INSTITUTO POLITÉCNICO NACIONAL

Escuela Superior de Cómputo

Predicción mediante algoritmos de clasificación para la aprobación de una tarjeta de crédito

Elaborado por:

Pineda Hernández Francisco

Ramirez Aguilar Rodrigo Vidal

Ciudad de México a 14 de enero del 2024

Índice

Tabla de contenido

Introducción

El fraude en tarjetas de crédito es un problema que ha aumentado con el paso de los años, con el aumento de este medio de pago, la forma de realizar fraude con estas también ha aumentado, por tanto, es necesario encontrar métricas que ayuden a las instituciones bancarias a detectar este tipo de problemática.

Descripción del dataset

El dataset *Credit Card Approval Prediction* fue tomado de Kaggle una plataforma para la comunidad de el área de Ciencia de Datos. Fue proporcionado por el usuario MoneyMan en el 2020, con el objetivo de predecir si un cliente es bueno o malo para la obtención de la tarjeta de crédito.

El dataset está compuesto de dos archivos csv (Comma Separated Values), *application_record.csv* y *credit_record.csv* con 18 y 3 columnas respectivamente, las cuales se enumeran a continuación

Tabla 1 Atributos de application_record

Application_record	
Atributo	Explicación
ID	Numero de cliente
CODE_GENDER	Sexo
FLAG_OWN_CAR	Tiene coche
FLAG_OWN_REALTY	Tiene propiedad
CNT_CHILDREN	Número de hijos
AMT_INCOME_TOTAL	Ingresos anuales
NAME_INCOME_TYPE	Categoría de ingresos
NAME_EDUCATION_TYPE	Nivel de estudios
NAME_FAMILY_STATUS	Estado Civil
NAME_HOUSING_TYPE	Tipo de vivienda
DAYS_BIRTH	Nacimiento contado hacia atrás desde el día actual (0), -1 significa ayer
DAYS_EMPLOYED	Fecha de comienzo de trabajo, contado hacia atrás desde el día actual (0). Si es positivo, significa que la persona está desempleada actualmente.
FLAG_MOBIL	Si hay un teléfono celular
FLAG_WORK_PHONE	Tiene teléfono de trabajo
FLAG_PHONE	Tiene teléfono de casa
FLAG_EMAIL	Tiene correo electrónico
OCCUPATION_TYPE	Ocupación
CNT_FAM_MEMBERS	Tamaño de la familia

Tabla 2 Atributos credit_record

Credit_record	
ID	Numero de cliente
MONTHS_BALANCE	El mes de los datos extraídos es el punto de partida, hacia atrás, 0 es el mes actual, -1 es el mes anterior, y así sucesivamente
STATUS	0: 1-29 días de demora 1: 30-59 días de demora 2: 60-89 días de demora 3: 90-119 días de demora 4: 120-149 días de demora 5: Deudas vencidas o incobrables, cancelaciones de más de 150 días C: cancelado ese mes X: Sin préstamo ese mes

El primero cuenta con 438557 patrones, mientras que el segundo cuenta con 1048575 de patrones.

Preprocesamiento de los datos

Exploración inicial

Como primer paso al decidir hacer clasificación necesitamos encontrar la etiqueta que nos permita conocer a que clase pertenece cada cliente, para ello, ya que el dataset no cuenta con una etiqueta especifica, decidimos comenzar el análisis con el archivo Credit_record y debido a que esta no cuenta con valores perdidos, realizamos una clasificación manual mediante el siguiente umbral:

Consideramos buenos aquellos clientes cuyos datos pertenezcan a las clases 0, C o X, y malos aquellos que pertenezcan a 1, 2, 3 o 4, quedando de la siguiente forma:

	ID	MONTHS_BALANCE	STATUS	Bueno o Malo
0	5001711	0	X	Bueno
1	5001711	-1	0	Bueno
2	5001711	-2	0	Bueno
3	5001711	-3	0	Bueno
4	5001712	0	C	Bueno

Ilustración 1 Creación del atributo Bueno Malo

Procedemos a contar los datos y organizarlos de acuerdo con el ID y a la etiqueta Bueno o Malo y guardarlo en una nueva columna, obteniendo:

	ID	Bueno o Malo	size
0	5001711	Bueno	4
1	5001712	Bueno	19
2	5001713	Bueno	22
3	5001714	Bueno	15
4	5001715	Bueno	60

Ilustración 2 Agregando columna size

Elegimos el valor más grande por cliente y es lo que terminamos guardando, por último, hacemos el umbral considerando la siguiente distribución:

- 0 para todos aquellos etiquetados como malos
- 1 para todos los clientes que han pagado bien los últimos 0 a 20 meses
- 2 para todos los clientes que han pagado bien los últimos 20 a 40 meses
- 2 para los clientes que han pagado bien los últimos 40 o más meses

Y procedemos a combinarla con el archivo application_record por medio del ID y obtenemos un dataset con las siguientes dimensiones: 36457 rows × 19 columns.

Valores perdidos:

En la última versión de datos encontramos que la única columna con valores perdidos es la de la ocupación de los clientes obteniendo un total de 11323 valores perdidos.

Haciendo un análisis superficial, podemos concluir rápidamente que la columna Ocupación no puede ser desechada de nuestro dataset, ya que esta es una de las más relevantes al momento de clasificar nuestros objetivos. Sabiendo lo anterior, procedemos a imputar los datos faltantes mediante un modelo clasificador de RandomForest. Este modelo, analiza las variables externas del dataset, y utilizando un conjunto de árboles de decisión aleatorios clasifica nuestra categoría.

La metodología que utilizamos para entrenar el modelo clasificador fue, seleccionar las columnas o variables de interés para la clasificación, y al ser estas variables categóricas, crear dummies con ellas. Una vez creados los dummies, se aplicó validación de datos por medio de train_test_split, que separa los datos de una forma no estratificada con una proporción de 80% de los datos para entrenamiento y 20% para pruebas. Con lo anterior, el modelo obtuvo una precisión del 93% y respeto la proporción original de los datos.

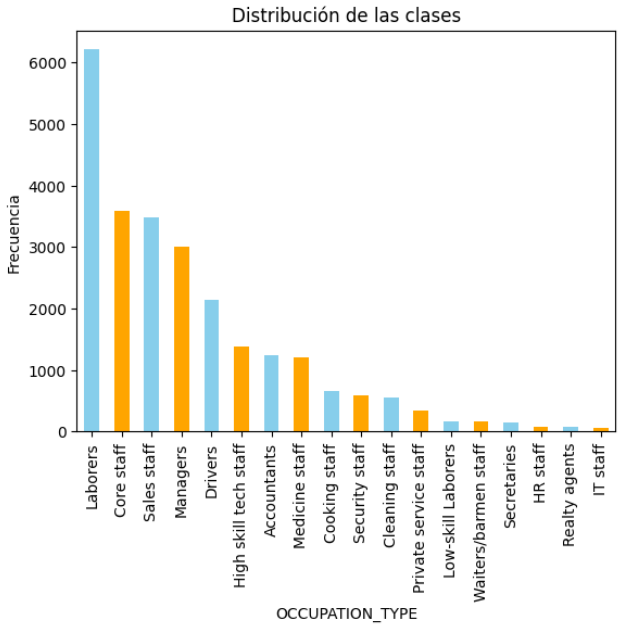

Ilustración 3 Distribución de Ocupación antes de la imputación

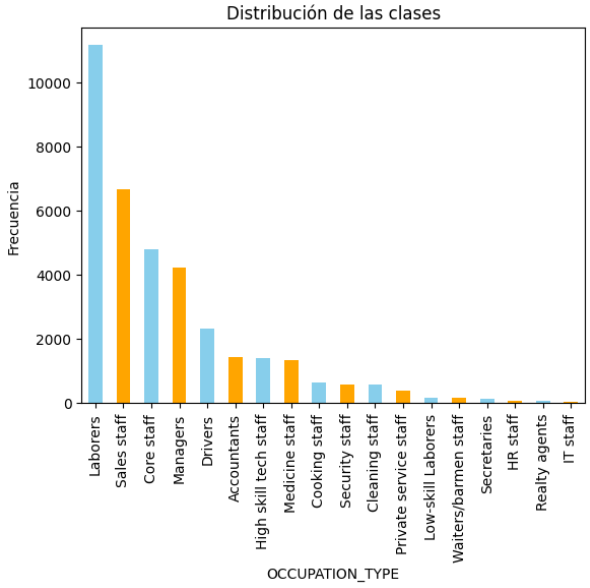

Ilustración 4 Distribución de Ocupación después de la imputación

Podemos observar que la imputación mantuvo la misma distribución por clase y eliminamos los valores nulos del dataset.

Proseguimos codificando las variables categóricas a variables numéricas, mediante un mapeo sencillo asignando un numero entre 0 y N clases de forma discreta.

Desbalance de clases

Ilustración 5 Desbalance de clases

Una vez preprocesado el dataset comenzamos la limpieza final comenzando con el desbalanceo de clases que vemos es muy superior al umbral permitido (véase figura 5), para ello decidimos ocupar el over_sampling de las clases minoritarias con 20000 datos sintéticos, dejando un 65-35 a las mismas respectivamente, pues al realizar un under_sampling la precisión de nuestros algoritmos de clasificación disminuyo alrededor de un 40%. Para realizar este proceso analizamos la distribución de cada atributo de la clase minoritaria. Obteniendo el siguiente resultado:

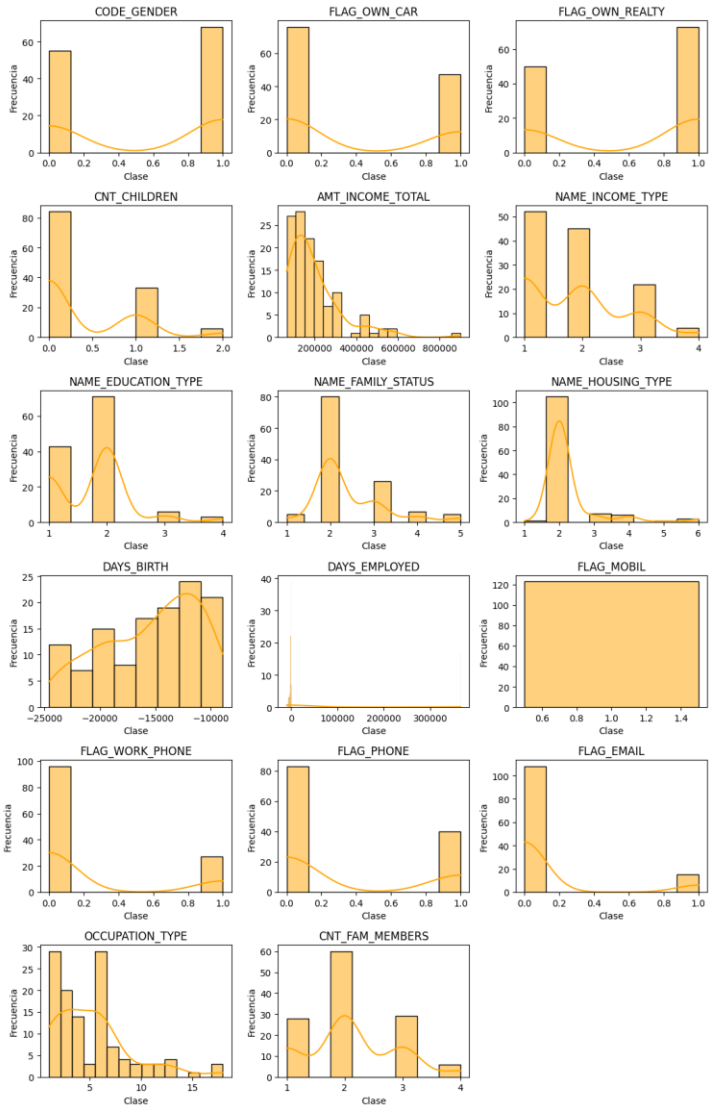

Ilustración 6 Distribución de la clase minoritaria 0

Podemos observar que en su mayoría para los atributos que no son biclase tenemos una distribución cercana a la exponencial, y esta es la que ocuparemos para la generación de datos sintéticos, decidimos crear un 35% de datos sintéticos, para así balancear el conjunto de datos obteniendo las siguientes distribuciones:

- Para los atributos con clases menores a 10 clases que seleccionará aleatoriamente datos de las clases existentes con la probabilidad del conjunto original
- Para el atributo “OCCUPATION_TYPE” calculamos las probabilidades dividiendo el conteo total entre las ocurrencias logrando así una distribución proporcional.

- Para el resto de las columnas creamos una distribución exponencial con una escala de 1/media si la media es mayor a 0 o 1 en otro caso

Obteniendo los siguientes resultados:

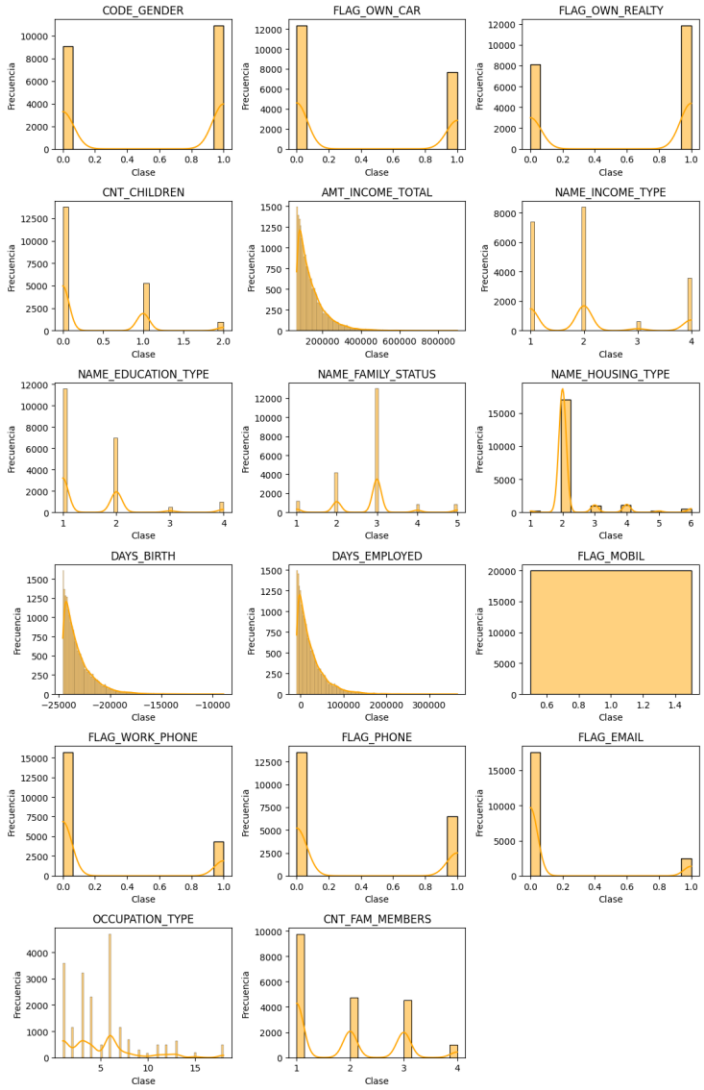

Ilustración 7 Distribución de la clase minoritaria 0 con datos sintéticos

En el caso de la segunda clase minoritaria:

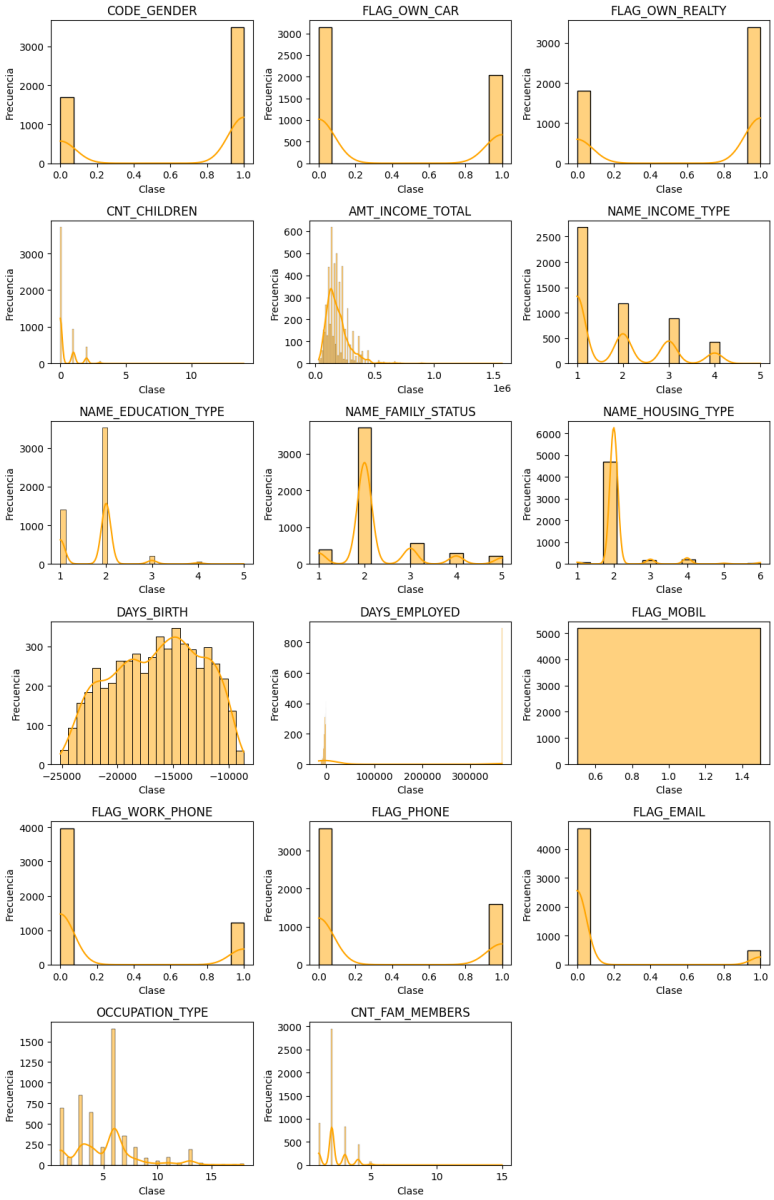

Ilustración 8 Distribución de la clase minoritaria 3

Observamos una distribución similar a la clase minoritaria anterior, a excepción de days_birth la cual se asemeja más a una distribución normal truncada, la cual es la única modificación que realizaremos a la función para calcular los valores sintéticos, obteniendo:

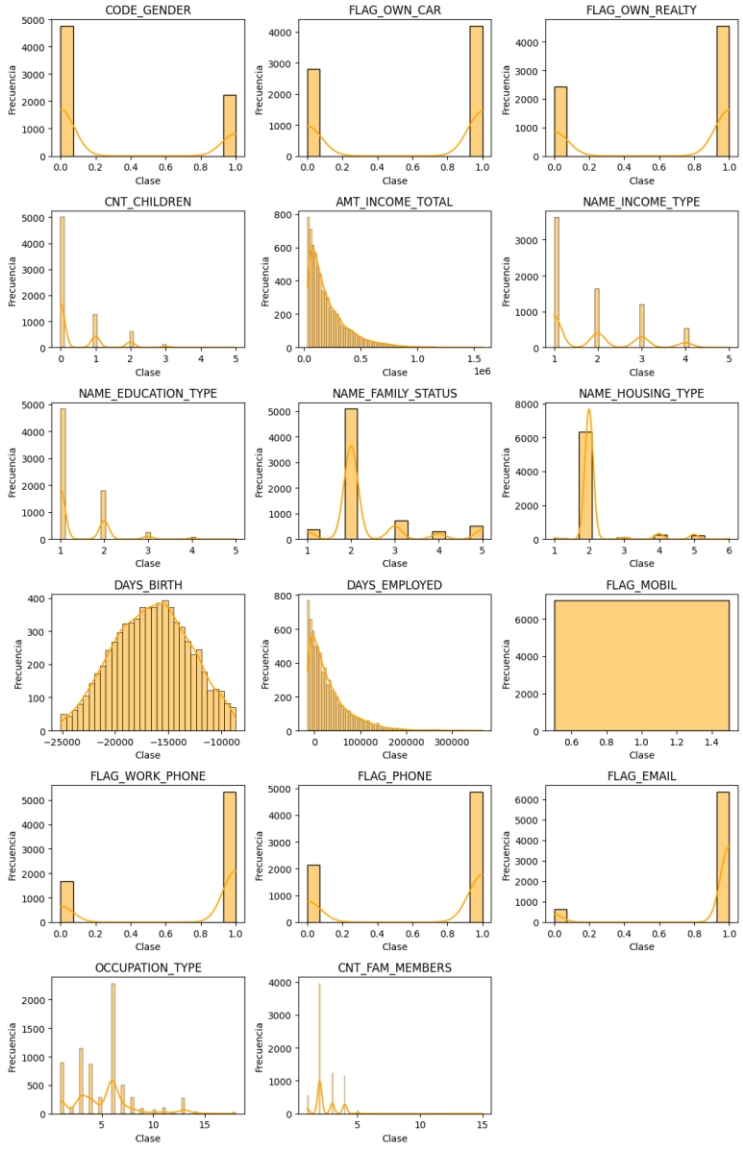

Ilustración 9 Distribución de las clases en la clase minoritaria 3 de valores sintéticos

Y fusionándolo con los valores reales obtenemos la siguiente distribución de clases:

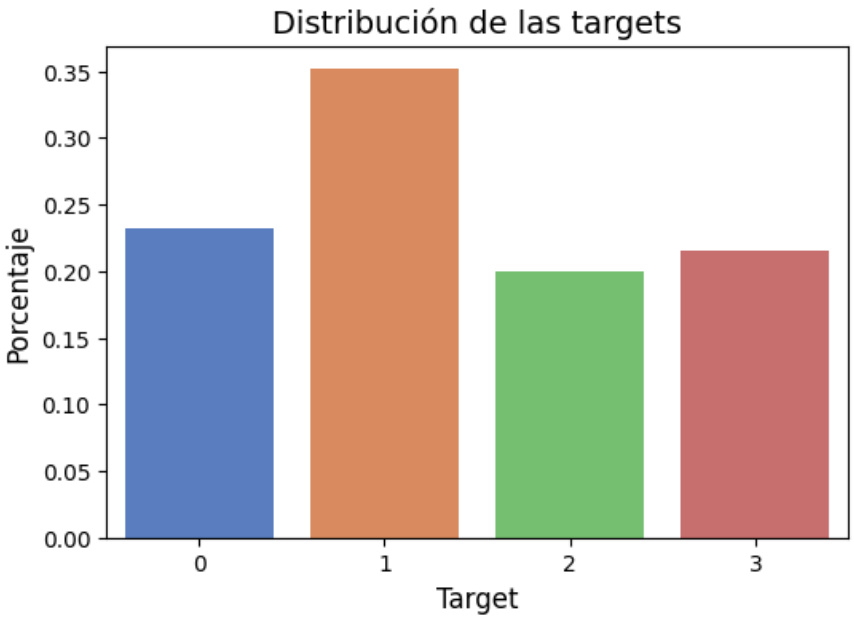

Ilustración 10 Desbalanceo de clases después del tratamiento

Podemos observar que el desbalanceo ha quedado resuelto.

Valores Atípicos

Una vez tratado el desbalanceo de clases, continuamos con los valores atípicos, calculando el IQR y graficándolo con una gráfica del tipo boxplot, obteniendo:

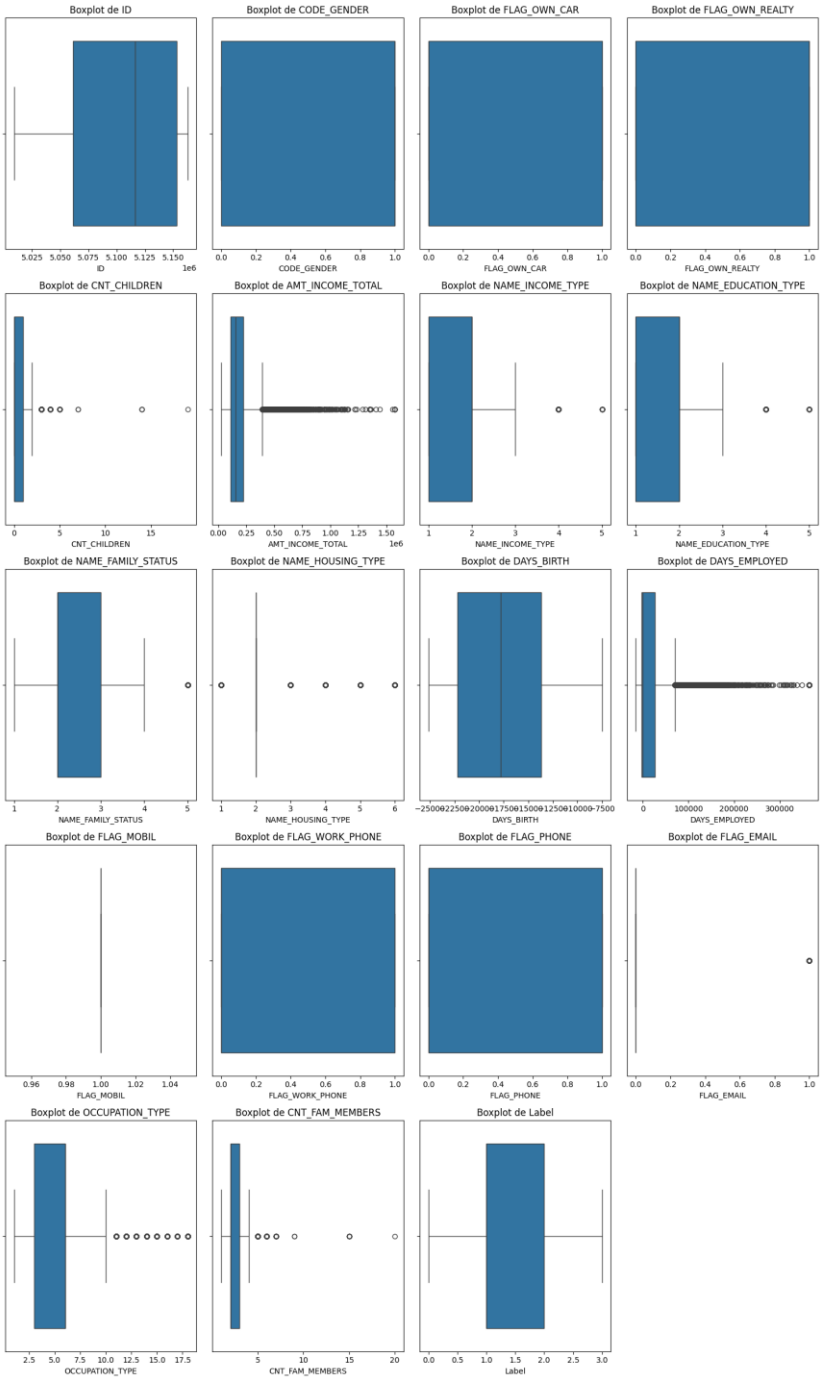

Ilustración 11 Valores Atípicos

Podemos observar que existen atributos con gran cantidad de valores atípicos, por lo cual no podíamos eliminarlos, decidiendo así ocupar la técnica Winzorization que los traslada al valor interno mas cercano, logrando los siguiente:

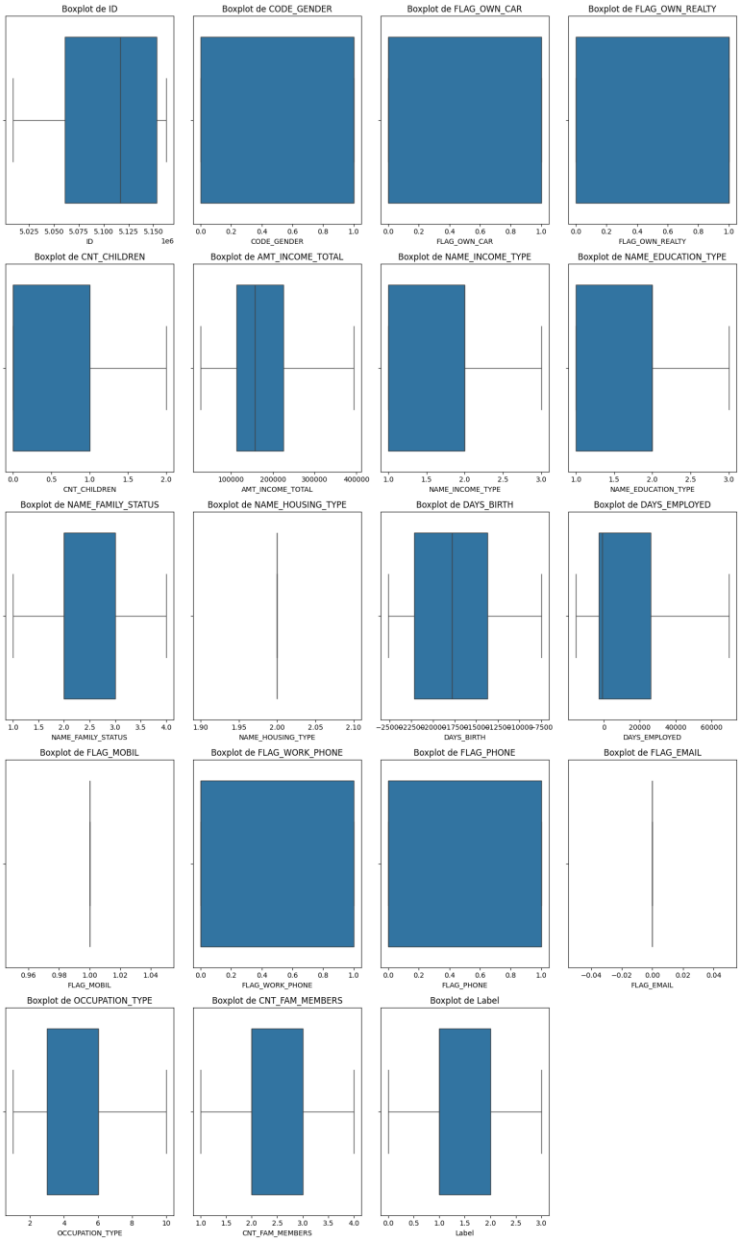

Ilustración 12 Valores atípicos después del tratamiento

Validación y normalización de los datos

Para la normalización de los datos decidimos hacer dos diferentes pruebas para conocer cual daba mejores resultados, las cuales fueron Hold Out estratificado 70-30 y K-Fold con 5 Folds, después de eso normalizamos los datos con StandardScaler para obtener mejores resultados.

Procesamiento de datos

Métricas de evaluación

Random Forest Classifier

Validación HoldOut

```
Matriz de Confusión - Random Forest:
[[2553   13    3   56]
 [   8 2961  734  265]
 [   4  928 1037  292]
 [  67  367  308 1695]]

Reporte de Clasificación - Random Forest:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      2625
           1       0.69      0.75      0.72      3968
           2       0.50      0.46      0.48      2261
           3       0.73      0.70      0.71      2437

    accuracy                           0.73     11291
   macro avg       0.72      0.72      0.72     11291
weighted avg       0.73      0.73      0.73     11291
```

Validación K-Fold

```
Entrenando el pliegue 1 con Random Forest...
Reporte de clasificación para el pliegue 1:
              precision    recall  f1-score   support

           0       0.97      0.98      0.97      2624
           1       0.66      0.70      0.68      3369
           2       0.53      0.51      0.52      2261
           3       0.74      0.70      0.72      2438

    accuracy                           0.73     10692
   macro avg       0.73      0.72      0.72     10692
weighted avg       0.73      0.73      0.73     10692


Entrenando el pliegue 2 con Random Forest...
Reporte de clasificación para el pliegue 2:
              precision    recall  f1-score   support

           0       0.97      0.98      0.97      2624
           1       0.67      0.71      0.69      3369
           2       0.52      0.51      0.51      2261
           3       0.74      0.68      0.71      2438

    accuracy                           0.73     10692
   macro avg       0.72      0.72      0.72     10692
weighted avg       0.73      0.73      0.73     10692


Entrenando el pliegue 3 con Random Forest...
Reporte de clasificación para el pliegue 3:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      2625
           1       0.67      0.71      0.69      3368
           2       0.52      0.50      0.51      2261
           3       0.74      0.70      0.72      2437

    accuracy                           0.73     10691
   macro avg       0.72      0.72      0.72     10691
weighted avg       0.73      0.73      0.73     10691
```

```
Entrenando el pliegue 4 con Random Forest...
Reporte de clasificación para el pliegue 4:
              precision    recall  f1-score   support

           0       0.97      0.98      0.97      2625
           1       0.66      0.70      0.68      3368
           2       0.52      0.50      0.51      2260
           3       0.74      0.69      0.71      2438

    accuracy                           0.73     10691
   macro avg       0.72      0.72      0.72     10691
weighted avg       0.72      0.73      0.72     10691


Entrenando el pliegue 5 con Random Forest...
Reporte de clasificación para el pliegue 5:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      2625
           1       0.66      0.69      0.68      3368
           2       0.51      0.50      0.50      2260
           3       0.72      0.70      0.71      2438

    accuracy                           0.72     10691
   macro avg       0.72      0.71      0.72     10691
weighted avg       0.72      0.72      0.72     10691
```

Gradient Boosting Classifier (XGBoost)

Usando validación Hold Out

Usando validación K-Fold

```
  warnings.warn(smsg, UserWarning)
Matriz de Confusión - XGBoost:
[[3812   40    9   76]
 [   9 4509  524   11]
 [   3 2609  767   12]
 [  91 1222  333 2011]]

Reporte de Clasificación - XGBoost:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      3937
           1       0.54      0.89      0.67      5053
           2       0.47      0.23      0.31      3391
           3       0.95      0.55      0.70      3657

    accuracy                           0.69     16038
   macro avg       0.73      0.66      0.66     16038
weighted avg       0.73      0.69      0.67     16038
```

```
Entrenando el pliegue 1 con Gradient Boosting...
Reporte de clasificación para el pliegue 1:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      2625
           1       0.55      0.99      0.71      3969
           2       0.44      0.02      0.05      2260
           3       0.96      0.56      0.70      2438

    accuracy                           0.70     11292
   macro avg       0.73      0.63      0.61     11292
weighted avg       0.72      0.70      0.64     11292


Entrenando el pliegue 2 con Gradient Boosting...
Reporte de clasificación para el pliegue 2:
              precision    recall  f1-score   support

           0       0.98      0.97      0.97      2625
           1       0.55      0.98      0.70      3969
           2       0.41      0.03      0.06      2260
           3       0.96      0.55      0.70      2438

    accuracy                           0.70     11292
   macro avg       0.72      0.63      0.61     11292
weighted avg       0.71      0.70      0.64     11292


Entrenando el pliegue 3 con Gradient Boosting...
Reporte de clasificación para el pliegue 3:
              precision    recall  f1-score   support

           0       0.97      0.98      0.97      2624
           1       0.55      0.98      0.70      3968
           2       0.44      0.04      0.07      2261
           3       0.98      0.55      0.70      2438

    accuracy                           0.70     11291
   macro avg       0.74      0.64      0.61     11291
weighted avg       0.72      0.70      0.64     11291

Entrenando el pliegue 4 con Gradient Boosting...
Reporte de clasificación para el pliegue 4:
              precision    recall  f1-score   support

           0       0.98      0.97      0.98      2624
           1       0.55      0.99      0.70      3968
           2       0.51      0.03      0.06      2261
           3       0.96      0.54      0.69      2438

    accuracy                           0.70     11291
   macro avg       0.75      0.63      0.61     11291
weighted avg       0.73      0.70      0.64     11291


Entrenando el pliegue 5 con Gradient Boosting...
Reporte de clasificación para el pliegue 5:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      2625
           1       0.55      0.98      0.70      3968
           2       0.42      0.04      0.07      2261
           3       0.96      0.55      0.70      2437

    accuracy                           0.70     11291
   macro avg       0.73      0.63      0.61     11291
weighted avg       0.71      0.70      0.64     11291
```

Decision Tree Classifier

Usando validación Hold Out

```
Matriz de Confusión - XGBoost:
[[3745   19   13  160]
 [  24 4058 1337  534]
 [  12 1222 1680  477]
 [ 137  516  471 2533]]

Reporte de Clasificación - XGBoost:
              precision    recall  f1-score   support

           0       0.96      0.95      0.95      3937
           1       0.70      0.68      0.69      5953
           2       0.48      0.50      0.49      3391
           3       0.68      0.69      0.69      3657

    accuracy                           0.71     16938
   macro avg       0.70      0.71      0.70     16938
weighted avg       0.71      0.71      0.71     16938
```

Usando validación K-Fold

```
Entrenando el pliegue 1 con Gradient Boosting...
Reporte de clasificación para el pliegue 1:
              precision    recall  f1-score   support

           0       0.96      0.96      0.96      2625
           1       0.71      0.70      0.71      3969
           2       0.48      0.50      0.49      2260
           3       0.70      0.68      0.69      2438

    accuracy                           0.72     11292
   macro avg       0.71      0.71      0.71     11292
weighted avg       0.72      0.72      0.72     11292


Entrenando el pliegue 2 con Gradient Boosting...
Reporte de clasificación para el pliegue 2:
              precision    recall  f1-score   support

           0       0.96      0.96      0.96      2625
           1       0.71      0.69      0.70      3969
           2       0.48      0.50      0.49      2260
           3       0.68      0.68      0.68      2438

    accuracy                           0.71     11292
   macro avg       0.70      0.71      0.71     11292
weighted avg       0.71      0.71      0.71     11292


Entrenando el pliegue 3 con Gradient Boosting...
Reporte de clasificación para el pliegue 3:
              precision    recall  f1-score   support

           0       0.95      0.96      0.95      2624
           1       0.70      0.68      0.69      3968
           2       0.47      0.48      0.47      2261
           3       0.67      0.69      0.68      2438

    accuracy                           0.71     11291
   macro avg       0.70      0.70      0.70     11291
weighted avg       0.71      0.71      0.71     11291

Entrenando el pliegue 4 con Gradient Boosting...
Reporte de clasificación para el pliegue 4:
              precision    recall  f1-score   support

           0       0.97      0.96      0.96      2624
           1       0.69      0.69      0.69      3968
           2       0.47      0.48      0.47      2261
           3       0.68      0.68      0.68      2438

    accuracy                           0.71     11291
   macro avg       0.70      0.70      0.70     11291
weighted avg       0.71      0.71      0.71     11291


Entrenando el pliegue 5 con Gradient Boosting...
Reporte de clasificación para el pliegue 5:
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      2625
           1       0.70      0.70      0.70      3968
           2       0.48      0.48      0.48      2261
           3       0.68      0.69      0.69      2437

    accuracy                           0.71     11291
   macro avg       0.71      0.71      0.71     11291
weighted avg       0.71      0.71      0.71     11291
```

Resultados de los modelos

Random Forest Classifier

Para los primeros experimentos se utilizo el algoritmo clasificador Random Forest. Este algoritmo, es un método de Machine Learning basado en el algoritmo Decision Tree, pero utilizando de forma extendida, ampliando así la eficacia y robustez del modelo.

Con nuestro enfoque, el método de aprendizaje automático fue utilizado con todas las variables del dataset, tratando así de predecir el *target* definido durante la limpieza de datos.

El modelo, fue tratado con dos enfoques distintos de validación de datos, dando con ellos resultados muy similares. Para el caso de la validación por medio de Hold Out Estratificado, se obtuvo una precisión promedio de las cuatro categorías de la variable objetivo del 73%. Por otro lado, usando la validación K-Fold, y definiendo 5 folds, obtuvimos un promedio de precisión del 73% en 4 de los 5 folds, siendo el diferente una precisión del 72%, la cual no se aleja para nada de los folds anteriores.

En cuanto a tiempos de ejecución, y recordando que el conjunto de datos sobrepasa los 50 mil registros, el algoritmo se comporto de manera eficiente y no llevo demasiado realizar estos experimentos.

Concluyendo así, que ambas validaciones aportan resultados similares y ninguna es realmente más eficaz que la otra, al menos utilizando este algoritmo clasificador.

Gradient Boosting Classifier (XGBoost)

Continuando con el segundo algoritmo clasificador, se implemento un modelo de Gradient Boosting XGB. A diferencia del algoritmo de GB común, este es un método avanzado de clasificación que se sirve de combinar modelos más débiles, por ejemplo, arboles de decisión. A diferencia del modelo anterior, este no construye los arboles de forma independiente, sino secuencial, corrigiendo así los errores del árbol anterior y obteniendo mayor precisión.

Se entreno el modelo, de igual forma, utilizando nuestros dos métodos de validación. Al utilizar el método Hold Out Estratificado, se obtuvo una precisión promedio de las cuatro categorías de 69%, por lo que podemos observar una disminución del 5% en la precisión respecto al modelo anterior. Esto se puede adjudicar directamente al desbalance de clases que aun cuando fue tratado previamente en la limpieza de datos, seguía existiendo para ciertas clases en el conjunto de datos. Al ser este un algoritmo secuencial, puede sufrir de sesgos en sus predicciones, por lo que nos quedamos con una disminución de precisión.

Por otra parte, utilizando la validación K-Folds, obtuvimos una precisión consistente de 70% en promedio de todas las categorías. Viendo así un aumento solo del 1% frente a la validación anterior.

Hablando de tiempos de ejecución, este algoritmo fue el más lento de los tres que fueron posibles de evaluar, no obstante, tampoco constituyo un tiempo de más de un par de minutos para realizar el experimento.

Podemos concluir así, que el algoritmo sufrió debido al desbalance de clases, pues aun cuando obtuvo una presión medianamente aceptable, observamos que la desventaja ante sus iguales es clara, siendo inferior en las métricas, y más costoso computacionalmente hablando.

Decision Tree Classifier

Para el ultimo algoritmo tratado se probó con Decision Tree, un modelo de aprendizaje supervisado que puede ser utilizando en regresiones o clasificaciones. Este algoritmo divide iterativamente subconjuntos de datos según características o variables, creando así una estructura de datos en forma de árbol.

Este modelo, fue probado en primera instancia con la validación Hold Out Estratificado, obteniendo los resultados de 71% de precisión promedio de las categorías de la variable objetivo. Siendo con ese número, superior al modelo del Gradient Boosting, pero, inferior al modelo de Random Forest. Teniendo sentido, pues este algoritmo marca un punto medio entre la deficiencia de Gradient Boosting al trabajar con conjuntos de datos desbalanceados, y, la superioridad de Random Forest al corregir los errores y sesgos individuales de arboles independientes.

Para la validación K-Fold, se obtuvieron 4 de los 5 folds con un resultado promedio de 71% de precisión, y un último fold con un resultado de 72%, siendo consistente a la validación anterior, y no mejorando realmente en los experimentos.

Por su parte, Decision Tree cuenta con un tiempo de ejecución y coste computacional promedio en muchos sentidos, no implicando mayor dificultad para experimentar con él.

En conclusión, este es un algoritmo promedio que puede ser bastante útil y versátil en más de un sentido, aportando consistencia y seguridad, mientras que se sacrifica un poco la precisión máxima que es posible obtener.

Support Vector Machine

Para este algoritmo, el cuál pretendía ser el último con el que se experimentaría, y se esperaba tener los mejores resultados, no fue posible realizar las pruebas debido al gran coste computacional que el algoritmo demanda.

Support Vector Machine o SVM, es un algoritmo utilizado tanto para clasificación o regresión. Trabaja creando un hiperplano que viva en la dimensión de la cantidad de variables que se estén utilizando, y que separe los datos en diferentes clases, para maximizar la distancia entre las categorías.

Al trabajar con altas dimensionalidades, este algoritmo es sumamente costoso para la computadora, por lo que, para nuestro dataset con una cantidad de registros considerable, este pierde abrumadoramente en tiempos de ejecución. Llevando así a los experimentos a ser imposibles de realizar en esta ocasión, y dejando la puerta abierta para futuros análisis más exhaustivos.

Conclusiones

- Pineda Hernández Francisco

A modo de cierre, podemos concluir muchas cosas respecto a este análisis e informe.

De primera mano, este análisis tuvo como objetivo crear un clasificador que pudiera predecir la aprobación de tarjetas de crédito utilizando modelos de Machine Learning y un preprocesamiento de datos completo y riguroso. Los resultados mostraron precisiones desde el 69% hasta el 75%, lo cual, basado en las métricas, nos da una perspectiva general del como trabajan los modelos clasificadores, como los mismos son sensibles a las correspondientes entradas con las que se les alimente. Observamos que, modelos como Random Forest son óptimos para trabajar datos que sufren de desbalance, mientras que otros como Gradient Boosting son menos precisos con los mismos.

De igual forma, pudimos verificar como el procesado de datos, la limpieza, la normalización y validación, son pasos sumamente importantes para el resultado final, pudiendo estos afectar enormemente a la calidad de un modelo, pues datos erróneos o inconsistentes pueden llevar a modelos sesgados o predicciones incorrectas. Es propio tratar los valores perdidos, los valores atípicos, la normalización, la codificación de variables categóricas, y los outliers para maximizar los resultados.

Por último, considero relevante mencionar el cómo un correcto análisis exploratorio de los datos es propio antes de comenzar a realizar cualquier trabajo, pues este puede ser clave a la hora de decidir si se darán manos a la obra, o si es mejor retirarse y buscar mejores fuentes. Tomar la decisión de realizar todo un proyecto conociendo las limitaciones puede implicar la perdida de recursos de muchos indoles; por lo que es importante tener en cuenta cada factor.

Este trabajo, puede proporcionar una base para el desarrollo de sistemas de predicción aplicables en instituciones bancarias o de sectores económicos, su implementación con los modelos de Machine Learning pueden facilitar la toma de decisiones estratégicas; pues, a pesar de las limitaciones sigue siendo un apoyo considerable para el sector.

- Ramirez Aguilar Rodrigo Vidal

El análisis de este dataset nos permitió desarrollar y mejorar técnicas para el análisis de datos, nuestro objetivo principal fue crear un clasificador que permitiera conocer si un cliente era candidato para obtener una tarjeta de crédito, al comienzo del análisis encontramos varios problemas dentro del dataset, en primera instancia este no contaba con un etiquetado para realizar la clasificación es por ello que decidimos crear un umbral, el segundo reto fue el basto desbalance de clases y encontrar una forma aceptable de abordarla, por ultimo dentro de la limpieza de datos encontramos una gran cantidad de valores perdidos dándonos cuenta que este al ser una cantidad extensa implementamos técnicas para tratarlos y no eliminarlos. Con esto pudimos darnos cuenta de la importancia que tiene la limpieza de datos y como esta afecta las fases posteriores, pues es indispensable tratarla de forma correcta antes de continuar.

Al implementar el algoritmo de Decision Tree encontré que este es un algoritmo que si bien no da los resultados más precisos, en términos de costos computacionales es más proporcional en costo beneficio, acerca del ultimo algoritmo utilizado, no pudimos obtener resultados debido al costo computacional de este, pero por la naturaleza del problema podría dar mejores resultados que los algoritmos anteriores pues teóricamente encontraría el hiperplano que divida las clases de forma más clara, es importante aclarar que para la utilización de este algoritmo en problemas similares es recomendable para una clasificación binaria y con pocos datos ya que puede tener una complejidad entre n^2 y n^3, por este motivo no pudo ser analizada de manera correcta.

Con el avance tecnológico y el aumento de las ventas en línea la demanda de tarjetas de créditos ha aumentado exponencialmente y es una necesidad de los bancos el encontrar formas de evadir los fraudes en este tipo de trámites para así minimizar las perdidas. Este modelo es útil para la vida real pues a pesar de no ser preciso puede dar una pauta a los agentes bancarios para la toma de una decisión a la hora de aceptar a un nuevo cliente. Una forma de mejorar los resultados podría ser el obtener mas datos reales que permitan un mejor balance en el banco de datos evitando así la sobreexplotación de datos sintéticos, también podría servir el uso de otros algoritmos como KNN para mejores resultados pues en el banco de datos se cuenta con fronteras poco establecidas.

Referencias bibliográficas

Díaz, R. (s. f.). Credit card approval prediction [Conjunto de datos]. Kaggle. Recuperado el 11 de enero de 2025, de https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction

Anexos

Tabla de ilustraciones