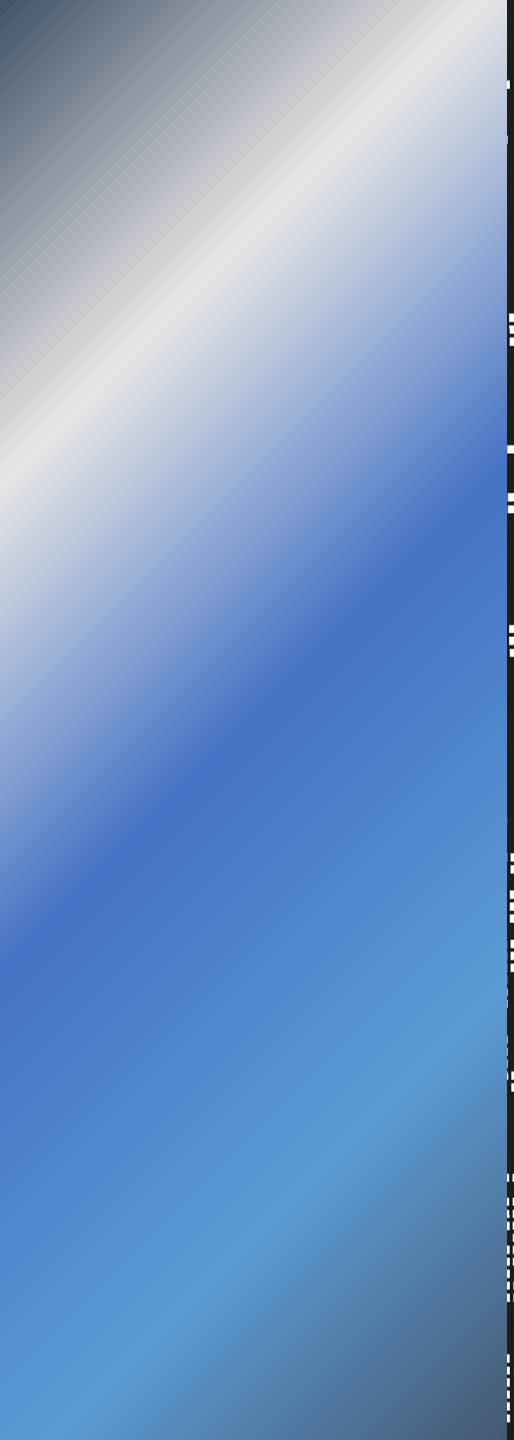


Prueba Técnica banco AV Villas

Manuel Alejandro Diaz Rubiano

Table of contents

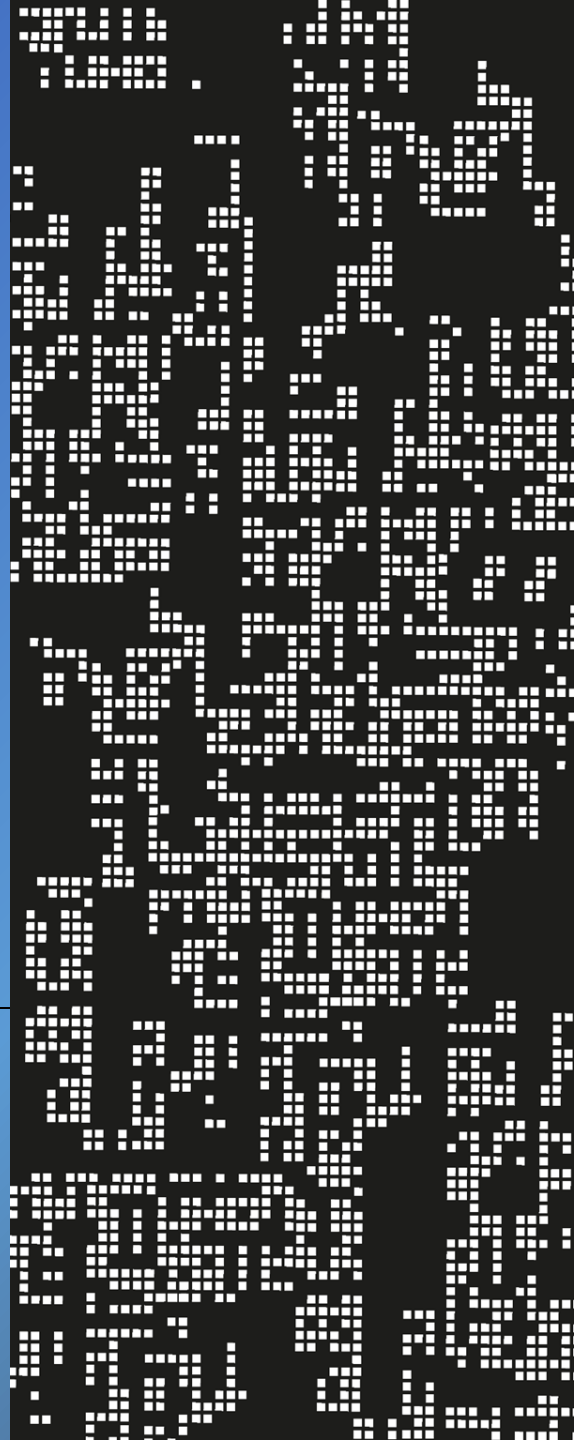
01 Project & strategy — You can describe the topic of the section here

02 Consulting proposal — You can describe the topic of the section here

03 Change management — You can describe the topic of the section here

04 Implementing changes — You can describe the topic of the section here

01 Comprensión del ejercicio

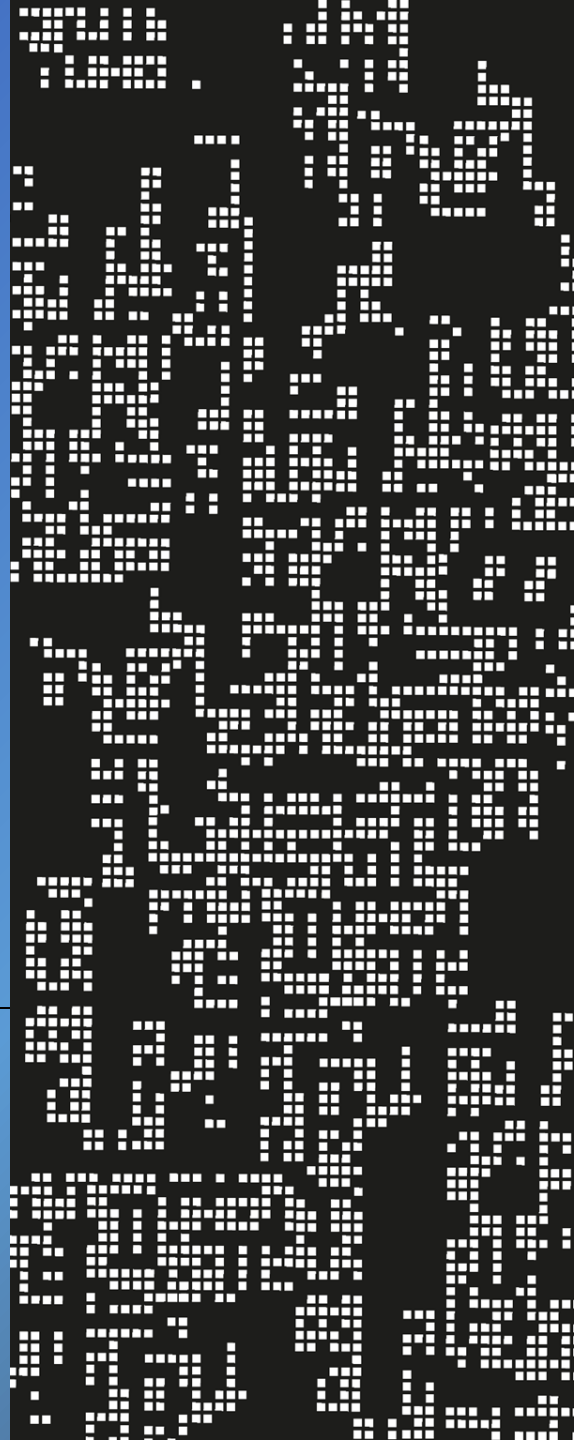


Comprension del ejercicio

El objetivo principal de este proyecto es desarrollar un modelo de aprendizaje automático que pueda predecir la probabilidad de incumplimiento de un cliente. Esta tarea es crucial para la gestión de riesgos y la toma de decisiones en el ámbito financiero, ya que permite a las instituciones anticipar y mitigar posibles pérdidas debidas a incumplimientos de pago.

El problema a abordar es un caso de clasificación binaria en el ámbito del crédito y riesgo financiero. Utilizando la base de datos proporcionada ("MDT_prueba.csv"), el modelo debe ser capaz de estimar la probabilidad de que un cliente incurra en incumplimiento en un periodo de 12 meses después de la "cosecha" (entendida como el momento de otorgamiento del crédito o servicio financiero).

02 Comprensión de los datos



Comprension de los Datos

La base de datos "MDT_prueba.csv" es el recurso principal para este proyecto. Contiene información detallada por cliente, que se utilizará para predecir la probabilidad de incumplimiento..

Archivo: MDT_prueba.csv

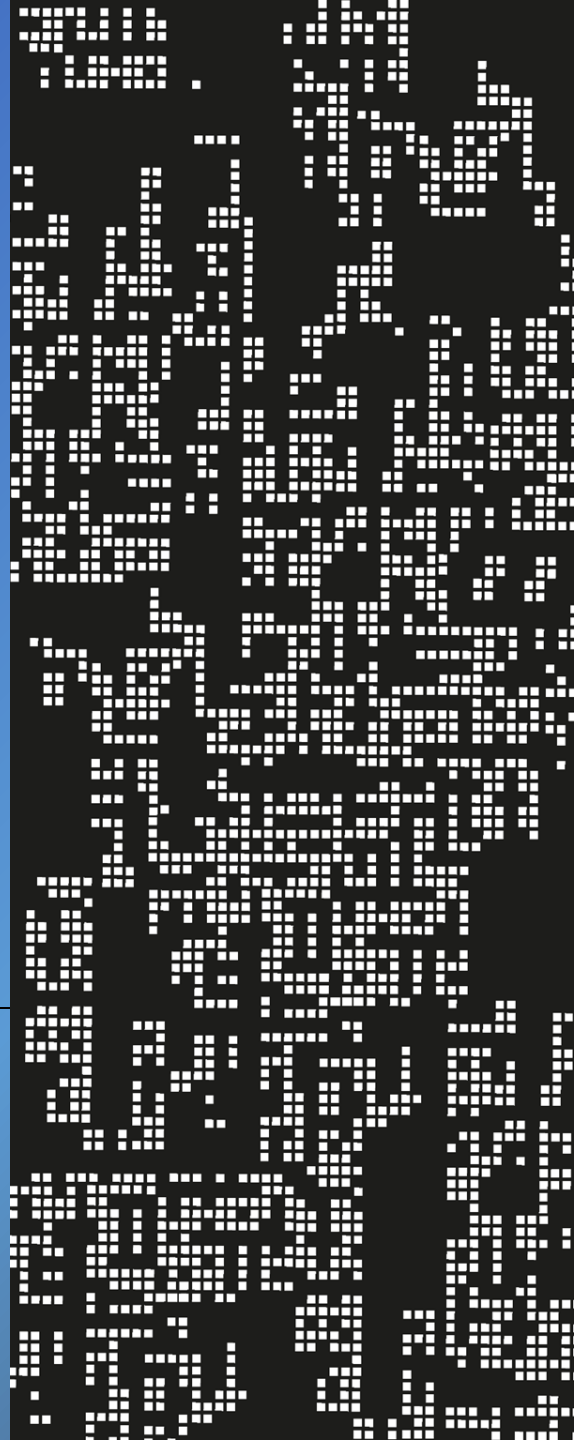
Variables: 100 variables en total.

Variable Objetivo: BGI_max (binaria: 1 indica incumplimiento, 0 no incumplimiento).

Variables Predictoras: 99 variables que pueden incluir datos demográficos, financieros, historial de crédito, comportamiento de pagos, entre otros.

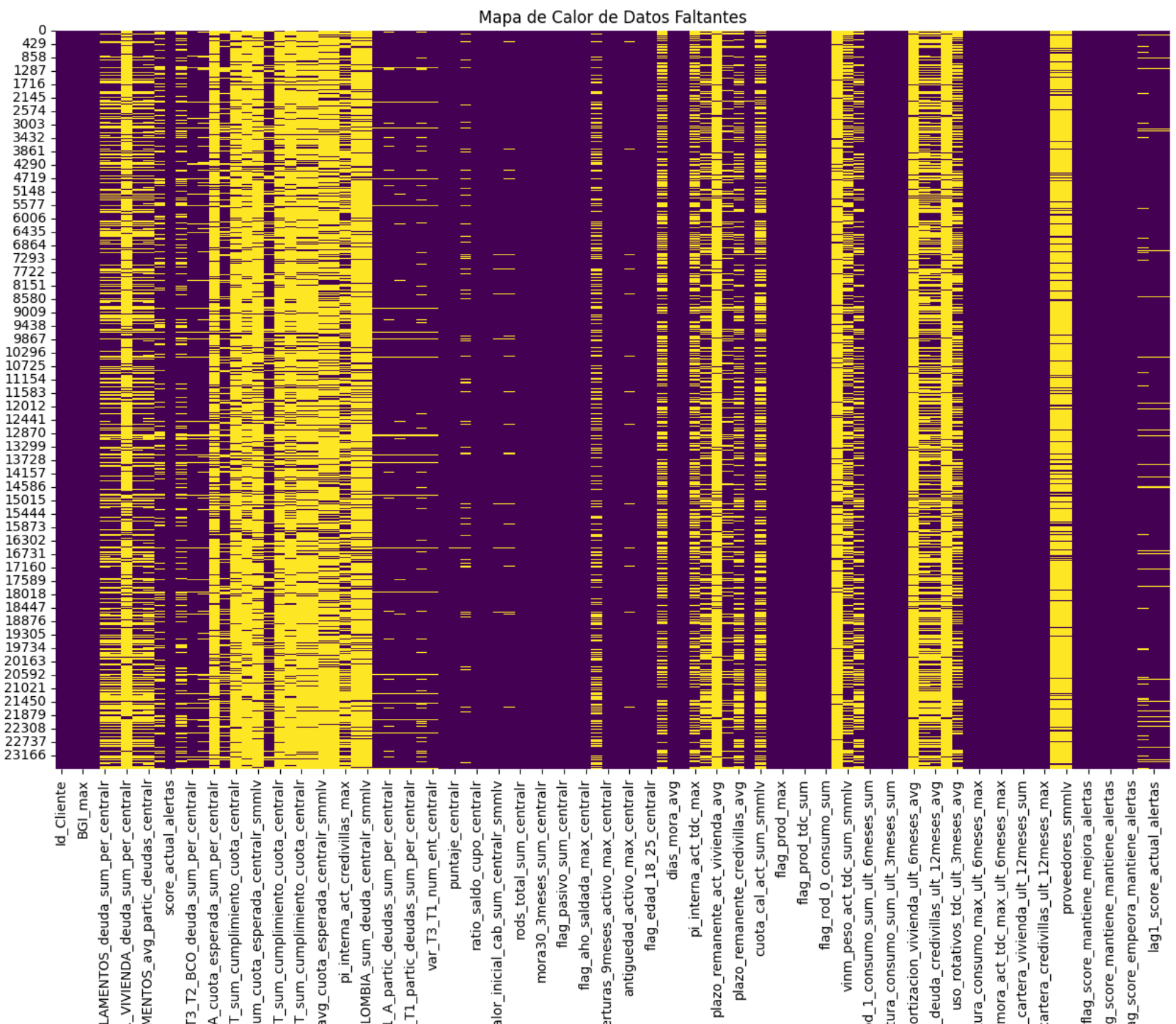
Cantidad de Registros: 23591

03 Preparación de los datos



Datos Faltantes

Se observa una gran cantidad de datos Nulos en ciertas columnas. Sabiendo que la base de datos tiene 23591 filas en total, las variables con 22343 valores Nulos son variables que no aportan casi nada al analisis.



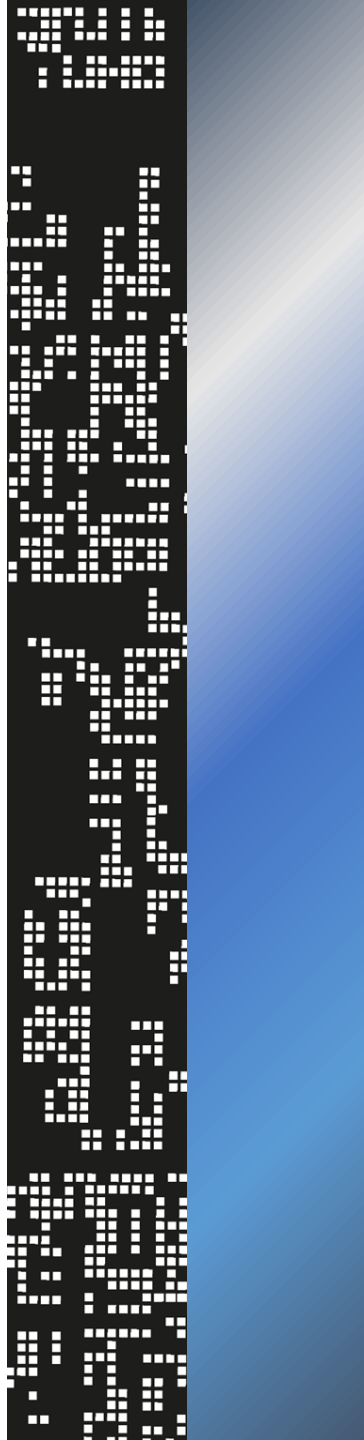
Imputacion de Datos Faltantes

Existen varias formas de tratar con los datos Nulos. Dos de las formas mas famosas son: Eliminar los datos nulos o imputarlos.

En este caso, se procedió eliminando aquellas columnas que tienen mas del 20% de los datos como nulos, y las que tienen menos del 20% de los datos nulos, se imputaran de la siguiente manera:

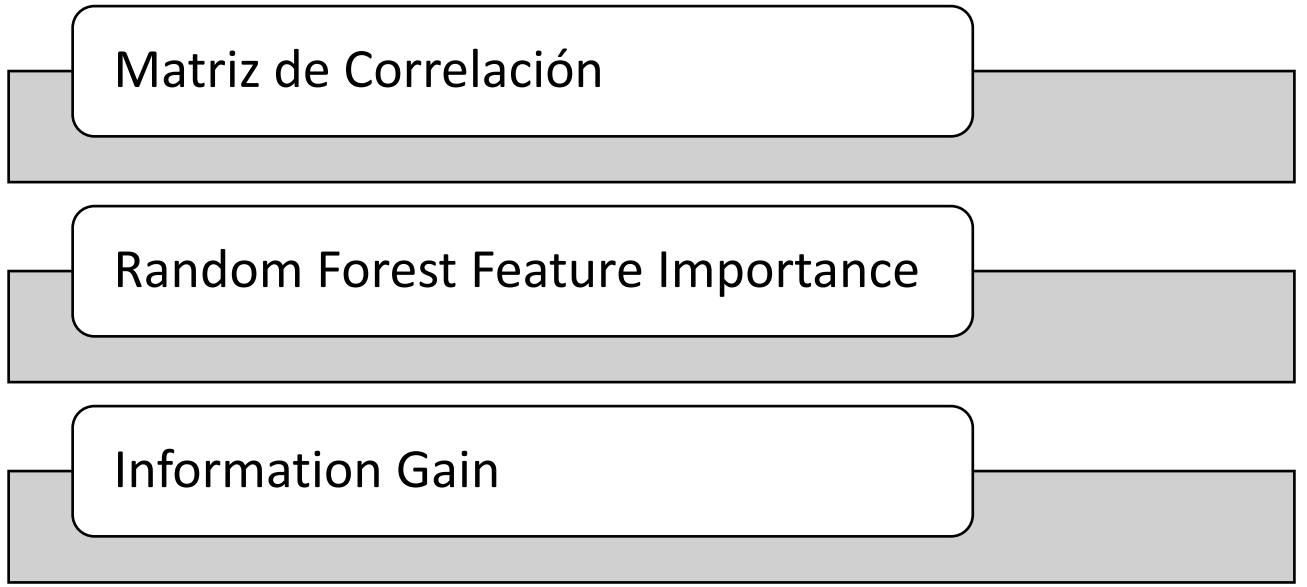
- los datos continuos se imputarán con la mediana
- los datos dicotómicos (por ejemplo 0 y 1), se creará un tercer código que será "99" y denota faltante

Se obtuvo que, de 102 variables iniciales, quedaron 65 variables que estamos seguros de que están completas.

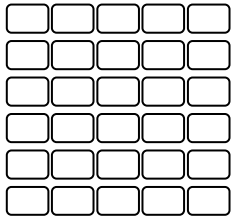


Importancia de las variables.

Aunque fue posible poder descartar una gran cantidad de variables debido a sus datos faltantes, 65 variables siguen siendo un gran número de variables. Por lo anterior, y sabiendo que nuestra base de datos resultante ya tiene una completitud deseada, se procede a analizar la importancia de las variables en relación a la variable objetivo que tenemos BGI_max, usando los siguientes procesos:



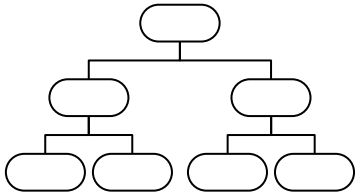
Metodos para evaluar Importancia de las Variables



01

Matriz de Correlacion

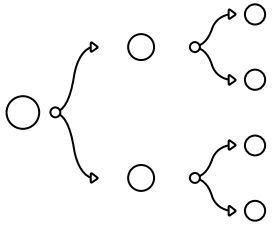
identifica la relación entre variables, seleccionando las más correlacionadas con la variable objetivo.



02

Random Forest

calcula la importancia de las características basándose en el aumento medio de la impureza de los nodos que utilizan la característica.



03

Information Gain

evalúa la reducción de la entropía (desorden) de una variable objetivo al conocer el valor de otras variables.

Variables Seleccionadas

- Para entender por qué ciertas variables fueron seleccionadas como las más importantes en la predicción del incumplimiento de clientes, es esencial considerar los tres métodos utilizados: Matriz de Correlación, Random Forest Feature Importance y Information Gain.

- Se seleccionaron en total, 32 variables, que es un numero de variables mas reducido y que es mas fácil lidiar, a diferencia de las 102 variables del inicio

Matriz de Correlación	Las variables seleccionadas, como puntaje_centralr y score_actual_alertas, mostraron una correlación significativa, ya sea positiva o negativa, con la variable objetivo.
Random Forest Feature Importance	Las variables más importantes, como puntaje_centralr, y balma_centralr, son aquellas que, en promedio, más contribuyen a mejorar la precisión de los árboles de decisión en el modelo.
Information Gain	Variables como flag_score_mantiene_mantiene_alertas y aperturas_9meses_activo_max_centralr tienen un alto Information Gain.

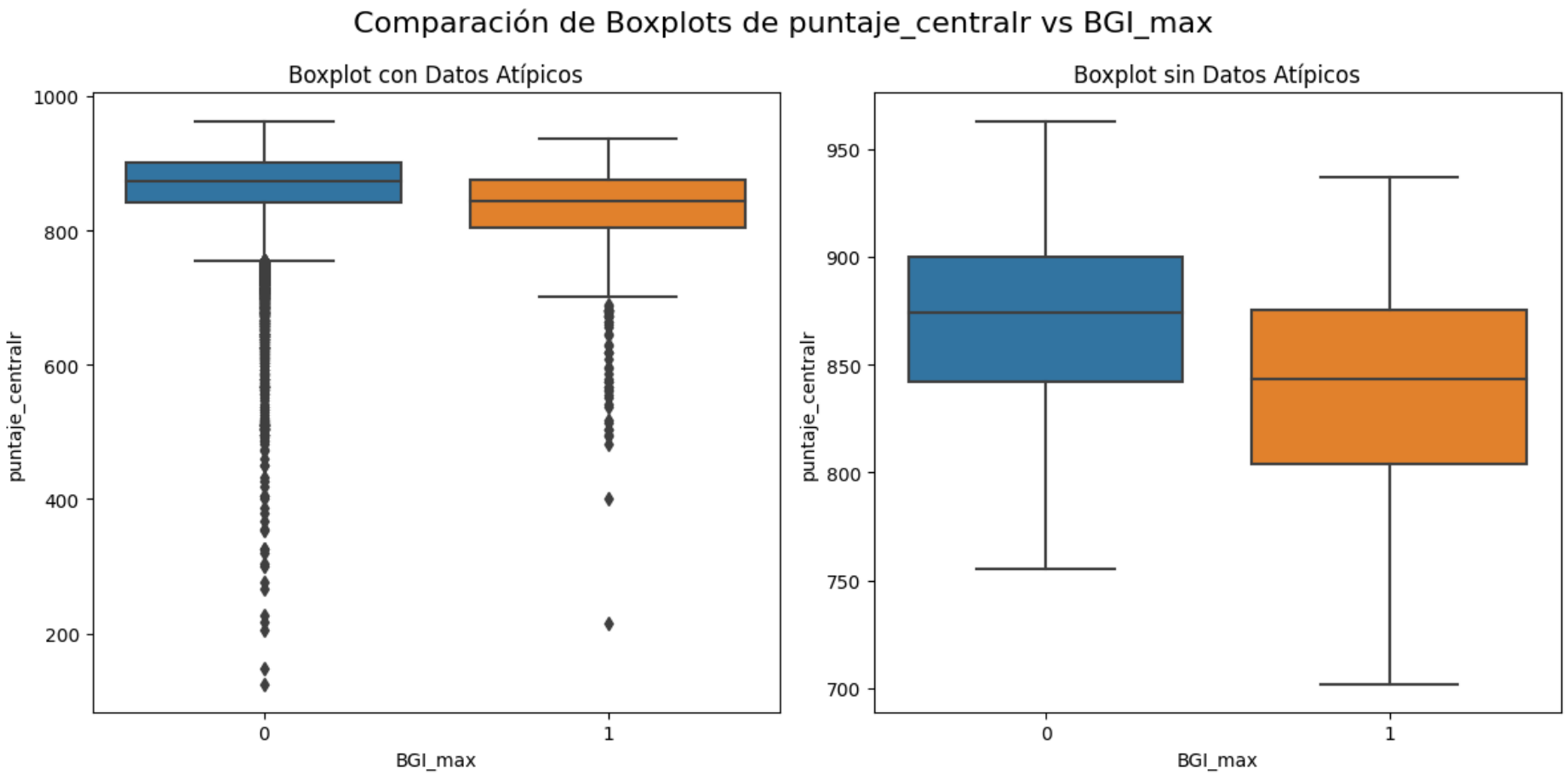
Outliers o Anomalías

En los datos seleccionados se observan varios indicadores que sugieren la presencia de datos atípicos en distintas columnas. Primero, se identifican valores máximos extremadamente altos en comparación con los valores del tercer cuartil en columnas como dias_mora_act_tdc_max_ult_6meses_max, balma_centralr, flag_rod_0_consumo_sum_ult_6meses_sum, Trim_3_AVVILLAS_sum_cumplimiento_cuota_centralr, y dias_mora_avg. Estos valores máximos son desproporcionadamente altos en relación con la mayoría de los datos, lo que sugiere la presencia de outliers.

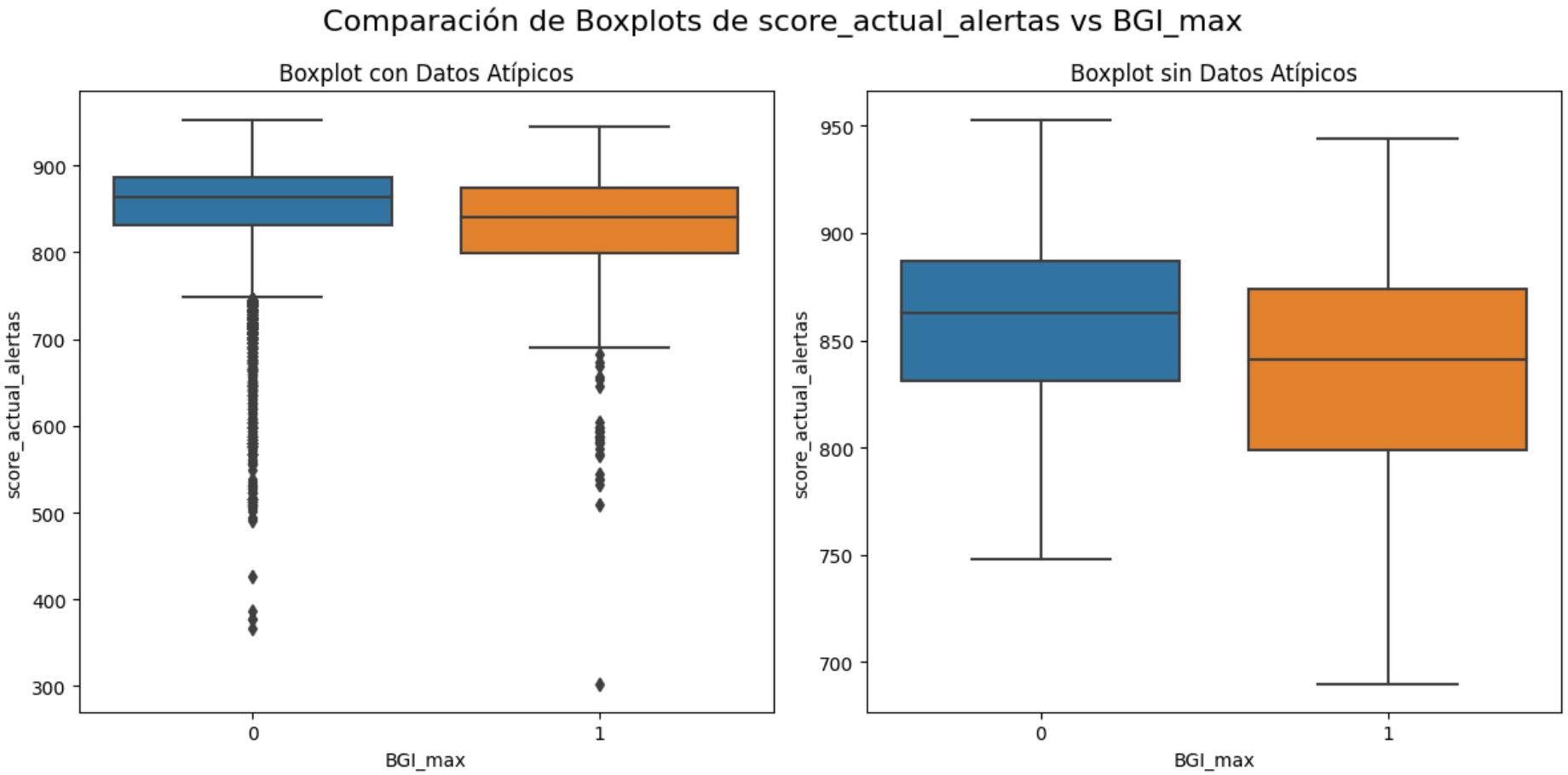
Se observan valores mínimos extremos en columnas como var_T3_T1_num_ent_centralr y plazo_remanente_consumo_avg, donde los valores mínimos son negativos y significativamente alejados de los valores del primer cuartil. Esto también indica la posible presencia de datos atípicos.

Por otro lado, se observa una gran dispersión en varias columnas, como se evidencia por las altas desviaciones estándar en comparación con los rangos intercuartílicos. Esto es particularmente notable en columnas como puntaje_centralr, score_actual_alertas, y lag1_score_actual_alertas, entre otras.

Outliers o Anomalías



Outliers o Anomalías



Outliers o Anomalías

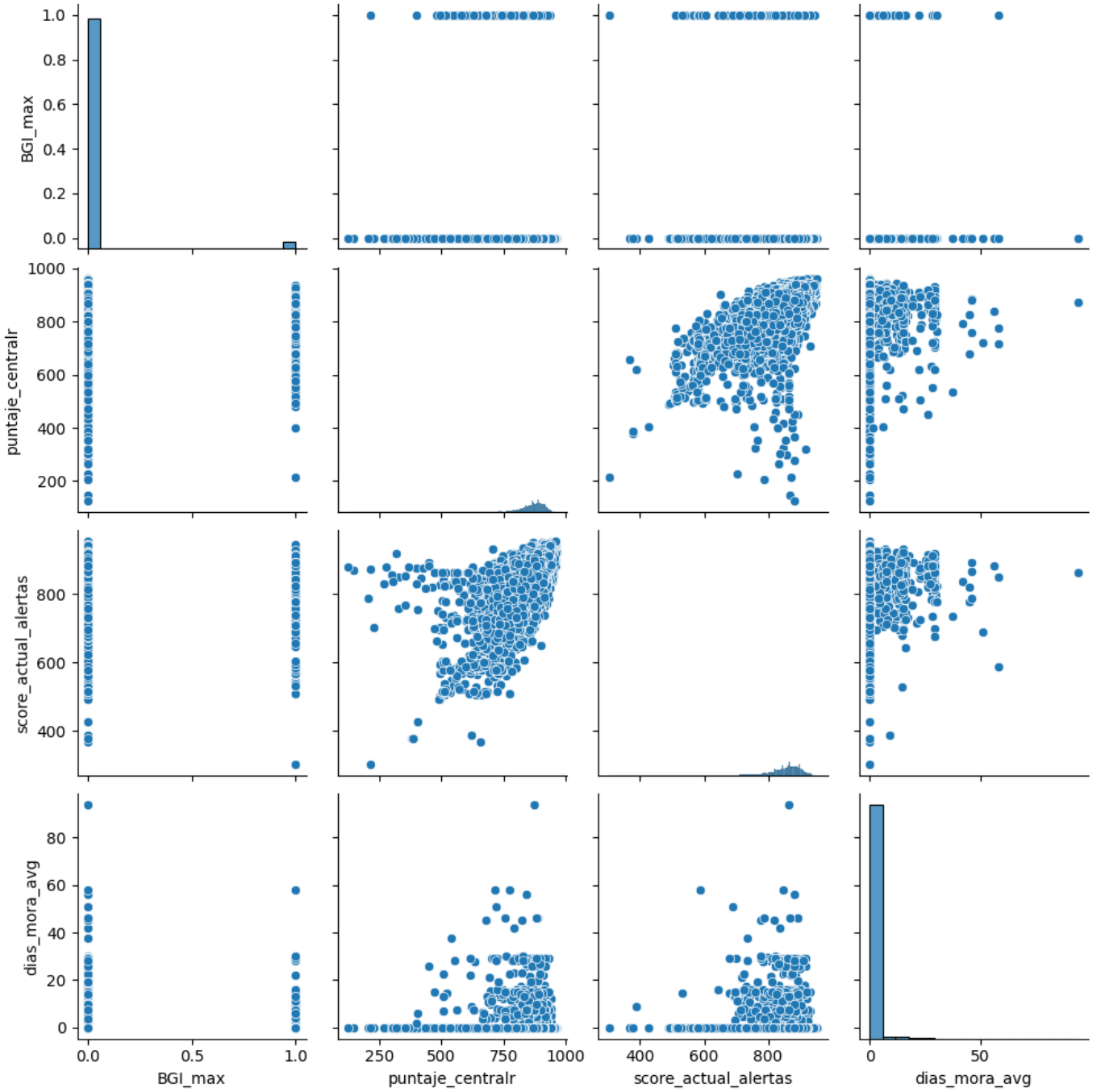


Gráfico de Conteos

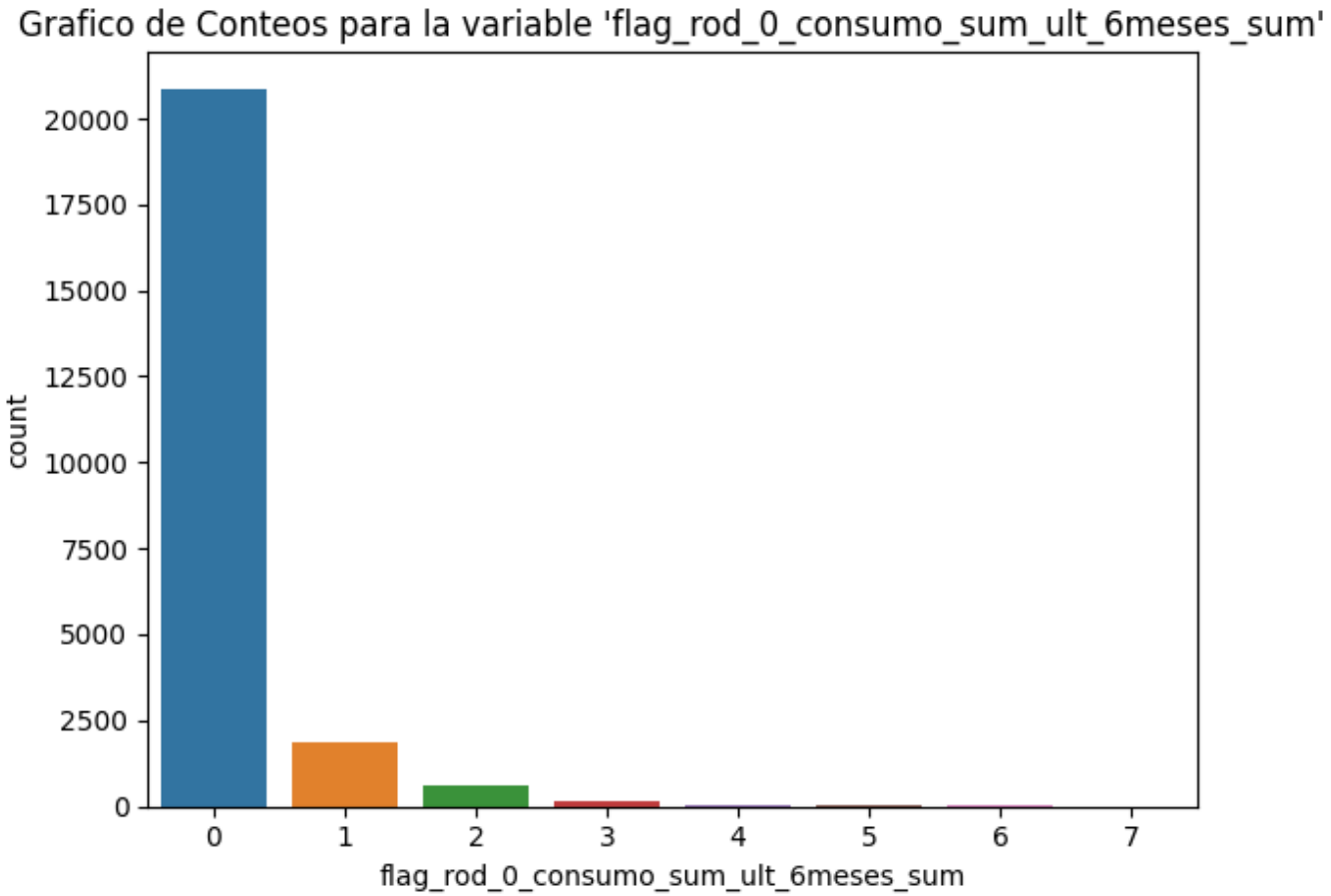
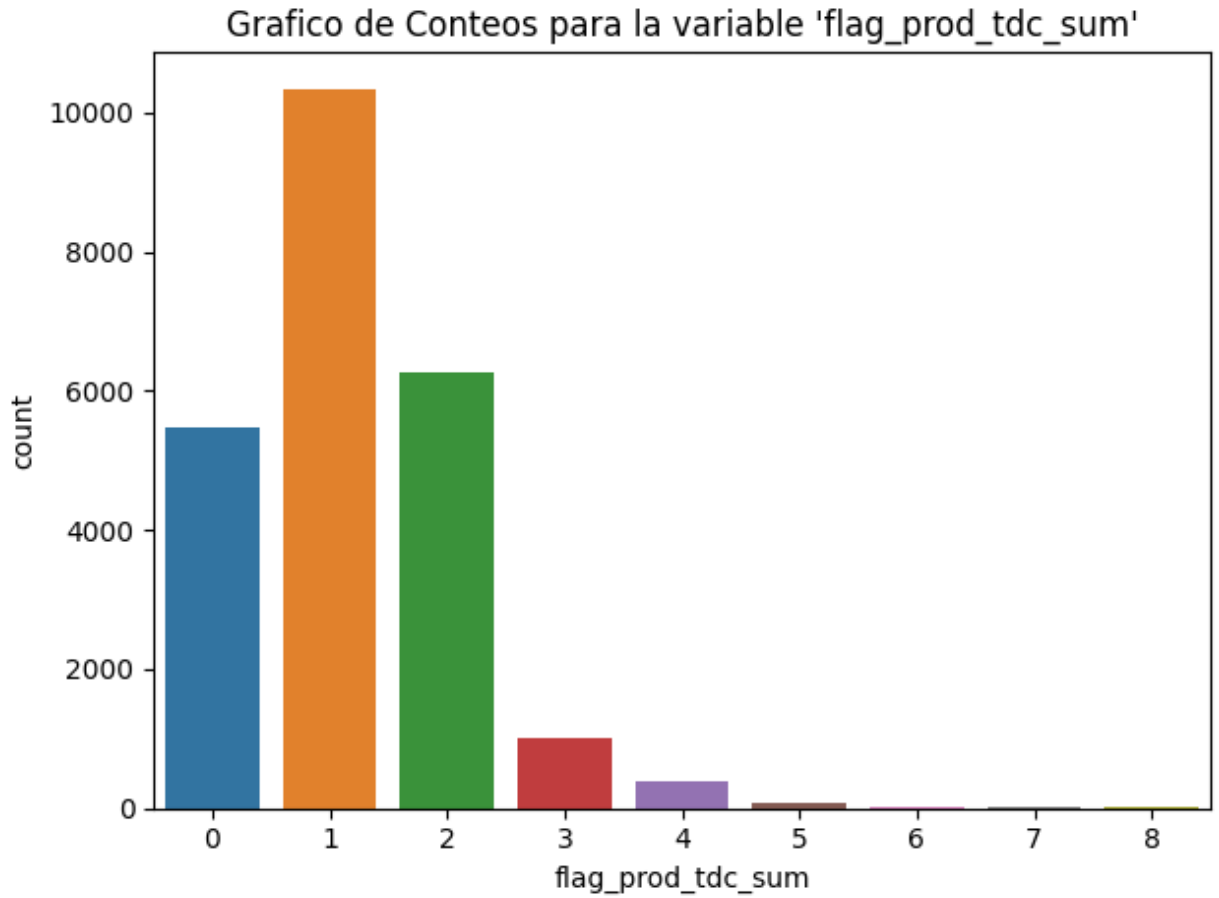
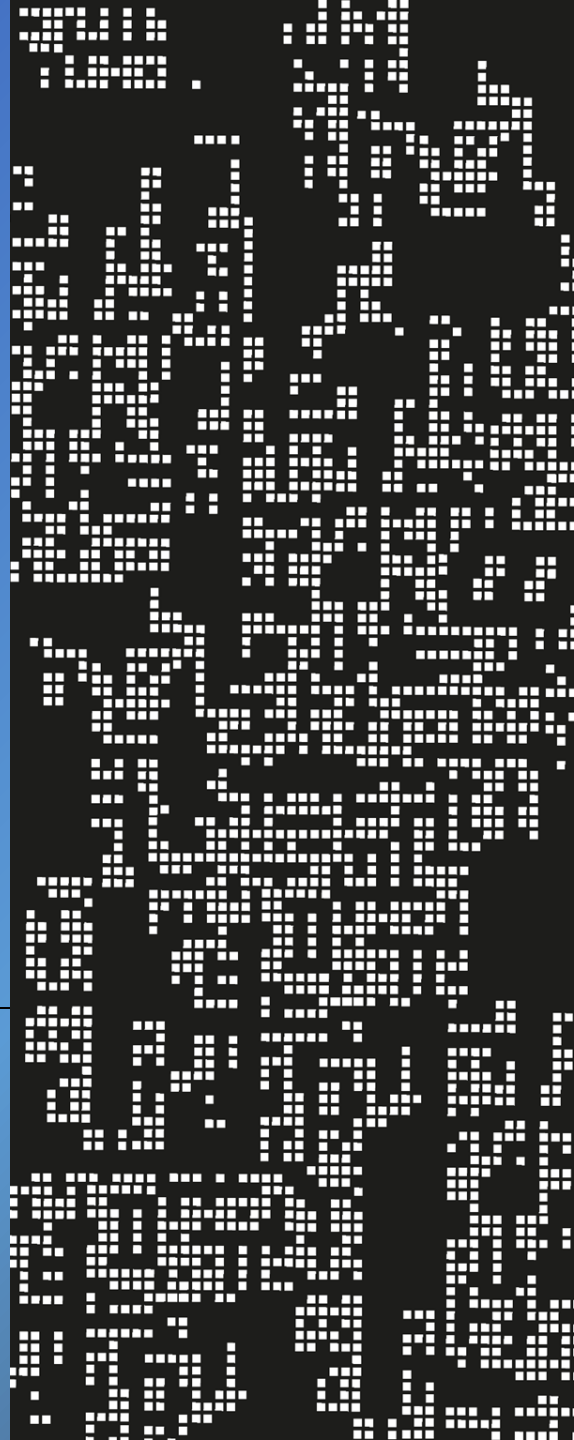


Gráfico de Conteos



04 Modelado



Muestreo para Desarrollo y Pruebas

En nuestro caso, en el que sabemos que nuestros datos no estan balanceados, el muestreo estratificado es crucial para garantizar que los conjuntos de entrenamiento y prueba reflejen adecuadamente la proporción de clientes que incumplen y los que no. Dado que existe un desequilibrio entre estas clases, el muestreo estratificado asegura que ambas clases estén representadas proporcionalmente en ambos conjuntos. Esto es vital para prevenir el sesgo en el modelo, mejorar su capacidad de generalización y asegurar una evaluación precisa de su rendimiento.

```
  # Separar las características y la variable objetivo
X = df_final.drop('BGI_max', axis=1)
y = df_final['BGI_max']

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=123,stratify=y)
```

Proceso de Modelado

En nuestro caso, de clases no balanceadas, se consideraron principalmente modelos de Boosting, esto debido a que son modelos que generalmente tienen mejor desempeño en contexto de modelado en bases no balanceadas. Para cada modelo, se siguen los siguientes pasos:

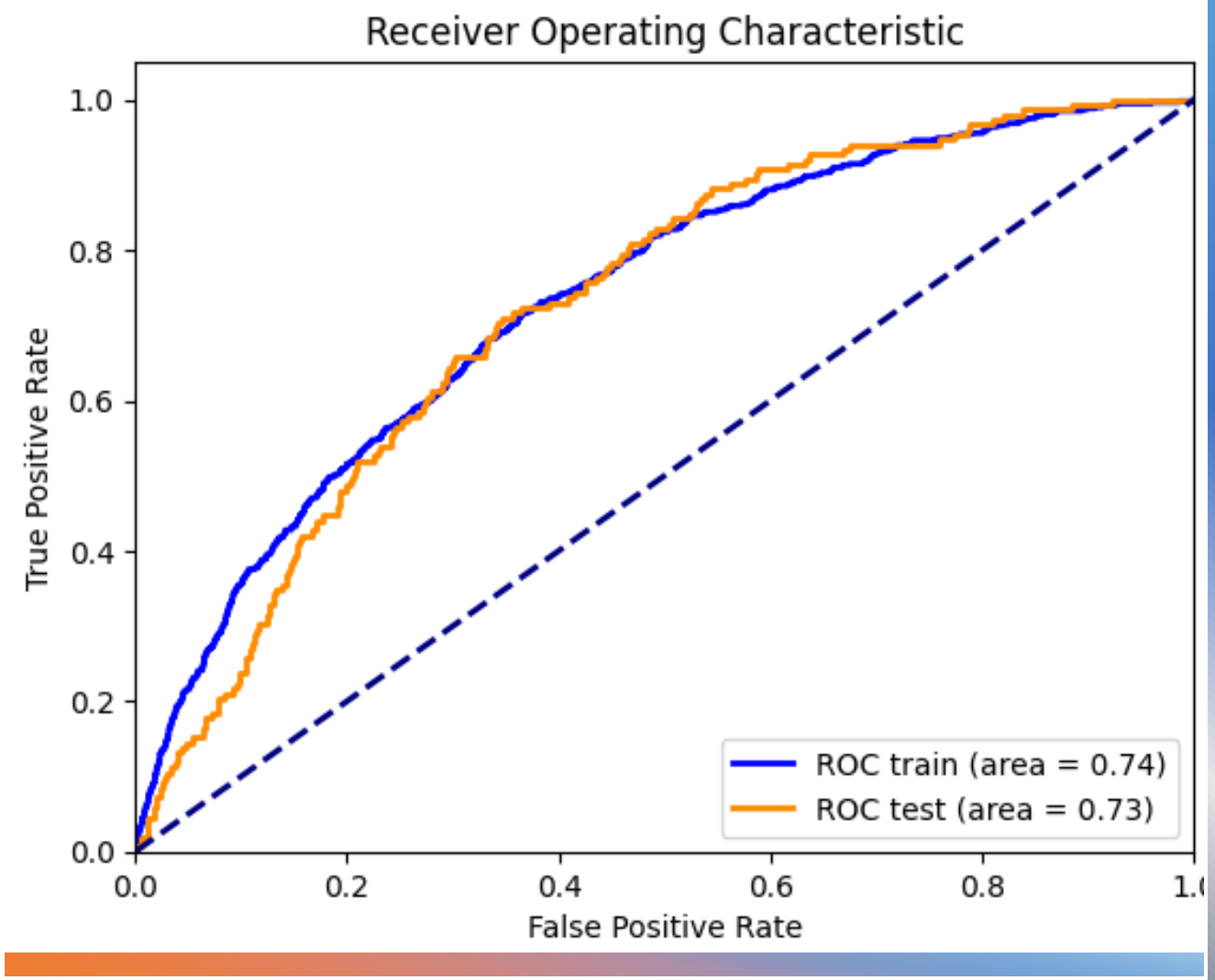
Búsqueda de Hiperparámetros	Utilizar **GridSearchCV** para encontrar los mejores hiperparámetros.
Selección de Características	Aplicar **RFECV** para seleccionar las características más relevantes.
Entrenamiento y Evaluación	Entrenar el modelo con las características seleccionadas y evaluar usando ROC AUC, con enfoque en el desbalanceo de las clases.
Informe de Clasificación	para entender el rendimiento del modelo en términos de precisión, recall y f1-score.

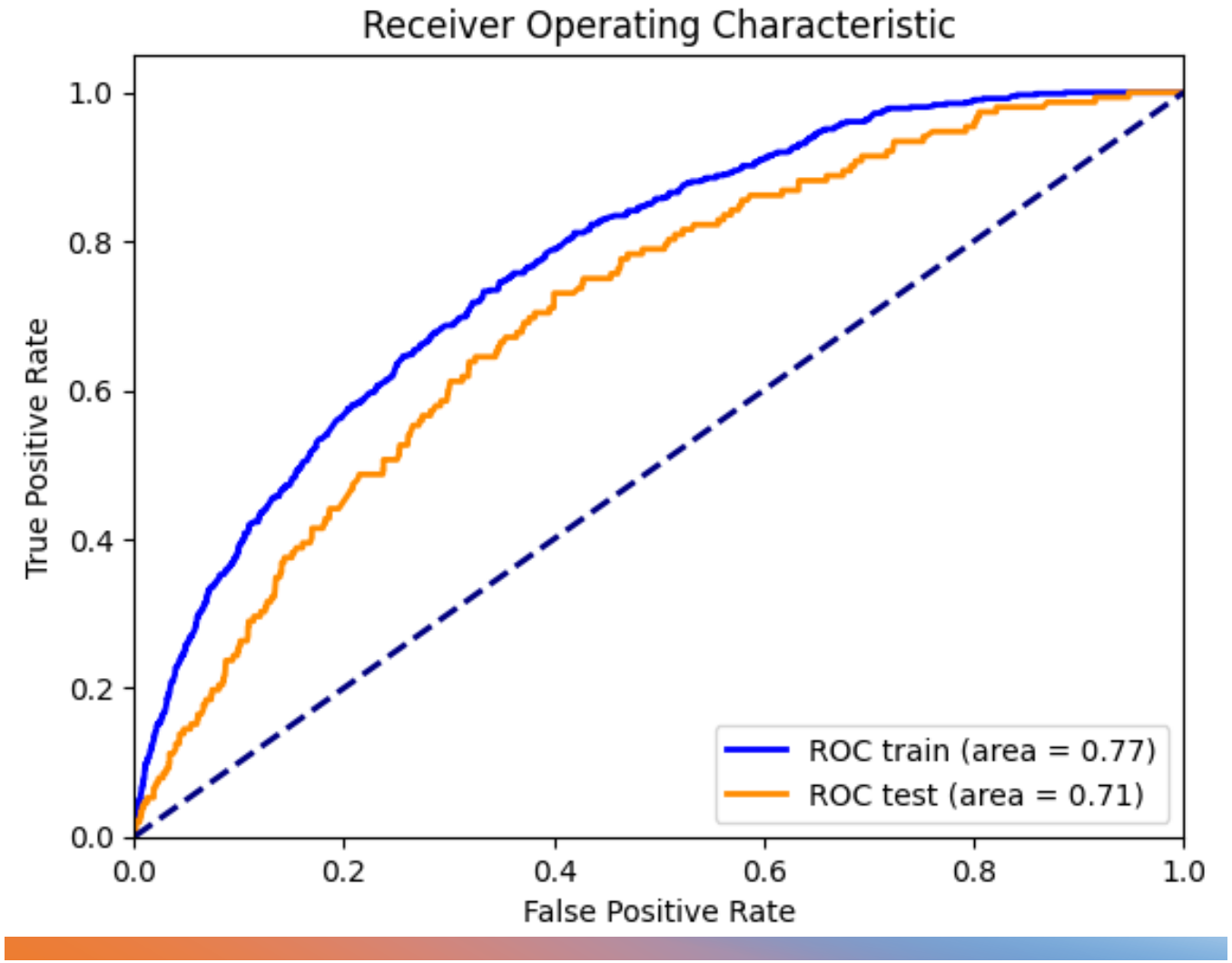
Modelo Logístico

- Se empleó un modelo de regresión logística. Este modelo, adecuado para clasificación binaria, estima la probabilidad de pertenencia a una categoría basándose en los predictores.

- Se realizó una búsqueda de hiperparámetros óptimos mediante GridSearchCV, enfocándose en la regularización (parámetro 'C') y el algoritmo de optimización ('solver'). La regularización ayuda a evitar el sobreajuste, crucial en un contexto de datos desequilibrados.

- Posteriormente, se aplicó RFECV para seleccionar las características más relevantes, optimizando así el modelo. Con las características seleccionadas, el modelo se entrenó y evaluó usando la métrica ROC AUC, que es efectiva en escenarios con clases desbalanceadas.
- mostró un AUC-ROC de 0.735 en el conjunto de entrenamiento y 0.728 en el conjunto de prueba, indicando una capacidad razonable para distinguir entre las clases y una capacidad de generalización aceptable.

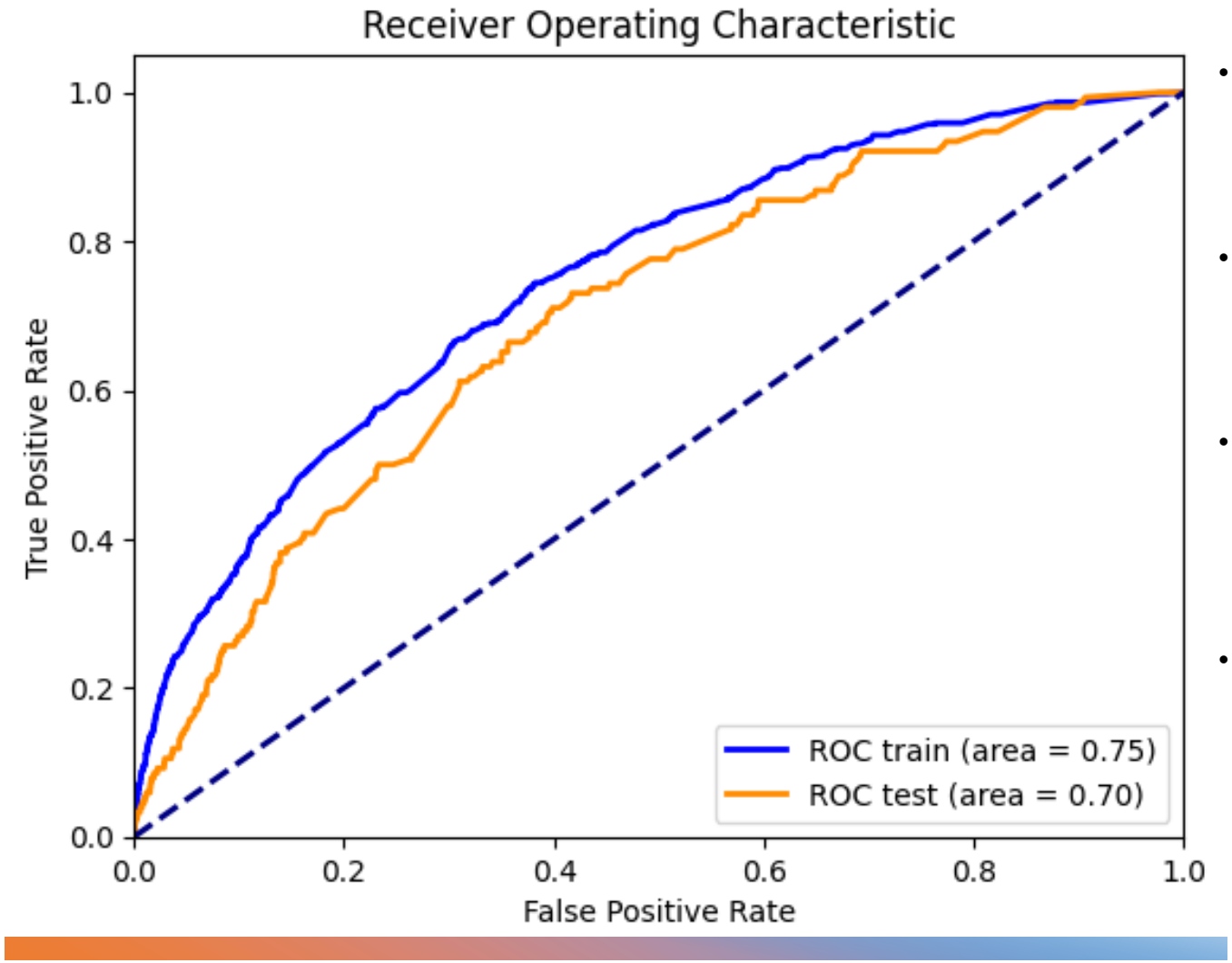




Modelo XGBoost

- XGBoost Classifier se destaca por su eficiencia y precisión. XGBoost, basado en árboles de decisión, mejora su rendimiento a través del 'boosting', corrigiendo errores de árboles anteriores en cada iteración.
- Dado el desequilibrio de clases en los datos, se ajustó el parámetro 'scale_pos_weight' para equilibrar la influencia de las clases en el entrenamiento. Este ajuste es crucial para mejorar la detección de la clase minoritaria, en este caso, los incumplimientos.
- La optimización del modelo comenzó con la selección de hiperparámetros mediante GridSearchCV, enfocándose en 'n_estimators', 'learning_rate' y 'max_depth'. Estos hiperparámetros controlan la complejidad del modelo y previenen el sobreajuste.
- Tras determinar los mejores hiperparámetros, se aplicó RFECV para seleccionar las características más relevantes, mejorando así la eficiencia del modelo. El modelo optimizado se evaluó con la métrica ROC AUC, que es efectiva en contextos de clases desequilibradas.

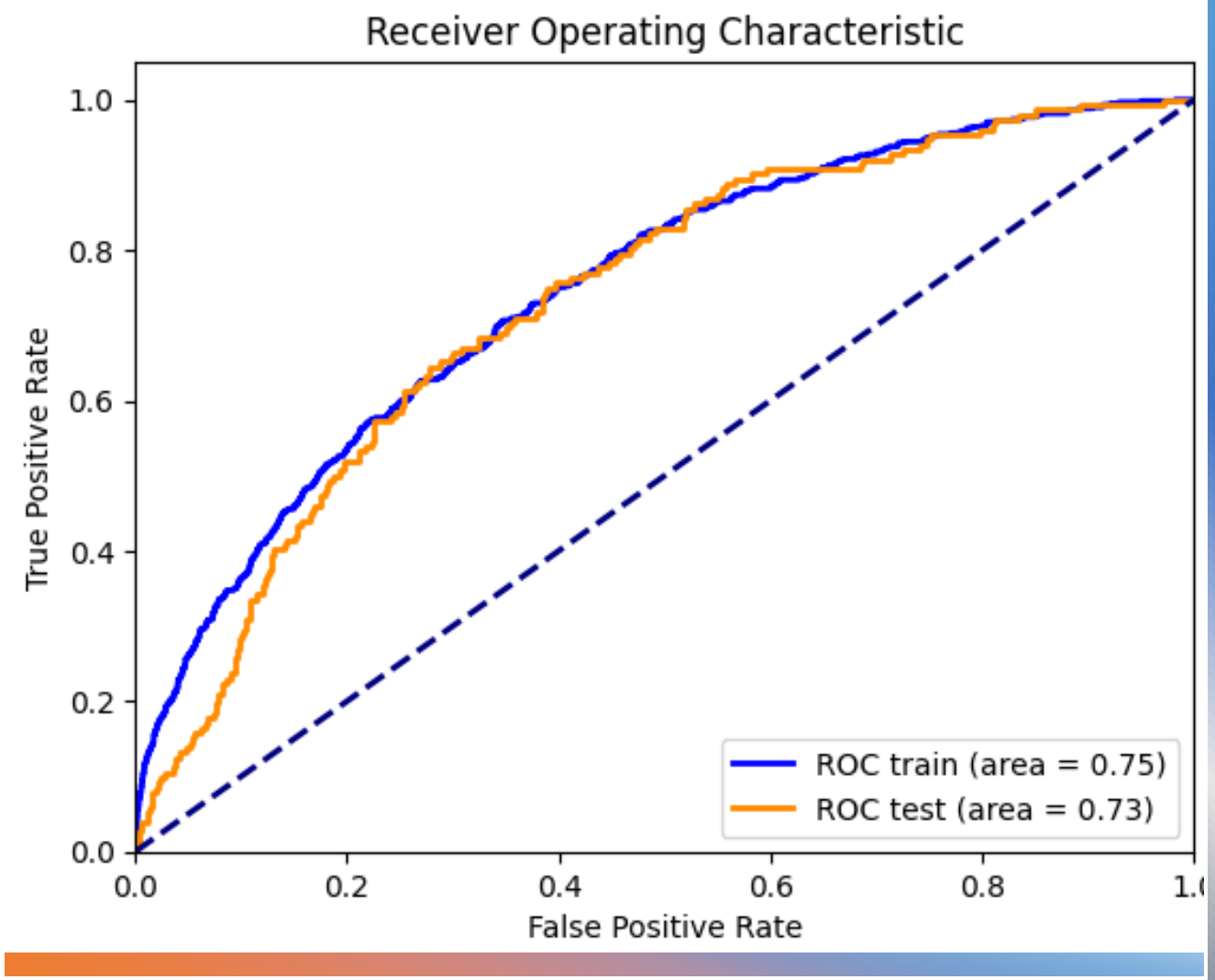
Modelo LightGBM



- LightGBM Classifier, conocido por su rapidez y eficiencia en el manejo de grandes conjuntos de datos. LightGBM utiliza un método de construcción de árboles basado en gradient boosting, optimizando tanto la velocidad como la precisión.
- Para abordar el desequilibrio de clases en los datos, se ajustó el parámetro 'scale_pos_weight', equilibrando así la influencia de las clases minoritarias y mayoritarias durante el entrenamiento. Este ajuste es crucial para mejorar la identificación de casos de incumplimiento.
- La optimización del modelo comenzó con la selección de hiperparámetros a través de GridSearchCV, enfocándose en 'n_estimators', 'learning_rate', 'max_depth' y 'num_leaves'. Estos hiperparámetros son fundamentales para controlar la complejidad del modelo y evitar el sobreajuste.
- Tras determinar los mejores hiperparámetros, se aplicó RFECV para seleccionar las características más influyentes, mejorando la eficiencia del modelo. El modelo optimizado fue evaluado utilizando la métrica ROC AUC, adecuada para contextos con clases desequilibradas, y se generó un informe de clasificación para analizar su rendimiento en términos de precisión, recall y f1-score. Estas métricas son vitales para evaluar la capacidad del modelo para identificar correctamente los casos de incumplimiento.

Modelo CatBoost

- CatBoost Classifier, destacado por su eficacia en el manejo de datos categóricos y su capacidad para manejar el desequilibrio de clases. CatBoost es un algoritmo de boosting basado en árboles de decisión que se enfoca en reducir el sobreajuste y mejorar la precisión del modelo.
- Para abordar el desequilibrio de clases, se calculó el 'scale_pos_weight', equilibrando la influencia de las clases mayoritarias y minoritarias. Este ajuste es crucial para mejorar la detección de casos de incumplimiento.
- La optimización del modelo comenzó con la selección de hiperparámetros mediante GridSearchCV, enfocándose en 'iterations', 'learning_rate' y 'depth'. Estos hiperparámetros son esenciales para controlar la complejidad del modelo, la velocidad de aprendizaje y la profundidad de los árboles, respectivamente.
- Tras determinar los mejores hiperparámetros, se aplicó RFECV para seleccionar las características más relevantes, mejorando así la eficiencia del modelo. Finalmente, el modelo optimizado fue evaluado utilizando la métrica ROC AUC, adecuada para contextos con clases desequilibradas, y se generó un informe de clasificación para analizar su rendimiento en términos de precisión, recall y f1-score. Estas métricas son fundamentales para evaluar la capacidad del modelo para identificar correctamente los casos de incumplimiento



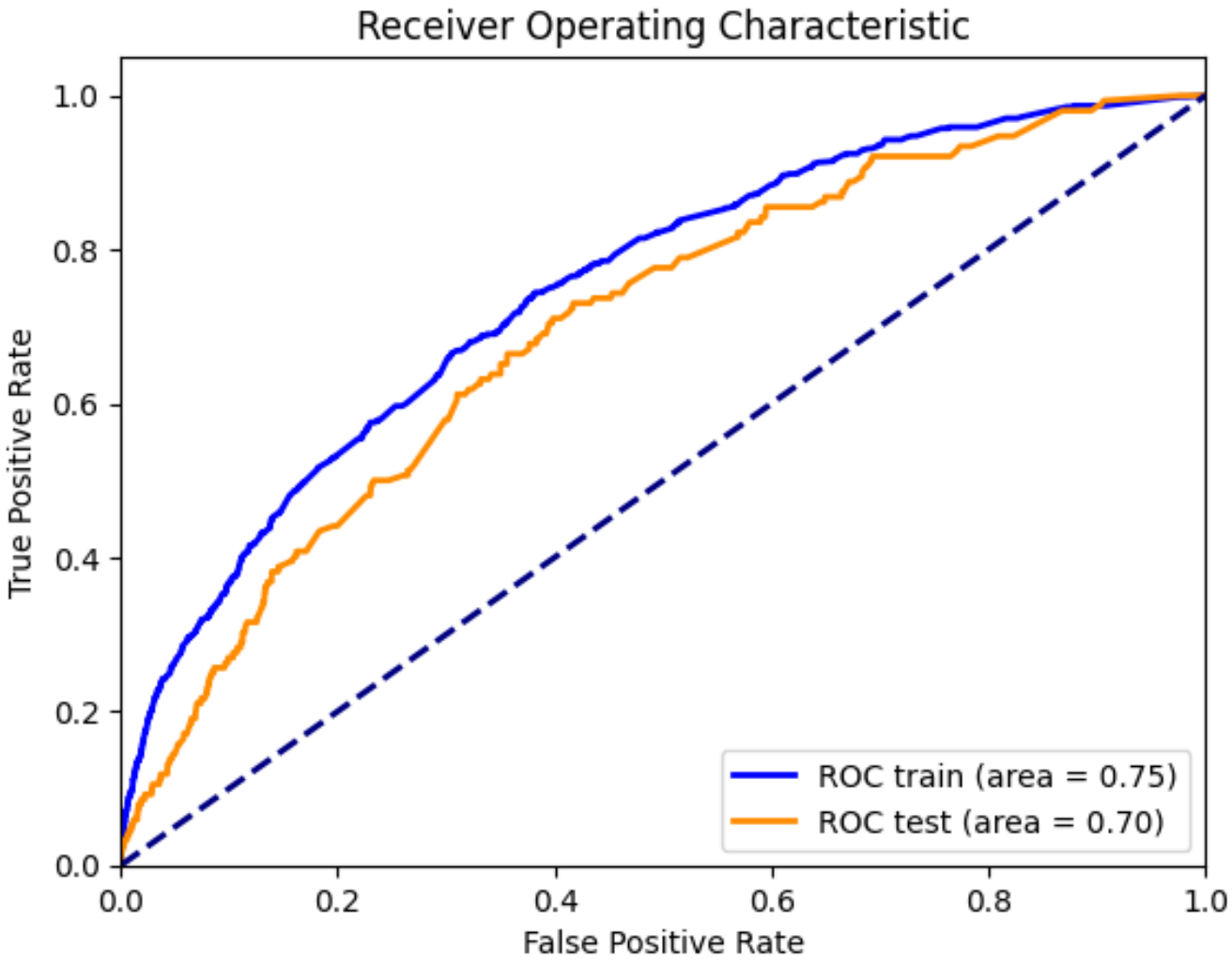
Modelo AdaBoost

- AdaBoost, un algoritmo de boosting, funciona combinando múltiples modelos débiles, en este caso, árboles de decisión, para crear un clasificador fuerte y más preciso.

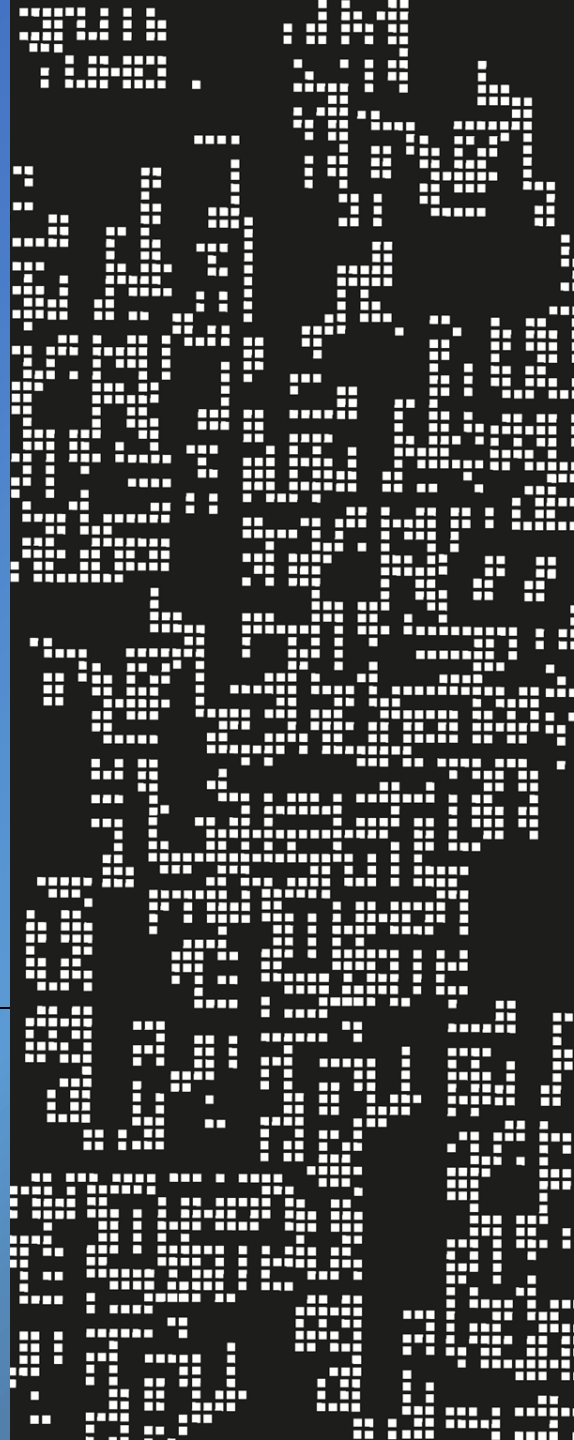
- Para abordar el desequilibrio de clases, se asignaron pesos a las clases, equilibrando así la influencia de las clases mayoritarias y minoritarias. Este ajuste es crucial en contextos con desequilibrio de clases, como en la predicción de incumplimientos.

- La optimización del modelo comenzó con la selección de hiperparámetros a través de GridSearchCV, enfocándose en 'n_estimators' y 'learning_rate'. Estos hiperparámetros son esenciales para controlar el número de modelos débiles en el ensamblaje y la tasa de aprendizaje del algoritmo, respectivamente.

- Tras determinar los mejores hiperparámetros, se aplicó RFECV para seleccionar las características más relevantes, mejorando así la eficiencia del modelo. Finalmente, el modelo optimizado fue evaluado utilizando la métrica ROC AUC, adecuada para contextos con clases desequilibradas, y se generó un informe de clasificación para analizar su rendimiento en términos de precisión, recall y f1-score. Estas métricas son fundamentales para evaluar la capacidad del modelo para identificar correctamente los casos de incumplimiento.



05 Evaluación



Resultados de la Evaluación

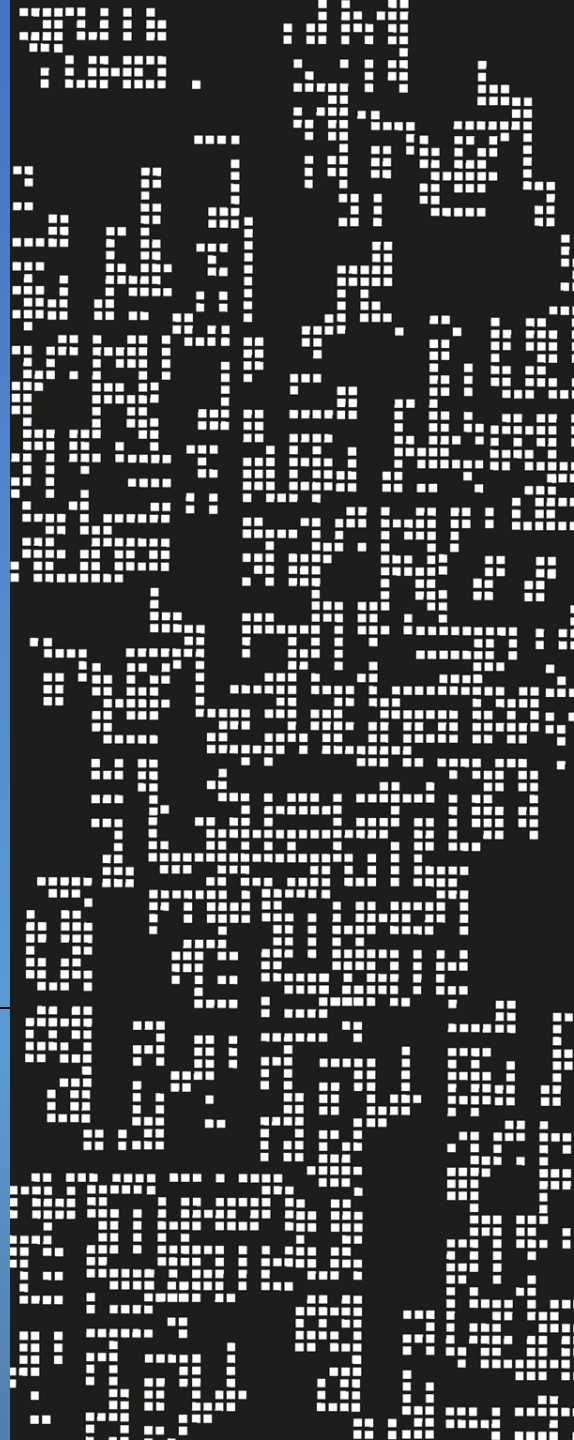
El AUC mide la capacidad de discriminación del modelo. La precisión y el recall son métricas de identificación correcta, útiles cuando los falsos positivos son costosos y no se deben perder positivos reales, respectivamente. El F1 Score equilibra precisión y recall, útil en desequilibrio de clases. El promedio macro calcula métricas por clase y promedia, tratando todas las clases por igual. El promedio ponderado pondera la contribución de cada clase por su tamaño. La precisión general mide la proporción de predicciones correctas, pero puede ser engañosa en conjuntos de datos desequilibrados.

Modelo	AUC Entrenamiento	AUC Prueba	F1 Score Clase 0	F1 Score Clase 1	Precision Clase 0	Precision Clase 1	Recall Clase 0	Recall Clase 1	Accuracy	Macro Avg F1	Weighted Avg F1
Regresión Logística	0.7353	0.7278	0.81	0.12	0.98	0.07	0.69	0.66	0.69	0.47	0.79
XGBoost Classifier	0.7721	0.7067	0.78	0.11	0.98	0.06	0.65	0.66	0.65	0.45	0.76
LightGBM Classifier	0.7480	0.6991	0.98	0.00	0.97	0.00	1.00	0.00	0.97	0.49	0.95
CatBoost Classifier	0.7479	0.7339	0.98	0.00	0.97	0.00	1.00	0.00	0.97	0.49	0.95
AdaBoost Classifier	0.7320	0.7063	0.78	0.11	0.98	0.06	0.64	0.68	0.64	0.44	0.76

Resultados de la Evaluación

1. El Modelo de Regresión Logística muestra un AUC moderadamente alto tanto en entrenamiento como en prueba, indicando una capacidad decente para distinguir entre clases. Aunque su precisión para la clase de incumplimiento es baja, su alto recall sugiere que es capaz de identificar una buena cantidad de incumplimientos reales, aunque a costa de un número significativo de falsos positivos. En nuestro contexto, es preferible errar por exceso en la identificación de posibles incumplimientos, ya que los costos asociados a no detectar un incumplimiento real pueden ser sustanciales.
2. El Modelo XGBoost Classifier, aunque muestra un alto AUC en entrenamiento, su rendimiento disminuye en la prueba, lo que podría indicar sobreajuste. Al igual que la regresión logística, tiene una baja precisión pero un alto recall para la clase de incumplimiento, lo que significa que, aunque identifica muchos incumplimientos reales, también genera muchos falsos positivos. Nuevamente, en nuestro contexto, este enfoque de "mejor prevenir que lamentar" puede ser más beneficioso.
3. El Modelo LightGBM Classifier y el Modelo CatBoost Classifier, a pesar de tener buenos AUC, fallan notablemente en identificar incumplimientos, mostrando una muy baja precisión y recall para esta clase. Esto los hace menos adecuados para nuestro objetivo, ya que es crítico detectar la mayoría de los incumplimientos reales, incluso si esto implica aceptar un mayor número de falsos positivos.
4. El Modelo AdaBoost Classifier presenta un equilibrio moderado en términos de AUC y un alto recall para la clase de incumplimiento, similar a la regresión logística y XGBoost. Aunque su precisión es baja, su capacidad para identificar incumplimientos es valiosa en nuestro contexto, donde el costo de no detectar un incumplimiento real puede ser significativo.
5. El modelo CatBoost Classifier muestra un AUC Consistentemente bueno tanto en entrenamiento como en prueba, pero para el Precisión y Recall para la clase 1, es Muy baja similar a LightGBM. A pesar de un buen AUC, el modelo no logra identificar efectivamente los incumplimientos.

05 Conclusión



Conclusión

Aunque ninguno de los modelos es perfecto, la Regresión Logística, XGBoost y AdaBoost ofrecen un mejor equilibrio entre AUC y recall para la clase de incumplimiento. En un escenario donde es crucial no pasar por alto los incumplimientos, estos modelos son preferibles, a pesar de que esto pueda llevar a un mayor número de falsos positivos. En nuestro contexto, es más prudente equivocarse por exceso en la identificación de posibles incumplimientos que no detectarlos.