LOGÍSTICA** que arrojó un porcentaje de predicción del 79%.

Con ayuda de *Python* se pudo realizar una gran cantidad de análisis (predictivos y descriptivos) que permitieron el entendimiento y comprensión de un problema por medio de las variables de la base de datos que se estudió. Gracias a estos análisis, se logró la integración de 2 herramientas que sirven para el análisis de datos. Con ayuda de *Python* se pudo hacer análisis y visualizar información por medio de gráficos que la herramienta de *Power BI* no tiene implementada, por lo tanto, fue un plus muy grande el hecho de tener un análisis realizado en *Python* y el poder utilizar todas las funcionalidades propias de *Power BI* como el manejo de filtros para enriquecer el análisis y potencializar el valor del reporte en donde el usuario final puede deslizarse por el reporte manipulando las funcionalidades del programa para obtener los resultados más relevantes de los análisis con el fin de tomar decisiones respecto a algún tema de negocio.

Referencias Bibliográficas

[1] Diego Iribarren Baró. modelo predictivo aplicado al análisis de datos climáticos capturados por una placa sparkfun. 2016. Consultada en (20 de septiembre del 2020). https://repositorio.comillas.edu/jspui/bitstream/11531/14322/1/TFG000958.pdf

[2] Federico Carlos Peralta. Proceso de Conceptualización del Entendimiento del Negocio para Proyectos de Explotación de Información. 2014.Consultada en (20 de septiembre del 2020). https://www.researchgate.net/publication/284215308_Proceso_de_Conceptualizacion_del_Entendimiento_del_Negocio_para_Proyectos_de_Explotacion_de_Informacion/figures?lo=1

[3] ¿Qué es PowerBI Desktop? Consultada en (5 de septiembre del 2020). https://docs.microsoft.com/es-es/power-bi/fundamentals/desktop-what-is-desktop.

[4] Predictor de infidelidad utilizando regresión logística en Python. 2019. Consultada en (23 de septimbre del 2020). https://empresas.blogthinkbig.com/predictor-de-infidelidad-iii-ejemplo-de/

[5] Thomas Wood. ¿What is a Random Forest? Consultada en (3 de septiembre del 2020). https://deepai.org/machine-learning-glossary-and-terms/random-forest.

[6] conjuntos de entrenamiento, validación y pruebas. Consultada en (01 de junio del 2020). https://es.qwe.wiki/wiki/Training,_validation,_and_test_sets

[7] Guzmán, Elizabeth. Minería de datos. Consultada en (15 de mayo del 2020). https://disi.unal.edu.co/~eleonguz/cursos/md/presentaciones/Sesion5_Metodologias.pdf

[8] Handling Imbalanced Data with *SMOTE* and Near Miss Algorithm in Python. Consultado en (26 de septiembre del 2020). https://www.geeksforgeeks.org/ml-handling-imbalanced-data-with-smote-and-near-miss-algorithm-in-python/#:~:text=SMOTE%20(synthetic%20minority%20oversampling%20technique)%20is%20one%20of%20the%20most,instances%20between%20existing%20minority%20instances.