PONTIFICIA UNIVERSIDAD CATOLICA DEL ECUADOR

FACULTAD DE HÁBITAT, INFRAESTRUCTURA Y CREATIVIDAD

MAESTRÍA EN SISTEMAS DE INFORMACIÓN MENCIÓN DATA SCIENCE



TRABAJO PREVIO PARA LA OBTENCIÓN DEL TÍTULO DE MAGISTER EN SISTEMAS DE INFORMACIÓN MENCIÓN DATA SCIENCE.

TEMA:

MODELO PREDICTIVO BASADO EN MACHINE LEARNING PARA LA DETECCIÓN DE TRANSACCIONES FRAUDULENTAS EN EL SISTEMA BANCARIO.

AUTOR:

ARACELY ELIZABETH CADENA CONDOR

DIRECTOR:

EDISON VICENTE MORA LONDOÑO

QUITO, AGOSTO 2026

AGRADECIMIENTOS

Agradezco profundamente a mis padres, quienes han sido el pilar fundamental en este camino académico y personal. Gracias por creer en mí, por brindarme siempre su apoyo incondicional y, especialmente, por haber hecho posible mi formación de maestría con su esfuerzo y sacrificio. Este logro también les pertenece.

A mi hermana, por estar presente en los momentos de cansancio, por las conversaciones, por las pequeñas distracciones que muchas veces ayudaron a aliviar la presión.

Finalmente, agradezco a todas las personas que formaron parte de este recorrido, y a mí misma por haber asumido este reto con constancia, disciplina y el deseo de seguir creciendo tanto personal como profesionalmente.

INDICE GENERAL

CAPÍTULO 1: INTRODUCCIÓN 1

1.1 Introducción 1

1.2 Objetivos 2

1.2.1 Objetivo general: 2

1.2.2 Objetivos específicos: 2

1.3 Justificación 2

1.3.1 Justificación Social 3

1.3.2 Justificación Científica 3

1.3.3 Justificación Empresarial 3

1.4 Preguntas de Investigación 4

1.4.1 Pregunta Principal 4

1.4.2 Preguntas Específicas 4

CAPÍTULO 2: DESARROLLO 4

2.1 Marco Teórico 4

2.1.1 Fraude en transacciones con tarjetas de crédito 4

2.1.2 Machine Learning en la detección de fraude 5

2.1.3 Aprendizaje supervisado en Machine Learning 6

2.1.4 Algoritmos de clasificación supervisada 6

2.1.4.1 Logistic Regression 6

2.1.4.2 Random Forest 8

2.1.4.3 XGBoost 9

2.1.4.4 LightGBM 10

2.1.5 Desbalance de clases en detección de fraude 11

2.1.6 Técnicas de balanceo de datos 12

2.1.6.1 SMOTE 12

2.1.7 Métricas de evaluación 13

2.1.7.1 Precision 13

2.1.7.2 Recall 14

2.1.7.3 F1-Score 14

2.1.7.4 ROC-AUC 15

2.1.7.5 PR-AUC 16

2.1.7.6 Matriz de confusión 16

2.1.8 Stratified 5-Fold Cross-Validation 17
2.1.9 Matriz de costos en la detección de fraude 18
2.2. Metodología 19
2.2.1 Diseño de investigación 19
2.2.2 Metodología CRISP-ML(Q) 19
2.2.3 Comprensión del negocio y los datos 21
2.2.4 Preparación de los datos. 23
2.2.4.1 Limpieza y transformación 23
2.2.4.2 Selección y adecuación de variables predictoras 24
2.2.4.3 Identificación del desbalance de clases 27
2.2.5 Modelado 28
2.2.5.1 Construcción de pipelines y escenarios experimentales. 28
2.2.5.2 Optimización de hiperparámetros mediante GridSearchCV 29
2.2.6 Evaluación y selección de modelos 31
2.2.6.1 Validación cruzada estratificada de cinco pliegues 31
2.2.6.2 Evaluación económica y criterio de selección 32
2.2.6.3 Evaluación final sobre el conjunto de prueba reservado 33
2.2.7 Fase de despliegue 33
2.2.8 Fase de monitoreo y mantenimiento 34
2.2.9 Garantía de calidad durante el ciclo CRISP-ML (Q) 35
2.3. Resultados y Análisis 38
2.3.1. Resultados del análisis exploratorio 38
2.3.2 Resultados del pre procesamiento de datos. 41
2.3.3 Resultados de la optimización de hiperparámetros 43
2.3.4 Evaluación de los modelos optimizados. 45
2.3.4.1 Regresión Logística 47
2.3.4.2 Random Forest 47
2.3.4.3 XGBoost 47
2.3.4.4 LightGBM 48
2.3.5 Comparación del desempeño de los modelos predictivos 48
2.3.6 Resultados de la evaluación económica 50
2.3.7 Selección del modelo predictivo 51
2.3.8 Evaluación final del modelo seleccionado 52
2.3.9 Identificación de Variables relevantes en el modelo seleccionado. 55
CAPÍTULO 3: CONCLUSIONES Y RECOMENDACIONES 57

3.1. Conclusiones 57
3.2. Recomendaciones 58
Referencias 59
Anexo A. Fragmentos principales del código 62
Preprocesamiento de datos 62
Construcción de pipelines 63
Optimización de hiperparámetros 65
Stratified 5-fold cross-validation. 66
Evaluación económica 68
Entrenamiento y evaluación del modelo final 69

INDICE DE TABLAS

Tabla 1. Estructura general de una matriz de confusión. 17
Tabla 2. Matriz conceptual de costos para la detección de transacciones fraudulentas 18
Tabla 3. Características generales de los archivos utilizados. 21
Tabla 4. Descripción de las Variables del conjunto de datos 22
Tabla 5. Actividades realizadas durante la etapa de limpieza y transformación. 23
Tabla 6. Análisis de cardinalidad y tratamiento de las variables 25
Tabla 7. Estructura de los pipelines para los escenarios experimentales. 29
Tabla 8. Espacios de búsqueda de hiperparámetros. 30
Tabla 9. Controles de garantía de calidad aplicados durante el desarrollo experimental de los modelos. 35
Tabla 10. Resumen de los resultados del pre-procesamiento de datos 41
Tabla 11. Características finales de los conjuntos de datos. 42
Tabla 12. Hiperparámetros seleccionados mediante GridSearchCV 43
Tabla 13. Resultados de las configuraciones seleccionadas durante la optimización 44
Tabla 14. Desempeño de los modelos optimizados mediante Stratified 5-fold cross-validation 45
Tabla 15. Resultados acumulados de clasificación en los cinco pliegues 46
Tabla 16. Variación de las métricas después de aplicar SMOTE 49
Tabla 17. Costos estimados de las configuraciones evaluadas 50
Tabla 18. Comparación entre XGBoost sin SMOTE y LightGBM con SMOTE 51
Tabla 19. Métricas finales de LightGBM con SMOTE 53
Tabla 20. Evaluación económica final de LightGBM con SMOTE 55

INDICE DE FIGURAS

Figura 1. Representación de la función sigmoide en Regresión Logística 7
Figura 2. Fases de la metodología CRISP-ML(Q) 20
Figura 3. Proceso de comprensión inicial del conjunto de datos realizado en Google Colab. 23
Figura 4. Distribución de las transacciones según la variable objetivo 39
Figura 5. Distribución de las transacciones fraudulentas por hora 40
Figura 6. Cantidad de transacciones fraudulentas por categoría comercial 40
Figura 7. Distribución del monto de las transacciones 41
Figura 8. Matriz de confusión del modelo final LightGBM con SMOTE 54
Figura 9. Curvas ROC y Precision-Recall del modelo final LightGBM con SMOTE 54
Figura 10. Diez variables más relevantes del modelo LightGBM con SMOTE 56

CAPÍTULO 1: INTRODUCCIÓN

1.1 Introducción

Dentro del sistema bancario, la detección de transacciones fraudulentas se ha convertido en un desafío relevante. El volumen creciente de operaciones dificulta la revisión manual y hace necesario el uso de herramientas capaces de analizar grandes cantidades de datos. Además, las modalidades de fraude cambian con frecuencia, lo que limita la efectividad de mecanismos basados únicamente en reglas fijas.

Además, en los conjuntos de datos financieros, las transacciones legítimas suelen ser mucho más numerosas que las fraudulentas. Esta diferencia podría orientar el aprendizaje a la clase mayoritaria y disminuir su capacidad para identificar correctamente los casos de fraude (Al-dahasi et al., 2025).

El objetivo de este trabajo es desarrollar un modelo predictivo basado en Machine Learning para la detección de transacciones fraudulentas en tarjetas de crédito. El estudio se realizó con los archivos fraudTrain.csv y fraudTest.csv, pertenecientes al conjunto de datos públicos Credit Card Fraud Detection, disponible en Kaggle (Bhadouria, n.d.)

Se utilizó la metodología CRISP-ML(Q), que permitió organizar las fases de comprensión del problema y de los datos, preparación, modelado y evaluación (Studer et al., 2021). En la fase de modelado se utilizaron 4 algoritmos de clasificación supervisada: Regresión Logística, Random Forest, XGBoost y LightGBM. Cada algoritmo fue analizado con la distribución original de los datos y mediante la aplicación de SMOTE.

Los hiperparámetros de cada alternativa se ajustaron mediante una búsqueda sistemática acotada con GridSearchCV y validación cruzada estratificada de tres pliegues, utilizando PR-AUC como criterio de selección. Las ocho configuraciones optimizadas fueron evaluadas mediante validación cruzada estratificada de cinco pliegues.

Los promedios y las desviaciones estándar obtenidos en este procedimiento se utilizaron posteriormente para comparar el desempeño de los modelos mediante Accuracy, Precision, Recall, F1-Score, ROC-AUC y PR-AUC. La selección también incorporó un análisis de costos relacionado con los fraudes no detectados y las alertas generadas.

Finalmente, la configuración seleccionada se entrenó con el conjunto completo de entrenamiento (fraudTrain.csv) y se evaluó una sola vez sobre el conjunto de prueba reservado (fraudTest.csv).

Debido al uso de datos públicos de Kaggle y a que la evaluación económica combina los montos registrados en las transacciones con supuestos sobre el costo de investigación de las alertas, los resultados se circunscriben al contexto experimental definido.

1.2 Objetivos

1.2.1 Objetivo general:

- Desarrollar un modelo predictivo basado en Machine Learning para la detección de transacciones fraudulentas en tarjetas de crédito mediante la metodología CRISP-ML(Q).

1.2.2 Objetivos específicos:

- Analizar y preparar el conjunto de datos de transacciones financieras mediante procesos de limpieza, transformación e ingeniería de características, para garantizar la calidad de la información antes del entrenamiento de modelos predictivos.
- Realizar un análisis exploratorio de los datos e identificar las variables más relevantes asociadas a la detección de transacciones fraudulentas.
- Desarrollar y entrenar modelos predictivos de Regresión Logística, Random Forest, XGBoost y LightGBM bajo dos escenarios, con la distribución original de las clases y con la aplicación de SMOTE.
- Evaluar y comparar el desempeño de los modelos mediante Precision, Recall, F1-Score, ROC-AUC y PR-AUC, complementando el análisis con la matriz de confusión para seleccionar el modelo con mejor capacidad predictiva.

1.3 Justificación

La detección de fraude en transacciones con tarjetas de crédito constituye uno de los desafíos más importantes dentro del sistema bancario actual. El uso masivo de tarjetas de crédito y la expansión de plataformas digitales han facilitado la realización de operaciones comerciales de forma más rápida y accesible. Sin embargo, este mismo avance ha

generado un incremento en los riesgos asociados a actividades fraudulentas que afectan tanto a las instituciones financieras como a sus usuarios.

1.3.1 Justificación Social

Desde una perspectiva social, esta investigación adquiere relevancia debido a que el fraude financiero impacta directamente en la seguridad y confianza de las personas. Las transacciones no autorizadas pueden ocasionar pérdidas económicas además de una percepción de inseguridad frente al uso de tarjetas de crédito.

Actualmente, un gran número de usuarios realiza pagos, transferencias y compras en línea de manera frecuente, por lo que la protección de la información financiera se ha convertido en una necesidad cada vez más importante.

En este sentido, el desarrollo de modelos predictivos basados en Machine Learning representa una alternativa para anticipar posibles comportamientos fraudulentos a partir del análisis de patrones presentes en los datos.

1.3.2 Justificación Científica

En el ámbito científico, esta investigación aporta evidencia experimental al campo de la ciencia de datos mediante el desarrollo y comparación de diferentes modelos predictivos con el propósito de identificar cuál presenta un mejor desempeño frente a un problema de clasificación dentro del conjunto de datos analizado.

Uno de los aspectos más relevantes del estudio se enfoca en el análisis de conjuntos de datos desbalanceados, una condición frecuente debido a que las transacciones fraudulentas representan una proporción considerablemente menor frente a las transacciones legítimas.

El estudio también contribuye a generar evidencia sobre la aplicación de SMOTE y su influencia en el desempeño de distintos modelos predictivos.

1.3.3 Justificación Empresarial

Desde la perspectiva empresarial, las instituciones financieras procesan diariamente grandes cantidades de transacciones, lo que dificulta detectar de manera inmediata

operaciones inusuales o potencialmente fraudulentas. La identificación tardía u omisión de este tipo de transacciones puede generar pérdidas económicas significativas y afectar la relación de confianza entre las entidades bancarias y sus clientes (Ali et al., 2022).

Ante esta situación, el uso de modelos predictivos puede ser de apoyo en el análisis de patrones de comportamiento e identificación de posibles casos de fraude. La implementación de soluciones basadas en Machine Learning podría facilitar la generación de alertas y fortalecer los procesos internos de prevención y control.

1.4 Preguntas de Investigación

1.4.1 Pregunta Principal

- ¿Cómo desarrollar un modelo predictivo basado en técnicas de aprendizaje automático que permita detectar transacciones fraudulentas en tarjetas de crédito a partir de un dataset público de Kaggle, mediante la preparación de datos y la evaluación del efecto del balanceo de clases?

1.4.2 Preguntas Específicas

- ¿Cómo preparar el conjunto de datos mediante procesos de limpieza y transformación para mejorar su calidad antes del entrenamiento?
- ¿Qué patrones y variables del conjunto de datos analizado se asocian con las transacciones fraudulentas?
- ¿Qué efecto produce la aplicación de SMOTE en el desempeño de los modelos?
- ¿Cuál es el desempeño de Regresión Logística, Random Forest, XGBoost y LightGBM sobre el conjunto de datos analizado?
- ¿Qué modelo presenta el desempeño más adecuado según Precision, Recall, F1-Score, ROC-AUC y PR-AUC?

CAPÍTULO 2: DESARROLLO

2.1 Marco Teórico

2.1.1 Fraude en transacciones con tarjetas de crédito

El fraude en transacciones con tarjetas de crédito constituye una modalidad de fraude financiero que afecta tanto a los usuarios como a las instituciones financieras. Ocurre cuando un tercero realiza operaciones no autorizadas utilizando información bancaria que

obtuvo de manera ilícita, generando pérdidas económicas para ambas partes (Bhattacharyya et al., 2011).

Las modalidades de fraude asociadas a tarjetas de crédito han evolucionado considerablemente en los últimos años. Entre las prácticas más comunes se encuentran la clonación física de tarjetas, ataques de phishing, filtración de datos personales y uso indebido de información obtenida a través de plataformas digitales vulneradas. La diversidad de estos mecanismos hace que la detección temprana sea cada vez más compleja, especialmente cuando los patrones fraudulentos logran imitar comportamientos legítimos de consumo (Ali et al., 2022; Lucas & Jurgovsky, 2020).

Uno de los principales desafíos radica en la velocidad con la que se procesan este tipo de operaciones. A diferencia de otros procesos financieros que pueden ser revisados posteriormente, las transacciones con tarjetas son ejecutadas en cuestión de segundos, lo que exige sistemas capaces de identificar patrones en tiempo real antes de autorizar una operación potencialmente fraudulenta (Lebichot et al., 2021).

En escenarios reales, las transacciones fraudulentas tienden a representar un porcentaje mínimo frente al volumen total de operaciones legítimas registradas diariamente. Esta característica genera conjuntos de datos altamente desbalanceados, esto en ocasiones dificulta a los modelos tradicionales aprender suficientes patrones asociados al fraude sin verse influenciados por la clase mayoritaria.

2.1.2 Machine Learning en la detección de fraude

El Machine Learning dentro del ámbito financiero permite analizar datos históricos e identificar relaciones entre las características de las transacciones. Estos patrones pueden utilizarse para diferenciar las operaciones legítimas de las fraudulentas, sin depender exclusivamente de reglas predefinidas (Ali et al., 2022).

A diferencia de los sistemas tradicionales basados en reglas estáticas, los algoritmos de Machine Learning pueden identificar relaciones complejas entre distintas variables presentes en una transacción, por ejemplo: montos inusuales, frecuencia de uso, ubicación geográfica o patrones de consumo. Esto permite mejorar la identificación de operaciones sospechosas incluso cuando los patrones de fraude cambian constantemente con el tiempo.

Diversas investigaciones han demostrado que la aplicación de Machine Learning dentro del sector bancario permite mejorar considerablemente los mecanismos de prevención frente a actividades fraudulentas. Los modelos predictivos han mostrado resultados superiores frente a enfoques convencionales, convirtiéndose en herramientas cada vez más utilizadas dentro de los sistemas de seguridad financiera y optimizar la detección temprana de fraude (Hashemi et al., 2023).

2.1.3 Aprendizaje supervisado en Machine Learning

Dentro del campo del Machine Learning, la clasificación supervisada consiste en que los algoritmos aprenden a partir de datos previamente etiquetados, con el objetivo de identificar patrones que permitan predecir la categoría a la que pertenece una nueva observación. Lo que permite ajustar su capacidad de predicción conforme se analiza la información disponible (Ali et al., 2022).

A diferencia de otros enfoques de aprendizaje automático, en la clasificación supervisada el modelo analiza la relación existente entre las características de cada observación y la categoría asociada. Luego ajusta internamente sus parámetros con el propósito de reducir errores de predicción y mejorar su aprendizaje a partir de los datos disponibles.

Entre los algoritmos más utilizados dentro de la clasificación supervisada se encuentran los modelos lineales, algoritmos basados en árboles de decisión y redes neuronales artificiales. La selección del algoritmo depende de la estructura del conjunto de datos, la complejidad del problema y los criterios definidos para evaluar su desempeño (Hashemi et al., 2023).

2.1.4 Algoritmos de clasificación supervisada

2.1.4.1 Logistic Regression

La Regresión Logística es un algoritmo de aprendizaje supervisado utilizado en problemas de clasificación binaria. Su objetivo consiste en predecir la probabilidad de ocurrencia de un evento dentro de dos categorías mutuamente excluyentes. A diferencia de la regresión lineal, que predice valores continuos, este algoritmo transforma la relación existente entre las variables predictoras y la variable objetivo en una probabilidad comprendida entre cero y uno (Trevor Hastie, Robert Tibshirani, 2009).

El modelo se fundamenta en la función logística, también conocida como función sigmoide, la cual permite modelar relaciones no lineales entre las variables de entrada y

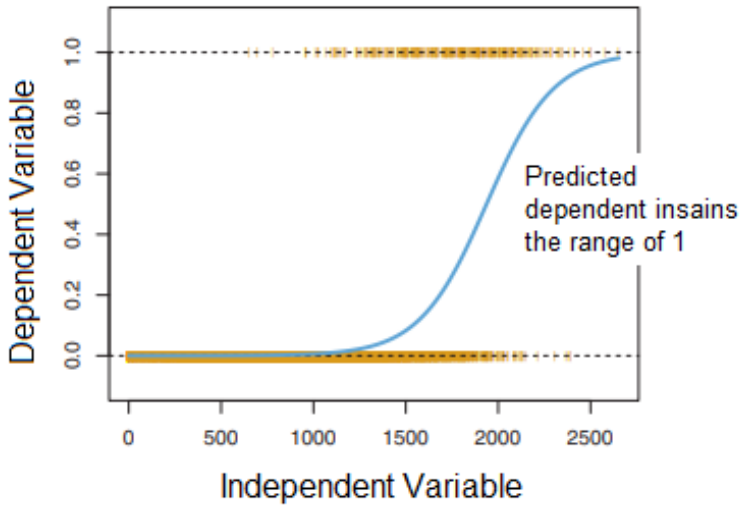
la probabilidad asociada a una determinada clase. Su representación matemática se expresa mediante la siguiente ecuación:

$$\rho = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n)}}$$

En esta expresión:

- ρ representa la probabilidad de la clase positiva (Fraude).
- $1 - \rho$ representa la probabilidad de la clase negativa (No es Fraude)
- β representan el peso asociado a cada variable predictora.

Figura 1. Representación de la función sigmoide en Regresión Logística



Nota. Imagen adaptada de Handbook of Quantile Regression (Koenker et al., 2017).

Durante el proceso de entrenamiento, el algoritmo ajusta iterativamente sus coeficientes con el propósito de minimizar el error de predicción. Generalmente utiliza métodos de optimización como Maximum Likelihood Estimation (MLE), técnica estadística que busca encontrar el conjunto de parámetros que maximiza la probabilidad de observar los datos disponibles (Hashemi et al., 2023).

En problemas de clasificación, la decisión final se establece mediante un umbral de probabilidad previamente definido, generalmente en 0.5. Cuando la probabilidad estimada supera este valor, la observación se asigna a la clase positiva; de lo contrario, se clasifica dentro de la clase negativa.

Las ventajas que presenta este modelo son: la rapidez durante el entrenamiento, facilidad de interpretación y bajo requerimiento computacional. Sin embargo, su desempeño puede

verse limitado cuando las relaciones entre variables presentan alta complejidad no lineal, situación en la que algoritmos más avanzados suelen ofrecer mejores resultados predictivos (Hashemi et al., 2023).

2.1.4.2 Random Forest

Random Forest es un algoritmo de aprendizaje supervisado basado en la combinación de múltiples árboles de decisión para resolver problemas de clasificación y regresión. Su funcionamiento se basa en Bootstrap Aggregating o bagging, técnica que permite entrenar cada árbol con una muestra diferente de los datos (Breiman, 2001).

Durante el entrenamiento de cada árbol, se selecciona una muestra aleatoria del conjunto de entrenamiento, permitiendo que una misma observación aparezca más de una vez. A partir de cada subconjunto se construye un árbol de decisión independiente, utilizando una selección aleatoria de variables en cada nodo. Este procedimiento favorece la diversidad entre los árboles y disminuye la correlación existente entre ellos, reduciendo el riesgo de sobreajuste (Breiman, 2001).

En problemas de clasificación, cada árbol emite una predicción independiente y la clase con mayor número de votos constituye el resultado final del algoritmo. Este mecanismo puede representarse mediante la siguiente expresión (Breiman, 2001):

$$\hat{y} = mode\{h_1(x), h_2(x), \dots, h_k(x)\}$$

Donde:

- $\hat{y}$ representa la predicción final
- $h_k(x)$ corresponde a la predicción realizada por el árbol k
- *K* indica el número total de árboles que conforman el bosque

La construcción de cada árbol de decisión se realiza mediante un proceso iterativo en el que el algoritmo divide progresivamente el conjunto de datos. Este proceso continúa hasta que se alcanza un criterio de parada previamente definido o los nodos contienen observaciones suficientemente homogéneas.

Para determinar cuál es la mejor división en cada nodo, Random Forest emplea criterios de impureza, lo que significa que cuantifican qué tan mezcladas se encuentran las clases dentro de un grupo de datos. Uno de los criterios más utilizados es el índice de Gini, el cual mide el grado de heterogeneidad de las observaciones en un nodo. Un valor de Gini

cercano a cero indica que la mayoría de las observaciones pertenecen a una misma clase, mientras que valores más altos significan una mayor mezcla entre transacciones fraudulentas y legítimas. Este índice se calcula mediante la siguiente expresión (Trevor Hastie, Robert Tibshirani, 2009):

$$G = 1 - \sum_{i=1}^{C} p_i^2$$

Donde:

- p_i representa la proporción de observaciones pertenecientes a la clase i
- C representa el número total de clases presentes en el nodo

Una de las principales ventajas de Random Forest es su capacidad para modelar relaciones complejas entre variables sin requerir una distribución estadística específica de los datos (Breiman, 2001).

2.1.4.3 XGBoost

XGBoost (Extreme Gradient Boosting) es un algoritmo de aprendizaje supervisado. Su funcionamiento se basa en el principio de gradient boosting. Consiste en entrenar múltiples modelos más débiles de manera secuencial (Chen & Guestrin, 2016).

A diferencia de algoritmos que realizan una predicción a partir de un único modelo. XGBoost genera una secuencia de árboles donde cada nuevo árbol intenta corregir los errores producidos por las iteraciones anteriores, reduciendo gradualmente el error de predicción.

El fundamento matemático de XGBoost se basa en una función objetivo, compuesta por dos elementos principales: la función de pérdida y un término de regularización orientado a controlar la complejidad del modelo. Esta función puede expresarse mediante la siguiente ecuación: (Chen & Guestrin, 2016)

$$obj(\theta) = L(\theta) + \Omega(\theta)$$

Donde:

- $L(\theta)$ representa la función de pérdida encargada de medir la diferencia entre las predicciones generadas y los valores reales.

- $\Omega(\theta)$ es el término de regularización utilizado para penalizar modelos excesivamente complejos y reducir el riesgo de sobreajuste.

Durante el proceso de entrenamiento, el algoritmo optimiza esta función objetivo utilizando descenso por gradiente. La regularización permite controlar el número de nodos, profundidad de árboles y peso asignado a cada estructura, mejorando la capacidad de generalización del modelo y evitando que el algoritmo aprenda patrones excesivamente específicos presentes únicamente en el conjunto de entrenamiento (Chen & Guestrin, 2016).

Investigaciones recientes también han evidenciado que XGBoost presenta un comportamiento robusto en escenarios donde existe desbalance entre clases. Este algoritmo obtuvo uno de los mejores resultados al compararse con otros modelos predictivos aplicados a transacciones financieras, manteniendo altos niveles de precisión y capacidad de detección (Al-dahasi et al., 2025) y (Lucas & Jurgovsky, 2020) .

2.1.4.4 LightGBM

LightGBM (Light Gradient Boosting Machine) es un algoritmo de aprendizaje supervisado basado en árboles potenciados por gradiente. Su diseño busca reducir el tiempo de entrenamiento y el consumo de memoria. Al igual que XGBoost, construye los árboles de manera secuencial. En cada iteración incorpora un nuevo árbol orientado a reducir los errores que permanecen en el conjunto anterior (Ke et al., 2017).

Una de las principales diferencias entre ambos algoritmos es su estrategia para construir árboles. Mientras XGBoost utiliza un crecimiento por niveles (level-wise growth), LightGBM implementa un crecimiento por hojas (leaf-wise growth), estrategia que selecciona en cada iteración la hoja cuya división produce la mayor reducción del error (Ke et al., 2017).

El crecimiento leaf-wise permite concentrar el proceso de aprendizaje en regiones del árbol donde pueda seguir mejorando. Esto reduce la función de pérdida y aumenta la precisión del modelo con un menor número de iteraciones. Sin embargo, se puede generar árboles más profundos y complejos. Los parámetros como max_depth y min_data_in_leaf permiten controlar el sobreajuste (Ke et al., 2017).

Además de su estrategia de crecimiento, LightGBM incorpora dos mecanismos de optimización que contribuyen a su eficiencia computacional:

El primero corresponde a Gradient-based One-Side Sampling (GOSS), técnica que reduce el número de observaciones utilizadas durante el entrenamiento conservando aquellas donde el modelo comete mayores errores de predicción (gradientes elevados). Esta estrategia reduce el tiempo de entrenamiento sin afectar de manera significativa la capacidad predictiva del algoritmo.

El segundo mecanismo corresponde a Exclusive Feature Bundling (EFB), técnica orientada a reducir la dimensionalidad del conjunto de datos mediante el agrupamiento de variables mutuamente excluyentes dentro de una misma representación. Al disminuir el número de características procesadas durante el entrenamiento, el algoritmo reduce el consumo de memoria y mejora la velocidad de ejecución, especialmente cuando se trabaja con conjuntos de datos que contienen un elevado número de variables categóricas o derivadas (Ke et al., 2017).

Las pruebas realizadas con LightGBM mostraron una reducción en el tiempo requerido para entrenar el modelo, mientras que su precisión se mantuvo cercana a la obtenida con GBDT convencional. Este resultado es conveniente cuando la eficiencia computacional es un aspecto relevante (Ke et al., 2017).

2.1.5 Desbalance de clases en detección de fraude

Se conoce como desbalance de clases cuando dentro de un conjunto de datos, una de sus clases es extremadamente mayoritaria respecto a la otra. Dependiendo del porcentaje de desbalance, en ocasiones puede representar dificultad frecuente en aplicaciones del Machine Learning, en escenarios donde el evento de interés ocurre con baja frecuencia, como sucede en la detección de fraude financiero.

Dentro del sistema financiero, las transacciones fraudulentas suelen representar un porcentaje mínimo frente al volumen total de operaciones legítimas procesadas diariamente. Esto genera conjuntos de datos altamente desbalanceados. Esta característica constituye uno de los principales desafíos en el desarrollo de modelos predictivos orientados a la detección de fraude financiero.

Por otro lado, el desbalance de clases puede afectar de manera diferente a cada algoritmo, depende de la sensibilidad de aprendizaje, la función de pérdida, hiperparámetros y configuraciones utilizadas. Aunque algunos modelos tienden a favorecer la clase mayoritaria debido a que esta domina estadísticamente el proceso de aprendizaje.

Como consecuencia, el clasificador presenta valores favorables al predecir correctamente la mayoría de observaciones, y un desempeño deficiente al momento de identificar registros pertenecientes a la clase minoritaria (Ngai et al., 2011).

Un ejemplo frecuente ocurre cuando un conjunto de datos contiene 99% de observaciones pertenecientes a una clase legítima y únicamente el 1% asociado a fraude. En este escenario, un modelo que clasifique todas las observaciones como transacciones legítimas alcanzaría aproximadamente 99% de accuracy, sin embargo, en términos prácticos sería incapaz de detectar un solo caso fraudulento. Se debe utilizar métricas como Recall, F1-Score para complementar su análisis, no basarse únicamente en Accuracy (Ngai et al., 2011).

2.1.6 Técnicas de balanceo de datos

De manera general, las técnicas de balanceo pueden agruparse en tres enfoques principales. El primero corresponde a undersampling, estrategia que reduce la cantidad de observaciones pertenecientes a la clase mayoritaria. El segundo corresponde a oversampling, método que incrementa artificialmente la cantidad de registros de la clase minoritaria replicando o generando nuevas observaciones. También existen métodos híbridos que combinan ambas estrategias buscando un equilibrio entre representatividad y reducción del sesgo durante el entrenamiento (Kabane, 2024).

2.1.6.1 SMOTE

SMOTE (Synthetic Minority Over-sampling Technique), es una técnica de sobremuestreo propuesta para tratar conjuntos de datos con clases desbalanceadas. A diferencia de replicar registros existentes, este método genera nuevas muestras sintéticas a partir de observaciones reales pertenecientes a la clase minoritaria. Esto permite ampliar el conjunto de entrenamiento sin introducir copias exactas de los datos originales (Chawla et al., 2002).

El funcionamiento de SMOTE se basa en el concepto de vecinos más cercanos (k-nearest neighbors). Para cada observación perteneciente a la clase minoritaria, el algoritmo selecciona uno de sus vecinos más próximos y genera una nueva muestra sintética mediante interpolación lineal entre ambos registros. De acuerdo con el estudio SMOTE for Learning from Imbalanced Data: Progress and Challenges, este proceso puede representarse mediante la siguiente expresión matemática:

$$x_{new} = x_i + \lambda(x_{nn} - x_i)$$

Donde:

- x_i representa la observación original de la clase minoritaria
- x_{nn} corresponde a uno de sus vecinos más cercanos, seleccionados aleatoriamente
- λ representa un valor aleatorio comprendido entre 0 y 1 que determina la posición del nuevo dato generado dentro del espacio de características.

La aplicación de SMOTE también puede introducir observaciones sintéticas poco representativas cuando las fronteras entre clases presentan alta superposición o cuando existen registros atípicos dentro de la clase minoritaria. Por esta razón, la selección de técnicas de balanceo debe analizarse cuidadosamente en función de las características particulares del conjunto de datos utilizado (Ahmed et al., 2025).

2.1.7 Métricas de evaluación

En problemas de detección de fraude, donde la cantidad de transacciones legítimas suele ser considerablemente mayor que la cantidad de transacciones fraudulentas; la evaluación de modelos de clasificación constituye una etapa fundamental.

En este tipo de escenarios, accuracy puede resultar insuficiente, ya que un modelo podría obtener un valor aparentemente alto al clasificar correctamente la mayoría de transacciones legítimas, pero fallar en la identificación de los casos fraudulentos. Por esta razón, la evaluación debe complementarse con Precision, Recall, F1-Score, ROC-AUC y PR-AUC (Wijaya et al., 2024) y (Breskuvienė & Dzemyda, 2024).

2.1.7.1 Precision

Precision indica qué proporción de las transacciones identificadas como fraudulentas realmente son casos de fraude. Un valor elevado refleja una menor presencia de falsos positivos entre las predicciones clasificadas como positivas. Una falsa alerta puede generar costos operativos o afectar la experiencia del cliente (Al-dahasi et al., 2025).

La Precision se define de la siguiente manera (Bishop, 2006):

$$Precision = \frac{TP}{TP + FP}$$

Donde:

- TP representa los verdaderos positivos, es decir, las transacciones fraudulentas correctamente identificadas
- FP corresponde a los falsos positivos, que son transacciones legítimas clasificadas erróneamente como fraudulentas.

En detección de fraude financiero, un valor alto de Precision indica que el modelo genera pocas falsas alarmas al marcar una transacción como fraudulenta. Sin embargo, esta métrica no debe analizarse de forma aislada, ya que un modelo podría ser muy preciso al detectar fraude, pero dejar sin identificar una cantidad importante de transacciones fraudulentas reales. Por ello, Precision suele interpretarse junto con Recall y F1-Score para obtener una evaluación más equilibrada del desempeño del modelo (Kabane, 2024).

2.1.7.2 Recall

Recall permite medir la capacidad que tiene un modelo para detectar correctamente las transacciones fraudulentas presentes dentro del conjunto de datos. Representa la proporción entre las transacciones fraudulentas correctamente detectadas por el modelo y el total real de transacciones fraudulentas existentes dentro del conjunto de datos (Bishop, 2006).

Su expresión matemática se representa mediante la siguiente ecuación

$$Recall = \frac{TP}{TP + FN}$$

Donde:

- TP corresponde a los casos fraudulentos que fueron correctamente clasificados.
- FN corresponde a los casos fraudulentos que el algoritmo no logró detectar y fueron clasificadas incorrectamente como operaciones legítimas.

En problemas caracterizados por conjuntos de datos altamente desbalanceados, optimizar Recall suele ser prioritario, ya que una transacción fraudulenta no detectada puede traducirse directamente en pérdidas económicas para la institución financiera. Por esta razón, esta métrica suele considerarse uno de los indicadores más relevantes en modelos orientados a la detección de fraude financiero (Kabane, 2024).

2.1.7.3 F1-Score

La métrica F1-Score permite evaluar el desempeño general de un modelo de clasificación mediante la combinación de Precision y Recall en un único indicador. Consiste en ofrecer

una medida equilibrada entre la capacidad del modelo para identificar correctamente transacciones fraudulentas y la precisión con la que realiza dichas predicciones.

En muchos casos, analizar únicamente Precision o Recall de forma individual puede generar interpretaciones incompletas sobre el comportamiento real del modelo. Un algoritmo podría detectar una gran cantidad de fraudes, pero al mismo tiempo generar demasiadas falsas alarmas, o, por el contrario, clasificar correctamente pocas transacciones fraudulentas, aunque con alta precisión. Debido a ello, F1-Score permite evaluar simultáneamente ambas métricas.

F1-Score se calcula a partir de la media armónica de Precision y Recall, como se muestra en la siguiente ecuación (Bishop, 2006):

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Esta formulación permite penalizar escenarios donde exista un desequilibrio considerable entre ambas métricas. Como resultado, un F1-Score elevado indica que el modelo mantiene un mejor equilibrio entre la detección de fraudes y la reducción de falsas alarmas.

2.1.7.4 ROC-AUC

ROC-AUC permite evaluar la capacidad del modelo para diferenciar las transacciones legítimas de las fraudulentas bajo distintos umbrales de decisión. A diferencia de métricas como Precision o Recall, que analizan una única predicción final, ROC-AUC permite observar el comportamiento global del modelo. Evalúa la capacidad de discriminación del modelo mediante el área situada bajo la curva ROC (Fawcett, 2006).

La curva ROC (Receiver Operating Characteristic) representa gráficamente la relación entre la tasa de verdaderos positivos y la tasa de falsos positivos en diferentes umbrales de clasificación. Estas relaciones pueden expresarse mediante las siguientes ecuaciones (Fawcett, 2006):

Tasa de verdaderos Positivos (True Positive Rate - TPR):

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

Donde:

- TP representa las transacciones fraudulentas correctamente detectadas
- FN corresponde transacciones fraudulentas que el modelo no logró detectar

Tasa de Falsos Positivos (False Positive Rate - FPR):

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

- FP representa operaciones legítimas clasificadas erróneamente como fraude
- TN corresponde a transacciones legítimas correctamente clasificadas.

Un valor cercano a 1 indica que el algoritmo posee una elevada capacidad para diferenciar entre fraude y transacciones legítimas, mientras que valores cercanos a 0.5 reflejan un comportamiento similar a una clasificación aleatoria, sin capacidad predictiva confiable.

2.1.7.5 PR-AUC

PR-AUC (Area Under the Precision-Recall Curve) evalúa la relación entre Precision y Recall a medida que varía el umbral de clasificación. Un umbral más bajo puede favorecer la detección de fraudes, sin embargo, también incrementa las falsas alertas. Por otro lado, un umbral más alto puede reducir los falsos positivos, pero dejar más casos fraudulentos sin identificar.

El área bajo la curva indica si el modelo mantiene un equilibrio favorable entre ambas métricas. Un valor elevado refleja una mayor capacidad para detectar la clase positiva sin perder considerablemente la confiabilidad de las alertas. Este resultado no representa un porcentaje directo de aciertos, sino el comportamiento del clasificador frente a diferentes umbrales.

PR-AUC resulta útil en conjuntos de datos desbalanceados porque concentra el análisis en la clase positiva. En la detección de fraude, permite valorar si el incremento de los casos identificados ocasiona una cantidad excesiva de falsas alertas. Complementa la información proporcionada por ROC-AUC y F1-Score (Saito & Rehmsmeier, 2015).

2.1.7.6 Matriz de confusión

La matriz de confusión es una herramienta utilizada para la comparación entre las clases reales y las predicciones realizadas por el algoritmo. Su principal finalidad consiste en identificar la cantidad de clasificaciones correctas e incorrectas obtenidas por el modelo (Bishop, 2006).

En problemas de detección de fraude financiero, la matriz de confusión permite analizar el comportamiento del modelo al clasificar transacciones legítimas y transacciones fraudulentas. A partir de esta matriz es posible identificar cuatro resultados fundamentales:

- Verdaderos positivos (TP): Transacciones fraudulentas correctamente detectadas.
- Verdaderos negativos (TN): Transacciones legítimas correctamente clasificadas.
- Falsos positivos (FP): Cuando una transacción legítima es clasificada erróneamente como fraudulenta.
- Falsos negativos (FN): Transacciones fraudulentas que el modelo no logra identificar.

A continuación, se muestra la estructura general de la matriz de confusión.

Tabla 1. Estructura general de una matriz de confusión.

	Legítima Predicción	Fraude Predicción
Transacción Legítima	TN	FP
Fraude	FN	TP

Nota. Elaboración propia

A partir de los valores contenidos en esta matriz se calculan las principales métricas utilizadas para evaluar modelos de clasificación, entre ellas Precision, Recall y F1-Score. Además, facilitan el análisis de los errores cometidos por el modelo, particularmente las falsas alarmas y los casos de fraude que no fueron identificados.

2.1.8 Stratified 5-Fold Cross-Validation

La validación cruzada estratificada de cinco pliegues es un procedimiento empleado para evaluar un modelo mediante distintas particiones de los datos. El conjunto se distribuye en cinco grupos y, en cada iteración, cuatro se destinan al entrenamiento y uno a la validación. Este proceso se repite cinco veces, de modo que cada pliegue participa una vez en la evaluación del modelo (Kohavi, 1995).

La estratificación procura que cada uno de los cinco pliegues conserve aproximadamente el mismo porcentaje de transacciones legítimas y fraudulentas que existe en el conjunto

original. Este procedimiento no balancea los datos, sino que mantiene en los pliegues una distribución similar a la observada en el conjunto original.

Las métricas obtenidas se resumen mediante su media y desviación estándar. La media representa el desempeño promedio, mientras que la desviación estándar permite examinar la variación de los resultados entre las particiones. En la detección de fraude, este procedimiento facilita la comparación de modelos sin depender de una sola división de los datos y ha sido empleado para evaluar algoritmos aplicados a transacciones financieras (Niu et al., 2019).

2.1.9 Matriz de costos en la detección de fraude

La matriz de costos relaciona los verdaderos positivos, falsos positivos, falsos negativos y verdaderos negativos con sus posibles efectos financieros. Una alerta, verdadera o falsa, puede incorporar el costo administrativo de analizar la operación. Por otro lado, cuando una transacción fraudulenta no es identificada, el costo puede representarse mediante el monto de la transacción. Este enfoque permite reconocer que la pérdida asociada con un falso negativo varía según cada operación y no necesariamente corresponde a un valor constante (Correa Bahnsen et al., 2016).

Tabla 2. Matriz conceptual de costos para la detección de transacciones fraudulentas

Clase real	**Predicción del modelo**	**Resultado**	**Consecuencia considerada**	**Costo asociado**
Fraudulenta	Fraude	Verdadero positivo	Fraude detectado que requiere revisión	Costo de investigación
Legítima	Fraude	Falso positivo	Falsa alerta que requiere revisión	Costo de investigación
Fraudulenta	Legítima	Falso negativo	Fraude no detectado	Monto de la transacción
Legítima	Legítima	Verdadero negativo	Operación procesada normalmente	Sin costo adicional

Nota. Adaptado de (Correa Bahnsen et al., 2016)

Este enfoque permite comparar los modelos considerando simultáneamente la pérdida potencial por fraudes no detectados y el esfuerzo requerido para revisar las alertas. En consecuencia, un modelo con mejores métricas estadísticas no siempre genera el menor

costo, debido a que el resultado también depende del monto de las operaciones omitidas y del número de alertas producidas.

2.2. Metodología

2.2.1 Diseño de investigación

La presente investigación tiene un enfoque cuantitativo, debido a que se basa en el análisis de datos transaccionales y en la evaluación numérica del desempeño de modelos predictivos para detectar transacciones fraudulentas en tarjetas de crédito.

El estudio es de tipo aplicado, no se limita únicamente a describir las características del dataset, sino que propone el desarrollo, evaluación y comparación de modelos predictivos capaces de clasificar transacciones como legítimas o fraudulentas.

Además, la investigación presenta un diseño experimental ya que implementa 4 algoritmos de clasificación supervisada: Regresión Logística, Random Forest, XGBoost LightGBM. Cada modelo se evaluó bajo dos escenarios de entrenamiento: con la distribución original de las clases y con datos balanceados mediante SMOTE. Además, se realizó una optimización sistemática de sus hiperparámetros.

Las configuraciones optimizadas se compararon mediante métricas para datos desbalanceados, validación cruzada estratificada y un análisis económico de los errores de clasificación. Finalmente, el modelo seleccionado se evaluó sobre el conjunto de pruebas reservado.

2.2.2 Metodología CRISP-ML(Q)

Se empleó la metodología CRISP-ML(Q) (Cross-Industry Standard Process for Machine Learning with Quality Assurance). Esta metodología amplía el alcance de CRIPS-DM al incorporar actividades de garantía de calidad y etapas como el despliegue, monitoreo y mantenimiento de los modelos.

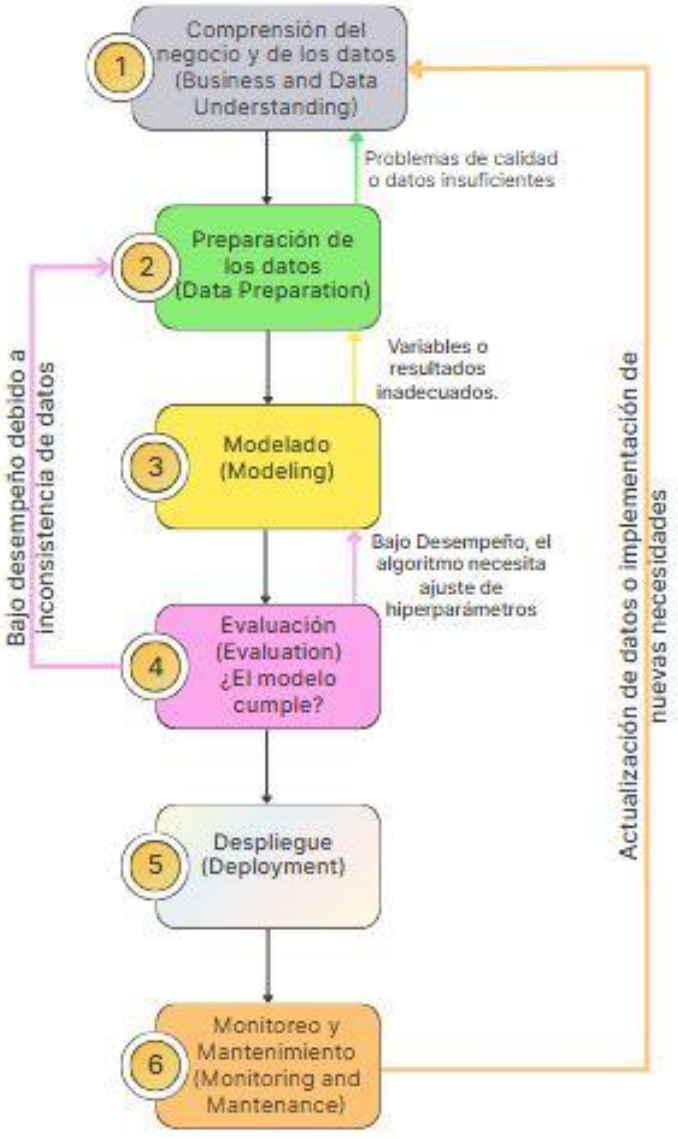
CRISP-ML(Q) organiza el proyecto en las fases de comprensión del negocio y de los datos, preparación de los datos, modelado, evaluación, despliegue y monitoreo. Además, facilita la planificación de las actividades, la trazabilidad entre los objetivos de investigación y los modelos desarrollados (Studer et al., 2021).

En la presente investigación se desarrollaron las cuatro primeras fases, debido a que el objetivo consiste en diseñar, entrenar y evaluar modelos predictivos para la detección de

transacciones fraudulentas en tarjetas de crédito. Las fases de despliegue, monitoreo y mantenimiento no forma parte del alcance del estudio. Sin embargo, se describirán como referencia para una posible aplicación posterior.

Esta metodología es iterativa, permitiendo regresar a fases previas cuando los resultados de la evaluación requieren realizar ajustes en la preparación de los datos o en el proceso de modelado. La Figura 2 presenta las fases que conforman la metodología CRISP-ML(Q).

Figura 2. Fases de la metodología CRISP-ML(Q)



Fuente: Elaboración propia a partir del modelo de procesos CRISP-ML(Q) descrito en la literatura (Studer et al., 2021)

2.2.3 Comprensión del negocio y los datos

El problema se centra en la detección de transacciones fraudulentas realizadas con tarjetas de crédito, considerando que el número reducido de operaciones fraudulentas respecto al total de transacciones, representa un desafío para la construcción de modelos predictivos.

A partir de este análisis el problema fue formulado como una tarea de clasificación binaria supervisada. La variable objetivo, is_fraud, cuyo valor 0 identifica una transacción legítima y el valor 1 representa una operación fraudulenta. El alcance del estudio comprendió el desarrollo y comparación de modelos de Regresión Logística, Random Forest, XGBoost y LightGBM bajo escenarios con la distribución original de las clases y con la aplicación de SMOTE.

Como criterio de éxito se estableció la selección de una configuración capaz de reducir los fraudes no detectados y la pérdida financiera potencial, manteniendo controladas las falsas alertas y conservando un desempeño predictivo competitivo.

Para ello, se consideraron errores de clasificación y costos asociados con las decisiones del modelo. Los procedimientos específicos de optimización, validación y evaluación se describen en los apartados posteriores.

Para el desarrollo experimental se utilizó el conjunto de datos público Credit Card Fraud Detection, disponible en Kaggle (Bhadouria, n.d.). Este está conformado por los archivos fraudTrain.csv y fraudTest.csv, cuyas características generales se presentan en la Tabla a continuación.

Tabla 3. Características generales de los archivos utilizados.

Dataset	Registros	Columnas	Transacciones legítimas	Transacciones fraudulentas	Fraude (%)
fraudTrain.csv	1296675	23	1289169	7506	0.5789
fraudTest.csv	555719	23	553574	2145	0.3860

Nota.: Elaboración propia. a partir del conjunto de datos Credit Card Fraud Detection.

Ambos archivos fueron cargados en el entorno de desarrollo Google Colab utilizando la biblioteca Pandas. La inspección inicial incluyó la verificación de sus dimensiones, nombres de columnas, tipos de datos, variable objetivo, valores faltantes, registros duplicados y consistencia general de la información. También se revisó la distribución de las clases para determinar la magnitud del desbalance existente.

Tabla 4. Descripción de las Variables del conjunto de datos

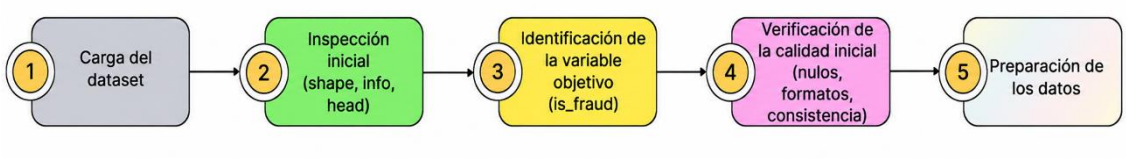
Número de Item	Variable	Tipo	Descripción
1	$Unnamed: 0$	int64	Número de ítem. No representa una característica propia de la transacción.
2	$trans_date_trans_time$	object	Fecha y hora en que se realizó la transacción.
3	cc_num	int64	Número de la tarjeta de crédito del cliente.
4	$merchant$	object	Comercio donde se efectuó la transacción.
5	$category$	object	Categoría comercial del establecimiento.
6	amt	float64	Monto de la transacción.
7	$first$	object	Nombre del titular de la tarjeta.
8	$last$	object	Apellido del titular de la tarjeta.
9	$gender$	object	Género del titular.
10	$street$	object	Dirección del titular.
11	$city$	object	Ciudad de residencia del titular.
12	$state$	object	Estado de residencia del titular.
13	zip	int64	Código postal del titular.
14	lat	float64	Latitud del domicilio del titular.
15	$long$	float64	Longitud del domicilio del titular.
16	$city_pop$	int64	Población de la ciudad de residencia.
17	job	object	Profesión del titular de la tarjeta.
18	dob	object	Fecha de nacimiento del titular.
19	$trans_num$	object	Identificador único asignado a cada transacción.
20	$unix_time$	int64	Marca temporal de la transacción en formato Unix.

21	$merch_lat$	float64	Latitud del comercio.
22	$merch_long$	float64	Longitud del comercio.
23	is_fraud	int64	Variable objetivo que indica si la transacción es fraudulenta (1) o legítima (0).

Nota.: Elaboración propia a partir del dataset Credit Card Fraud Detection

Esta revisión permitió confirmar que el conjunto de datos era adecuado para desarrollar un problema de clasificación supervisada y definir las actividades que se ejecutarían durante la fase de preparación de los datos. Las características técnicas del conjunto de datos se describen más adelante.

Figura 3. Proceso de comprensión inicial del conjunto de datos realizado en Google Colab.



Nota. Elaboración propia.

2.2.4 Preparación de los datos.

2.2.4.1 Limpieza y transformación

La etapa de limpieza y transformación, nos orienta a eliminar información que consideramos no relevante para el estudio, además se transformaron variables y adecuó el conjunto de datos al formato de cada algoritmo de clasificación.

Tabla 5. Actividades realizadas durante la etapa de limpieza y transformación.

Actividades	Variables	Objetivo
Eliminación de Variables	$Unnamed: 0$ cc_num $first$ $last$ $street$ $trans_num$ zip	Reducir la dimensionalidad del conjunto de datos. Se eliminaron 7 variables correspondientes a datos personales e identificadores que no aportan capacidad predictiva al modelo.
Transformación de variables temporales		Las variables de fecha y hora fueron procesadas para

	$trans_date_trans_time$	generar nuevas características relacionadas con el comportamiento temporal de las transacciones.
Ingeniería de características	dob	En base a esta variable, se calcula una nueva "age", obteniendo la edad del titular de la tarjeta.
Ingeniería geográfica	lat $long$ $merch_lat$ $merch_long$	Calcular la distancia entre el cliente y el comercio donde se realizó la transacción.
Eliminación de variables redundantes, las originales	$trans_date_trans_time$ dob	Una vez generadas las nuevas características, se eliminaron las variables originales que dejaron de aportar información al proceso de modelado.

2.2.4.2 Selección y adecuación de variables predictoras

Una vez terminada la fase de limpieza y transformación, se revisó la estructura de las variables, con el propósito de conservar información relevante que ayude en la predicción y evitar la transformación innecesaria de variables categóricas.

En primer lugar, se realizó un análisis de cardinalidad, en el cual se expresa la cantidad de categorías distintas presentes en una variable. Si utilizamos One Hot Encoding en categorías con alta cardinalidad, esto provocaría un consumo excesivo de memoria, prolongación del entrenamiento, generación de variables.

A continuación, se muestra una tabla con el resumen del proceso de selección y adecuación de variables predictoras.

Tabla 6. Análisis de cardinalidad y tratamiento de las variables

Variables involucradas	Números de categorías	Tratamiento	Justificación metodológica
$gender$	2	Conservada y codificada	Presenta una cardinalidad mínima, por lo que su codificación mediante One-Hot Encoding con $drop_first = True$, genera una variable binaria. Se conserva información demográfica sin aumentar de forma relevante la dimensionalidad. Además, puede ser una variable influyente dentro de los modelos.
$category$	14	Conservada y codificada	Representa el tipo de actividad comercial asociada a la transacción. Su cardinalidad es moderada y permite dar seguimiento al comportamiento entre categorías sin producir una expansión excesiva del conjunto de datos.
$state$	51	Conservada y codificada	Esta información geográfica, presenta una cantidad de categorías aún manejable frente al tamaño total del conjunto de datos. Su conservación permite captar variaciones territoriales, lo cual se considera de importancia.
job	494	Eliminada	Su codificación habría generado cientos de variables binarias y una matriz dispersa. Además, el análisis auxiliar de importancia por ganancia de LightGBM ocupó la posición 17, con apenas el 0,0053 % de la

			importancia total, lo que indica una contribución predictiva prácticamente nula frente a su elevada cardinalidad.
$merchant$	693	Eliminada	Al tener una elevada cardinalidad, podía hacer que el modelo aprendiera comportamientos particulares de comercios concretos en lugar de patrones generales de fraude. Además, dentro del análisis auxiliar de LightGBM ocupó la posición 15 y representó solamente el 0,0285 % de la importancia total, por lo que su limitada contribución no compensó la complejidad de incorporarla en los modelos.
$city$	894	Eliminada	Fue la variable categórica con mayor cardinalidad. Su codificación habría producido una matriz considerablemente más dispersa y costosa. Por otro lado, dentro de LightGBM ocupó la posición 16, con apenas el 0,0187 % de la importancia total. Además, parte de su información geográfica ya se encontraba representada mediante $state$, lat, $long$, $merch_lat$ y $merch_long$.

Nota. Elaboración propia.

Como análisis complementario para examinar la contribución de las variables categóricas de alta cardinalidad, se entrenó un modelo auxiliar de LightGBM. Este modelo fue ajustado únicamente con las variables predictoras y la variable objetivo is_fraud

procedentes de fraudTrain.csv. El archivo fraudTest.csv no intervino en el entrenamiento del modelo auxiliar ni en la obtención de las importancias.

Para este análisis, LightGBM procesó directamente las variables categóricas mediante su tratamiento nativo y su importancia se estimó mediante el criterio de ganancia ($gain$).

Para eliminar las variables $merchant$, $city$ y job, se consideró su alta cardinalidad, costo computacional y la posibilidad de redundancia en la información. Por ejemplo, la información que contiene $city$ se encuentra parcialmente representada en las variables geográficas.

Además, se consideró que la elevada cardinalidad podría incrementar la complejidad de los modelos y el riesgo de que los modelos aprendieran relaciones demasiado vinculadas con las categorías específicas del conjunto de datos.

Dicha decisión también fue fundamentada en base a la importancia por ganancia del modelo LightGBM auxiliar. Aunque el algoritmo utilizó las tres variables, estas ocuparon las últimas posiciones y su contribución conjunta representó aproximadamente el 0,0525 % de la importancia total.

Por otro lado, las variables $gender$, $category$ y $state$ fueron transformadas mediante One-Hot Encoding. Se utilizó $drop_first = True$ para reducir redundancia en la representación.

Después de la codificación, los conjuntos de entrenamiento y prueba conservan la misma estructura y número de variables.

Finalmente, se separó la variable objetivo is_fraud y se conformaron las matrices de variables predictoras.

2.2.4.3 Identificación del desbalance de clases

El análisis de a variable is_fraud evidenció un desbalance de clases muy evidente dentro del conjunto de entrenamiento. Las transacciones legítimas representan el 99.42 %, mientras que las transacciones fraudulentas representan apenas un 0.57% del total. A partir de este resultado se definieron dos escenarios de modelado: uno conservó la distribución original y otro incorporó SMOTE.

El balanceo no se ejecutó directamente sobre todo el conjunto de entrenamiento durante esta fase, sino que se integró posteriormente dentro de los Pipeline de los modelos

correspondientes. Con esto, se pudo controlar el momento en que se generaron las observaciones sintéticas y evitar que los datos de validación o prueba intervinieran.

2.2.5 Modelado

En esta fase se implementaron cuatro algoritmos de clasificación supervisada: Regresión Logística, Random Forest, XGBoost y LightGBM. Cada algoritmo fue evaluado bajo dos escenarios: con la distribución original de las clases y mediante la aplicación de SMOTE. De esta manera, se obtuvieron ocho configuraciones experimentales.

Para el desarrollo del proyecto se organizaron dos etapas. Primero, se construyeron pipelines para integrar de manera ordenada las transformaciones requeridas, el tratamiento del desbalance con SMOTE en los escenarios correspondientes y el algoritmo asignado.

Dentro de la segunda etapa, se realizó una búsqueda sistemática de hiperparámetros. Para cada algoritmo se evaluó un conjunto acotado de configuraciones y se utilizó PR-AUC como criterio para identificar la alternativa con mejor desempeño de validación. Las actividades desarrolladas en ambas etapas se describen en los apartados siguientes.

2.2.5.1 Construcción de pipelines y escenarios experimentales.

Se construyeron ocho pipelines mediante la biblioteca imbalanced-learn, correspondientes a los cuatro algoritmos evaluados, con la distribución original y con la aplicación de SMOTE. Esto permitió mantener la misma secuencia de procesamiento durante la optimización de hiperparámetros y la evaluación de los modelos.

Se debe aclarar que la optimización y la evaluación interna de estas configuraciones se realizaron exclusivamente con los datos procedentes de fraudTrain.csv, mientras que fraudTest.csv se mantuvo reservado para la evaluación final.

En la Regresión Logística se incorporó StandardScaler, debido a que este algoritmo es sensible a la diferencia de escalas entre las variables. Para los escenarios donde se usa SMOTE, el escalado se efectuó antes de generar las muestras sintéticas.

Random Forest, XGBoost y LightGBM no requirieron escalado, puesto que su aprendizaje se basa en árboles de decisión. Por esta razón, sus pipelines estuvieron compuestos únicamente por el modelo en el escenario original y por SMOTE seguido del modelo en el escenario balanceado. La Tabla siguiente resume la estructura utilizada.

Tabla 7. Estructura de los pipelines para los escenarios experimentales.

Algoritmo	Distribución original	Uso de SMOTE
Regresión Logística	Escalado – Regresión Logística	Escalado – SMOTE – Regresión Logística
Random Forest	Random Forest	SMOTE - Random Forest
XGBoost	XGBoost	SMOTE - XGBoost
LightGBM	LightGBM	SMOTE - LightGBM

Nota. Elaboración propia

Dentro de cada pliegue, los datos se dividieron primero en las particiones de entrenamiento y validación. Posteriormente, SMOTE se aplicó únicamente a la partición utilizada para entrenar el modelo, mientras que los datos de validación conservaron su distribución original.

Las ocho configuraciones utilizaron las mismas variables predictoras, las mismas divisiones estratificadas y la semilla random_state=42 cuando el algoritmo lo permitió. Además, en cada iteración se creó una copia independiente del pipeline para evitar la reutilización de modelos previamente entrenados.

2.2.5.2 Optimización de hiperparámetros mediante GridSearchCV

Para evitar que la comparación dependiera únicamente de configuraciones predeterminadas, se realizó una búsqueda sistemática acotada mediante GridSearchCV. Los hiperparámetros que controlan aspectos del aprendizaje deben definirse antes del entrenamiento del modelo, como la intensidad de regularización, la cantidad de árboles o su nivel de complejidad.

La optimización se efectuó sobre fraudTrain.csv. Cada configuración de hiperparámetros fue evaluada mediante validación cruzada estratificada de tres pliegues, utilizando las mismas divisiones y la semilla random_state=42.

En cada iteración, dos pliegues se destinaron al entrenamiento y el restante a la validación. En los escenarios balanceados, SMOTE permaneció dentro del pipeline y se aplicó solamente a los dos pliegues utilizados para entrenar.

Como criterio de optimización se usó PR-AUC debido al desbalance de la variable objetivo. Para cada alternativa se calculó el PR-AUC promedio de validación y se seleccionó la combinación que alcanzó el valor más alto dentro del espacio examinado.

También se conservaron los resultados de entrenamiento, la desviación entre pliegues y la diferencia entre entrenamiento y validación para identificar posibles señales de sobreajuste.

Se empleó el mismo conjunto de alternativas en los escenarios con y sin SMOTE, con el propósito de mantener condiciones comparables. La siguiente Tabla presenta las configuraciones evaluadas.

Tabla 8. Espacios de búsqueda de hiperparámetros.

Algoritmo	**Orden de los hiperparámetros**	**Función**	**Combinaciones evaluadas**
Regresión Logística	(c)	Controla la intensidad de la regularización.	(0.1); (1); (10)
Random Forest	$(n_estimators,$ max$_depth,$ $min_samples_leaf)$	Cantidad de árboles Profundidad de árboles Número mínimo de registros por hoja.	(100, None, 1); (100, 20, 5); (150, 20, 5)
XGBoost	$(n_estimators,$ $learning_rate,$ max$_depth,$ $subsample)$	Cantidad de árboles Regula la contribución de cada árbol. Profundidad de árboles Proporción de registros utilizada en cada iteración.	(100, 0.3, 6, 1.0); (100, 0.1, 3, 1.0); (200, 0.1, 6, 0.8); (200, 0.05, 6, 1.0); (300, 0.05, 3, 0.8)
LightGBM	$(n_estimators,$ $learning_rate,$ $num_leaves)$	Cantidad de árboles Regula la contribución de cada árbol. Número máximo de hojas.	(100, 0.1, 31); (100, 0.1, 15); (200, 0.05, 31); (200, 0.05, 15); (200, 0.1, 63)

Nota. En la Regresión Logística se mantuvieron $solver = lbfgs$ y penalización L2. Las alternativas fueron delimitadas considerando las características de cada algoritmo, el tamaño del conjunto de datos y los recursos computacionales disponibles. Elaboración propia.

GridSearchCV se configuró para evaluar una alternativa a la vez y no reajustar automáticamente el modelo después de la búsqueda. De esta manera, su función se limitó a identificar los mejores hiperparámetros.

Una vez seleccionadas las configuraciones, los ocho pipelines optimizados fueron reconstruidos y evaluados posteriormente mediante validación cruzada estratificada de cinco pliegues.

2.2.6 Evaluación y selección de modelos

Una vez identificados los mejores hiperparámetros para cada algoritmo, se reconstruyeron los ocho pipelines optimizados y se evaluó su desempeño. Esta fase tuvo como propósito comparar la capacidad predictiva de los cuatro algoritmos bajo los escenarios con y sin SMOTE, utilizando la información de fraudTrain.csv.

La evaluación comprendió tres actividades. Primero, se aplicó validación cruzada estratificada de cinco pliegues para obtener métricas de desempeño y errores de clasificación.

Estos resultados se complementaron con una evaluación económica orientada a estimar las consecuencias de las alertas generadas y de los fraudes no detectados. Finalmente, la configuración seleccionada se entrenó con la totalidad del conjunto de entrenamiento y se evaluó una sola vez sobre fraudTest.csv.

2.2.6.1 Validación cruzada estratificada de cinco pliegues

Las ocho configuraciones optimizadas se evaluaron mediante validación cruzada estratificada de cinco pliegues sobre fraudTrain.csv. Para ello, los datos fueron divididos en cinco particiones que conservaron aproximadamente la proporción original de transacciones legítimas y fraudulentas. En cada iteración se utilizaron cuatro pliegues para entrenar el pipeline y el restante para validarlo.

Las mismas divisiones fueron utilizadas en todas las configuraciones, y se establecieron: shuffle=True y random_state=42 para garantizar la reproductibilidad del proceso. En cada iteración se creó una copia independiente del pipeline optimizado. Cuando la configuración incluía SMOTE, las muestras sintéticas se generaron únicamente a partir de los cuatro pliegues de entrenamiento; el pliegue de validación mantuvo la distribución original de las clases.

Para obtener las clases predichas se empleó un umbral de decisión de 0.5. A partir de estas predicciones se calcularon Accuracy, Precision, Recall y F1-Score. Además de los verdaderos negativos, falsos positivos, falsos negativos y verdaderos positivos. ROC-

AUC y PR-AUC se calcularon utilizando las probabilidades asignadas a la clase fraudulenta.

Los resultados de los cinco pliegues se resumieron mediante el promedio y la desviación estándar de cada métrica. Además, se acumularon los componentes de las matrices de confusión y los montos asociados con los fraudes detectados y no detectados. Esta información se utilizó posteriormente para comparar las configuraciones y desarrollar la matriz de costos.

2.2.6.2 Evaluación económica y criterio de selección

La evaluación predictiva se complementó con un análisis económico ya que los errores de clasificación no generan las mismas consecuencias. Es decir que, una falsa alerta requiere la revisión de una transacción legítima, mientras que un fraude no detectado puede ocasionar una pérdida equivalente al monto de la operación (Bahnsen et al., 2013).

En base a esto, se consideró que las consecuencias de un error dependen tanto del tipo de predicción como del monto de la transacción involucrada. Para ello, se utilizaron las matrices de confusión y los montos registrados en la variable amt.

Los verdaderos positivos y falsos positivos se les asignó un costo administrativo por revisión. En los falsos negativos, la pérdida financiera potencial correspondió al monto de la transacción fraudulenta no detectada, mientras que los verdaderos negativos no generaron un costo adicional. Los montos asociados con los verdaderos positivos se interpretaron como fraude potencialmente evitado.

El costo total estimado para cada configuración se calculó mediante la siguiente expresión:

$$C_{Total} = \sum Monto\ de\ falsos\ negativos_i + C_{unitario}(FP + TP)$$

Donde:

- X_i representa el costo unitario de investigar una alerta
- $FP + TP$ representan el total de alertas generadas.

El conjunto de datos contiene el monto de las transacciones (amt), pero no especifica cuánto cuesta investigar una alerta. Por esta razón, se plantearon tres escenarios hipotéticos de 1, 5 y 10 por investigación.

El ahorro estimado se obtuvo al restar el costo total del monto acumulado de las transacciones fraudulentas. Además, se examinó qué configuración presentó el menor costo dentro de cada pliegue y escenario, con el propósito de comprobar la estabilidad de la comparación económica y reducir la influencia de transacciones aisladas de alto valor.

La selección no se basó en una única métrica. Se priorizó la reducción de fraudes no detectados y de la pérdida financiera potencial, procurando mantener controladas las falsas alertas y conservar valores competitivos de Precision, Recall, F1-Score y PR-AUC.

2.2.6.3 Evaluación final sobre el conjunto de prueba reservado

Después de seleccionar la configuración con el mejor equilibrio entre desempeño predictivo y costo estimado, se construyó nuevamente su pipeline con los hiperparámetros determinados mediante GridSearchCV. El modelo se entrenó con la totalidad del archivo fraudTrain.csv.

Después se procedió a evaluar sobre fraudTest.csv. Este conjunto se mantuvo separado durante la optimización de hiperparámetros, la validación cruzada y la selección del modelo, y conservó la distribución original de las clases.

La evaluación final incluyó Accuracy, Precision, Recall, F1-Score, ROC-AUC, PR-AUC y la matriz de confusión. También se calcularon la pérdida financiera potencial y el costo total estimado para los escenarios de 1, 5 y 10 unidades monetarias por alerta investigada.

2.2.7 Fase de despliegue

La fase de despliegue no se limita a la simple ejecución del algoritmo. Consiste en integrar el modelo predictivo dentro de un entorno operativo, incluyendo dentro de un mismo flujo, el pre-procesamiento de datos, la generación y selección de características, el modelo predictivo en sí y la presentación de los resultados (Studer et al., 2021).

Es decir que el despliegue supondría conectar nuestro modelo al sistema de procesamiento de transacciones dentro de una entidad financiera. Cada operación en tiempo real sería sometida a todo el tratamiento de datos realizado en la etapa de entrenamiento y posteriormente generaría una probabilidad de fraude. Esta debería

compararse con un umbral de decisión establecido por la entidad financiera para determinar si la transacción requiere de una alerta o puede continuar con el flujo normal.

También implica definir una arquitectura, comprobar compatibilidad con los sistemas existentes, y en base a esto, establecer una estrategia de implementación y evaluación del comportamiento del modelo bajo condiciones reales de producción. (Studer et al., 2021).

Existen aspectos que no pueden determinarse únicamente con las métricas obtenidas sobre el conjunto de pruebas, como: tiempos de respuesta, cantidad de transacciones procesadas simultáneamente, disponibilidad del servicio, políticas de privacidad, impacto operativo que está dispuesto a aceptar la entidad financiera respecto a falsos positivos y falsos negativos. Los mecanismos de contingencia frente a errores, interrupciones o predicciones anómalas.

En base a esto, dentro de esta investigación no se contó con acceso a la infraestructura tecnológica de una entidad financiera o a un sistema real de procesamiento de pagos. Tampoco se obtuvo información sobre políticas de seguridad y aspectos ya mencionados antes. Por consiguiente, no se reunieron las condiciones necesarias para integrar y validar el modelo dentro de un sistema bancario en operación.

Por ende, se presenta la fase únicamente de manera metodológica y no ejecutada.

2.2.8 Fase de monitoreo y mantenimiento

El monitoreo inicia una vez que el modelo haya sido desplegado y empiece hacer predicciones sobre nuevos datos. Esta fase permite verificar que su desempeño se mantenga dentro de los criterios establecidos inicialmente.

Esta fase no pudo realizarse en el presente estudio porque el conjunto de datos empleado es estático y la evaluación no se desarrolló dentro de un entorno productivo real. Es decir que no hubo un periodo de operación durante el cual pudiera medirse la evolución del desempeño. Sin estas condiciones, no es posible detectar cambios en los datos, degradación predictiva o necesidad de reentrenamiento. En consecuencia, no sería metodológicamente correcto afirmar que el modelo fue monitoreado.

Sin embargo, se describirá de manera teórica.

Dentro de un sistema de detección de fraude, los patrones transaccionales iniciales no necesariamente permaneces estables, pueden aparecer nuevas estrategias de fraude,

modificación de hábitos de consumo, incorporación de nuevos comercios o variaciones geográficas. Estos cambios pueden generar una desviación entre la distribución de datos de entrenamiento y los datos futuros.

El monitoreo requeriría registrar continuamente las variables de entada, las probabilidades generadas, las decisiones tomadas y las etiquetas reales confirmadas. A partir de esta información se debería controlar (Studer et al., 2021):

- Cambios en la distribución de las variables de entrada.
- Variaciones en Precisión, Recall, F1-Score, ROC-AUC y PR-AUC.
- Incrementos en falsos positivos y falsos negativos
- Incumplimiento de esquemas o formatos esperados de datos.
- Diferencia entre el desempeño actual y el histórico.

Cuando el monitoreo identifica algún cambio significativo dentro de la información que estaba controlando, se inicia el mantenimiento. Este implica una nueva recopilación, etiquetado reciente, revisión de características y ajuste del umbral de clasificación (Studer et al., 2021).

Esta nueva versión debe evaluarse y compararse con la anterior, ya que una incorporación de datos reciente no garantiza una mejora. Por lo que sugiere conservar las versiones anteriores del modelo y mantener la trazabilidad.

2.2.9 Garantía de calidad durante el ciclo CRISP-ML (Q)

La garantía de calidad constituye un componente presente durante todo el proyecto, por lo que no está limitado a la evaluación final del modelo, se incorpora en todas las fases ejecutadas. (Studer et al., 2021).

Dentro de esta investigación, se aplicaron controles orientados a mantener la consistencia de los datos, prevenir la fuga de información, tratar correctamente el desbalance de clases y garantizar una comparación homogénea entre los modelos.

Tabla 9. Controles de garantía de calidad aplicados durante el desarrollo experimental de los modelos.

Fase o proceso	**Riesgo Identificado**	**Control de calidad aplicado**	**Propósito**	**Consecuencia de no aplicar el control**

Comprensión de datos	Desconocimiento de la estructura y distribución del dataset.	Revisión de dimensiones, tipo de variables, valores nulos, estadísticos descriptivos, distribución de la variable objetivo.	Detectar nulos, problemas de calidad, reconocer la existencia del desbalance de clases.	Selección de procesos inadecuados, comprometiendo la validez de los resultados.
Preparación de datos	Presencia de valores faltantes, formatos incompatibles o atributos con escasa utilidad predictiva.	Limpieza, tratamiento de valores ausentes o nulos, transformación, ingeniería y selección de características.	Obtener una base de datos limpia y consistente.	Los algoritmos podrían aprender relaciones poco generalizables, o tener baja capacidad predictiva.
División de datos	Fuga de información entre el entrenamiento y evaluación	Utilización de fraudTrain.csv para entrenar y de fraudTest.csv exclusivamente para evaluar	Mantener de forma independiente: el conjunto de pruebas únicamente para medir la capacidad de generalización. Y el de entrenamiento para preparar el modelo.	Las métricas podrían resultar artificialmente elevadas y no representar el desempeño esperado ante datos no observados.

Tratamiento del desbalance	Contaminación del conjunto de prueba con observaciones sintéticas.	La aplicación de técnicas de balanceo únicamente sobre el conjunto de entrenamiento.	Examinar el efecto del balanceo sin alterar los datos usados en la evaluación.	Aplicar SMOTE antes de la división de datos podría ocasionar una sobreestimación del desempeño.
Modelado comparativo	Desigualdad entre algoritmos y escenarios	Entrenamiento de todos los algoritmos, con y sin SMOTE, utilizando las mismas variables predictoras y el mismo conjunto de prueba.	Asegurar que las diferencias se relacionen directamente al algoritmo y escenario de entrenamiento.	No se podría establecer si las variaciones se deben al modelo, al balanceo o a diferencias en los datos utilizados
Evaluación del desempeño	Interpretación sesgada por el uso aislado de Accuracy	Usar métricas adecuadas como: Recall, F1-Score, ROC-AUC, Matriz de confusión.	Analizar la detección de la clase fraudulenta y el equilibrio entre falsos positivos y falsos negativos.	Un modelo podría considerarse adecuado por su exactitud, pese a presentar limitaciones para identificar fraudes.
Reproductibilidad	Variaciones en los resultados	Se establece el uso de random_state=	Controlar los componentes pseudoaleatori	Los resultados podrían variar después de

	tras varias ejecuciones.	42 en algoritmos que admiten este parámetro.	os y favorecer la repetición del experimento.	cada ejecución.
Trazabilidad	Imposibilidad de reconstruir o verificar el procedimiento.	Conservación del código y resultados. Incluyendo: transformaciones, variables seleccionadas y configuraciones.	Documentar y facilitar una revisión de la evidencia obtenida.	No sería posible determinar cómo se produjeron los resultados.

Nota. Elaboración propia con base en el enfoque de garantía de calidad de CRISP-ML(Q) propuesto por (Studer et al., 2021).

2.3. Resultados y Análisis

En el desarrollo de esta sección, se presentan los resultados obtenidos después del entrenamiento de cada modelo. Se analiza el desempeño de cada algoritmo, así como el efecto de la técnica SMOTE sobre la capacidad de clasificación de los modelos.

2.3.1. Resultados del análisis exploratorio

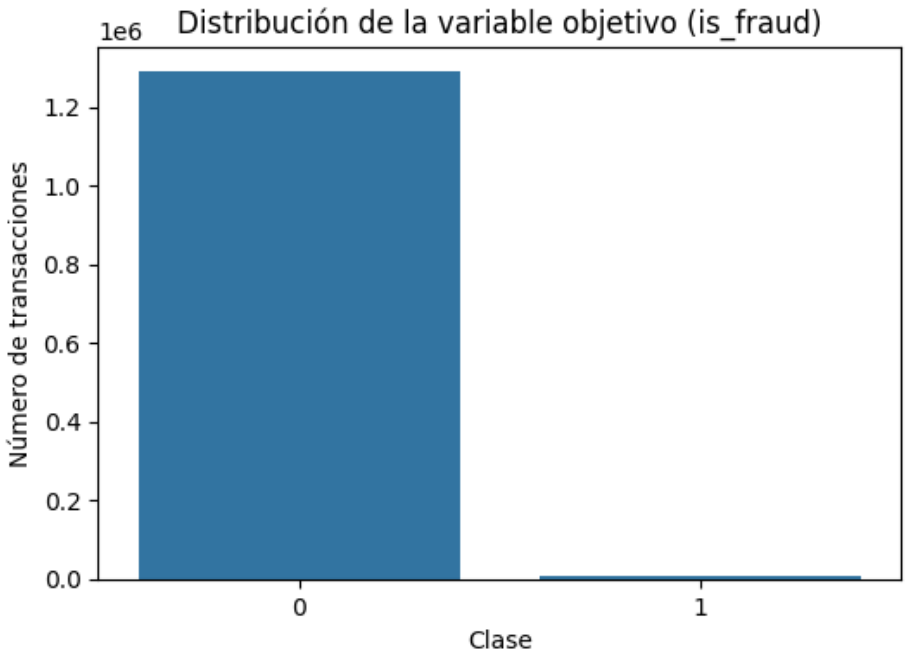
El análisis exploratorio constituye la primera etapa de comprensión de la estructura y el comportamiento del conjunto de datos antes del desarrollo de los modelos predictivos.

Es importante mencionar que durante la fase de exploración no se encontraron valores nulos dentro del dataset.

- **Distribución de la variable objetivo.**

Como primera instancia, se presenta el análisis de la variable objetivo, la cual está definida por el nombre is_fraud.

Figura 4. Distribución de las transacciones según la variable objetivo



Nota. Elaboración propia, la categoría denominada como legítima tiene el valor de 0, mientras que las clasificadas como fraudulentas tienen un valor de 1.

En la Figura 5. Distribución de la variable is_fraud. Se hace evidente un escenario de clasificación altamente desbalanceado, propio de estos datasets. Las transacciones legítimas (0) representan un 99.42% de todo el conjunto, mientras que las transacciones fraudulentas (1) apenas representan un 0.578% del total.

Desde la perspectiva del aprendizaje automático, esta distribución representa un desafío para los modelos predictivos, ya que un algoritmo entrenado directamente con los datos desbalanceados, tiende a favorecer a la clase mayoritaria. En base a este análisis se considera necesario el uso de SMOTE como técnica de balanceo.

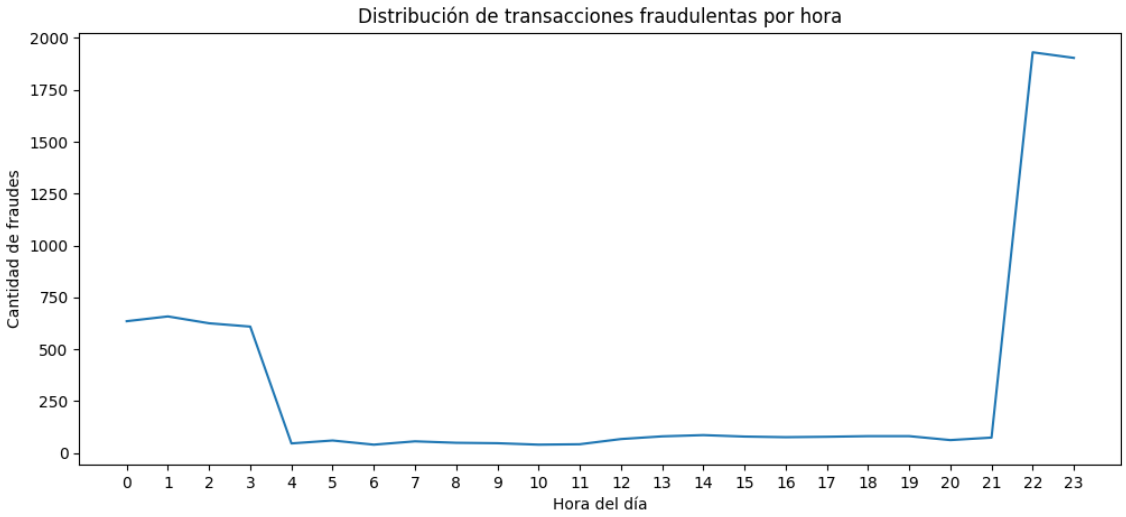
- **Distribución temporal de las transacciones fraudulentas**

La distribución horaria evidencia que las transacciones fraudulentas se concentraron principalmente durante la noche y parte de la madrugada, entre las 10 PM Y 3AM.

A partir de las primeras horas de la mañana (4am), la cantidad de fraudes disminuye y se mantiene durante el resto del día. Este comportamiento muestra la existencia de un patrón temporal y respalda la inclusión de la variable transaction_hour como variable predictora.

La gráfica presenta valores absolutos y no la proporción de fraudes respecto del total de transacciones de cada hora.

Figura 5. Distribución de las transacciones fraudulentas por hora



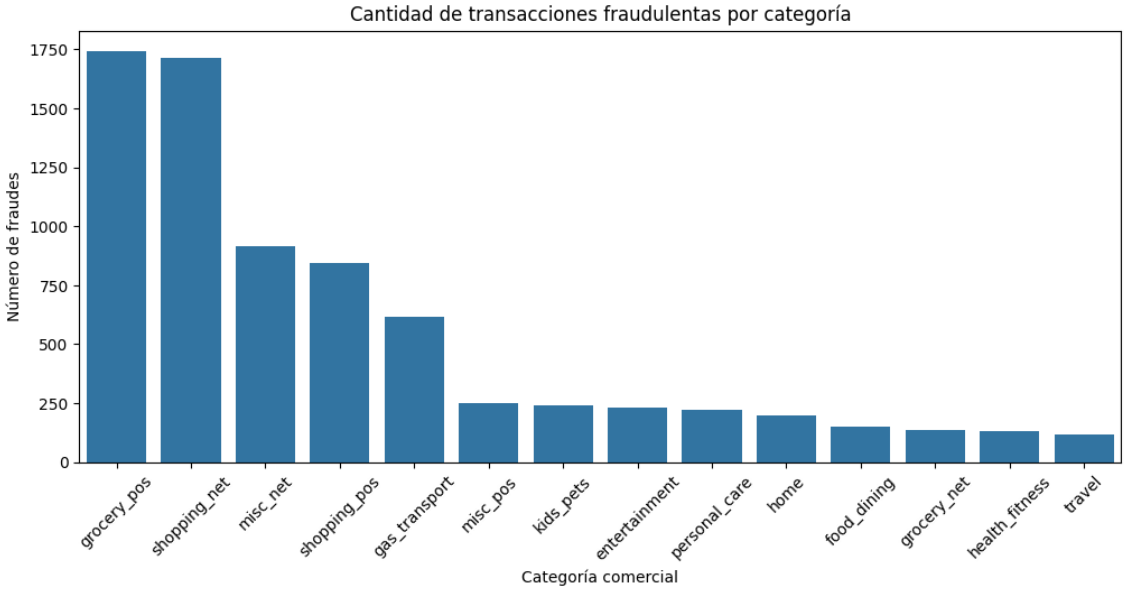
Nota: Elaboración propia a partir del conjunto de datos Credit Card Fraud Detection.

- **Transacciones fraudulentas por categoría comercial**

El análisis realizado por categoría comercial, nos mostró que *grocery_pos* y *shopping_net* concentran las mayores cantidades de transacciones fraudulentas. En contraste, las categorías como *travel*, *health_fitness* y *food_dining* registraron una menor cantidad de fraudes.

Este análisis permite tomar medidas de seguridad frente a estas categorías de comercio.

Figura 6. Cantidad de transacciones fraudulentas por categoría comercial

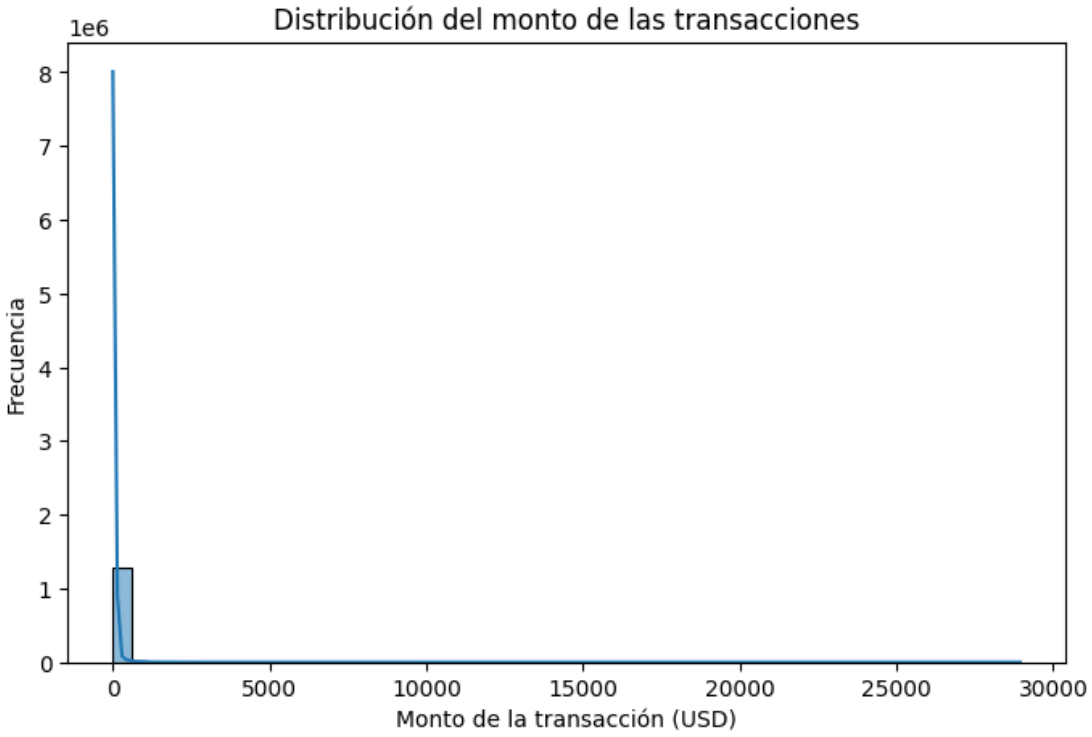


Nota. *Elaboración propia a partir del conjunto de datos* Credit Card Fraud Detection.

- **Distribución del monto de las transacciones**

La mayoría de las transacciones fraudulentas se concentró en valores bajos, mientras que un número reducido registró montos considerablemente elevados. Como se puede observar, el monto no sigue una distribución normal.

Figura 7. Distribución del monto de las transacciones



Nota. *Elaboración propia a partir del conjunto de datos* Credit Card Fraud Detection.

2.3.2 Resultados del pre procesamiento de datos.

El preprocesamiento permitió obtener conjuntos de entrenamiento y prueba con la misma estructura. Las principales actividades realizadas y sus resultados se resumen en la siguiente Tabla.

Tabla 10. Resumen de los resultados del pre-procesamiento de datos

Proceso Realizado	Resultado obtenido
Eliminación de identificadores y datos personales	Se eliminaron $Unnamed$: 0, cc_num, $first$, $last$, $street$, $trans_num$ y zip.
Transformación de variables temporales	A partir de $trans_date_trans_time$ se crearon $transaction_hour$,

	$transaction_day$, $transaction_month$ y $transaction_dayofweek$.
Cálculo de la edad	La variable dob se transformó en age, considerando el año de la transacción y el año de nacimiento del titular.
Eliminación de variables originales	Después de obtener las variables derivadas, se eliminaron $trans_date_trans_time$ y dob para evitar información redundante.
Tratamiento de alta cardinalidad	Se eliminaron $merchan$t, $city$ y job
Codificación de variables categóricas	Las variables $gender$, $category$ y $state$ se transformaron mediante One-Hot Encoding.
Alineación de columnas	Los conjuntos de entrenamiento y prueba fueron alineados para conservar las mismas variables y el mismo orden.

Nota: Elaboración propia a partir del pre-procesamiento aplicado al conjunto de datos Credit Card Fraud Detection.

La incorporación de variables derivadas, permitió ampliar el contexto de cada transacción, mientras que la eliminación de atributos con elevada cardinalidad evitó generar una excesiva cantidad de variables durante la codificación.

Después de aplicar One-Hot Encoding, cada conjunto quedó conformado por 77 columnas: 76 variables predictoras y la variable objetivo is_fraud. Las dimensiones finales y la verificación de calidad se presentan en la siguiente tabla.

Tabla 11. Características finales de los conjuntos de datos.

Característica	**Entrenamiento**	**Prueba**
Número de registros	1296675	555 719
Variables predictoras	76	76
Valores faltantes	0	0
Transacciones legítimas	1289169	553574
Transacciones fraudulentas	7506	2145
Porcentaje de fraude	0,5789 %	0,3860 %

Nota. *Elaboración propia a partir del conjunto de datos* Credit Card Fraud Detection.

Los resultados confirmaron el marcado desbalance de clases. SMOTE no se aplicó dentro de los pipelines correspondientes y unicamente sobre los datos de entrenamiento utilizados en cada iteración. De esta manera, los pliegues de validación y el conjunto de prueba conservaron la distribución original de las clases.

2.3.3 Resultados de la optimización de hiperparámetros

La búsqueda de hiperparámetros se realizó mediante GridSearchCV sobre el conjunto de entrenamiento, utilizando validación cruzada estratificada de tres pliegues. Para cada algoritmo se evaluaron los escenarios con y sin SMOTE. La mejor configuración se seleccionó en base al mayor PR-AUC promedio en los pliegues de validación.

Los hiperparámetros seleccionados se presentan en la Tabla a continuación.

Tabla 12. Hiperparámetros seleccionados mediante GridSearchCV

Modelo	**Balanceo**	**Hiperparámetros seleccionados**
Regresión Logística	Sin SMOTE	$C = 10$
Regresión Logística	Con SMOTE	$C = 0{,}1$
Random Forest	Sin SMOTE	$n_estimators = 100$; $max_depth = None$; $min_samples_leaf = 1$
Random Forest	Con SMOTE	$n_estimators = 100$; $max_depth = None$; $min_samples_leaf = 1$
XGBoost	Sin SMOTE	$n_estimators = 200$; $learning_rate = 0{,}1$; $max_depth = 6$; $subsample = 0{,}8$
XGBoost	Con SMOTE	$n_estimators = 100$; $learning_rate = 0{,}3$; $max_depth = 6$; $subsample = 1{,}0$

LightGBM	Sin SMOTE	$n_estimators = 200$; $learning_rate = 0{,}05$; $num_leaves = 15$
LightGBM	Con SMOTE	$n_estimators = 200$; $learning_rate = 0{,}1$; $num_leaves = 63$

Nota. *Elaboración propia a partir de los resultados obtenidos mediante GridSearchCV.*

Los resultados de PR-AUC obtenidos por las configuraciones seleccionadas se representan en la Tabla 13. La diferencia entre entrenamiento y validación permitió identificar posibles señales de sobreajuste, mientras que la desviación estándar reflejó la estabilidad del desempeño entre los tres pliegues.

Tabla 13. Resultados de las configuraciones seleccionadas durante la optimización

Modelo	**Balanceo**	**PR-AUC entrenamiento**	**PR-AUC validación**	**Desviación estándar**	**Diferencia entrenamiento-validación**
Regresión Logística	Sin SMOTE	0,2460	0,2449	0,0046	0,0011
Regresión Logística	Con SMOTE	0,1760	0,1755	0,0042	0,0005
Random Forest	Sin SMOTE	1	0,9138	0,0027	0,0862
Random Forest	Con SMOTE	1	0,8983	0,0022	0,1017
XGBoost	Sin SMOTE	0,9794	**0,9385**	0,0062	0,0409
XGBoost	Con SMOTE	0,9093	0,8816	0,0023	0,0277
LightGBM	Sin SMOTE	0,9433	0,8863	0,0047	0,0570
LightGBM	Con SMOTE	0,9528	0,9148	0,0072	0,0380

Nota. Los valores corresponden al promedio obtenido mediante validación cruzada estratificada de tres pliegues. Elaboración propia.

XGBoost sin SMOTE alcanzó el mayor PR-AUC de validación durante la optimización, con un valor de 0,9385. Seguido por LightGBM con SMOTE, con 0,9148, y Random Forest sin SMOTE, con 0,9138. Estos resultados muestran que la aplicación de SMOTE no produjo el mismo efecto en todos los algoritmos.

Random Forest obtuvo un PR-AUC de 1 en entrenamiento en ambos escenarios, sin embargo, este valor disminuyó a 0,9138 sin SMOTE y a 0,8983 con SMOTE durante la validación. Este comportamiento es una señal de un sobreajuste en los datos de entrenamiento, aunque la baja desviación entre pliegues indica que su desempeño de validación fue estable.

Por otra parte, XGBoost y LightGBM presentaron diferencias menores entre entrenamiento y validación.

La Regresión Logística registró los valores más bajos de PR-AUC, tanto con la distribución original como con SMOTE. Aunque sus diferencias entre entrenamiento y validación fueron reducidas, esto no representó un mejor desempeño.

Los hiperparámetros seleccionados en esta etapa se incorporaron posteriormente en los ocho pipelines.

2.3.4 Evaluación de los modelos optimizados.

Después de incorporar los hiperparámetros seleccionados mediante GridSearchCV, las ocho configuraciones fueron evaluadas mediante validación cruzada estratificada de cinco pliegues. La Tabla resumen presenta el promedio y la desviación estándar de las métricas obtenidas.

Tabla 14. Desempeño de los modelos optimizados mediante Stratified 5-fold cross-validation

Modelo	**Balanceo**	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**ROC-AUC**	**PR-AUC**
Regresión Logística	Sin SMOTE	0.9939 ± 0.0000	0.1732 ± 0.0262	0.0157 ± 0.0040	0.0288 ± 0.0071	0.8214 ± 0.0096	0.2446 ± 0.0104
Regresión Logística	Con SMOTE	0.8818 ± 0.0013	0.0362 ± 0.0003	0.7583± 0.0122	0.0691 ± 0.0005	0.9141 ± 0.0024	0.1739 ± 0.0069
Random Forest	Sin SMOTE	0.9983 ± 0.0000	**0.9869 ± 0.0022**	0.7116 ± 0.0088	0.8269 ± 0.0057	0.9920 ± 0.0007	0.9194 ± 0.0031

Random Forest	Con SMOTE	0.9984 ± 0.0001	0.9227 ± 0.0086	0.7872± 0.0097	0.8496 ± 0.0085	0.9913 ± 0.0008	0.8869 ± 0.0040
XGBoost	Sin SMOTE	**0.9988 ± 0.0001**	0.9590 ± 0.0058	0.8309 ± 0.0049	**0.8904 ± 0.0051**	**0.9984 ± 0.0004**	**0.9447 ± 0.0054**
XGBoost	Con SMOTE	0.9967 ± 0.0002	0.6640 ± 0.0127	0.8716 ± 0.0094	0.7537 ± 0.0105	0.9951 ± 0.0007	0.8902 ± 0.0082
LightGBM	Sin SMOTE	0.9983 ± 0.0002	0.9069 ± 0.0314	0.7950 ± 0.0095	0.8472 ± 0.0185	0.9949 ± 0.0014	0.8611 ± 0.0606
LightGBM	Con SMOTE	0.9980 ± 0.0001	0.8031 ± 0.0099	**0.8778 ± 0.0052**	0.8387 ± 0.0044	0.997 ± 0.0007	0.9204 ± 0.0030

Nota. Los resultados se expresan como promedio ± desviación estándar de los cinco pliegues. Elaboración propia.

La Tabla 15 presenta los resultados acumulados de las matrices de confusión obtenidas durante la validación cruzada.

Tabla 15. Resultados acumulados de clasificación en los cinco pliegues

Modelo	**Balanceo**	**TN**	**FP**	**FN**	**TP**
Regresión Logística	Sin SMOTE	1288614	555	7 388	118
Regresión Logística	Con SMOTE	1137703	151466	1814	5692
Random Forest	Sin SMOTE	1289098	71	2165	5341
Random Forest	Con SMOTE	1288674	495	1597	5909
XGBoost	Sin SMOTE	1288902	267	1269	6237
XGBoost	Con SMOTE	1285857	3312	964	6542
LightGBM	Sin SMOTE	1288552	617	1539	5967
LightGBM	Con SMOTE	1287552	1617	917	6589

Nota. TN: verdaderos negativos; FP: falsos positivos; FN: falsos negativos; TP: verdaderos positivos. Los valores corresponden a las predicciones de validación acumuladas de los cinco pliegues; cada registro fue evaluado una sola vez. Elaboración propia.

2.3.4.1 Regresión Logística

La Regresión Logística sin SMOTE, configurada con ($C = 10$), alcanzó un Accuracy de 0,9939. Sin embargo, su Recall fue apenas de 0,0157, lo que evidencia una capacidad limitada para reconocer las transacciones fraudulentas. En los cinco pliegues identificó correctamente 118 de los 7506 fraudes, lo que significa que dejó pasar 7388. Por lo tanto, se puede decir que el valor de Accuracy es influenciado por la gran cantidad de transacciones legítimas que el modelo clasificó correctamente.

Con la aplicación de SMOTE y ($C = 0{,}1$), el Recall aumentó a 0,7583 y el modelo logró detectar 5692 fraudes. Sin embargo, la Precision disminuyó a 0,0362 debido al incremento de falsas alertas, que pasaron de 555 a 151466. Este comportamiento también produjo una reducción del Accuracy a 0,8818.

En conclusión, pese a que SMOTE mejoró el ROC-AUC de 0,8214 a 0,9141, el PR-AUC disminuyó de 0,2446 a 0,1739. El F1-Score permaneció bajo en ambos escenarios. SMOTE mejoró la detección de fraudes, pero generó demasiadas falsas alertas.

2.3.4.2 Random Forest

Random Forest utilizó 100 árboles, sin límite de profundidad y con un registro mínimo de 1 por hoja en ambos escenarios. Dentro del escenario sin SMOTE, alcanzó una Precision de 0,9869 y un Recall de 0,7116. En las predicciones de validación, el modelo detectó 5341 fraudes, dejó sin identificar 2165 y generó únicamente 71 falsas alertas.

Con SMOTE, el Recall aumentó a 0,7872, se identificaron 568 fraudes más que con la distribución original. Los fraudes no detectados disminuyeron a 1597; sin embargo, las falsas alertas aumentaron de 71 a 495 y la Precision se redujo a 0,9227.

El F1-Score mejoró de 0,8269 a 0,8496 con SMOTE, mientras que el PR-AUC disminuyó de 0,9194 a 0,8869. El ROC-AUC presentó una variación minima entre ambos escenarios. En este modelo, SMOTE favoreció la detección de fraudes y el equilibrio medido por F1-Score, a cambio de una menor precisión y un mayor número de falsas alertas.

2.3.4.3 XGBoost

XGBoost sin SMOTE, configurado con 200 árboles, una tasa de aprendizaje de 0,1, profundidad máxima de 6 y subsample de 0,8, obtuvo una Precision de 0,9590 y un Recall de 0,8309. En las predicciones acumuladas de los cinco pliegues, identificó 6237 fraudes, dejó sin detectar 1269 y generó 267 falsas alertas.

Con SMOTE se utilizaron 100 árboles, una tasa de aprendizaje de 0,3, profundidad máxima de 6 y subsample de 1,0. El Recall aumentó a 0,8716 y se identificaron 305 fraudes más que en el escenario sin balanceo. Sin embargo, las falsas alertas aumentaron de 267 a 3312 y la Precision disminuyó a 0,6640.

Sin SMOTE, el modelo obtuvo mejores resultados en F1-Score, con 0,8904, y PR-AUC, con 0,9447. Con SMOTE, estas métricas disminuyeron a 0,7537 y 0,8902, respectivamente. En conclusión, el balanceo permitió detectar más fraudes, pero redujo la precisión de las alertas y el equilibrio general entre Precision y Recall.

2.3.4.4 LightGBM

LightGBM sin SMOTE, configurado con 200 árboles, una tasa de aprendizaje de 0,05 y 15 hojas, alcanzó una Precision de 0,9069 y un Recall de 0,7950. En las predicciones acumuladas de los cinco pliegues, detectó correctamente 5967 fraudes, dejó sin identificar 1539 y generó 617 falsas alertas.

Con SMOTE se utilizaron 200 árboles, una tasa de aprendizaje de 0,1 y 63 hojas. El Recall aumentó a 0,8778 y el modelo identificó 6589 fraudes, es decir, 622 casos más que en el escenario sin balanceo. Los fraudes no detectados disminuyeron de 1539 a 917. Sin embargo, las falsas alertas aumentaron de 617 a 1617 y la Precision se redujo a 0,8031.

El PR-AUC mejoró de 0,8611 a 0,9204 con SMOTE y presentó una menor variación entre los cinco pliegues. Por su parte, el F1-Score disminuyó ligeramente de 0,8472 a 0,8387. En este modelo, SMOTE incrementó la detección de fraudes y el desempeño sobre la clase minoritaria, además de un mayor número de falsas alertas.

2.3.5 Comparación del desempeño de los modelos predictivos

El análisis general de los resultados evidenció diferencias importantes entre los modelos. XGBoost sin SMOTE obtuvo el mayor F1-Score, con 0,8904, y el mayor PR-AUC, con 0,9447. Random Forest sin SMOTE alcanzó la Precision más alta, con 0,9869, mientras que LightGBM con SMOTE presentó el mayor Recall, con 0,8778. Es decir, que ninguna de las configuraciones obtuvo los mejores resultados en todas las métricas.

La Tabla 16 muestra los cambios producidos en las principales métricas después de aplicar SMOTE. Los valores positivos representan un incremento y los negativos una disminución respecto al escenario sin balanceo.

Tabla 16. Variación de las métricas después de aplicar SMOTE

Modelo	Variación en Precisión	Variación en Recall	Variación en F1-Score	Variación en PR-AUC
Regresión Logística	−0,1370	+0,7426	+0,0403	−0,0707
Random Forest	−0,0642	+0,0756	+0,0227	−0,0325
XGBoost	−0,2950	+0,0407	−0,1367	−0,0545
LightGBM	−0,1038	+0,0828	−0,0085	+0,0593

Nota. Las variaciones se calcularon restando el resultado sin SMOTE al resultado con SMOTE. Elaboración propia.

Es fácil identificar dentro de la tabla que, SMOTE incrementó el Recall en los cuatro algoritmos, lo que demuestra una mayor capacidad para identificar transacciones fraudulentas. Sin embargo, la Precision se vio reducida en todos los casos debido al aumento de falsas alertas. Esto fue más evidente en la Regresión Logística y XGBoost, mientras que Random Forest y LightGBM conservaron valores de Precision superiores a 0,80.

El efecto del balanceo tampoco fue uniforme en F1-Score y PR-AUC. Random Forest mejoró su F1-Score, mientras que XGBoost presentó una reducción considerable. LightGBM fue el único algoritmo cuyo PR-AUC aumentó al usar SMOTE, al pasar de 0,8611 a 0,9204. Además, la desviación de esta métrica disminuyó de 0,0606 a 0,0030, lo que indica un comportamiento más estable entre los cinco pliegues.

La aplicación de SMOTE hizo que los modelos clasificaran una mayor cantidad de transacciones como fraudulentas. Como resultado, aumentaron los fraudes detectados, pero también las falsas alarmas. El efecto de SMOTE dependió del algoritmo y no debe considerarse favorable únicamente por el incremento del Recall. Debe analizarse conjuntamente mediante Recall, Precision, F1-Score y PR-AUC.

Desde una perspectiva predictiva, XGBoost sin SMOTE presentó el mejor equilibrio general debido a sus resultados de Precision, Recall, F1-Score y PR-AUC. Sin embargo, esta comparación todavía no considera el monto de los fraudes no detectados ni el costo de investigar las alertas. Estos elementos se incorporan en la evaluación económica del apartado siguiente.

2.3.6 Resultados de la evaluación económica

La evaluación económica incorporó el monto de los fraudes no detectados y el costo de investigar las alertas generadas.

Para facilitar la interpretación de los resultados, el costo total de cada configuración se calculó mediante la siguiente expresión:

$$Costo\ total = Monto\ fraudes\ no\ detectados + (Alertas\ generadas\ x\ Costo\ de\ alerta)$$

Donde las alertas generadas son la suma de TP + FP

Por ejemplo: LightGBM con SMOTE presentó:

- 8206 alertas generadas.
- 91701,01 en fraudes no detectados.

Entonces: Costo total cuando "costo por alerta es 1" es igual a.

$$C_1 = 91701.01 + (8206x1) = 99907.01$$

Para cada configuración se calcularon tres escenarios, asignando costos de 1, 5 y 10 unidades por alerta. Los resultados acumulados de los cinco pliegues se presentan en la siguiente Tabla.

Tabla 17. Costos estimados de las configuraciones evaluadas

Modelo	Balanceo	Alertas generadas	Pérdida financiera potencial	Si cada alerta cuesta 1	Si cada alerta cuesta 5	Si cada alerta cuesta 10
Regresión Logística	Sin SMOTE	673	3875619,10	3876292,10	3878984,10	3882349,10
Regresión Logística	Con SMOTE	157158	55781,91	212939,91	841571,91	1627361,91
Random Forest	Sin SMOTE	5412	633094,32	638506,32	660154,32	687214,32
Random Forest	Con SMOTE	6404	328362,73	334766,73	360382,73	392402,73
XGBoost	Sin SMOTE	6504	265813,29	272317,29	298333,29	330853,29
XGBoost	Con SMOTE	9854	102225,19	112079,19	151495,19	200765,19

LightGBM	Sin SMOTE	6584	396550,29	403134,29	429470,29	462390,29
LightGBM	Con SMOTE	8206	91701,01	**99907,01**	**132731,01**	**173761,01**

Nota. La pérdida financiera potencial corresponde a la suma de los montos de los fraudes no detectados. El costo total incorpora dicha pérdida y el costo de investigar las alertas. Elaboración propia.

LightGBM con SMOTE presentó el menor costo total en los tres escenarios. Aunque generó más alertas que su configuración sin balanceo, la reducción de los fraudes no detectados permitió disminuir la pérdida financiera potencial. XGBoost con SMOTE obtuvo el segundo menor costo, con valores de 112079,19, 151495,19 y 200765,19, respectivamente.

La Regresión Logística con SMOTE registró la menor pérdida por fraudes no detectados. Sin embargo, generó 157158 alertas, por lo que su costo total aumentó considerablemente a medida que se incrementó el costo unitario de la investigación. Esto demuestra que reducir únicamente la pérdida por falsos negativos no garantiza el menor costo total si el modelo produce una cantidad elevada de falsas alertas.

Para comprobar que el resultado no dependiera exclusivamente de la suma global de los montos, también se identificó la configuración con menor costo en cada pliegue.

2.3.7 Selección del modelo predictivo

Para la selección del modelo, se consideró tanto el desempeño predictivo como los resultados de la evaluación económica. Desde una perspectiva exclusivamente estadística, XGBoost sin SMOTE presentó el mejor equilibrio general, al obtener el mayor F1-Score, y el mayor PR-AUC. Sin embargo, LightGBM con SMOTE alcanzó el Recall más alto, y produjo la menor pérdida financiera potencial.

La Tabla 18 compara las dos configuraciones con los resultados más relevantes para la decisión final.

Tabla 18. Comparación entre XGBoost sin SMOTE y LightGBM con SMOTE

Criterio	XGBoost sin SMOTE	LightGBM con SMOTE
Precision	**0,9590**	0,8031
Recall	0,8309	**0,8778**
F1-Score	**0,8904**	0,8387

PR-AUC	**0,9447**	0,9204
Fraudes detectados	6237	**6589**
Fraudes no detectados	1269	**917**
Falsas alertas	**267**	1617
Pérdida financiera potencial	265813,29	**91701,01**
Costo total con costo unitario de 1	272317,29	**99907,01**
Costo total con costo unitario de 5	298333,29	**132731,01**
Costo total con costo unitario de 10	330853,29	**173761,01**

Nota. Elaboración propia

LightGBM con SMOTE detectó 352 fraudes más que XGBoost sin SMOTE y redujo en 174112,28 unidades monetarias el monto asociado con fraudes no identificados. Sin embargo, esto implicó 1350 falsas alertas adicionales y una disminución de Precision, F1-Score y PR-AUC.

En la evaluación económica, LightGBM con SMOTE obtuvo el menor costo acumulado en los tres escenarios y presentó el resultado más favorable en 13 de las 15 comparaciones realizadas por pliegue. Esta estabilidad permitió comprobar que su ventaja no dependió únicamente del monto acumulado en una sola partición.

En base a este análisis, se seleccionó LightGBM con SMOTE como modelo predictivo final. La decisión priorizó la reducción de fraudes no detectados y de la pérdida financiera potencial, manteniendo un nivel competitivo de número de alertas inferior al de otras alternativas con Recall elevado.

Esta selección se limita a los datos y a los escenarios de costos definidos en la investigación, por lo que no implica que LightGBM con SMOTE sea superior en todos los contextos.

2.3.8 Evaluación final del modelo seleccionado

Una vez seleccionado LightGBM con SMOTE, el pipeline se entrenó nuevamente utilizando la totalidad de fraudTrain.csv y los hiperparámetros determinados mediante GridSearchCV: 200 árboles, una tasa de aprendizaje de 0,1 y un máximo de 63 hojas.

El modelo fue evaluado una sola vez sobre fraudTest.csv, conjunto que no intervino en la optimización ni en la selección.

- **Métricas de desempeño**

Los resultados obtenidos sobre el conjunto de prueba se presentan a continuación.

Tabla 19. Métricas finales de LightGBM con SMOTE

Métrica	Resultado
Accuracy	0,9977
Precision	0,6688
Recall	0,8163
F1-Score	0,7353
ROC-AUC	0,9931
PR-AUC	0,8443

Nota. Elaboración propia

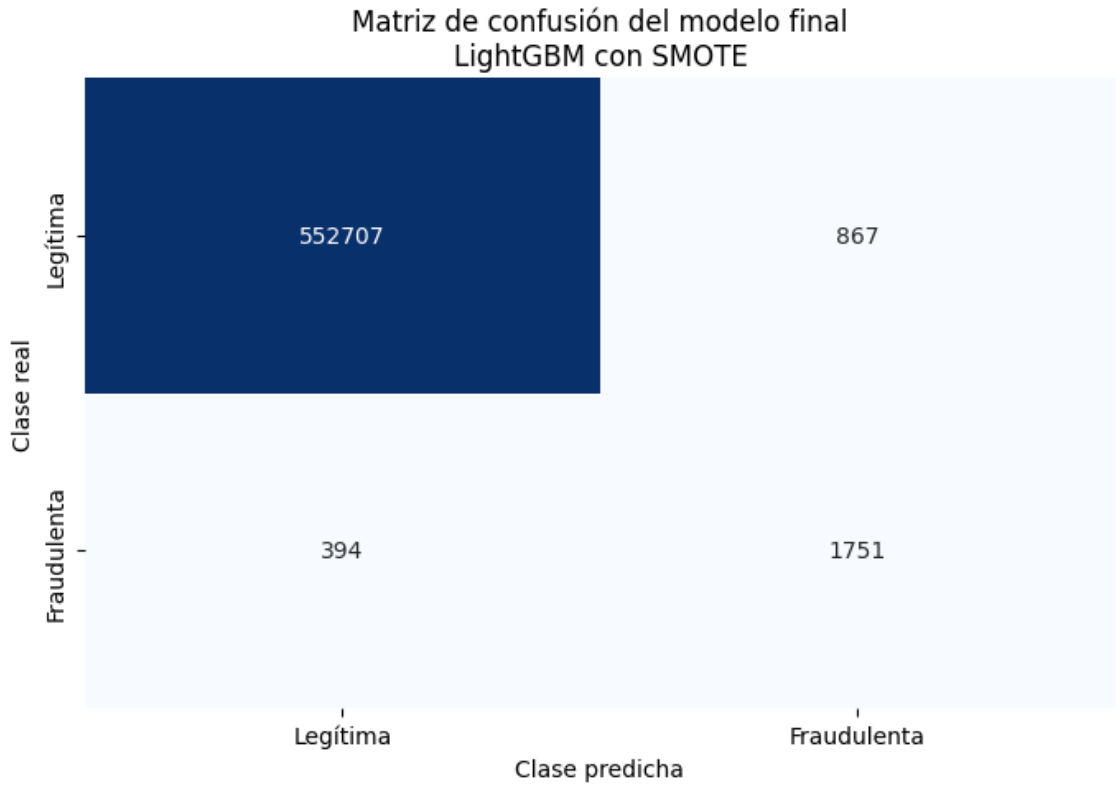
El modelo identificó el 81,63 % de las transacciones fraudulentas presentes en el conjunto de prueba. La Precision de 0,6688 indica que aproximadamente dos de cada tres alertas generadas correspondieron realmente a fraudes. Por su parte, los valores de ROC-AUC y PR-AUC muestran que el modelo mantuvo una capacidad aceptable para diferenciar las clases.

En comparación con la validación cruzada, las métricas del conjunto de prueba presentaron una reducción, especialmente en Precision, F1-Score y PR-AUC. Esta diferencia refleja el desempeño del modelo ante transacciones que no participaron en las etapas de optimización, validación y selección.

- **Matriz de Confusión**

De las 2145 transacciones fraudulentas, el modelo identificó correctamente 1751 y dejó sin detectar 394. En la clase legítima, clasificó correctamente 552707 operaciones y generó 867 falsas alertas. En total, se emitieron 2618 alertas, correspondientes a la suma de los verdaderos positivos y los falsos positivos.

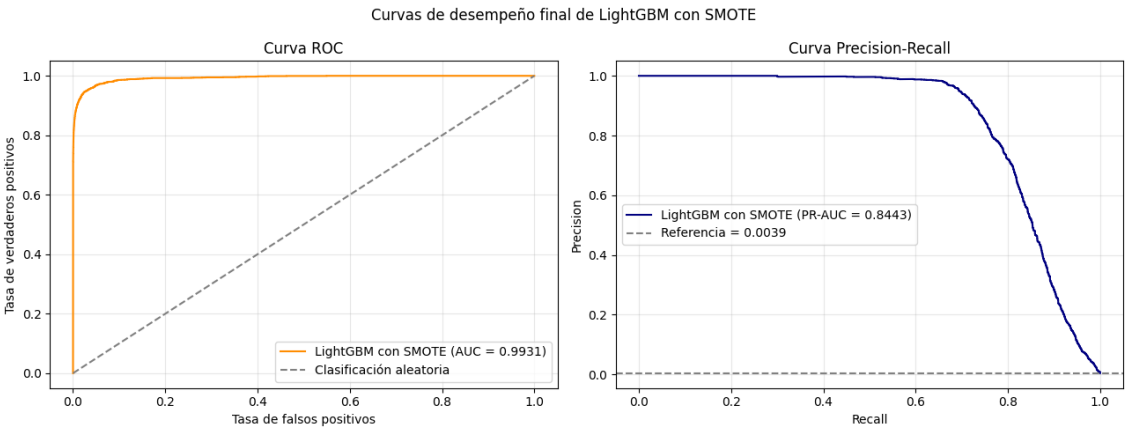
Figura 8. Matriz de confusión del modelo final LightGBM con SMOTE



Nota. Elaboración propia a partir del conjunto de prueba Credit Card Fraud Detection.

- **Curvas ROC y Precision-Recall**

Figura 9. Curvas ROC y Precision-Recall del modelo final LightGBM con SMOTE



Nota. Elaboración propia a partir del conjunto de prueba Credit Card Fraud Detection.

La curva ROC alcanzó un área de 0,9931, esto muestra una elevada capacidad para diferenciar transacciones legítimas y fraudulentas. No obstante, debido al desbalance de clases, el PR-AUC de 0,8443 proporciona una valoración más representativa del desempeño sobre los fraudes.

La curva Precision-Recall muestra que, al incrementar el Recall, la Precision disminuye debido al aumento de falsas alertas. Con el umbral de 0,5, el modelo alcanzó un Recall de 0,8163 y una Precision de 0,6688, lo que refleja el equilibrio obtenido entre la detección de fraudes y la precisión de las alertas.

- **Evaluación económica**

Los 394 fraudes no detectados representaron un monto acumulado de 50843,92 unidades monetarias. Por otro lado, los 1751 fraudes identificados correctamente alcanzaron un monto de 1082480,76. La Tabla 20 presenta el costo total y el ahorro estimado en los tres escenarios de investigación.

Tabla 20. Evaluación económica final de LightGBM con SMOTE

Costo por investigación	Alertas generadas	Pérdida financiera potencial	Costo total estimado	Ahorro estimado	Ahorro estimado %
1	2618	50843,92	53461,92	1079862,76	95,28 %
5	2618	50843,92	63933,92	1069 390,76	94,36 %
10	2618	50843,92	77023,92	1056300,76	93,20 %

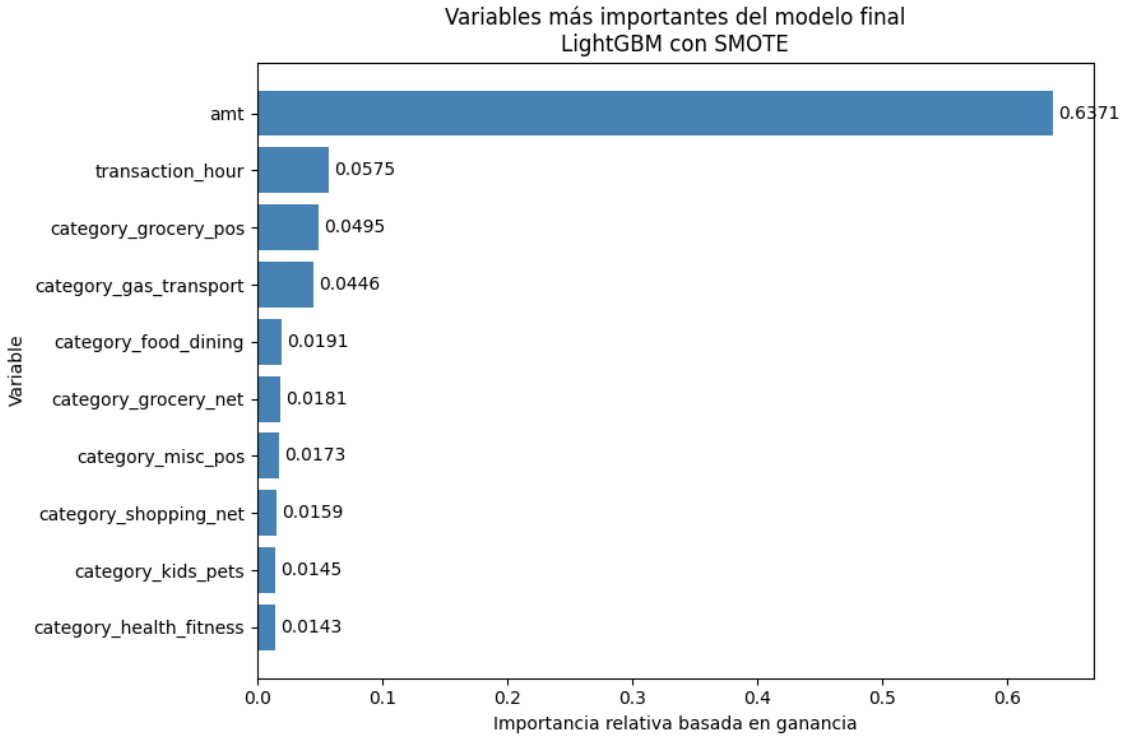
Nota. El ahorro se estimó respecto al monto total de las transacciones fraudulentas del conjunto de prueba, equivalente a 1133324,68 unidades monetarias. Los costos de investigación son hipotéticos. Elaboración propia

El modelo mantuvo un ahorro estimado superior al 93 % en los tres escenarios. Estos resultados muestran que LightGBM con SMOTE conservó una capacidad favorable para detectar fraudes y reducir la pérdida financiera potencial al aplicarse sobre el conjunto de prueba reservado.

2.3.9 Identificación de Variables relevantes en el modelo seleccionado.

Después de entrenar LightGBM con SMOTE, se calculó la importancia de las variables mediante el criterio de ganancia. La contribución de cada característica a la reducción del error durante la construcción de los árboles se ve representada en la figura 10, donde se muestran las 10 variables más relevantes.

Figura 10. Diez variables más relevantes del modelo LightGBM con SMOTE



Nota. Elaboración propia a partir del conjunto de prueba Credit Card Fraud Detection

La importancia obtenida corresponde a LightGBM entrenado con. Además, muestra cuánto contribuyó cada variable al modelo, pero no indica si su efecto incrementó o redujo la probabilidad de que una transacción fuera clasificada como fraudulenta.

El monto de la transacción (amt) presentó la mayor importancia relativa, con 63,71 %. Le siguen la hora de la transacción (transaction_hour), con 5,75 %, y las categorías grocery_pos y gas_transport, con 4,95 % y 4,46 %, respectivamente.

En conjunto, las diez variables principales concentraron el 88,78 % de la importancia total del modelo.

También se identificaron como relevantes las categorías food_dining, grocery_net, misc_pos, shopping_net, kids_pets y health_fitness. Estos resultados indican que el modelo utilizó principalmente el monto, el momento de la operación y el tipo de establecimiento para distinguir patrones asociados con las transacciones fraudulentas.

CAPÍTULO 3: CONCLUSIONES Y RECOMENDACIONES

3.1. Conclusiones

- La investigación permitió desarrollar y evaluar un modelo predictivo para la detección de transacciones fraudulentas mediante la metodología CRISP-ML(Q). El procedimiento incluyó la preparación de los datos, la construcción de pipelines, la optimización de hiperparámetros, la validación cruzada estratificada y una evaluación económica. Esto permitió seleccionar el modelo considerando tanto su desempeño predictivo como las consecuencias asociadas con sus errores de clasificación.
- Los resultados demostraron que SMOTE incrementó el Recall en los cuatro algoritmos, pero también redujo la Precision debido al aumento de falsas alertas. XGBoost sin SMOTE obtuvo el mayor F1-Score y PR-AUC, mientras que LightGBM con SMOTE alcanzó el Recall más alto. Por tanto, el balanceo mejoró la detección de fraudes, pero su efecto dependió del algoritmo y no produjo una mejora general en todas las métricas.
- LightGBM con SMOTE fue seleccionado porque ofreció el mejor equilibrio entre detección de fraude y costo económico. En el conjunto de prueba identificó más del 81 % de las operaciones fraudulentas y aproximadamente dos de cada tres alertas generadas correspondieron a fraudes reales. Además, produjo el menor costo en los escenarios analizados, lo que demuestra que su mayor capacidad de detección compensó el incremento de falsas alertas.
- El monto de la transacción fue la variable con mayor contribución al modelo seleccionado, seguido por la hora de la operación y varias categorías comerciales. No obstante, los resultados corresponden a datos públicos y costos de investigación hipotéticos. Su aplicación en un entorno bancario requeriría validación con información institucional, costos operativos reales y mecanismos de monitoreo.
- Accuracy no fue suficiente para valorar el desempeño en este problema desbalanceado. Por ejemplo, la Regresión Logística sin SMOTE alcanzó un Accuracy de 0,9939, pero solo identificó el 1,57 % de los fraudes. Esto confirmó la necesidad de priorizar Precision, Recall, F1-Score y PR-AUC.

- La validación cruzada de cinco pliegues permitió comprobar la estabilidad de los resultados y evitó que la comparación dependiera de una única división de los datos.

3.2. Recomendaciones

- Se recomienda validar el modelo seleccionado con transacciones reales de una institución financiera. Esto permitiría comprobar si su desempeño se mantiene ante cambios en los hábitos de los usuarios y en las modalidades de fraude.
- Se debería determinar el costo real de investigar una alerta y la pérdida asociada con los fraudes no detectados. Con esta información se podría ajustar el umbral de clasificación según las prioridades operativas, detección de fraudes y la generación de falsas alertas.
- Futuras investigaciones podrían comparar SMOTE con otras estrategias de balanceo de datos, como ponderación de clases, Borderline-SMOTE o métodos combinados de sobremuestreo y limpieza. Así como analizar la posible superposición entre las clases para determinar por qué las muestras sintéticas afectan de manera diferente a cada algoritmo.

Referencias

Ahmed, K. H., Axelsson, S., Li, Y., & Sagheer, A. M. (2025). A credit card fraud detection approach based on ensemble machine learning classifier with hybrid data sampling. *Machine Learning with Applications*, *20*, 100675. https://doi.org/10.1016/j.mlwa.2025.100675

Al-dahasi, E. M., Alsheikh, R. K., Khan, F. A., & Jeon, G. (2025). Optimizing fraud detection in financial transactions with machine learning and imbalance mitigation. *Expert Systems*, *42*(2), e13682. https://doi.org/10.1111/EXSY.13682;WGROUP:STRING:PUBLICATION

Ali, A., Abd Razak, S., Othman, S. H., Eisa, T. A. E., Al-Dhaqm, A., Nasser, M., Elhassan, T., Elshafie, H., & Saif, A. (2022). Financial Fraud Detection Based on Machine Learning: A Systematic Literature Review. *Applied Sciences (Switzerland)*, *12*(19). https://doi.org/10.3390/APP12199637

Bahnsen, A. C., Stojanovic, A., Aouada, D., & Ottersten, B. (2013). Cost Sensitive Credit Card Fraud Detection Using Bayes Minimum Risk. *2013 12th International Conference on Machine Learning and Applications*, 333–338. https://doi.org/10.1109/ICMLA.2013.68

Bhadouria, T. (n.d.). *Credit Card Fraud Detection*. Kaggle. Retrieved August 13, 2026, from https://www.kaggle.com/datasets/tusharbhadouria/credit-card-fraud-detection?select=fraudTrain.csv

Bhattacharyya, S., Jha, S., Tharakunnel, K., & Westland, J. C. (2011). Data mining for credit card fraud: A comparative study. *Decision Support Systems*, *50*(3), 602–613. https://doi.org/10.1016/J.DSS.2010.08.008

Bishop, C. M. (2006). *Reconocimiento de patrones y aprendizaje automático. Springer, Berlín. - Referencias - Publicaciones de investigación científica*. https://www.scirp.org/reference/referencespapers?referenceid=3523326

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324/METRICS

Breskuvienė, D., & Dzemyda, G. (2024). Enhancing credit card fraud detection: highly imbalanced data case. *Journal of Big Data 2024 11:1*, *11*(1), 182-. https://doi.org/10.1186/S40537-024-01059-5

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. https://doi.org/10.1613/JAIR.953

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-*

17-August-2016, 785–794. https://doi.org/10.1145/2939672.2939785;CSUBTYPE:STRING:CONFERENCE

Correa Bahnsen, A., Aouada, D., Stojanovic, A., & Ottersten, B. (2016). Feature engineering strategies for credit card fraud detection. *Expert Systems with Applications*, *51*, 134–142. https://doi.org/10.1016/j.eswa.2015.12.030

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, *27*(8), 861–874. https://doi.org/10.1016/J.PATREC.2005.10.010

Hashemi, S. K., Mirtaheri, S. L., & Greco, S. (2023). Fraud Detection in Banking Data by Machine Learning Techniques. *IEEE Access*, *11*, 3034–3043. https://doi.org/10.1109/ACCESS.2022.3232287

Kabane, S. (2024). *Impact of Sampling Techniques and Data Leakage on XGBoost Performance in Credit Card Fraud Detection*. https://arxiv.org/pdf/2412.07437

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). *LightGBM: A Highly Efficient Gradient Boosting Decision Tree*. https://github.com/Microsoft/LightGBM.

Koenker, R., Chernozhukov, V., He, X., & Peng, L. (2017). Handbook of quantile regression. In *Handbook of Quantile Regression*. https://doi.org/10.1201/9781315120256

Kohavi, R. (1995). *A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. BT - Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes* (pp. 1137–1145). http://ijcai.org/Proceedings/95-2/Papers/016.pdf

Lebichot, B., Verhelst, T., Le Borgne, Y. A., He-Guelton, L., Oble, F., & Bontempi, G. (2021). Transfer Learning Strategies for Credit Card Fraud Detection. *IEEE Access*, *9*, 114754–114766. https://doi.org/10.1109/ACCESS.2021.3104472

Lucas, Y., & Jurgovsky, J. (2020). *Credit card fraud detection using machine learning: A survey*. https://arxiv.org/pdf/2010.06479

Ngai, E. W. T., Hu, Y., Wong, Y. H., Chen, Y., & Sun, X. (2011). The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems*, *50*(3), 559–569. https://doi.org/10.1016/J.DSS.2010.08.006

Niu, X., Wang, L., & Yang, X. (2019). *A Comparison Study of Credit Card Fraud Detection: Supervised versus Unsupervised*. https://doi.org/10.48550/arXiv.1904.10604

Saito, T., & Rehmsmeier, M. (2015). The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLOS ONE*, *10*(3),

e0118432. https://doi.org/10.1371/journal.pone.0118432

Studer, S., Bui, T. B., Drescher, C., Hanuschkin, A., Winkler, L., Peters, S., & Müller, K. R. (2021). Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology. *Machine Learning and Knowledge Extraction 2021, Vol. 3, Pages 392-413*, *3*(2), 392–413. https://doi.org/10.3390/MAKE3020020

Trevor Hastie, Robert Tibshirani, J. F. (2009). *The Elements of Statistical Learning Data Mining, Inference and Prediction*.

Wijaya, M. G., Pinaringgi, M. F., Zakiyyah, A. Y., & Meiliana. (2024). Comparative Analysis of Machine Learning Algorithms and Data Balancing Techniques for Credit Card Fraud Detection. *Procedia Computer Science*, *245*, 677–688. https://doi.org/10.1016/J.PROCS.2024.10.294

Anexos

Anexo A. Fragmentos principales del código

Preprocesamiento de datos

- Carga de datos.

```
train = pd.read_csv("fraudTrain.csv") # Dataset para
entrenamiento
test = pd.read_csv("fraudTest.csv") # Dataset para pruebas
```

- Limpieza y transformación de datos

```
# Eliminación de variables que no aportan capacidad predictiva
columnas_eliminar = [
    'Unnamed: 0',
    'cc_num',
    'first',
    'last',
    'street',
    'trans_num',
    'zip'
]
train.drop(columns=columnas_eliminar, inplace=True)
test.drop(columns=columnas_eliminar, inplace=True)
```

```
# Transformación de variables temporales
train['transaction_day'] = train['trans_date_trans_time'].dt.hour
train['transaction_day'] = train['trans_date_trans_time'].dt.day
train['transaction_month'] =
train['trans_date_trans_time'].dt.month
train['transaction_dayofweek'] =
train['trans_date_trans_time'].dt.dayofweek
test['trans_date_trans_time'] =
pd.to_datetime(test['trans_date_trans_time'])
test['transaction_hour'] = test['trans_date_trans_time'].dt.hour
test['transaction_day'] = test['trans_date_trans_time'].dt.day
test['transaction_month'] =
test['trans_date_trans_time'].dt.month
test['transaction_dayofweek'] =
test['trans_date_trans_time'].dt.dayofweek
```

```
# Cálculo de la edad del titular
```

```
# Convertir la fecha de nacimiento a formato datetime
train['dob'] = pd.to_datetime(train['dob'])
test['dob'] = pd.to_datetime(test['dob'])

# Calcular la edad al momento de la transacción
train['age'] = (
    train['trans_date_trans_time'].dt.year - train['dob'].dt.year
)

test['age'] = (
    test['trans_date_trans_time'].dt.year - test['dob'].dt.year
)
```

- Codificación de variables

```
# Codificación One-Hot encoding de las variables categóricas
train = pd.get_dummies(
    train,
    columns=['gender', 'category', 'state'],
    drop_first=True )

test = pd.get_dummies(
    test,
    columns=['gender', 'category', 'state'],
    drop_first=True )
```

- Separación de las variables predictoras y variable objetivo

```
# Separación de variables predictoras y variable objetivo
X_train = train.drop(columns='is_fraud')
y_train = train['is_fraud']

X_test = test.drop(columns='is_fraud')
y_test = test['is_fraud']
```

Construcción de pipelines

- Definición de los estimadores para la optimización

```
# Regresión Logística
# lbfgs permite evaluar penalizaciones L2
estimador_lr = LogisticRegression(
    solver='lbfgs',
    penalty='l2',
```

```
    max_iter=1000,
    random_state=42)
# Random Forest
estimador_rf = RandomForestClassifier(
    random_state=42,
    n_jobs=-1)
# XGBoost
# tree_method='hist' agiliza el entrenamiento
# sobre conjuntos de datos extensos
estimador_xgb = XGBClassifier(
    random_state=42,
    eval_metric='logloss',
    tree_method='hist',
    n_jobs=-1)
# LightGBM
estimador_lgbm = LGBMClassifier(
    random_state=42,
    verbosity=-1,
    n_jobs=-1)
```

- Pipeline para las ocho configuraciones

```
# Diccionario que almacenará los Pipeline
# utilizados durante GridSearchCV
pipelines_busqueda = {

    # Regresión Logística con distribución original
    ('Regresión Logística', 'Sin SMOTE'): Pipeline([
        (
            'escalado',
            StandardScaler()
        ),
        (   'modelo',
            clone(estimador_lr) )
    ]),
    # Regresión Logística con escalado y SMOTE
    ('Regresión Logística', 'Con SMOTE'): Pipeline([
        (
            'escalado',
            StandardScaler()
        ),
        (
            'smote',
            SMOTE(random_state=42)
        ),
        (   'modelo',
            clone(estimador_lr))
```

```
    ]),
    # Random Forest con distribución original
    ('Random Forest', 'Sin SMOTE'): Pipeline([
        (   'modelo',
            clone(estimador_rf)
        )
    ]),
    # Random Forest con SMOTE
    ('Random Forest', 'Con SMOTE'): Pipeline([
        (   'smote',
            SMOTE(random_state=42)
        ),
        (  'modelo',
            clone(estimador_rf)
        )
    ]),
    # XGBoost con distribución original
    ('XGBoost', 'Sin SMOTE'): Pipeline([
        (   'modelo',
            clone(estimador_xgb)
        )
    ]),
    # XGBoost con SMOTE
    ('XGBoost', 'Con SMOTE'): Pipeline([
        (   'smote',
            SMOTE(random_state=42)
        ),
        (  'modelo',
            clone(estimador_xgb)
        )
    ]),
    # LightGBM con distribución original
    ('LightGBM', 'Sin SMOTE'): Pipeline([
        (  'modelo',
            clone(estimador_lgbm) )
    ]),
    # LightGBM con SMOTE
    ('LightGBM', 'Con SMOTE'): Pipeline([
        (   'smote',
            SMOTE(random_state=42)
        ),
        (   'modelo',
            clone(estimador_lgbm)
        )
    ])
}
```

Optimización de hiperparámetros

- Configuración de la validación cruzada

```
# Validación utilizada durante la búsqueda
# sistemática de hiperparámetros
cv_busqueda = StratifiedKFold(
    n_splits=3,
    shuffle=True,
    random_state=42
)
```

- Criterio de optimización basado en PR-AUC

```
# Función utilizada por GridSearchCV para calcular
# el área bajo la curva Precision-Recall
def calcular_pr_auc_busqueda(y_real, y_prob):
    precision, recall, _ = precision_recall_curve(
        y_real,
        y_prob
    )

    return auc(
        recall,
        precision
    )
# Convertir la función en un criterio compatible
# con GridSearchCV utilizando probabilidades
scorer_pr_auc = make_scorer(
    calcular_pr_auc_busqueda,
    response_method='predict_proba'
)
```

Stratified 5-fold cross-validation.

- Configuración de la validación cruzada

```
# Validación utilizada posteriormente para evaluar
# las configuraciones optimizadas
cv_evaluacion = StratifiedKFold(
    n_splits=5,
    shuffle=True,
    random_state=42
)
```

- Construcción de los ocho Pipelines optimizados

```
from sklearn.base import clone
# Relacionar cada archivo guardado con su Pipeline
mapa_configuraciones = {
    'lr_sin_smote': (
        'Regresión Logística',
        'Sin SMOTE'
    ),
    'lr_con_smote': (
```

```
        'Regresión Logística',
        'Con SMOTE'
    ),
    'rf_sin_smote': (
        'Random Forest',
        'Sin SMOTE'
    ),
    'rf_con_smote': (
        'Random Forest',
        'Con SMOTE'
    ),
    'xgb_sin_smote': (
        'XGBoost',
        'Sin SMOTE'
    ),
    'xgb_con_smote': (
        'XGBoost',
        'Con SMOTE'
    ),
    'lgbm_sin_smote': (
        'LightGBM',
        'Sin SMOTE'
    ),
    'lgbm_con_smote': (
        'LightGBM',
        'Con SMOTE'
    )
}
```

- Función para evaluar una configuración con 5-fold C

```
def evaluar_configuracion(nombre, balanceo, pipeline, X, y, cv):
    # Almacenar los resultados obtenidos
    # en cada uno de los cinco pliegues
    resultados = []
    # Recorrer las cinco divisiones estratificadas
    for pliegue, (indices_train, indices_validacion) in
enumerate(
        cv.split(X, y),
        start=1
    ):
        # Seleccionar los registros destinados
        # al entrenamiento del pliegue actual
        X_train_pliegue = X.iloc[indices_train]
        y_train_pliegue = y.iloc[indices_train]
        # Seleccionar los registros destinados
        # a la validación del pliegue
        X_validacion_pliegue = X.iloc[indices_validacion]
        y_validacion_pliegue = y.iloc[indices_validacion]
```

```
        # Crear una copia independiente del Pipeline
        # para cada iteración de la validación cruzada
        modelo_pliegue = clone(pipeline)
```

Evaluación económica

- Resultados acumulados de las matrices de confusión y montos

```
resultados_confusion_optimizados = (
    resultados_cv_optimizados.groupby(
        ['Modelo', 'Balanceo'],
        as_index=False
    )[[
        'TN',
        'FP',
        'FN',
        'TP',
        'Monto fraude detectado',
        'Monto fraude no detectado'
    ]].sum()
)
# Calcular totales derivados
resultados_confusion_optimizados[
    'Total'
] = (
    resultados_confusion_optimizados['TN']
    + resultados_confusion_optimizados['FP']
    + resultados_confusion_optimizados['FN']
    + resultados_confusion_optimizados['TP']
)
resultados_confusion_optimizados[
    'Alertas generadas'
] = (
    resultados_confusion_optimizados['FP']
    + resultados_confusion_optimizados['TP']
)
resultados_confusion_optimizados[
    'Monto fraude total'
] = (
    resultados_confusion_optimizados[
        'Monto fraude detectado'
    ]
    + resultados_confusion_optimizados[
        'Monto fraude no detectado'
    ]
)
```

- Análisis económico

```
costos_investigacion = [1, 5, 10]
filas_costos = []
for _, fila in resultados_confusion_optimizados.iterrows():
    for costo_unitario in costos_investigacion:
        # Costo de revisar todas las alertas generadas
        costo_investigacion_total = (
            fila['Alertas generadas']
            * costo_unitario
        )
        # Pérdida por fraude no detectado más
        # el costo de investigar las alertas
        costo_total_estimado = (
            fila['Monto fraude no detectado']
            + costo_investigacion_total
        )
        # Diferencia respecto al monto total de fraude
        ahorro_estimado = (
            fila['Monto fraude total']
            - costo_total_estimado
        )
        ahorro_porcentual = (
            ahorro_estimado
            / fila['Monto fraude total']
            * 100        )
```

Entrenamiento y evaluación del modelo final

- Preparación del modelo final seleccionado.

```
modelo_final_lgbm_smote = clone(pipeline_final)
inicio_entrenamiento = time.time()
modelo_final_lgbm_smote.fit(
    X_train,
    y_train )
tiempo_entrenamiento = (
    time.time() - inicio_entrenamiento
) / 60
```