Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Magíster en Ciencia de Datos

Proyecto BCI

Laboratorio de Programación Científica para Ciencia de Datos

Integrantes: Junwei He Mai
Sofia Capibara Chávez Bastidas
Profesor: Ignacio Meza De la Jara
Sebastián Tinoco

Fecha de entrega: 12 de Diciembre de 2024
Santiago de Chile

1. Introducción

El problema planteado busca predecir la probabilidad de morosidad de los clientes del Banco de Comisiones Infinitas (BCI). Específicamente, se intenta identificar con anticipación a aquellos clientes que podrían incumplir con sus pagos, permitiendo al banco tomar decisiones informadas para disminuir el impacto de los clientes que no pagarán los prestamos. Sin embargo, el modelo no solo debe enfocarse en evitar los impagos, sino también en identificar clientes que puedan cumplir con los pagos, maximizando los ingresos por tasas de interés sin poner en riesgo la estabilidad financiera del banco. Por lo tanto, el desafío radica en desarrollar un modelo predictivo preciso e interpretable que equilibre la reducción del riesgo crediticio con la maximización de la rentabilidad.

En este proyecto se trabajarán con la misma base pero en diferente temporalidad, es decir, se tienen las mismas variables pero distintas filas. Dentro de las variables se encuentran datos temporales como fechas de transacciones, cantidad de dinero depositado y retirado, factores de riesgo del mercado e indicadores de mercado.

Los modelos se centrarán en las métricas de F1-Score (combinando precision y recall) ya que se está trabajando con clases desbalanceadas. Una condición de los modelos es que las métricas para ambas clases sean aceptables, pues el banco a pesar de querer predecir los morosos, también valora a los potenciales clientes.

Dentro de los modelos que se usarán estan XGBoost y LightGBM. Estos se ajustan al problema planteado de predicción binaria. Durante el pre procesamiento se tomaron en cuenta varios factores como la estandarización de los datos, reemplazos de outliers y la eliminación de columnas que aportan poca información.

Los resultados de los diversos modelos fueron satisfactorios a nivel local, sin embargo, al ser subidos a la plataforma del concurso, se observó un deterioro en la calidad de las métricas. A pesar de este inconveniente, se lograron F1-Score superiores a 0.7 para ambas clases en el entorno local y un AUC-PR de 0.85 en la fase 2, lo que permitió posicionarse en el tercer lugar de la competencia.

2. Preprocesamiento

2.1. Análisis Exploratorio de Datos

2.1.1. Separación temporal de las variables

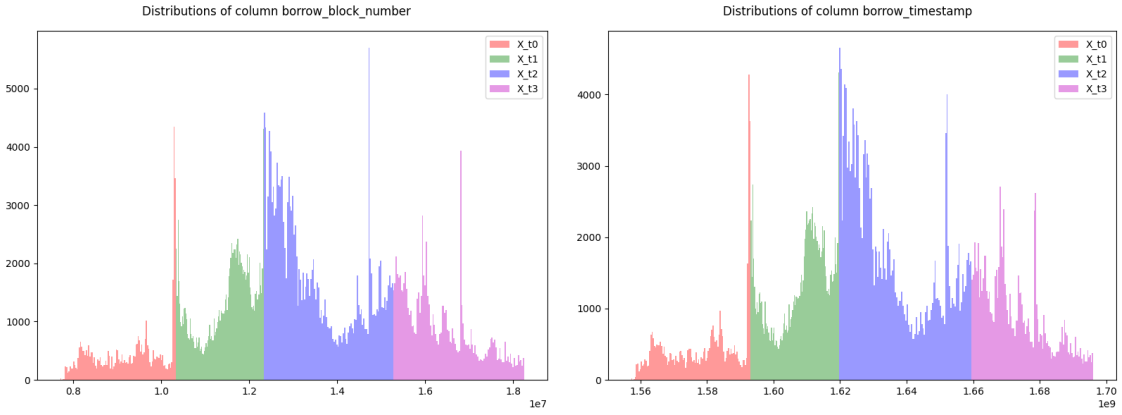


Figura 1: Distribuciones de `Borrow_block_number` y `borrow_timestamp`

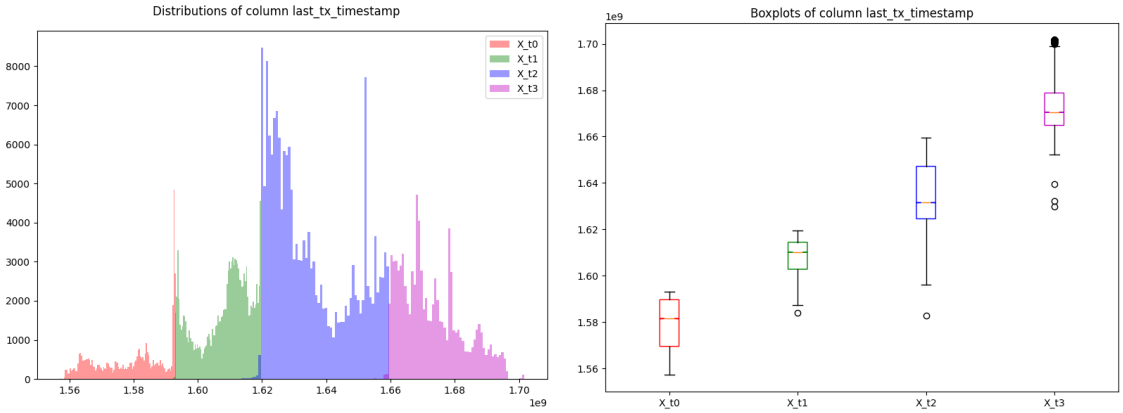


Figura 2: Distribución de `last_tx_timestamp`

Se observa que los datos de las iteraciones corresponden a distintos períodos sucesivos, con muy pocos solapamientos en `last_tx_timestramp_diff`. El pipeline de entrenamiento debería eliminar estas columnas si se busca un modelo independiente del tiempo y conservarla en caso de que la temporalidad sea importante.

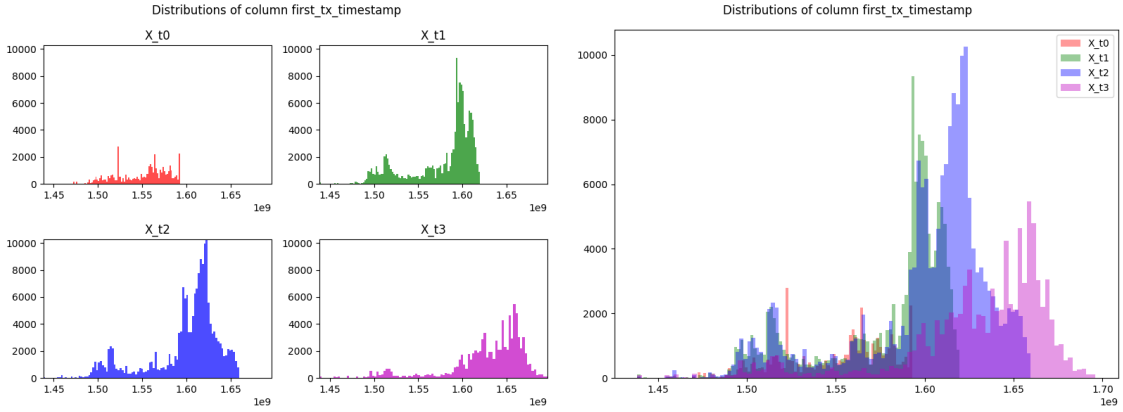


Figura 3: Distribución de `first_tx_timestamp`

Las distribuciones se solapan y se desplazan a medida que las iteraciones avanzan, lo que tiene sentido considerando que datos más recientes incluyen cuentas que comenzaron a transaccionar en periodos nuevos, así como también antiguos.

2.1.2. Variables con escala logarítmica

Existen muchas variables que se concentran fuertemente en valores iniciales y su escala se extiende incluso entre órdenes de magnitud, por ejemplo la variable `risky_sum_outgoing_amount_eth`:

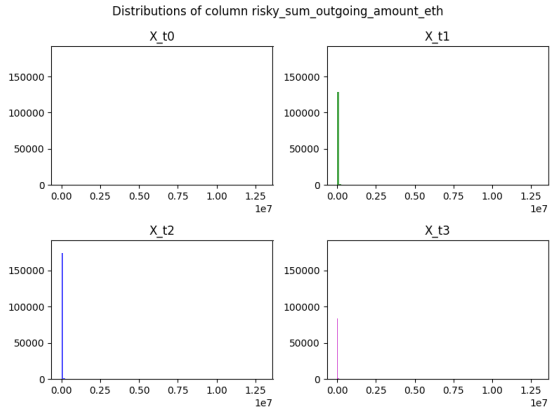


Figura 4: Distribución de `risky_sum_outgoing_amount_eth`

Si bien una posibilidad es quitar los valores extremos considerándolos outliers, incluso filtrando el 5 % de los datos inferiores y superiores, la variable sigue fuertemente concentrada y con valores extremos muy altos. Por lo tanto, es mejor alternativa realiza una transformación utilizando escala logarítmica, como se aprecia en la siguiente figura:

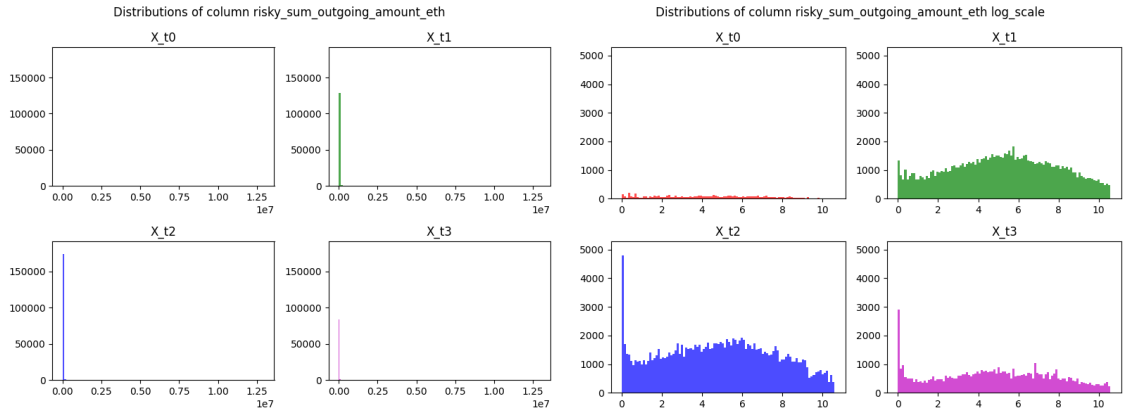


Figura 5: Distribución de risky_sum_outgoing_amount_eth

Variables con esta forma de distribución son:

- wallet_age
- incoming_tx_count
- outgoing_tx_count
- total_gas_paid_eth
- risky_tx_count
- risky_unique_contract_count
- outgoing_tx_sum_eth
- incoming_tx_sum_eth
- outgoing_tx_avg_eth
- risky_last_tx_timestramp
- incoming_tx_avg_eth
- max_eth_ever
- risk_factor
- total_collateral_eth
- total_avg_collateral_eth
- total_available_borrows_eth
- total_available_borrows_avg_eth
- risk_factor_above_threshold_daily_count

- risk_factor_above_threshold_daily_count
- avg_risk_factor
- borrow_amount_avg_eth
- borrow_count
- repay_amount_sum_eth
- repay_amount_avg_eth
- repay_count
- borrow_repay_diff_eth
- deposit_count
- deposit_amount_sum_eth
- liquidation_amount_sum_eth
- time_since_first_deposit
- withdraw_amount_sum_eth

Sus visualizaciones están en el anexo, en general tienen forma:

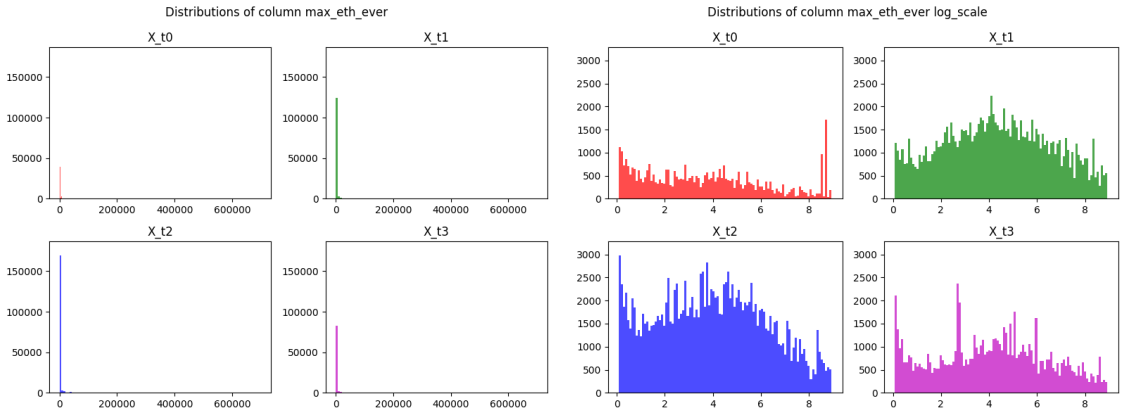


Figura 6: Distribución de max_eth_ever

2.1.3. Variables temporales

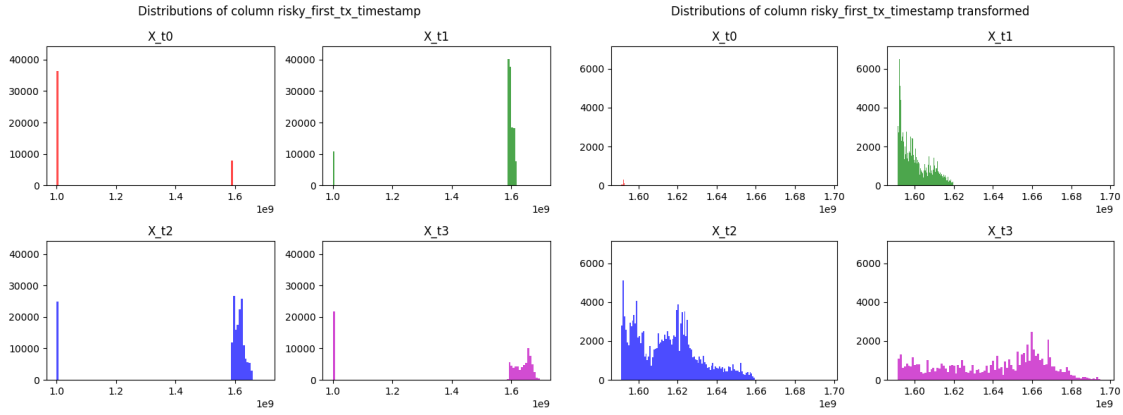


Figura 7: Distribución de `risky_first_tx_timestramp`

La mayoría de las unidades temporales en el conjunto de datos están en formato Unix Timestamp, cuyos valores oscilan entre `999999999` y `1619643423`, dependiendo del dataset. Al convertir estos valores a fechas, corresponden a un rango entre el 09/11/2001 y el 28/04/2021. Sin embargo, la primera fecha no es consistente con el contexto del análisis, ya que se está trabajando exclusivamente con datasets a partir del año 2020. Además, si bien los datos con esta fecha constituyen una fracción relevante del total, su valor se aleja completamente del resto en la distribución, es evidente que corresponden a un error.

Es por esto que, para la variable `risky_first_tx_timestamp`, se analiza la presencia de valores iguales a `999999999`. Para poder categorizar este error, se genera una nueva variable binaria denominada `has_risky_first_tx`, que indica si la transacción tiene o no una fecha anómala. Posteriormente, los valores `999999999` en la columna `risky_first_tx_timestamp` se reemplazan por `NaN`.

Con ello, el resultado es:

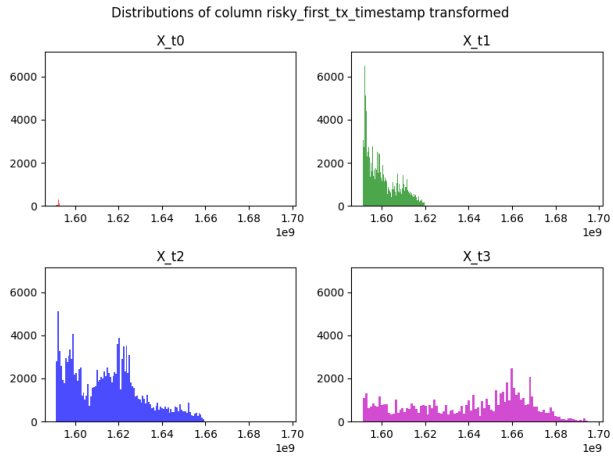


Figura 8: `risky_first_tx_timestamp`

Se realiza un proceso análogo para las variables `risky_last_tx_timestamp` y `time_since_last_liquidated`.

2.1.4. Variables con outliers

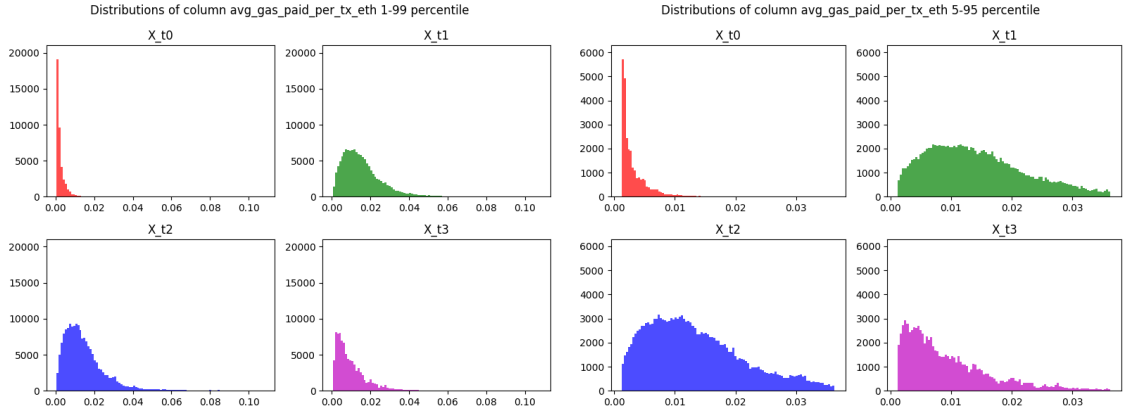


Figura 9: Distribución de `avg_gas_paid_eth`

En este caso, filtrando outliers se obtiene un resultado aceptable, el único caso problemático corresponde a los datos de a `X_t0`. Aplicar log scale directamente no lo soluciona debido que los valores son de escala muy pequeña, por otro lado, transformaciones de forma

$$\log(\texttt{avg\_gas\_paid\_eth}/\max(\texttt{avg\_gas\_paid\_eth}))$$

podrían alterar las distribuciones de `avg_gas_paid_eth` para los otros datasets.

Otras variables son más sencillas y simplemente filtrando outliers se obtiene un resultado aceptable:

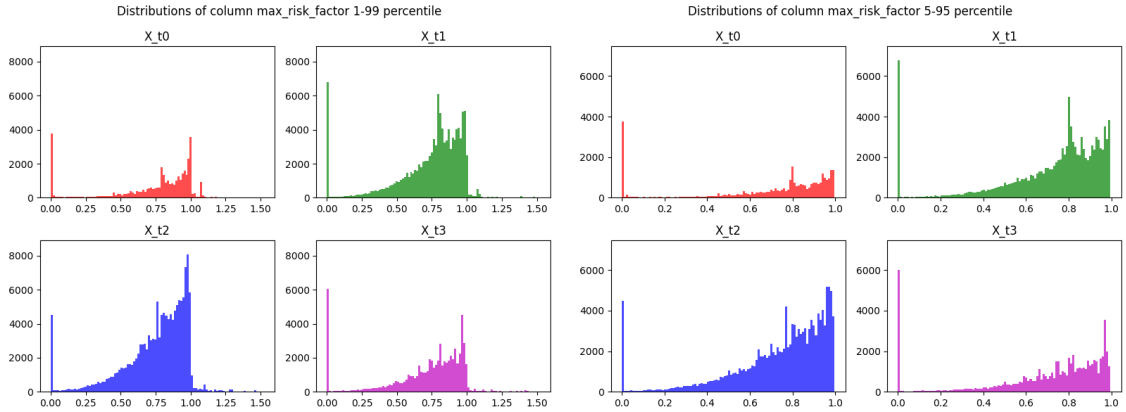


Figura 10: Distribución de `max_risk_factor`

Estas variables son:

- liquidation_count
- market_adx
- market_adxr
- market_correl
- market_linearreg_slope
- market_macd_macdext
- market_macd_macdfix
- market_macdsignal_macdfix
- market_macdsignal
- market_natr
- market_plus_dm
- market_ppo

2.1.5. Otras variables

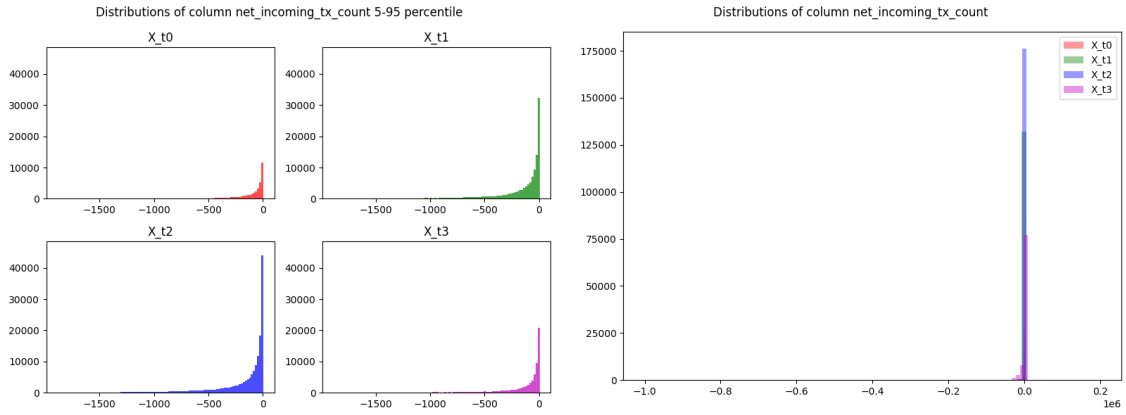


Figura 11: Distribución de net_incomoing_tx_count

El caso de net_incomoing_tx_count, por varios motivos. No es completamente negativa, por lo que una transformación de tipo $log(1 - \texttt{net\_incomoing\_tx\_count})$ no conduce a buenos resultados. Además su valor máximo es 97831 en X_t0, 195555 en X_t1, 196374 en X_t2 y 39966 en X_t3, por lo que es complejo saber cuál es un threshold para una transformación de tipo $log(1 - \texttt{net\_incomoing\_tx\_count} - \texttt{threshold})$ que permita aplicar una escala logarítmica.

Además, su distribución no se asemeja a un normal como para utilizar un z-score y utilizar otro métodos como robust scaler o filtrar un % de outliers no solucionan el problema de su concentración.

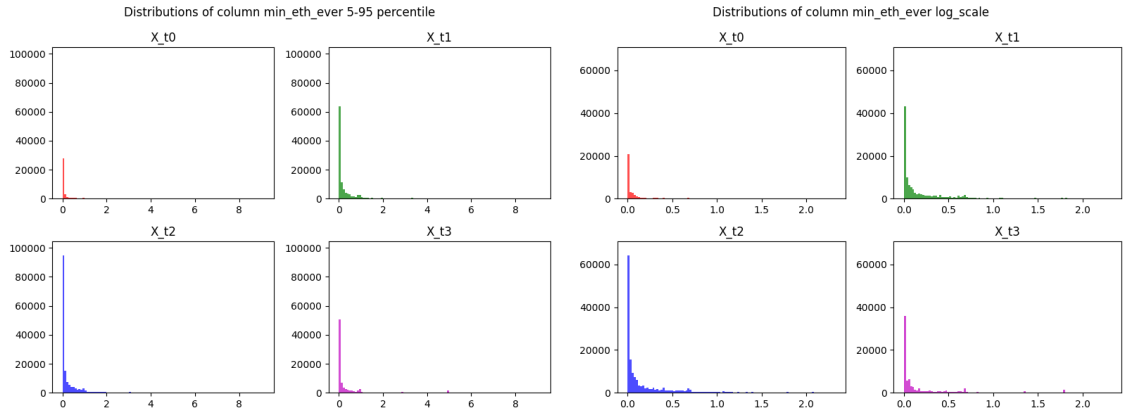


Figura 12: Distribución de `min_eth_ever`

Por otro lado, existen variables tan concentradas que cortando el 5 % de outliers o aplicando transformaciones logarítmicas no se logra reducir su grado de concentración.

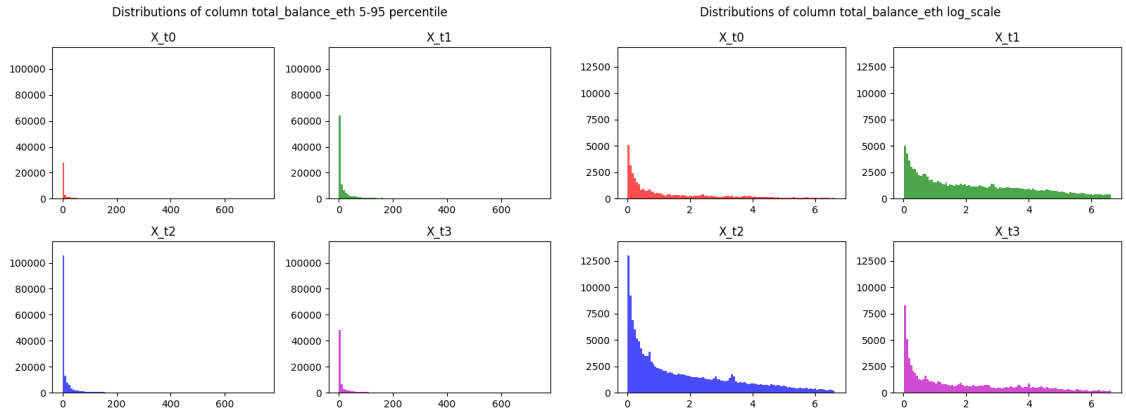


Figura 13: Distribución de `total_balance_eth`

Sin embargo, exsten casos en que, si bien se mantiene una alta concentración. la transformación logarítmica los mejora, como en la figura anterior.

2.1.6. Variables sin transformación

Existen varibales que no presentan outliers y están en una escala razonable (considerando después utilizar un MinMaxScaler). Por ejemplo:

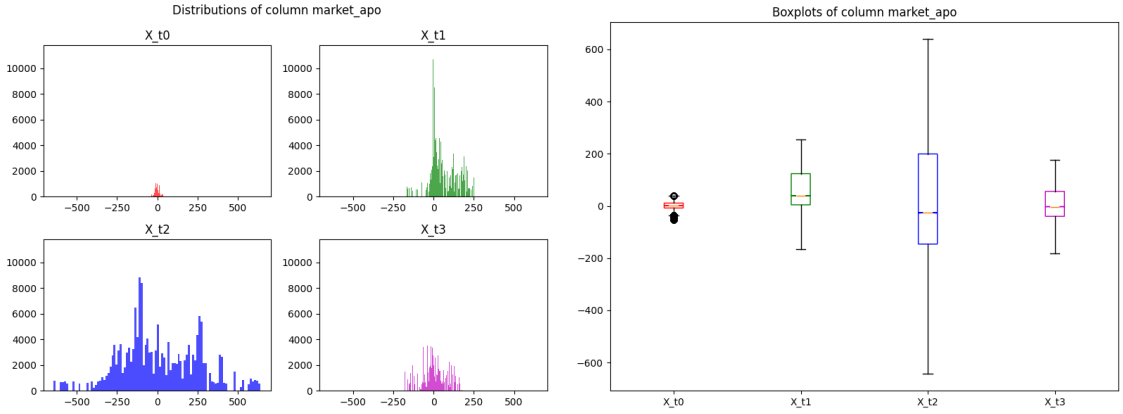


Figura 14: Distribución de `market_apo`

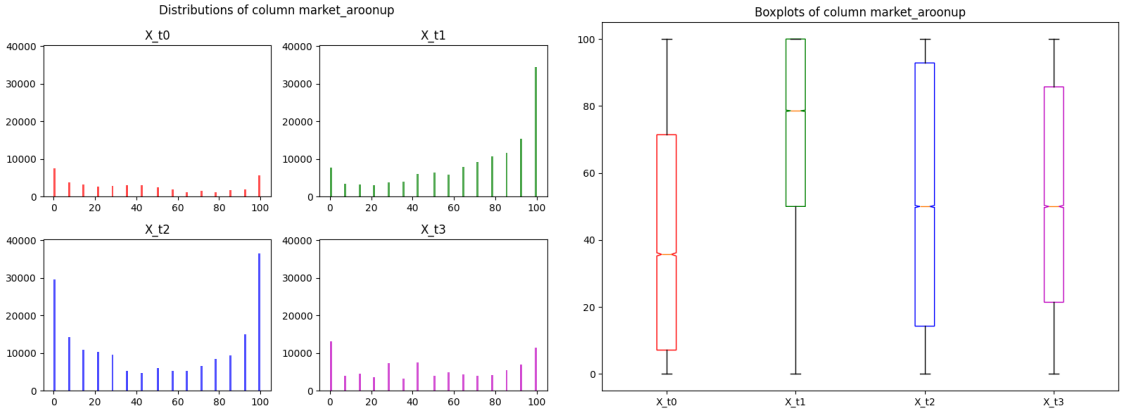


Figura 15: Distribución de `market_aroonup`

Estas variables son:

- `market_aroonosc`
- `market_atr`
- `market_cci`
- `market_cmo`
- `market_dx`
- `market_fask`
- `market_macd`
- `market_macdsignal_macdext`
- `market_max_drawdown_365d`

- market_plus_di
- market_rocp
- market_rocr
- unique_borrow_protocol_count
- unique_lending_protocol_count

2.1.7. Variables categóricas

Existen variables categóricas que no requieren mayores transformaciones dada su baja cantidad de clases, por ejemplo:

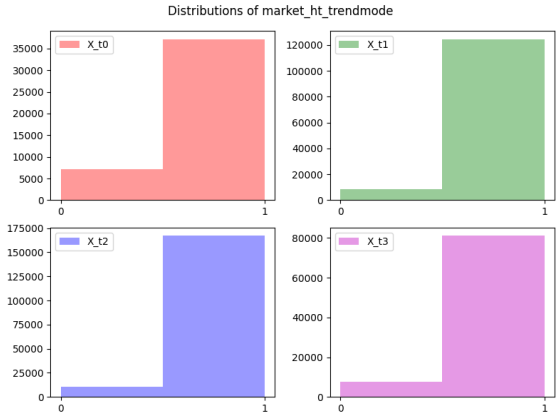


Figura 16: Distribución de market_ht_trendmode

2.2. Preprocesamiento de Datos

2.2.1. Transformaciones de columnas

Se implementaron ColumnTransformers para realizar las transformaciones descritas en la sección anterior. En particular, los valores NaN resultantes de la transformación de variables temporales después se reemplazaron por 0.

2.2.2. Outliers

Para mitigar el impacto de los outliers en el análisis, se utilizó la técnica de winsorización, que ajusta los valores atípicos a los límites correspondientes al percentil 1 y 99, preservando así la estructura de los datos mientras se reducen los efectos distorsionadores de estos valores extremos. Esto garantiza que las variables mantengan un rango representativo y robusto para el análisis posterior.

2.2.3. Reducción de dimensionalidad

Se utilizaron dos métodos para reducir la dimensionalidad:

1. **Varianza:**

   En el proceso de reducción de dimensionalidad, se eliminaron aquellas variables cuya varianza era inferior a 0.05. Esto se debe a que estas variables, al tener valores muy similares entre sí, no aportan información relevante ni diferenciadora para los futuros modelos predictivos. La baja varianza limita su capacidad para explicar patrones o contribuir significativamente al desempeño del modelo. Esta eliminación permite simplificar el conjunto de datos y enfocar el análisis en las variables que realmente aportan valor al modelo.

2. **Variables con alta correlación:**

   Se eliminaron las variables que presentaban una correlación mayor a 0.95 con otras variables del conjunto de datos. Esta decisión se tomó para reducir la multicolinealidad, ya que una alta correlación indica que dichas variables contienen información redundante. Al eliminar las variables altamente correlacionadas, se garantiza que el modelo sea más eficiente y robusto, conservando únicamente las variables que aportan información única y relevante.

2.2.4. Estandarización

Se implementó MinMaxScaler para preprocesar las variables numéricas resultantes de los pasos anteriores, como ingresos, montos de deuda y saldos. Esta técnica escala los datos restando la mediana y dividiendo por el rango intercuartílico (IQR), lo que garantiza que los valores extremos no dominen el análisis. Esto resulta especialmente útil para modelos como XGBoost, ya que mejora la precisión del modelo al mitigar el impacto de los outliers, asegurando una escala uniforme entre las características.

3. Modelamiento

3.1. Baseline

Como baseline se usó un modelo de XGBoost simple, sin hiperparámetros ni escalamiento de datos. Usando los datos de X-t0 Y-t0 se llegaron a las siguientes métricas:

Se eliminaron las variables con una varianza menor a 0.1 y las variables con una correlación mayor a 0.8.

Tabla 1: Reporte de Clasificación de la Baseline

Clase	Precisión	Recall	F1-Score	Soporte
0	0.84	0.92	0.88	6456
1	0.91	0.83	0.87	6833
Accuracy		0.87		
Promedio macro	0.88	0.87	0.87	13289
Promedio ponderado	0.88	0.87	0.87	13289

3.2. Modelos de Machine Learning

Se implementaron tres modelos distintos de aprendizaje automático con el objetivo de evaluar su desempeño en relación con el problema planteado. Cada modelo presenta ventajas y desventajas específicas, por lo que se realizó un análisis comparativo para determinar cuál de ellos se adapta mejor a las características y requerimientos del problema.

- **Regresión logística:**
  - Ideal para problemas lineales y bien definidos.
  - Útil como línea base para comparar con modelos más complejos.
- **Árbol de decisión:**
  - Fácil de interpretar y entender.
  - Captura relaciones no lineales en los datos.
  - Sensible a datos desequilibrados sin ajustes.
- **XGBoost:**
  - Modelo avanzado que combina múltiples árboles para mejorar precisión y reducir sobreajuste.
  - Ideal para datos grandes y complejos, con capacidad de manejar interacciones no lineales y datos desbalanceados.

3.2.1. XGBoost

Con todo el preprocesamiento anterior se hizo un modelo XGBoost con una proporción 70/30, para un problema 'binary logistic' y función de pérdida logloss. Por otro lado, al trabajar con datos desbalanceados se usa "scale pos weight"para compensar el desbalanceo. Finalmente se busca el mejor threshold de clasificación.

AUC-PR: 0.9140

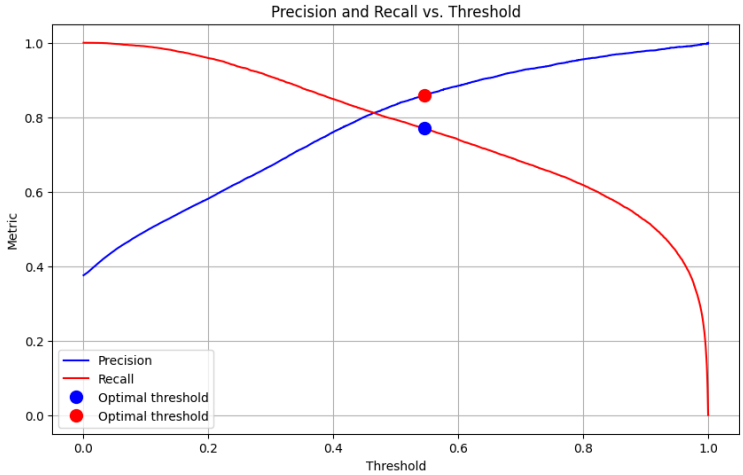


Figura 17: Threshold vs Recall y Precision.

Tabla 2: Resumen de las métricas de desempeño XGBoost.

Clase	**Precisión**	**Recall**	**F1-Score**	**Soporte**
0	0.87	0.92	0.90	24892
1	0.86	0.77	0.81	14975
Promedio Macro	0.86	0.85	0.85	39867
Promedio Ponderado	0.87	0.87	0.86	39867

3.2.2. Regresión Logística

Se realizó un modelo de Regresión Logística entrenando la base con una proporción 30/70 y usando transformaciones como SimpleImputer (transformando NAs en mean) y StandardScaler (Para normalizar).

Tabla 3: Classification Report Regresión Logística

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.76	0.85	0.80	7799
1	0.74	0.61	0.67	5490
Macro avg	0.75	0.73	0.74	13289
Weighted avg	0.75	0.75	0.75	13289

AUC-PR: 0.9140

3.2.3. Decision Tree

Para Decision Tree se entrena el modelo con una proporción 30/70, usando las mismas transformaciones que el modelo anterior. Se agregan los hiperparámetros:

- **criterion:** 'gini'
- **max_depth:** 5
- **min_samples_split:** 2
- **random_state:** 42

Tabla 4: Classification Report Decision Tree

Class	Precision	Recall	F1-score	Support
0	0.86	0.85	0.85	24892
1	0.75	0.76	0.76	14975
Accuracy			0.82	
Macro Avg	0.80	0.81	0.81	39867
Weighted Avg	0.82	0.82	0.82	39867

AUC-PR: 0.9103

Todos los modelos superan a la baseline.

El mejor modelo es el de XGBoost, con un F1-Score de 0.9 y 0.81 para las clases 0 y 1, respectivamente, y un AUC-PR de 0.914.

Uno de los factores que explica la selección de XGBoost son sus métricas balanceadas para ambas clases, ya que BCI valora tanto a los clientes morosos como a los que cumplen con sus pagos. En comparación, los otros modelos presentan métricas más bajas y desbalanceadas.

Aunque los tres modelos tienen tiempos de entrenamiento similares, al escalar los datos e incluir hiperparámetros más grandes, XGBoost es el que más se demora, mientras que la regresión logística es el más rápido.

Por estos motivos, se selecciona el modelo XGBoost para las siguientes iteraciones y optimizaciones.

3.3. Optimización de Modelos

La métrica que se decidió optimizar fue AUC-PR. Esta métrica es ideal datos desbalanceados ya que combina precision y recall. Además, se eligió debido a que el modelo de la competencia también busca optimizar esta misma métrica, permitiendo una comparación directa y efectiva.

El código establece un límite de 30 ensayos (2 minutos aproximadamente). Sin embargo, aumentar tanto el número de ensayos como la capacidad de los rangos de búsqueda de los parámetros podría explorar un espacio más amplio y detallado, incrementando la probabilidad de encontrar combinaciones óptimas.

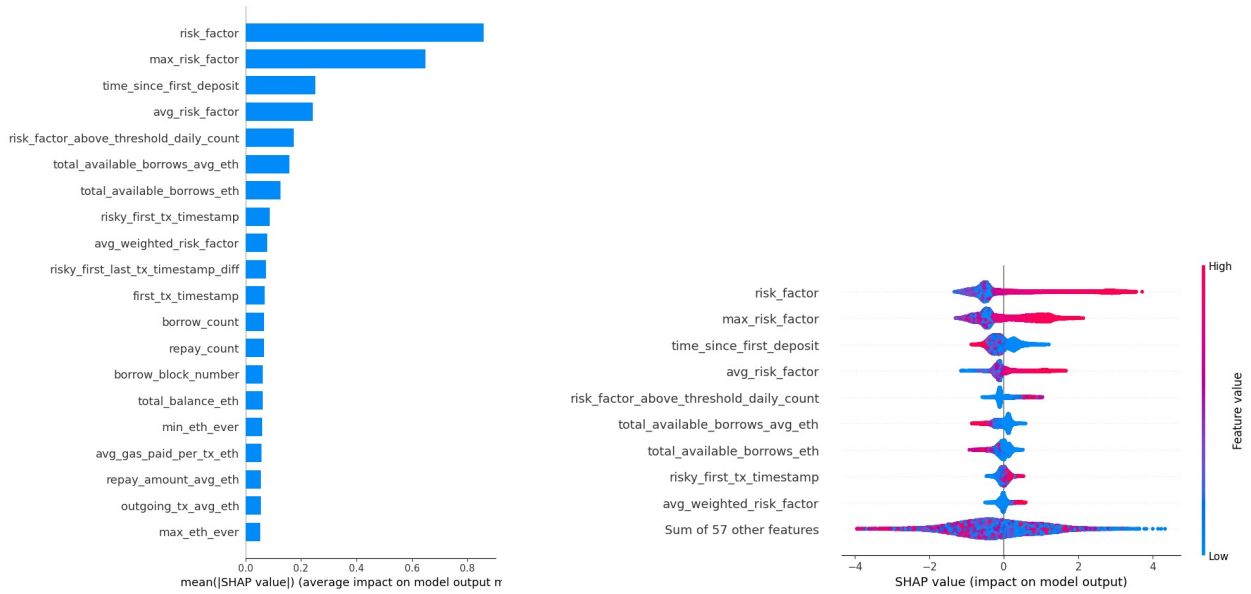
El parámetro que tuvo mayor impacto en el desempeño del modelo fue max depth, ya que una profundidad de 15 permitió capturar patrones más complejos en los datos, mejorando significativamente las predicciones. Profundidades menores, como 6 u 8, limitaron la capacidad del modelo, resultando en un rendimiento inferior. Además, la tasa de aprendizaje (learning rate) también influyó considerablemente, con valores moderados entre 0.1 y 0.2 que proporcionaron un aprendizaje estable y efectivo. Tasas más bajas o altas resultaron menos eficientes debido al subaprendizaje o a la inestabilidad. El parámetro min child weight, con valores bajos entre 1 y 7, permitió divisiones más finas en los nodos, contribuyendo a un mejor ajuste del modelo.

3.4. Interpretabilidad

3.4.1. XGBoost

En las siguientes figuras se presentan los valores SHAP del modelo XGBoost. La variable más importante identificada es risk factor (nivel de factor de riesgo), lo que sugiere que, a medida que aumenta el nivel de riesgo, también incrementa la probabilidad de incumplimiento en los pagos.

Sin embargo, se observa un posible sesgo en la variable total available borrows avg eth, ya que, según los resultados, un mayor número de préstamos parece estar asociado con una mayor probabilidad de cumplimiento de pagos. Esto contrasta con lo que el sentido común indicaría, dado que se esperaría que un mayor endeudamiento incrementara el riesgo de incumplimiento.



(a) Shap Values Promedios XGBoost

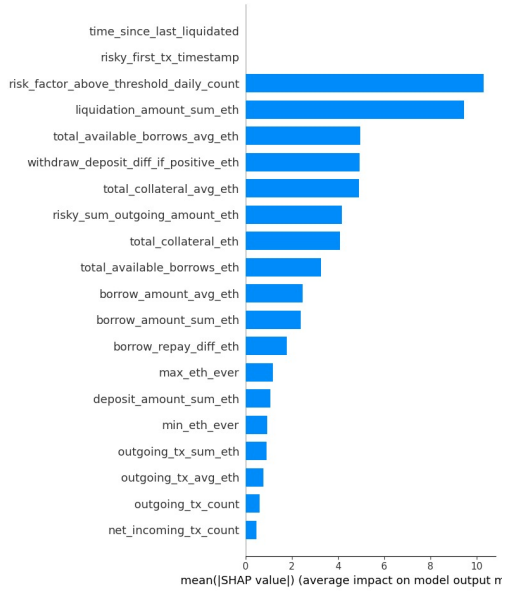
(b) Shap Values XGBoost

Figura 18: Shap Values para XGBoost: Promedio y Distribución.

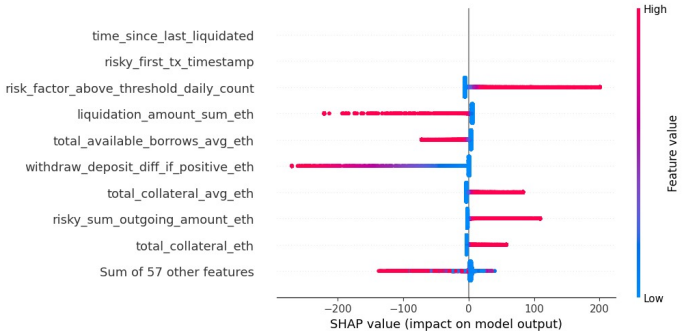
3.4.2. Logistic Regresion

En las siguientes figuras se pueden analizar los Shaps Values de Logistic Regresion. Donde la variable más importante es risk factor above threshold daily count"(Número de días en que el factor de riesgo estuvo por encima de un umbral). Esto indica que mientras más días una persona/cartera tiene un factor de riesgo encima del umbral, más probabilidad tiene de incumplir los pagos.

Es posible que exista sesgo un "total available borrows avg ethza que mientras más cantidad de prestamos tiene, más probabilidad tiene de cumplir con sus pagos. Porque el sentido común indica lo contrario.



(a) Shap Values Promedios Logistic Regression



(b) Shap Values Logistic Regression

Figura 19: Shap Values para Logistic Regression: Promedio y Distribución.

4. MLOps

4.1. Tracking con MLFlow

Se configura MLFlow con los códigos de creación de modelos. Se abre un puerto local 'http://127.0.0.1:5000' donde se documentan todos los resultados de los modelos. Se muestran los resultados de AUC-PR y los hiperparámetros cuando se usan.

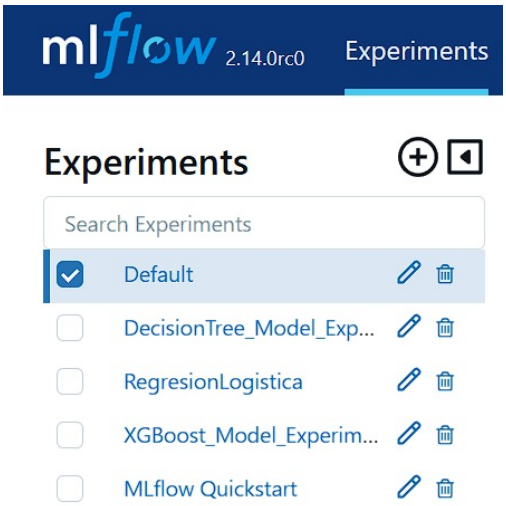


Figura 20: Modelos documentados

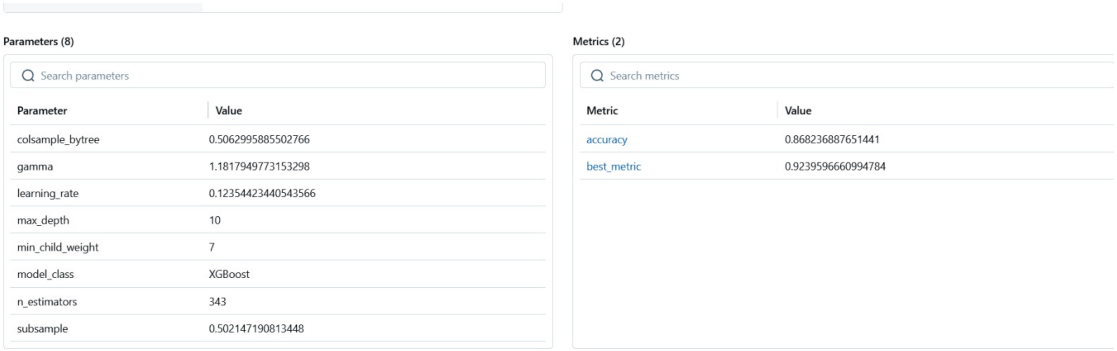


Figura 21: Modelos documentados

4.2. Monitoreo

Para detectar data drift, se puede usar una distribución estadística llamada Kolmogorov-Smirnov para comparar la distribución de las características del nuevo conjunto de datos con el conjunto de datos original. Donde la hipótesis nula H0 representa que ambas muestras provienen de la misma distribución mientras que H1 representa que vienen de distribuciones diferentes.

Para la evaluación continua se debe asegurar que el modelo mantenga un rendimiento constante, incluso en condiciones cambiantes. Para esto se analizarán las métricas claves como F1-Score, Recall y Precision.

- ¿Se detecta data drift? ¿Cómo se abordará?: Si se detecta data drift se reentrenaran los modelos con los nuevos datos para mantener la consistencia de las métricas. Asi el modelo puede seguir aprendiendo nuevos patrones.

- ¿Qué sucede si los datos nuevos son más desbalanceados?: Si los datos son más desbalanceados se puede hacer Oversampling o Undersampling para balancear ambas clases. Todo depende de la proporción del desbalance, pues si es 40/60 no se requiere mayor balance. En cambio, si existe una proporción 20/80 a favor de la clase 0, lo ideal sería usar SMOTE para duplicar filas con clase 1.

5. Resultados

En cada iteración las métrica fueron mejorando. Para la primera fase del torneo se obtuvo un AUC-PR aproximado de 0.97 en local pero a la hora de subirlo a la competencia solamente se obtuvo un 0.75. Esto usando solamente reducción de dimensionalidad (varianza menor a 0.1 y alta correlación 0.8).

Para la segunda iteración se llegó a un resultado local de 0.93 de AUC-PR y 0.85 en competencia, mejorando notoriamente el modelo al agregar procesamiento de outliers y optimización de thresholds. Además calibrando los umbrales de reducción de dimensionalidad (varianza menor a 0.05 y correlación mayor a 0.95)

Para la última iteración se intentó usar log para normalizar las variables llegando a un AUC-PR de 0.43 en la competencia.

Tabla 5: Reporte de clasificación XGBoost final.

Clase	Precisión	Recall	F1-Score	Soporte
0	0.86	0.95	0.90	42,541
1	0.90	0.78	0.84	28,333
Accuracy		0.88 (70,874)		
Promedio Macro	0.88	0.86	0.87	70,874
Promedio Ponderado	0.88	0.88	0.88	70,874

Hiperparámetros que explican los resultados

- **Profundidad máxima (`max_depth`)**:
  - La profundidad óptima fue 15, lo que permitió al modelo capturar patrones complejos en los datos.
  - Profundidades menores (6 u 8) limitaron esta capacidad, mientras que profundidades excesivas pudieron haber causado *overfitting*.
- **Learning rate**:
  - Valores entre 0.1 y 0.2 facilitaron un aprendizaje estable y efectivo, evitando tanto el subaprendizaje como la inestabilidad de tasas más altas o más bajas.
- `scale_pos_weight`:

  - Ajustó la influencia de las clases en los datos desbalanceados, mejorando el *recall* de la clase minoritaria (clientes morosos).

- **`min_child_weight`**:

  - Valores entre 1 y 7 permitieron divisiones más finas, ajustándose mejor a las características de los datos sin introducir ruido.

Factores que contribuyeron al *overfitting* o *underfitting*

- **Overfitting**:

  - El riesgo de *overfitting* fue mitigado con técnicas como la eliminación de variables altamente correlacionadas (> 0.95) y con baja varianza (< 0.05). Esto redujo la complejidad del modelo al eliminar redundancia y ruido. Sin embargo, profundidades muy altas (superiores a 15) habrían podido ocasionar overfitting al capturar ruido en lugar de patrones útiles.

- **Underfitting**:

  - Tasas de aprendizaje bajas (< 0.1) o profundidades insuficientes (< 6) resultaron en *underfitting*, donde el modelo no capturó suficiente información para realizar predicciones precisas.

- **Elección del mejor modelo**:

  - XGBoost fue seleccionado debido a sus métricas balanceadas entre ambas clases y su capacidad para manejar datos desbalanceados. Aunque toma más tiempo de entrenamiento, su flexibilidad para modelar relaciones no lineales y manejar interacciones lo hizo superior a la regresión logística y el árbol de decisión.

Implicaciones prácticas de los resultados

- El XGBoost optimizado ofrece una herramienta sólida para identificar clientes en riesgo de morosidad, maximizando tanto la recuperación de ingresos como la capacidad de toma de decisiones del banco.

- Las métricas equilibradas aseguran un buen desempeño en ambas clases, crucial en este contexto, ya que tanto identificar clientes solventes como prevenir riesgos crediticios son prioritarios para el BCI.

Este análisis destaca la importancia del preprocesamiento, el ajuste de hiperparámetros y la elección del modelo para equilibrar precisión y robustez en problemas de clasificación con datos desbalanceados.

6. Conclusiones

El modelo XGBoost desarrollado ha demostrado ser eficaz para predecir tanto a los clientes con riesgo de morosidad como a aquellos que cumplen con sus pagos. A pesar de enfrentar desafíos debido a la presencia de fechas incorrectas y variables con baja varianza que no aportan información significativa, se hizo un correcto preprocesamiento consiguiendo un rendimiento sólido. Mediante la optimización de hiperparámetros con Optuna, se logró encontrar el mejor modelo, al que se le realizó un análisis de interpretabilidad utilizando SHAP Values. Este análisis reveló conclusiones clave, como que un mayor número de días por encima del umbral de riesgo incrementa la probabilidad de morosidad, mientras que un mayor monto liquidado reduce dicha probabilidad. Además, gracias a MLFlow, es posible realizar un seguimiento detallado de los experimentos, lo que permite almacenar información crucial sobre los hiperparámetros y las métricas obtenidas, como el AUC-PR, garantizando así una trazabilidad completa del proceso de modelado.

El modelo es útil para identificar y clasificar a los clientes según su probabilidad de morosidad, permitiendo a la organización tomar decisiones informadas para gestionar riesgos y optimizar la recuperación de pagos. La capacidad de seguimiento de los experimentos mediante MLFlow facilita la iteración del modelo, asegurando una mejora continua a lo largo del tiempo. Sin embargo, la calidad del modelo depende de la calidad de los datos, y la presencia de fechas incorrectas o variables irrelevantes puede afectar el rendimiento, lo que sugiere que se debe realizar un esfuerzo continuo para limpiar y mejorar los datos. Además, el modelo puede ser sensible a los cambios en la distribución de los datos, lo que requiere un monitoreo constante y la actualización periódica para evitar problemas como el "data drift".