

LEAD University

Bachillerato en Ingeniería en Ciencia de Datos

2025- I TTCT0018 Datos Masivos

Profesor:

José Alberto Aguilar Romero

**Proyecto final**

Detección de Fraude por medio de transacciones financieras

Estudiantes:

Carolina Salas Moreno

Deykel Bernard Salazar

Esteban Ramírez Montero

Kristhel Porras Mata

Fecha de Entrega: agosto 18, 2025

Tabla de contenido

[I Introducción 4](#_Toc206449802)

[1.1 Descripción breve del proyecto 4](#_Toc206449803)

[1.2 Justificación 4](#_Toc206449804)

[1.3 Objetivos del Proyecto 5](#_Toc206449805)

[1.3.1 Objetivo general del proyecto 5](#_Toc206449806)

[1.3.2 Objetivos Específicos 5](#_Toc206449807)

[1.4 Propuesta para el Desarrollo del Proyecto 5](#_Toc206449808)

[1.4.1 Herramientas (Apache Spark, Python, Delta Lake, etc.) 5](#_Toc206449809)

[1.4.2 Plataforma (Databricks, Google Colab, local, etc.) 5](#_Toc206449810)

[1.4.3 ¿Cómo van a procesar los datos? (con Spark, Pandas, SQL, UDFs, etc.) 6](#_Toc206449811)

[ll. Extract, Transform and Load (ETL) 8](#_Toc206449812)

[2.1 Extracción: Descripción de la fuente de los datos y como los van a cargar 8](#_Toc206449813)

[2.2 Transformación: 8](#_Toc206449814)

[2.3 Carga: Destino de los datos (base de datos relacional, Data Lake, Delta Tables, base de datos de grafos, etc.) 10](#_Toc206449815)

[lll. Métricas utilizadas 10](#_Toc206449816)

[IV. Datasets y fuentes de datos 11](#_Toc206449817)

[V. Arquitectura y Herramientas utilizadas 12](#_Toc206449818)

[5.1 Fuentes de Datos 12](#_Toc206449819)

[5.2 Almacenamiento Lakehouse 12](#_Toc206449820)

[5.3 Cómputo y Orquestación 12](#_Toc206449821)

[5.4 Preparación de Features 13](#_Toc206449822)

[5.5 Entrenamiento y Evaluación 13](#_Toc206449823)

[5.6 Optimización de Umbral 13](#_Toc206449824)

[5.7 Artefactos y Registro 13](#_Toc206449825)

[5.8 Serving y Consumo 13](#_Toc206449826)

[VI. Metodología 15](#_Toc206449827)

[6.1 Análisis Exploratorio de los datos 15](#_Toc206449828)

[6.1.1 Caracterización inicial del dataset 15](#_Toc206449829)

[6.1.2 Análisis univariado 15](#_Toc206449830)

[6.1.3 Análisis univariado 16](#_Toc206449831)

[6.1.4 Visualización de patrones relevantes 16](#_Toc206449832)

[6.1.5 Hallazgos clave del EDA 16](#_Toc206449833)

[6.2 Preprocesamiento 17](#_Toc206449834)

[6.3 Modelos / Algoritmos 18](#_Toc206449835)

[6.4 Evaluación 18](#_Toc206449836)

[VII. Desarrollo 19](#_Toc206449837)

[VIII. Resultados 24](#_Toc206449838)

[8.1 Modelos evaluados y desempeño 25](#_Toc206449839)

[8.2 Comparación con la estrategia anterior (undersampling) 25](#_Toc206449840)

[8.3 Resultados con la estrategia optimizada (class weights + umbral ajustado) 25](#_Toc206449841)

[8.4 Interpretación de resultados 26](#_Toc206449842)

[IX. Conclusiones y trabajo futuro 27](#_Toc206449843)

[X. Referencias 29](#_Toc206449844)

[XI. Anexos 31](#_Toc206449845)

# I Introducción

## 1.1 Descripción breve del proyecto

Este proyecto tiene como objetivo desarrollar un sistema de detección de fraude a partir del análisis de transacciones financieras. Para ello, se emplea un conjunto de datos sintético que simula 5 millones de transacciones, diseñado específicamente para representar comportamientos reales y facilitar la identificación de actividades fraudulentas.

El dataset incluye información detallada de cada transacción (ID, fecha, cuentas involucradas, monto y tipo), características de comportamiento (como el tiempo desde la última transacción y patrones de gasto), metadatos (ubicación, dispositivo, dirección IP) e indicadores de fraude (etiqueta binaria y tipo de fraude).  
Este conjunto de datos está disponible públicamente en Kaggle bajo el siguiente nombre y dirección URL: [Financial Transactions Dataset for Fraud Detection.](https://www.kaggle.com/datasets/aryan208/financial-transactions-dataset-for-fraud-detection)

## 1.2 Justificación

La elección de este conjunto de datos corresponde a su conveniencia para simular escenarios reales de detección de fraude financiero, gracias a su volumen considerable y a la complejidad de sus variables, que reflejan el entorno operativo de las instituciones bancarias actuales. Este tipo de análisis permite aplicar técnicas avanzadas de aprendizaje automático en un contexto donde la precisión la velocidad de respuesta son críticas. La identificación oportuna de patrones anómalos no solo contribuye a mitigar pérdidas económicas, sino que también fortalece la confianza en los sistemas financieros digitales y promueve la seguridad en las transacciones electrónicas.

Abordar este tipo de problema representa una oportunidad para integrar conocimientos teóricos con herramientas y metodologías prácticas utilizadas en la industria, mediante el análisis de grandes volúmenes de datos para detectar fraude que permitirá desarrollar competencias clave como la ingeniería de datos, el modelado predictivo, el procesamiento eficiente de información y la toma de decisiones basada en evidencia, todo en un dominio de alta demanda profesional como el sector financiero.

## 1.3 Objetivos del Proyecto

### 1.3.1 Objetivo general del proyecto

* Diseñar un modelo de clasificación que permita identificar transacciones fraudulentas utilizando información transaccional.

### 1.3.2 Objetivos Específicos

* Identificar patrones y características comunes en las transacciones fraudulentas mediante el análisis exploratorio de los datos.
* Desarrollar un modelo predictivo que permita anticipar comportamientos sospechosos y clasificar nuevas transacciones como fraude o no fraude.

## 1.4 Propuesta para el Desarrollo del Proyecto

### 1.4.1 Herramientas (Apache Spark, Python, Delta Lake, etc.)

Para el desarrollo de este proyecto se emplearon tanto Apache Spark como Python, Apache Spark por ser una herramienta ampliamente reconocida por su capacidad para procesar grandes volúmenes de datos de manera rápida y distribuida y Python puro, haciendo uso de su ecosistema de bibliotecas especializadas en ciencia de datos. La implementación se realizará principalmente en Python, debido a su versatilidad, facilidad de uso y amplio ecosistema de bibliotecas especializadas en análisis de datos y aprendizaje automático, tales como Pandas, Scikit-learn, Matplotlib y PySpark.

Asimismo, se incorporará Delta Lake como sistema de almacenamiento, lo cual permitirá gestionar los datos de forma estructurada, confiable y optimizada para consultas eficientes. Esta combinación de tecnologías resulta especialmente adecuada para manejar el volumen y la complejidad de los datos involucrados en este proyecto.

### 1.4.2 Plataforma (Databricks, Google Colab, local, etc.)

El proyecto será desarrollado utilizando Databricks, una plataforma en la nube que permite trabajar de forma gratuita con Apache Spark. Esta opción ha sido seleccionada debido a la experiencia previa del equipo con la herramienta, lo cual aporta confianza y facilita un avance ágil en el desarrollo.

Además, Databricks ofrece un entorno colaborativo altamente eficiente, en el que todos los integrantes del equipo pueden visualizar, editar y ejecutar código en tiempo real. Esta funcionalidad contribuye a una dinámica de trabajo grupal más fluida y organizada, permitiendo que el enfoque principal se mantenga en el análisis de datos y el desarrollo del modelo, sin obstáculos técnicos innecesarios.

### 1.4.3 ¿Cómo van a procesar los datos? (con Spark, Pandas, SQL, UDFs, etc.)

El procesamiento de los datos lo haremos principalmente con Apache Spark, aprovechando su capacidad para manejar grandes volúmenes de información de forma rápida y distribuida. Utilizaremos Spark DataFrames para la mayoría de las transformaciones y análisis, ya que nos permiten trabajar con los datos de manera estructurada, similar a SQL, pero con la ventaja de que escalan muy bien en Databricks.

Pasos para desarrollo con una breve descripción de cada uno (limpieza, análisis, visualización, etc.)

Antes de cualquier análisis o modelado, es fundamental asegurar la calidad y coherencia de los datos. Los pasos principales que se seguirán son:

1. Limpieza de datos: Se eliminarán valores nulos o registros duplicados que puedan afectar la precisión del análisis. Se corregirán errores de formato y se normalizarán las variables para garantizar uniformidad. También se identificarán y gestionarán outliers que puedan distorsionar los resultados. Este proceso asegura que los datos sean fiables y aptos para el modelado.
2. Análisis exploratorio de datos (EDA): Se realizará un estudio exhaustivo para entender la estructura y características del dataset. Esto incluye análisis univariado para examinar la distribución de cada variable, y análisis bivariado para identificar relaciones relevantes entre variables, especialmente con la variable objetivo (fraude o no fraude). Se emplearán estadísticas descriptivas y técnicas gráficas para detectar patrones, tendencias y anomalías.
3. Visualización de datos: Se crearán gráficos y diagramas como histogramas, boxplots, diagramas de dispersión y mapas de calor para facilitar la interpretación visual de los datos. Estas visualizaciones ayudan a comunicar hallazgos clave y a detectar comportamientos inusuales que pueden indicar fraude. La visualización también es útil para validar supuestos previos y apoyar la selección de variables para el modelado.
4. Ingeniería de características: Basado en el análisis previo, se diseñarán y transformarán variables que puedan mejorar la capacidad predictiva del modelo. Esto incluye la creación de variables temporales, banderas para eventos especiales, agregaciones por usuario y codificación adecuada de variables categóricas.
5. Balanceo de datos: Dado que las transacciones fraudulentas suelen ser una minoría, se aplicarán técnicas para corregir el desbalance de clases, como sobre muestreo, submuestreo o métodos sintéticos (por ejemplo, SMOTE). Esto permitirá que el modelo aprenda correctamente ambas clases sin sesgo hacia la mayoría.
6. División de datos: Se separará el conjunto de datos en subconjuntos de entrenamiento, validación y prueba para evaluar el rendimiento del modelo de forma objetiva y evitar el sobreajuste.
7. Entrenamiento y validación del modelo: Se entrenarán varios modelos supervisados con técnicas como Naive Bayes, Regresión Logística, K-Nearest Neighbors y Random Forest, evaluando su desempeño con métricas específicas. La validación cruzada se usará para asegurar la robustez de los resultados.
8. Evaluación y selección del modelo: Se compararán los modelos utilizando métricas como precisión, sensibilidad, F1-score y ROC AUC, eligiendo aquel que ofrezca el mejor equilibrio entre detección de fraude y minimización de falsos positivos.
9. Implementación y monitoreo: Finalmente, se implementará el modelo seleccionado para la detección en tiempo real o análisis en lotes, y se definirá un plan de monitoreo continuo para evaluar su desempeño en producción y actualizarlo según sea necesario.

# ll. Extract, Transform and Load (ETL)

### 2.1 Extracción: Descripción de la fuente de los datos y como los van a cargar

Para la fase de extracción de datos en este proyecto se contemplan tres enfoques principales. La primera opción consiste en automatizar la descarga del conjunto de datos desde Kaggle utilizando una API Key gestionada de forma segura mediante los *secret scopes* de Databricks; esta estrategia permite obtener el archivo directamente desde la fuente y reproducir el proceso de forma eficiente. La segunda alternativa es la carga manual, que implica descargar el archivo CSV desde Kaggle y subirlo directamente al entorno de trabajo de Databricks, ya sea al sistema de archivos del workspace o a DBFS, para luego ser procesado. Finalmente, la tercera opción utiliza Amazon S3 como repositorio intermedio: el archivo se carga en un bucket de S3 para continuar las transformaciones en Databricks. Cada enfoque garantiza la disponibilidad del conjunto de datos dentro del entorno analítico, adaptándose a diferentes niveles de automatización y escalabilidad.

### 2.2 Transformación:

Como parte de la etapa de Transformación del flujo ETL, se aplicarán diversas operaciones para preparar y enriquecer los datos antes del modelado. El proceso se iniciará con un análisis exploratorio (EDA), tanto univariado como bivariado, con el objetivo de comprender la distribución de las variables, identificar relaciones relevantes y detectar posibles anomalías. Con base en estos hallazgos, se ejecutará una fase de limpieza de datos, que incluirá el tratamiento de valores nulos, el manejo del desbalance en la variable objetivo (is\_fraud) y la corrección de sesgos en variables numéricas.

Posteriormente, se implementarán técnicas de ingeniería de características, que incluirán:

* Creación de banderas para fechas clave, como fines de semana o temporadas especiales (por ejemplo, Navidad o época de reembolsos de impuestos).
* Extracción temporal: derivación de variables como hora, día, mes, día de la semana y bandera de fin de semana a partir del campo timestamp, con el fin de capturar patrones temporales relevantes.
* Análisis secuencial por cuenta (sender\_account): cálculo del tiempo entre transacciones (delta\_time) y conteo de transacciones en ventanas móviles, útil para detectar aumentos repentinos en la frecuencia de uso.
* Codificación de variables categóricas: se aplicará one-hot encoding o embeddings a campos como transaction\_type, merchant\_category, location, device\_used y payment\_channel. Para variables de alta cardinalidad como ip\_address y device\_hash, se considerará el uso de técnicas de hashing o agrupamiento. En el caso de receiver\_account, se podrán generar indicadores de recurrencia o relación con otros usuarios.
* Generación de indicadores derivados: banderas como is\_large\_transaction (transacciones fuera del percentil 95), device\_mismatch (cambio frecuente de dispositivo) y geo\_change (variación significativa en la ubicación geográfica) ayudarán a capturar señales de comportamiento anómalo.
* Agregaciones por usuario: cálculo del promedio, desviación estándar del monto por cuenta, así como la frecuencia de uso de ciertos dispositivos o canales de pago, para establecer perfiles base por usuario.
* Balanceo de clases: en caso de fuerte desbalance en la variable objetivo, se considerará la aplicación de técnicas de submuestreo, sobremuestreo o métodos sintéticos como SMOTE.
* Escalado de variables numéricas: variables como amount, spending\_deviation\_score y velocity\_score serán normalizadas mediante MinMaxScaler o StandardScaler, especialmente si se utilizan modelos sensibles a la escala de los datos.

Las transformaciones mencionadas con anterioridad al igual que otras permitirán construir un conjunto de datos sólido y optimizado para detectar patrones complejos de fraude.

### 2.3 Carga: Destino de los datos (base de datos relacional, Data Lake, Delta Tables, base de datos de grafos, etc.)

Después de aplicar las transformaciones, los datos fueron almacenados en Delta Tables dentro de la plataforma Databricks, aprovechando la robustez de Delta Lake como capa de almacenamiento transaccional sobre Apache Spark. Esta elección se justifica por su capacidad de garantizar la consistencia e integridad de los datos mediante propiedades ACID, permitir el versionado y la trazabilidad de cada carga para auditar y reproducir experimentos, y optimizar el acceso, lo cual mejora el desempeño en consultas analíticas.

Asimismo, las Delta Tables ofrecen la escalabilidad necesaria para manejar grandes volúmenes de información sin sacrificar eficiencia, constituyéndose así en un repositorio confiable y reproducible que sirve de base tanto para el análisis exploratorio como para la ingeniería de características, el entrenamiento de modelos y su posterior monitoreo en producción.

# lll. Métricas utilizadas

Los algoritmos utilizados serán evaluados con las siguientes métricas:

* Precisión: Mide la proporción de todas las clasificaciones positivas del modelo que realmente son positivas.
* Sensibilidad: Mide la proporción de clasificaciones positivas que capturan el modelo.
* F1-Score: Es la combinación entre precisión y sensibilidad que equilibra el objetivo reducir tantos los falsos positivos como los falsos negativos.

Con estas métricas, se podrá determinar cuál algoritmo es mejor en términos de detección de fraudes reales y minimizacion de falsos positivos que pongan en duda las alertas del sistema.

# IV. Datasets y fuentes de datos

El dataset seleccionado fue el “Financial Transactions Dataset for Fraud Detection”, que se encuentra en la plataforma de Kaggle y contiene 5,000,000 de registros donde cada registro cuenta con 18 atributos, todo almacenado en un archivo CSV con un tamaño casi a 800MB.

* Fuente: Kaggle
* URL: <https://www.kaggle.com/datasets/aryan208/financial-transactions-dataset-for-fraud-detection>

Justificación de su elección

El aumento de las transacciones bancarias digitales trae consigo múltiples beneficios para los usuarios, pero también ha abierto nuevas oportunidades para los ciberdelincuentes, quienes constantemente desarrollan técnicas más sofisticadas para realizar fraudes sin ser detectados. Esto genera pérdidas patrimoniales significativas y afecta la confianza en el sistema financiero.

El fraude financiero es un gran problema que ataca de manera global y no se limita a actores individuales con conocimientos técnicos en informática, muchos ciberdelincuentes forman parte de organizaciones altamente estructuradas, e incluso cuentan con apoyo de gobiernos autoritarios y acceso a recursos avanzados.

Por ello, los sistemas de detección de fraude requieren el uso de técnicas de avanzadas y actualizadas como puede ser el Machine Learning, con el objetivo de prevenir fraudes al momento. Por ello se optó por un dataset relacionado al tema como base para desarrollar y evaluar un modelo de detección de fraudes, abordando los principales desafíos que surgen en su implementación.

Descripción del dataset

* Variable objetivo: is\_fraud
* Valores faltantes en 2 columnas, el resto están completas
* El dataset está fuertemente desbalanceado respecto a la variable objetivo
* Tipos de variables: 10 categóricos, 5 numéricos, 1 binaria, 1 serie de tiempo y 1 categórica multinomial.

# V. Arquitectura y Herramientas utilizadas

La solución propuesta se basa en una arquitectura modular que combina distintas capas de procesamiento y almacenamiento, cada una soportada por tecnologías específicas que permiten manejar grandes volúmenes de datos, garantizar reproducibilidad y facilitar la implementación de modelos de Machine Learning para la detección de fraude.

## 5.1 Fuentes de Datos

* Transacciones financieras (CSV/Parquet): Dataset sintético de Kaggle con 5 millones de registros y 18 variables.
* Datos complementarios (Dispositivos, IP, Ubicación): Se utilizan como señales adicionales para enriquecer el modelado y detectar patrones anómalos.

## 5.2 Almacenamiento Lakehouse

* Delta Lake (Parquet + ACID): Se emplea como capa de almacenamiento transaccional para garantizar consistencia, trazabilidad y versionado de los datos.
* Databricks Workspace (Repos/Notebooks): Entorno colaborativo en la nube que centraliza el desarrollo de notebooks, la ejecución de código y la gestión de pipelines.

## 5.3 Cómputo y Orquestación

* Databricks Jobs / Schedules: Automatizan la ejecución recurrente de pipelines de ETL, feature engineering y scoring.
* Apache Spark: Motor distribuido para procesar grandes volúmenes de datos de manera eficiente.
* Python Runtime: Lenguaje principal para el análisis y modelado, soportado por librerías como Pandas, NumPy y Scikit-learn.

## 5.4 Preparación de Features

* Pandas y NumPy: Limpieza, normalización y generación de variables numéricas.
* Scikit-learn: Split estratificado, escalado robusto (RobustScaler) y codificación categórica (One-Hot y Frequency Encoding).
* SciPy: Representación en matrices dispersas (CSR) para optimizar memoria y cómputo.
* Imbalanced-learn / class\_weight: Técnicas de balanceo para corregir el fuerte desbalance de clases.

## 5.5 Entrenamiento y Evaluación

* Modelos en Scikit-learn: Regresión Logística, Random Forest y Gradient Boosting.
* Métricas de evaluación: PR curve, F1-score, ROC-AUC, con énfasis en recall y precisión balanceada para el caso de fraude.

## 5.6 Optimización de Umbral

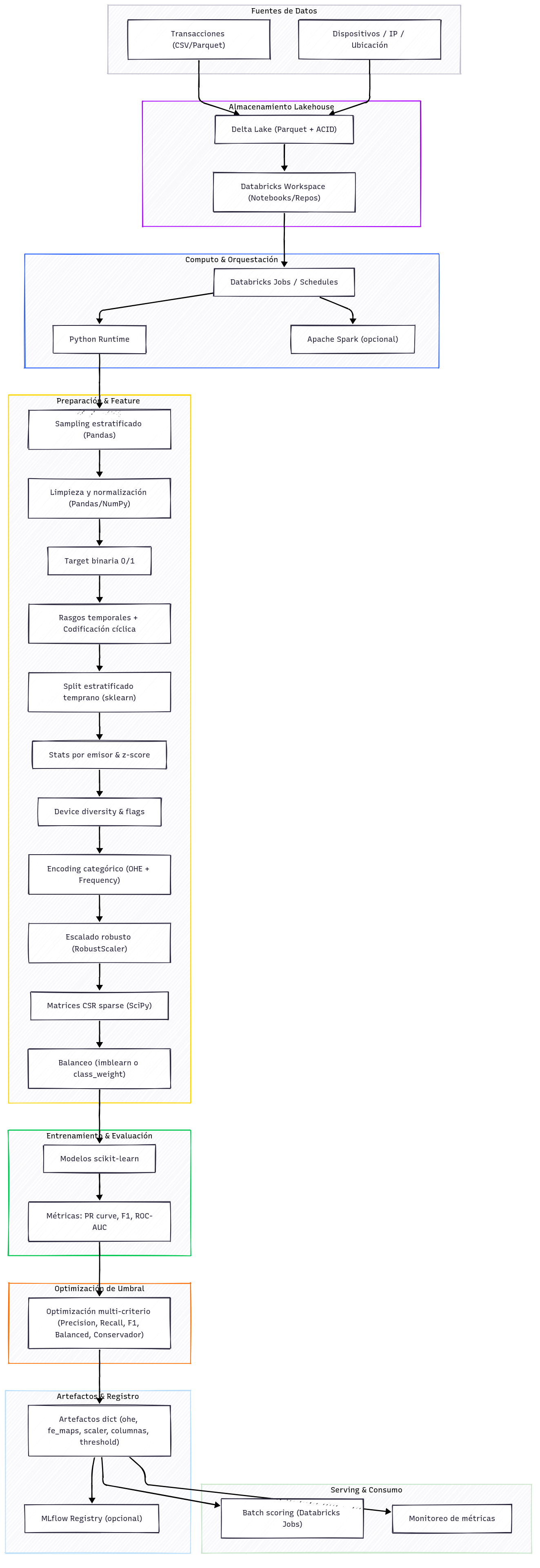
* Optimización multi-criterio: Se ajustan thresholds bajo distintos objetivos de negocio (precisión, recall, F1, conservador) para minimizar falsos positivos sin perder sensibilidad ante fraudes.

## 5.7 Artefactos y Registro

* Artefactos registrados: Incluyen escaladores, encoders, listas de columnas y thresholds, asegurando reproducibilidad.
* MLflow Registry (opcional y deseado): Permite versionar y gestionar modelos de manera centralizada dentro de Databricks.

## 5.8 Serving y Consumo

* Batch scoring (Databricks Jobs): Ejecución periódica para clasificar nuevas transacciones.
* Monitoreo de métricas: Seguimiento del desempeño del modelo y ajuste de umbrales en producción.



# VI. Metodología

## Análisis Exploratorio de los datos

Mediante el proceso del análisis exploratorio de los datos (EDA) se logra comprender la estructura del dataset, evaluar su calidad y extraer patrones preliminares útiles para el diseño posterior de características y modelos, para ello, se siguió un enfoque estructurado que combinó técnicas estadísticas, gráficas y computacionales, implementadas principalmente en Databricks con Apache Spark y Python.

### Caracterización inicial del dataset

El conjunto de datos analizado está compuesto por 5 millones de transacciones financieras y 18 variables. Se verificó la presencia de la variable objetivo “is\_fraud”, identificando un alto desbalance de clases, con fraudes representando solo una fracción mínima respecto a transacciones legítimas. Asimismo, se detectaron dos columnas con valores faltantes, lo que orientó el proceso de limpieza y preparación.

### Análisis univariado

Se calcularon estadísticos descriptivos (media, mediana, desviación estándar, percentiles) y se evaluaron distribuciones mediante histogramas y boxplots. Entre los hallazgos más relevantes:

* La variable “amount” presentó alta asimetría, con valores extremos y colas largas.
* La mayoría de las transacciones ocurrieron en horarios regulares, aunque se detectaron picos en horas nocturnas y fines de semana.
* Algunas variables categóricas como “transaction\_type” o “merchant\_category” evidenciaron alta cardinalidad, lo que requiere tratamiento en etapas posteriores.

### Análisis univariado

Se exploraron las relaciones de las variables con el objetivo “is\_fraud”, encontrando que:

* Los fraudes se concentran en montos elevados y en cuentas con alta frecuencia de transacciones en periodos cortos.
* La correlación entre variables numéricas resultó moderada, sin problemas críticos de multicolinealidad.
* El cruce entre variables de dispositivo y ubicación reveló patrones de cambio inusual, asociados a potenciales fraudes.

### Visualización de patrones relevantes

Se utilizaron mapas de calor, diagramas de dispersión y series temporales, identificando:

* Reincidencia de cuentas emisoras en múltiples fraudes.
* Picos de fraude en horarios no laborales.
* Formación de clusters de transacciones sospechosas con características similares.

### Hallazgos clave del EDA

* Confirmación del desbalance de clases, justificando el uso posterior de técnicas de re-muestreo o ajuste de pesos.
* Identificación de outliers legítimos que deben considerarse como señales de riesgo en lugar de eliminarlos.
* Detección de variables con alto poder discriminativo, como monto, frecuencia de transacciones y diversidad de dispositivos.

En la sección de anexos se incluyen los gráficos resultantes del análisis exploratorio, los cuales respaldan y complementan los hallazgos descritos en este documento.

## 6.2 Preprocesamiento

El pipeline de preprocesamiento inicia en preprocesamiento\_fraude\_big\_data(...) con dos principios clave: evitar fuga de información y eficiencia en big data. Para esto, incorpora un bloque de muestreo estratificado opcional, que permite trabajar con subconjuntos representativos sin alterar la prevalencia del fraude, y un módulo de optimización de memoria, que convierte columnas a category cuando es rentable. Además, se implementa un filtro de variables tipo “etiqueta disfrazada” usando normalización de strings y expresiones regulares, con el fin de excluir columnas que puedan anticipar la clase.

En la fase de limpieza y normalización, las columnas numéricas se procesan con funciones vectorizadas (to\_float\_vectorized), eliminando símbolos y forzando conversiones robustas. La variable objetivo se estandariza siempre a binario (0/1), asegurando consistencia aun cuando llega en formato texto, categórico o numérico. Si existe un timestamp, se extraen rasgos temporales (hora, día de la semana, bandera de fin de semana y horarios críticos), además de una codificación cíclica mínima de la hora. Antes de cualquier cálculo de estadísticas se realiza un split estratificado temprano, de manera que todas las transformaciones se ajustan en train y se aplican en test, protegiendo contra fuga.

El feature engineering agrega señales económicas pero potentes: promedios, conteos y desviaciones por emisor, coeficientes de variación y z-scores de montos, además de indicadores de diversidad de dispositivos y banderas de múltiples dispositivos. Estas características se calculan solo con train y se transfieren a test vía mapeo, nunca recalculando sobre datos de evaluación.

En la preparación de features, se eliminan identificadores, timestamps, IPs y toda columna no modelable, junto con las detectadas como label-like. Las variables categóricas se dividen en baja y alta cardinalidad: las primeras se one-hot-encodean y las segundas reciben frequency encoding. Para variables numéricas, se imputan medianas y se agrega una señal de “transacción inusualmente grande” comparada contra un umbral de cuantiles. Las variables numéricas se escalan con RobustScaler, más resistente a outliers típicos de fraude.

El pipeline contempla el balanceo de clases mediante undersampling o sample weights, según el caso, y construye tanto representaciones sparse como densas para máxima compatibilidad con distintos algoritmos. Finalmente, todos los objetos, listas de columnas, thresholds, encoders y escaladores se registran en un diccionario de artefactos para garantizar reproducibilidad y trazabilidad. El resultado final es un conjunto consistente y seguro: X\_train\_sparse\_res, X\_test\_sparse, y\_train\_res, y\_test y artefactos, listos para modelar y optimizar los umbrales de decisión según el criterio de negocio.

## 6.3 Modelos / Algoritmos

Este proyecto contemple un problema de clasificación binaria por lo que, para cumplir con el objetivo, se utilizarán diferentes técnicas supervisadas cómo:

* Regresión Logística: Utilizando funciones de regresión calcula la probabilidad (entre 0 y 1) de pertenecer a una clase.
* Random Forest: Mediante árboles de decisión en paralelo usando muestras aleatorias de los datos, cada árbol indica una clase y la que tenga más votos es la predicción.
* Gradient Boosting: Utiliza un conjunto de árboles de decisión entrenados de forma secuencial, donde cada nuevo árbol corrige los errores del anterior mediante la optimización de una función de pérdida. Maneja de una mejor forma datos que estén desbalanceados.

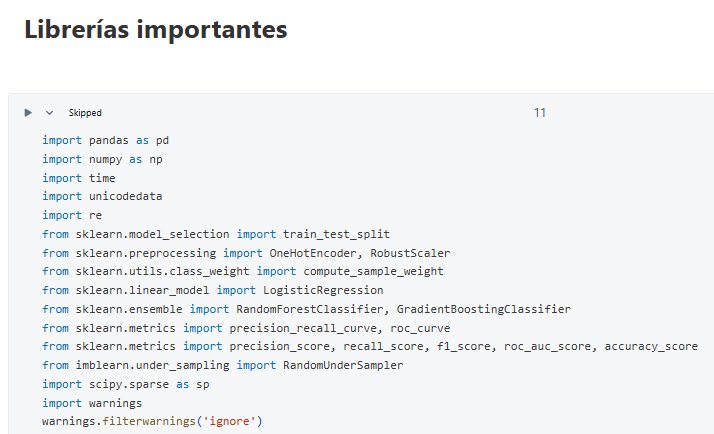
## 6.4 Evaluación

Cada algoritmo de clasificación se evalúa frente al set de prueba una sola vez, no hay cross-validation. Luego utiliza las métricas de evaluación anteriormente mencionadas en cada modelo para luego guardar los resultados en un diccionario.

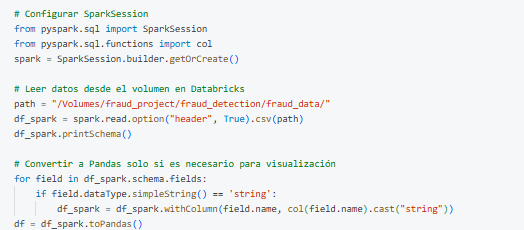
Se determinó que las evaluaciones de sensibilidad o recall y F1-score son óptimas para determinar la efectividad del algoritmo en este escenario. Se busca que el algoritmo detecte la mayor cantidad de fraudes pero que no sature con muchos falsos positivos.

# VII. Desarrollo

Algo importante a tomar en cuenta son las múltiples librerías que se utilizaron para el desarrollo en general.



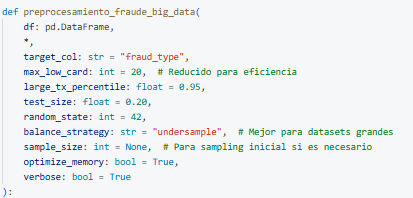
El proceso de desarrollo incluye la carga de datos en la que se utiliza el siguiente código:



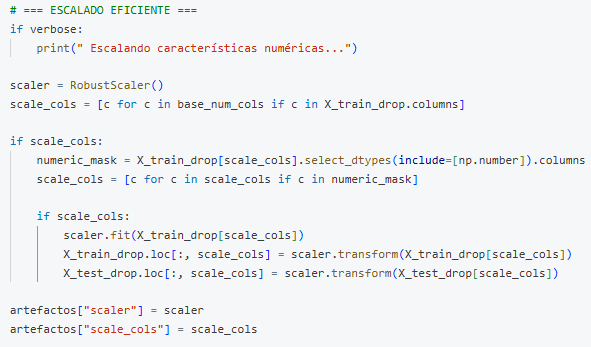
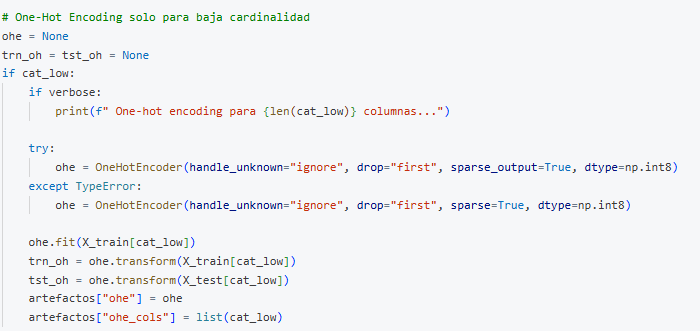
Después de la carga se procede con el EDA que se corre a través del siguiente comando, pues es un notebook creado dentro del workspace.



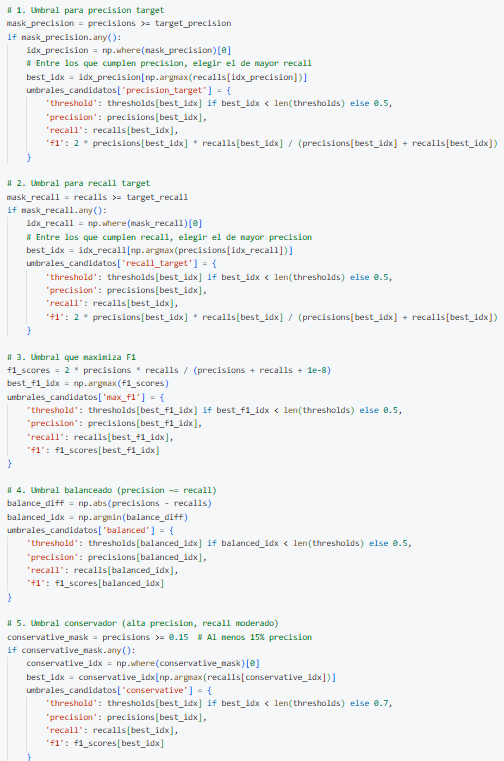
El EDA fue la base para el preprocesamiento el cual se describe en una de las secciones anteriores. Sin embargo, se puede resaltar el uso de parámetros dentro de la función general como por ejemplo large\_tx\_percentile que se utilizó para definir qué es una “transacción grande”. (5% restante que son outliers) o max\_low\_card que corresponse al umbral para decidir qué variables categóricas son de baja cardinalidad (menor o igual a 20).



En esta etapa (preprocesamiento) también se hizo uso de técnicas para escalado de la mano de RobustScaler y one hot encoding para bajas cardinalidades.



Durante la optimización se usó varios tipos de umbrales con el objetivo de evitar extremos como overfitting y underfitting. Estos se demuestran en la siguiente figura.



Un entrenamiento en base a modelos capaces de tener class weight debido a limitaciones de capacidad.



También se incluyó un análisis de costo beneficio para entender los impactos en un negocio en base a supuestos.



Durante el desarrollo del proyecto enfrentamos varios retos técnicos, principalmente relacionados con el desbalance de clases, la escala de los datos y las limitaciones de recursos en la versión Free Edition de Databricks. Para manejar el desbalance se implementaron técnicas como undersampling y el uso de class\_weight="balanced", además de la optimización de umbrales bajo distintos criterios (precision target, recall target, F1, entre otros) para alinear los resultados con los objetivos de negocio. En cuanto a la elección entre Pandas y Spark, se optó por trabajar mayormente con Pandas optimizado (vectorización, reducción de memoria, matrices sparse, frequency encoding), dejando a Spark como alternativa para futuros escenarios de mayor escala.

Otro reto importante fue el tamaño del dataset, que ocasionaba errores de memoria y tiempos excesivos de ejecución. Para resolverlo, se implementó un subset estratificado opcional que permitió prototipar manteniendo la proporción real de fraude, junto con el uso de tipos de datos compactos y matrices dispersas que redujeron el consumo de recursos. Además, se incluyeron procesos robustos de limpieza y normalización para columnas ruidosas y se evitó la fuga de información aplicando todos los cálculos estadísticos únicamente sobre el conjunto de entrenamiento. Estas medidas, combinadas con el modularidad del pipeline, permitieron obtener resultados confiables pese a las restricciones, manteniendo una ruta clara de escalabilidad hacia Spark y Databricks ML.

# VIII. Resultados

Se evaluó un pipeline optimizado para la detección de fraude, diseñado para maximizar el Retorno de Inversión (ROI) y minimizar las alertas innecesarias que podrían saturar los sistemas de revisión manual de transacciones. Este pipeline incluyó técnicas de preprocesamiento avanzadas, balanceo de clases, ingeniería de características y ajuste de umbrales de decisión.

## 8.1 Modelos evaluados y desempeño

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Modelo | Precisión | Recall | Falsos positivos | ROI estimado | Comentarios |
| Logistic Regression | 3.6% | 65% | Alto | 20% | Mejoró ligeramente la precisión, pero genera muchas alertas falsas. |
| Random Forest | 4.4% | 68% | Alto | 25% | Similar a la regresión logística, sin un impacto significativo en la eficiencia operativa. |
| Gradient Boosting (estrategia conservadora) | 15.4% | 35% | Muy bajo (13 alertas) | 81.8% | Modelo más eficiente: alta precisión y ROI, aunque detecta menos fraudes (recall bajo). |

## 8.2 Comparación con la estrategia anterior (undersampling)

* La estrategia basada en undersampling generaba una **precisión muy baja** (3.5–4.4%) y un **recall moderado** (63–69%).
* Esto resultaba en **más de 46,000 falsas alertas**, lo que saturaba el sistema de revisión y aumentaba los costos operativos.
* Aunque el modelo detectaba muchos fraudes, la gran cantidad de falsos positivos hacía que su implementación práctica fuera limitada y poco eficiente.

## 8.3 Resultados con la estrategia optimizada (class weights + umbral ajustado)

* **Logistic Regression y Random Forest:** mejoraron ligeramente la precisión, pero todavía generaban demasiadas alertas falsas, lo que limitaba su utilidad operativa.
* **Gradient Boosting Conservador:** logró reducir drásticamente las alertas innecesarias, limitándolas a solo 13. Alcanzó una **precisión del 15.4%**, lo que indica que la mayoría de las alertas generadas correspondían a fraudes reales. Sin embargo, el **recall fue bajo**, detectando menos fraudes totales, pero maximizando el retorno de inversión y la eficiencia de la operación.

8.4 Interpretación de resultados

1. **Precisión vs Recall:** La estrategia conservadora prioriza la **calidad de las alertas** (precisión alta) sobre la cantidad de fraudes detectados (recall más bajo). Esto es crucial en entornos financieros donde las revisiones manuales son costosas y saturar el sistema puede generar pérdidas adicionales.
2. **Impacto en operaciones:** Reducir los falsos positivos significa menor tiempo de revisión y menos recursos desperdiciados, aumentando la eficiencia operativa.
3. **ROI:** La optimización de pesos de clase y umbral de decisión permite alinear métricas técnicas con objetivos de negocio, logrando un ROI de **81.8%** y un beneficio neto estimado de **$900**, lo que demuestra que la estrategia conservadora es la más rentable.
4. **Selección de modelo:** Aunque Gradient Boosting tiene un recall más bajo, su mayor precisión y menor número de alertas lo convierte en el **modelo más recomendado** para implementación real. Esto muestra que, en problemas de fraude financiero, el **balance entre métricas y criterios de negocio** es clave para la toma de decisiones.

# IX. Conclusiones y trabajo futuro

El proyecto logró cumplir con los objetivos planteados, tanto el general como los específicos. Se diseñó un pipeline integral de análisis de datos y modelado predictivo capaz de detectar transacciones fraudulentas, aplicando técnicas de limpieza, ingeniería de características, balanceo de clases y entrenamiento de modelos supervisados.

Entre los hallazgos más relevantes:

* La variable objetivo “is\_fraud” mostró un fuerte desbalance de clases, lo que condicionó el rendimiento de los modelos y justificó la aplicación de técnicas como undersampling, class weights y optimización de umbrales.
* Gradient Boosting, bajo una estrategia conservadora, presentó el mejor desempeño en términos de Retorno de Inversión (ROI), logrando un balance adecuado entre la detección de fraudes y la minimización de falsos positivos.
* Las características más discriminativas para identificar fraude incluyeron el monto de la transacción, la frecuencia de uso de la cuenta, la diversidad de dispositivos y cambios geográficos inusuales.

Se sugiere utilizar el modelo que haya demostrado mayor ROI en el análisis de negocio, aplicando el umbral de decisión optimizado para maximizar la detección de fraudes con el menor impacto en falsos positivos. Además, es esencial monitorear continuamente la precisión en producción y ajustar el umbral cuando sea necesario para adaptarse a cambios en el comportamiento de los datos.

Como siguiente paso, se recomienda aprovechar Databricks ML para la gestión del ciclo de vida de los modelos y evaluar la posibilidad de escalar la infraestructura con una licencia empresarial de Databricks, lo que permitirá integrar más fácilmente pipelines de Machine Learning en producción. Finalmente, se sugiere realizar una comparativa de técnicas de optimización (como Grid Search, Random Search y Algoritmos Genéticos) junto con un dashboard interactivo para validar la robustez del modelo y asegurar que la estrategia seleccionada sea la más eficiente en términos de precisión y costo computacional.

El trabajo desarrollado demuestra la importancia de combinar metodologías estadísticas, computacionales y de negocio para abordar problemas reales en el ámbito financiero. La integración de Databricks, Spark y Python permitió manejar grandes volúmenes de datos y construir un pipeline reproducible y escalable. En conclusión, el proyecto no solo logró desarrollar un sistema de detección de fraude efectivo, sino que también estableció una base sólida para mejoras futuras y la implementación de soluciones de Machine Learning confiables y alineadas con los objetivos estratégicos del sector financiero.

# X. Referencias

Arabella. (2023). *Preprocesamiento de datos: Definición y explicación*. TechLib. <https://techlib.net/techedu/preprocesamiento-de-datos/>

Chauhan, N. S. (2023). *Métricas de evaluación de modelos en el aprendizaje automático*. DataSource.ai. <https://www.datasource.ai/es/data-science-articles/metricas-de-evaluacion-de-modelos-en-el-aprendizaje-automatico>

Flejoles, R. P. (2019). *Introduction to algorithms*. Ashland: Arcler Press. <https://research.ebsco.com/linkprocessor/plink?id=d9109a62-84db-30b9-9a27-ad2b9bbdc2a4>

Jafari, R. (2022). *Hands-On Data Preprocessing in Python: Learn how to effectively prepare data for successful data analytics*. Birmingham: Packt Publishing. <https://research.ebsco.com/linkprocessor/plink?id=868b15cc-ad12-3625-8028-d2983f141d6d>

López Murphy, J. J. y Zarza, G. (2017). La ingeniería del big data: cómo trabajar con datos. Editorial UOC. <https://elibro.net/es/lc/ulead/titulos/59093>

Microsoft. (2023). *Recomendaciones para optimizar el rendimiento de los datos*. Microsoft Learn. <https://learn.microsoft.com/es-es/azure/well-architected/performance-efficiency/optimize-data-performance>

Microsoft. (2025). *Get free Databricks training*. Microsoft Learn. <https://learn.microsoft.com/es-es/azure/databricks/getting-started/free-training>

Microsoft. (2025). *ETL para datos relacionales*. Microsoft Learn. <https://learn.microsoft.com/es-es/azure/architecture/data-guide/relational-data/etl>

Macías, M. y Gómez, M. (2015). Introducción a Apache Spark: para empezar a programar el big data. Editorial UOC. Clásico última edición <https://elibro.net/es/lc/ulead/titulos/58454>

Nelli, F. (2018). *Python Data Analytics: with Pandas, NumPy, and Matplotlib*. Apress. <https://ulead.kohahosting.info/cgi-bin/koha/opac-detail.pl?biblionumber=1406>

Orallo, J., Quintana, M., & Ramírez, C. (2004). *Introducción a la minería de datos*. Madrid: Pearson Prentice Hall. <https://ulead.kohahosting.info/cgi-bin/koha/opac-detail.pl?biblionumber=1350>

Pykes, K. (2025). *Data preprocessing: A complete guide with Python examples*. DataCamp. <https://www.datacamp.com/blog/data-preprocessing>

Servicetec. (2024). *Interpretación y comunicación de modelos de clasificación en R*. Servicetec. <https://servicetec.es/interpretacion-y-comunicacion-de-modelos-de-clasificacion-en-r/>

# XI. Anexos

En adición a el archivo de databricks también se puede acceder al código desde el siguiente repositorio en GitHub: <https://github.com/KrisSo03/Proyecto_Big-Data_Grupo-4>

Análisis exploratorio salidas:

Gráfico 1: Cantidad de fraudes por mes

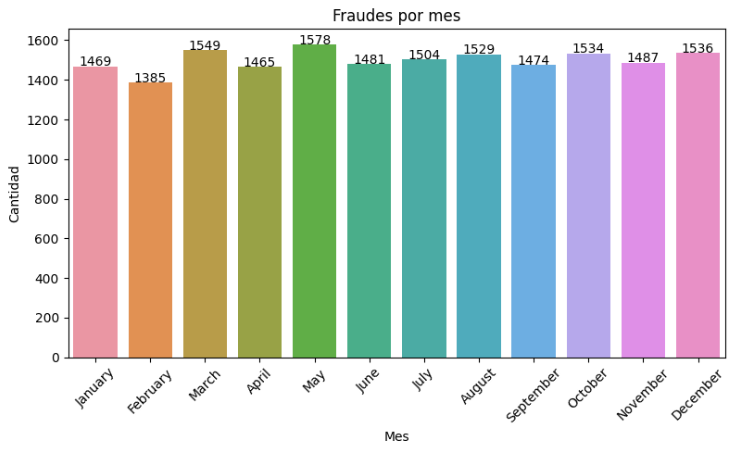


Gráfico 2: Fraudes por franja horaria

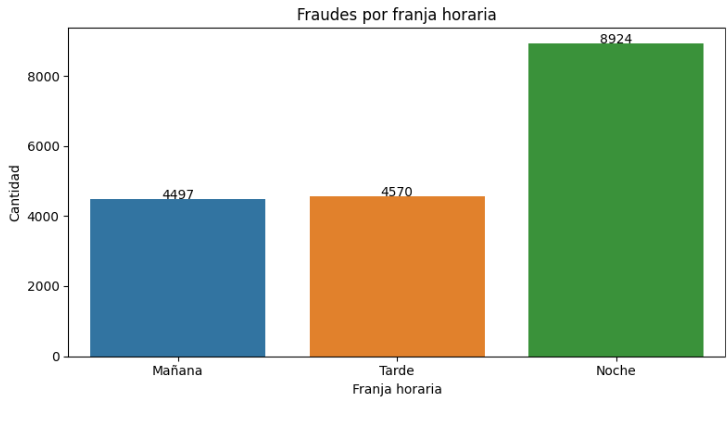


Gráfico 3: Distribución de transacciones de acuerdo con la muestra utilizada.

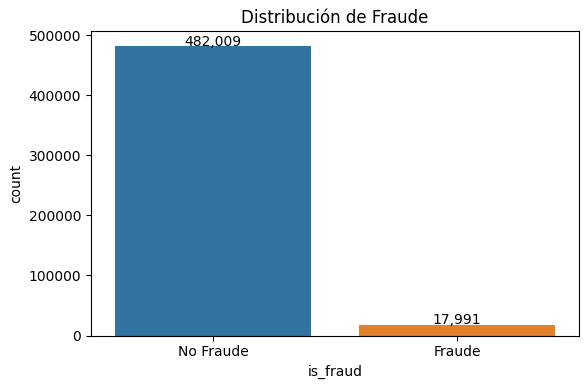


Gráfico 4. Ubicaciones con mayor número de fraudes.

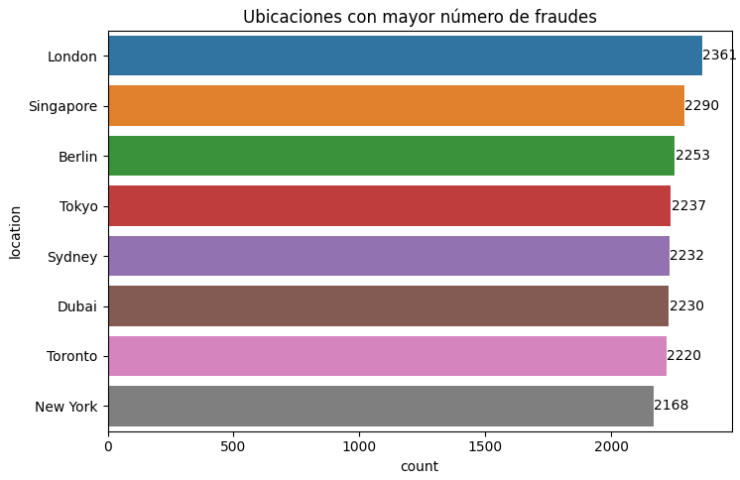


Gráfico 5. Dispositivos más utilizados en transacciones fraudulentas.

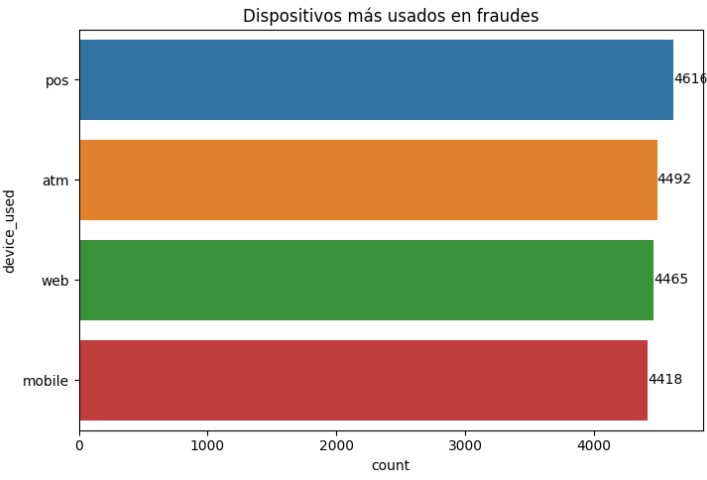


Gráfico 6. Matriz de correlación con la variable objetivo “is\_fraud” para las variables numéricas

