PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE HÁBITAT, INFRAESTRUCTURA Y CREATIVIDAD

CARRERA DE: Ingeniería en Sistemas de Información



Trabajo de Titulación

Tema: Desarrollo de un modelo predictivo basado en aprendizaje automático para la detección de fraudes en transacciones con tarjeta de crédito

AUTOR:

Fabricio Sebastián Montalvo Valdiviezo

DIRECTOR:

Ing. Fabián de la Cruz

QUITO D.M. JULIO 2026

Contenido

Tema del Proyecto 4
Capítulo 1. Introducción 4
1.1 Planteamiento del Problema 4
1.2 Justificación 5
1.3 Objetivos 7
Objetivo General 7
Objetivos Específicos 7
1.4 Alcance del Proyecto 8
1.5 Estructura del documento 9
Capítulo 2. Marco Teórico y Conceptual 10
2.1 Fraude financiero y el impacto en los sistemas de pago 10
2.2 Fundamentos del aprendizaje automático supervisado 10
2.3 Desbalance de clases en datos transaccionales 11
2.4 Modelos de clasificación 12
2.5 Métricas de evaluación de modelos 13
2.6 Operaciones de Machine Learning (MLOps) 14
2.7 Herramientas utilizadas 14
Capítulo 3. Metodología 16
3.1 Definición de CRISP-DM 16
3.2 Características de la metodología 17
3.3 Fases de CRISP-DM 17
3.4 Importancia de CRISP-DM en proyectos de Machine Learning 18
Capítulo 4. Implementación y Despliegue (Aplicación CRISP-DM) 20
4.1 Enfoque metodológico 20
4.2 Fuente de datos 20
4.3 Fase 1 Comprensión del Negocio 21
4.4 Fase 2 Comprensión de los datos 21
4.5 Fase 3 Preparación de los Datos 26
4.6 Fase 4: Modelado 33
4.6.1 Modelo 1: Regresión Logística 37
4.6.2 Modelo 2: Random Forest 39

4.6.3 Modelo 3: XGBoost 41
4.7 Fase 5: Evaluación 44
4.7.1 Matrices de Confusión 45
4.7.2 Reportes de Clasificación 48
4.7.3 Curvas ROC-AUC 51
4.8 Fase 6: Despliegue 54
4.8.1 Arquitectura general del sistema 54
4.8.2 Tecnologías utilizadas en el despliegue 56
4.8.3 Simulación de autenticación y control de acceso 57
4.9 Módulo complementario para detección de lavado de activos 64
4.9.1 Contexto del módulo AML 65
4.9.2 Preparación resumida del dataset IBM AML 66
4.9.3 Evaluación comparativa de modelos 66
4.9.4 Análisis visual de variables relevantes 71
4.9.5 Integración del módulo AML en el dashboard ejecutivo 73
Capítulo 5. Conclusiones y Recomendaciones 76
5.1 Conclusiones 76
5.2 Recomendaciones 77
Referencias bibliográficas 79

Tema del Proyecto

Desarrollo de un modelo predictivo basado en aprendizaje automático para la detección de fraudes en transacciones con tarjeta de crédito

Capítulo 1. Introducción

En este capítulo se presenta una introducción general al problema de la detección de fraude en transacciones financieras, abordando el contexto actual, la problemática existente y la necesidad de aplicar soluciones basadas en el aprendizaje automático. Además, se da la justificación del proyecto, así como los objetivos que guían al desarrollo del modelo propuesto.

1.1 Planteamiento del Problema

En la actualidad, el enorme crecimiento de las tecnologías tanto digitales como el comercio electrónico han transformado la forma en como las personas realizan transacciones financieras o como usan y mueven su dinero. Las plataformas de pago en línea como servicios bancarios digitales como la Banca Móvil del Banco Pichincha aquí en Ecuador procesan millones de operaciones por segundo, generando un entorno muy dinámico y complejo. Sin embargo, este notable avance trae consigo múltiples vulnerabilidades que pueden y han sido aprovechadas por actores maliciosos que desarrollan de forma regular estrategias para cometer fraude.

Los métodos tradicionales de detección de fraude, basados en las reglas fijas o revisiones manuales actualmente presentan considerables carencias y limitantes frente a la sofisticada forma de ataque actual por los actores maliciosos. Estos sistemas suelen depender de condiciones predefinidas, como montos máximos o ubicaciones geográficas, lo que los hacen muy rígidos y poco flexibles ante las nuevas formas y estrategias de fraude. Estos pueden generar una alta tasa de falsos positivos, afectando la experiencia del usuario, o bien fallar en la detección temprana de transacciones fraudulentas reales, ocasionando pérdidas económicas considerables.

Adicionalmente, la naturaleza masiva de los datos financieros dificulta en gran parte a la identificación manual de patrones sospechosos. En este escenario surge la necesidad de implementar soluciones automatizadas capaces de analizar grandes volúmenes de información

con predicción inmediata sobre datos cargados por archivo CSV y detectar comportamientos anómalos de forma eficiente.

Con esta problemática, el aprendizaje automático se convierte en una alternativa viable, ya que permite a los sistemas a aprender patrones a partir de los datos históricos y así lograr adaptarse a nuevas técnicas, tendencias o estrategias de fraude. Sin embargo, el desarrollo e implementación de estos modelos no solo implica en entrenar el algoritmo sino lograr que su integración en sistemas funcionales sea factible para el uso de entornos reales.

Ante este contexto, se plantea la siguiente pregunta de investigación:

¿Cómo diseñar e implementar un sistema basado en aprendizaje automático que permita detectar de manera temprana y eficiente las transacciones fraudulentas con tarjeta de crédito en grandes volúmenes de datos financieros, integrando modelos predictivos en un entorno desplegable y adaptable a nuevos datos?

El presente trabajo busca responder está pregunta mediante el diseño e implementación de un modelo predictivo que permita identificar patrones sospechosos, clasificar transacciones como legítimas o fraudulentas y evaluar la efectividad del modelo mediante métricas de precisión, recall y AUC-ROC.

De esta manera, el sistema propuesto podrá servir como una herramienta de apoyo para entidades financieras o plataformas digitales que deseen mejorar sus mecanismos de prevención y seguridad en línea.

1.2 Justificación

El crecimiento acelerado del comercio electrónico y de las transacciones digitales ha traído consigo un aumento significativo en los casos de fraude financiero. Miles de usuarios son víctimas de transacciones no autorizadas, suplantaciones de identidad o manipulación de datos bancarios, lo que genera pérdidas económicas y afecta la confianza en los sistemas de pago en línea.

Frente a esta problemática, surge la necesidad de aplicar tecnologías inteligentes que permitan detectar patrones de comportamiento anómalos en grandes volúmenes de datos. El uso de algoritmos de aprendizaje automático (Machine Learning) permite desarrollar modelos capaces de identificar patrones ocultos en grandes volúmenes de datos, mejorando significativamente la capacidad de detección de fraudes.

Además, los algoritmos de clasificación supervisada ofrecen la posibilidad de analizar

transacciones históricas y aprender de ellas, permitiendo predecir la probabilidad de que sea fraude en nuevas operaciones. Esto ayuda en la automatización de procesos de detección y reduce en gran medida la dependencia de la intervención humana. Constantemente las formas de fraude evolucionan en gran medida que herramientas como el aprendizaje automático (Machine Learning) no son algo opcional sino algo indispensable para el mundo actual.

Este proyecto busca desarrollar un modelo predictivo utilizando técnicas de clasificación supervisada.

Justificación técnica

El incremento de la complejidad en patrones de fraude financiero ha puesto en evidencia las limitaciones de los métodos tradicionales de detección. Los sistemas que están basados en reglas no son capaces de adaptarse de forma constante a las nuevas formas de fraude, lo que a la larga reduce su efectividad en entornos flexibles o dinámicos. En este contexto, el uso de técnicas de aprendizaje automático permite desarrollar modelos que son capaces de identificar patrones ocultos en grandes volúmenes de datos, mejorando de forma significativa la capacidad de detección de fraudes.

Además, los algoritmos contribuyen al desarrollo de competencias en el diseño e implementación de sistemas inteligentes, así como la evaluación de modelos predictivos mediante métricas adecuadas.

Justificación académica

Desde el punto de vista académico, el desarrollo de este proyecto me permite aplicar de manera práctica todos mis conocimientos adquiridos durante mi formación en la carrera de Ingeniería en Sistemas de Información, adaptando e integrando áreas de minería de datos que se profundizo en la cátedra de BI (Inteligencia de negocios), estadísticas, inteligencia artificial y el desarrollo e ingeniería de software. Asimismo, el uso de la metodología CRISP-DM proporciona un visión o enfoque más estructurado para soluciones basadas en datos debido a que esta metodología sigue un proceso de 6 pasos muy detallados que ayudan a alinear el objetivo de instituciones financieras con la problemática de minería de datos.

El proyecto también contribuye al desarrollo de competencias en el diseño e implementación de sistemas inteligentes y en la evaluación de modelos predictivos mediante métricas adecuadas.

Justificación social

A nivel social, el fraude financiero representa una problemática que afecta tanto a instituciones como a usuarios individuales. Las pérdidas económicas, la vulneración de datos personales y la pérdida de confianza en los sistemas de información digitales son algunas de las principales consecuencias de este fenómeno que si no es controlado puede traer problemas aún más grandes.

El desarrollo de sistemas más eficientes para la detección de fraude contribuye a fortalecer la seguridad digital, reducir riesgos y así generar más confianza en el usuario con el uso de plataformas financieras. En este punto, el presente proyecto tiene un gran impacto positivo al proponer una solución tecnológica que está orientada en mejorar la protección de los usuarios con sus datos más sensibles y a las organizaciones.

1.3 Objetivos

Objetivo General

Desarrollar un sistema inteligente desplegable basado en las técnicas de aprendizaje automático que permita detectar transacciones fraudulentas integrando un pipeline automatizado en el procesamiento de datos, entrenamiento dinámico del modelo y un módulo de inferencia accesible mediante interfaz web.

Objetivos Específicos

Analizar y preprocesar el dataset seleccionado para identificar variables relevantes, corregir valores atípicos y preparar la información para los algoritmos de clasificación.

Implementar modelos supervisados de clasificación para la detección de fraude en transacciones financieras.

Comparar el desempeño de los algoritmos mediante métricas como precisión, recall, F1-Score y AUC-ROC, con el fin de determinar el modelo más eficiente.

Diseñar e implementar un prototipo funcional mediante una interfaz gráfica que permitirá realizar predicciones sobre nuevas transacciones, tanto de forma individual o mediante la carga de archivos csv.

Desarrollar pipelines automatizados para el procesamiento de datos, entrenamiento y evaluación de modelo.

Implementar un mecanismo de actualización del modelo que permita incorporar nuevos datos y generar nuevas versiones del modelo sin intervención directa en entornos de desarrollo.

Evaluar el comportamiento del modelo mediante comparación de métricas con

versiones anteriores.

Evaluar el comportamiento del modelo en la fase de despliegue en el entorno del prototipo, mediante el análisis de métricas e indicadores generados a partir de nuevas predicciones.

Evaluar de manera complementaria, la aplicabilidad de los modelos de aprendizaje automático en la detección de posibles operaciones de lavado de activos mediante el conjunto de datos IBM AML.

1.4 Alcance del Proyecto

Este proyecto implica el diseño, desarrollo y evaluación de un sistema inteligente basado en aprendizaje automático para detectar transacciones fraudulentas con tarjetas de crédito, organizado en torno a la metodología CRISP-DM. Como complemento al sistema principal, se incorpora un módulo complementario para la detección de posibles operaciones de lavado de activos (AML), con el propósito de analizar la aplicabilidad de las mismas técnicas de aprendizaje automático en otro escenario financiero relacionado con la prevención de delitos económicos.

INCLUYE

Procesamiento y transformación de datos (Pipeline): Se creó rutinas automatizadas en Python para limpiar datos, manejar excepciones mediante un escalado robusto y aplicar por separado técnicas de equilibrio agregado durante la fase de capacitación.

Entrenamiento y selección de modelos supervisados: Implementación y evaluación comparativa de tres algoritmos predictivos de clasificación, selección de modelo óptimo basado en métricas orientadas al negocio (Recall y AUC-ROC).

Módulo de predicción: Desarrollo de un entorno que procese lotes de transacciones diarias, verifique la integridad del formato frente a los patrones esperados y genere alertas de fraude utilizando el modelo predictivo entrenado.

Módulo de reentrenamiento del modelo: Se desplegó una interfaz que permite cargar nuevos conjuntos de datos históricos para que el modelo éste actualizado. Este módulo ejecuta el pipeline de forma automática para el procesamiento de datos, incluyendo partes esenciales como la limpieza, transformación y preparación de variables, seguido del reentrenamiento del modelo seleccionado. De esta forma es posible generar versiones actualizadas del modelo sin la necesidad de ir directamente a un Jupyter o intervenir en el entorno del desarrollo.

GUI funcional: Construcción de una aplicación web interactiva mediante Streamlit para interactuar con los módulos de predicción y reentrenamiento, presentando los resultados

y las alertas mediante tablas e indicadores visuales.

Módulo complementario de detección de lavado de activos (AML): Implementación de un módulo de análisis utilizando el conjunto de datos IBM AML, con fines de validación y demostración, para evidenciar la aplicabilidad de los modelos de aprendizaje automático en un segundo escenario de riesgo financiero relacionado con la prevención del lavado de activos.

EXCEPCIONES

Procesamiento de transacciones en tiempo real (transmisión): El sistema procesará las transacciones descargando archivos grandes (procesamiento por lotes) y no estará conectado a un bus de datos activo ni a un sistema de pago instantáneo.

Admite variables no numéricas: El sistema está diseñado para extraer patrones sólo de datos cuantitativos continuos o discretos. El modelo no procesará variables como texto libre, nombres de clientes o fechas sin procesar.

Sistema de autenticación empresarial: Al tratarse de un prototipo funcional orientado al análisis y validación del modelo predictivo, la aplicación no implementa mecanismos avanzados de autenticación, gestión de usuarios, control de los roles o administración segura de credenciales como los utilizados en entornos bancarios.

1.5 Estructura del documento

En el presente proyecto se ahondará de forma estructurada y completa el análisis de los datos financieros, la descripción del problema, la metodología que se utilizará y la solución a lo planteado en los diferentes capítulos que se presentará ahora:

Capítulo 1. Introducción: presenta la descripción del problema, la justificación y el alcance del trabajo.

Capítulo 2. Marco Teórico y Conceptual: describe los fundamentos conceptuales relacionados con el fraude financiero, el aprendizaje automático, la detección de anomalías y las tecnologías utilizadas en el desarrollo del sistema.

Capítulo 3. Metodología: explica de forma detallada la metodología que se utilizará a lo largo del trabajo que es CRISP-DM con cada una de sus fases, y su importancia dentro del trabajo.

Capítulo 4. Implementación y Despliegue: presenta la aplicación de CRISP-DM dentro del trabajo la estructura y solución con los datos utilizados, explicación detallada de métricas y uso de los modelos dentro del sistema creado.

Capítulo 5. Conclusiones y Recomendaciones: resume los principales hallazgos de la

investigación, las conclusiones derivadas de los resultados obtenidos y las recomendaciones para futuras mejoras o trabajos relacionados.

Capítulo 2. Marco Teórico y Conceptual

En este capítulo se va a presentar los fundamento teóricos y conceptuales necesarios para el desarrollo del trabajo, abordando temas relacionados al fraude financiero, el aprendizaje automático, modelos de clasificación y métricas de evaluación.

2.1 Fraude financiero y el impacto en los sistemas de pago

Los fraudes en transacciones electrónicas constituyen una de las principales amenazas para los sistemas financieros. Las entidades bancarias procesan grandes volúmenes de transacciones diariamente, lo que incrementa el riesgo de actividades ilícitas como la suplantación de identidad, transacciones no autorizadas o la manipulación de datos (Bhattacharyya et al., 2011).

Entre los tipos de fraude más comunes están el fraude con tarjetas de crédito en entornos sin presencia física del titular, el phishing orientado a la obtención de credenciales bancarias y la clonación de tarjetas mediante dispositivos de captura de datos. Estos ataques están en constante evolución lo que dificulta la identificación mediante métodos tradicionales basados en reglas estáticas (Delamaire, Abdou & Pointon, 2009).

Según **Tom M. Mitchell** (1997), el análisis de grandes volúmenes de datos necesita métodos automatizados capaces de identificar patrones ocultos. En este sentido, el uso de técnicas de aprendizaje automático permite analizar una gran cantidad de información transaccional y detectar comportamientos sospechosos que no serían fáciles de encontrar con el análisis manual.

La enorme evolución de los atacantes en sus métodos exige la implementación de mecanismos avanzados que permitan detectar comportamientos sospechosos mediante predicciones generadas a partir de datos cargados por archivos CSV. De acuerdo con **Foster Provost** y **Tom Fawcett** (2013), el buen uso de modelos predictivos en los sistemas financieros permite mejorar la detección de eventos raros, como el fraude, reduciendo grandes pérdidas económicas y fortaleciendo la seguridad digital.

2.2 Fundamentos del aprendizaje automático supervisado

El aprendizaje automático o mejor conocido como machine learning, es una rama de la inteligencia artificial que ayuda a los sistemas aprender patrones a partir de los datos sin ser

programados explícitamente. Según **Tom M. Mitchell** (1997), un programa logra aprender de la experiencia si mejora su efectividad en una tarea específica.

Dentro de este campo, el aprendizaje automático puede clasificarse en supervisado y el no supervisado. El aprendizaje supervisado usa generalmente datos con etiquetas, donde el modelo aprende a partir de ejemplos previamente clasificados. Por el otro lado, los modelos supervisados buscan identificar estructuras ocultas en los datos sin etiquetas (Géron, 2019).

En el contexto de la detección de fraude financiero, el aprendizaje supervisado resulta ser útil, debido a que permite modelos utilizando transacciones previamente clasificadas como legítimas o fraudulentas (Provost & Fawcett, 2013).

El proceso general del aprendizaje supervisado incluye algunas etapas de recolección de datos, el preprocesamiento, el entrenamiento del modelo, la validación y predicción. Según Géron (2019), la calidad de los datos y la adecuada selección del modelo son clave para el éxito de los sistemas que están basados en el aprendizaje automático.

De igual forma el uso de estos modelos ayuda a superar las limitaciones de los sistemas tradicionales basados en reglas que generalmente suelen ser muy rígidos y poco flexibles frente a las técnicas de fraude.

2.3 Desbalance de clases en datos transaccionales

Un principal desafío en la detección de fraude es el desbalance de clases, ya que las transacciones fraudulentas representan un mínimo en comparación a las legítimas. Este problema puede afectar de forma significativa el desempeño de los modelos predictivos.

En escenarios reales, es común que los conjuntos de datos tengan el 95% de sus transacciones legítimas, esto puede llevar a que un modelo aprenda a clasificar todas las observaciones como normales para maximizar la precisión. Pero, este enfoque resulta ser ineficiente, ya que no permite identificar de forma correcta los casos que son o pueden ser fraude.

Según **Jiawei Han**, **Micheline Kamber** y **Jian Pei** (2012), en conjuntos de datos desbalanceados para lograr mejorar la detección de esa clase minoritaria se recomienda aplicar técnicas de remuestreo para lograr mejorar la detección. Entre estas técnicas se encuentra el submuestreo, reduce la cantidad de datos de la clase mayoritaria, y el sobremuestreo, que incrementa artificialmente los ejemplos de la clase minoritaria.

Por eso el tratamiento adecuado del desbalance de clases es imprescindible para lograr que los modelos sean más precisos y confiables a la hora de detectar fraudes financieros.

2.4 Modelos de clasificación

Los modelos de clasificación son muy utilizados en problemas de predicción donde se necesita asignar una etiqueta a cada observación. Entre los modelos tenemos la regresión logística, los árboles de decisión y los métodos de ensamble.

Regresión Logística

La regresión logística es uno de los modelos más utilizados en problemas de clasificación binaria. Este modelo estima la probabilidad de que una observación pertenezca a una clase determinada utilizando una función logística.

Una de sus principales ventajas es que interpreta mejor la relación entre las variables de entrada y la probabilidad de ocurrencia del evento. Además, es un modelo eficiente en términos computacionales lo que hace que sea muy utilizado en muchos problemas de clasificación.

Sin embargo, su principal limitante es la capacidad de modelar relaciones lineales, lo que puede afectar en su rendimiento o desempeño en problemas complejos (Hosmer, Lemeshow & Sturdivant, 2013).

Random Forest

Random Forest es un modelo basado en el uso de múltiples árboles de decisión, combinados mediante una técnica de ensamble. Este enfoque permite que la precisión mejore ayudando a reducir el sobreajuste en comparación a un solo árbol de decisión.

El modelo construye múltiples árboles de diferentes subconjuntos de datos y variables, luego combina los resultados mediante votación. Está diversidad en los árboles permite capturar patrones más complejos en los datos.

Entre sus ventajas, posee una considerable robustez frente al ruido, su capacidad para manejar grandes volúmenes de datos y su gran desempeño en problemas de clasificación. Por estas razones es muy utilizado en la detección de fraude financiero (Breiman, 2001).

XGBoost

XGBoost (*Extreme Gradient Boosting*) es un algoritmo basado en las técnicas de boosting, el cual constituye modelos secuencialmente, en donde cada nuevo modelo intenta corregir los errores del anterior.

Este enfoque permite obtener modelos altamente precisos y eficientes, siendo este uno de los algoritmos más populares en competencias de ciencia de datos y aplicaciones reales.

XGBoost es muy eficaz en problemas de datos estructurados y desbalanceados.

Entre sus principales ventajas es su alta capacidad de regularización, su eficiencia computacional y su alto rendimiento en tareas de clasificación, que lo ponen entre las mejores opciones en la detección de fraude financiero (Chen & Guestrin, 2016).

2.5 Métricas de evaluación de modelos

La evaluación de los modelos predictivos necesita de un uso de métricas adecuadas especialmente en contextos en donde existe desbalance de clases. La precisión (*accuracy*) por sí sola puede resultar insuficiente, ya que no refleja correctamente el desempeño del modelo en la detección de la clase minoritaria.

Entre las métricas más utilizadas se encuentran:

- **Precisión (Precision):** mide la proporción de predicciones positivas correctas.
- **Recall:** mide la capacidad del modelo para identificar correctamente casos positivos.
- **F1-score:** combina precisión y recall en una sola métrica.
- **AUC-ROC:** evalúa la capacidad del modelo para distinguir entre clases.

En el contexto del fraude financiero, el recall adquiere especial relevancia, debido a que permite minimizar los falsos negativos, los casos de fraude que no son detectados por el modelo.

Detectar el fraude es generalmente más importante que clasificar correctamente una transacción legítima.

Por otra parte, la curva ROC y el área bajo la curva (AUC) permiten evaluar el desempeño del modelo de manera global (Fawcett, 2006). Como se logra observar en la **Figura 1**, un modelo con mejor capacidad de clasificación presenta una curva más cercana a la esquina superior izquierda, lo que indica una tasa alta de verdaderos positivos (TPR) y una baja tasa de falsos positivos (FPR). En contraste, la línea diagonal representa un modelo con desempeño aleatorio.

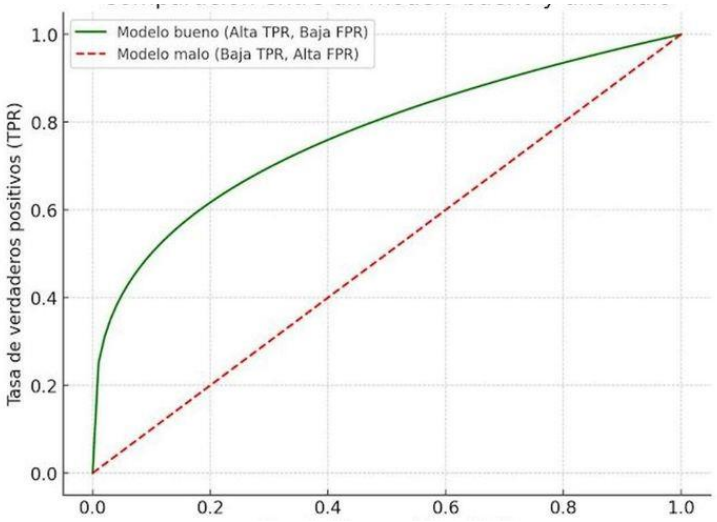


Figura 1 Ejemplo ROC y área bajo la curva (AUC)

2.6 Operaciones de Machine Learning (MLOps)

El concepto de MLOps hace referencia a la integración de prácticas de desarrollo de software y aprendizaje automático para gestionar el ciclo de vida de los modelos.

Este enfoque incluye procesos como la construcción de pipelines de datos, el entrenamiento automatizado, la validación, el despliegue y monitoreo de los modelos en producción. Según Géron (2019), la automatización de estos procesos permite gestionar y mejorar la reproducibilidad, escalabilidad y el mantenimiento de los sistemas basados en machine learning, facilitando su despliegue e implementación en entornos reales.

En el presente trabajo, la implementación de un prototipo funcional mediante herramientas como Streamlit ayuda a evidenciar la gran importancia del despliegue de modelos, facilitando su uso en entornos reales. Además, la posibilidad de actualizar datos y reentrenar modelos da constancia de la aplicabilidad práctica del sistema desarrollado (Treveil & Omont, 2020).

2.7 Herramientas utilizadas

Para el desarrollo del presente proyecto se emplearon diversas herramientas tecnológicas:

- Python, como lenguaje principal de programación.
- Jupyter Notebook, para el análisis exploratorio de datos.
- Scikit-learn, para la implementación de modelos de aprendizaje automático.
- Pandas y NumPy, para la manipulación y procesamiento de datos.
- Streamlit, para el desarrollo de la interfaz gráfica del prototipo.

Estás herramientas permitieron integrar de manera correcta y eficiente las distintas fases del proyecto, desde el análisis de datos hasta la implementación de un sistema funcional.

Capítulo 3. Metodología

En el presente capítulo se describe el enfoque metodológico que guía el desarrollo de este proyecto. Dado que es un trabajo orientado al análisis de datos y la construcción de los modelos de aprendizaje automático, para este caso fue necesario seleccionar una metodología que estructure de una forma ordenada y lógica cada etapa del proceso, desde la comprensión del problema hasta la entrega del producto final.

La metodología que fue elegida es CRISP-DM, este es un marco de trabajo muy reconocido en el área de la ciencia de datos y la minería de datos. En este capítulo se presentará su definición, sus características principales y cada una de sus fases, con el fin de justificar las decisiones metodológicas tomadas a lo largo del proyecto.

3.1 Definición de CRISP-DM

CRISP-DM, cuyas siglas significan Cross-Industry Standard Process for Data Mining, es una metodología de procesos estándar para proyectos de la minería de datos y el análisis de información. Está fue desarrollada a finales de los años noventa con el fin de proporcionar una guía estructurada que está lista para aplicarse en cualquier sector o dominio independientemente del tipo de datos o las herramientas tecnológicas usadas (Chapman et al., 2000).

En términos generales CRISP-DM es una ruta o un mapa que da pasos para seguir cuando se trabaja con los datos con ello extraer conocimiento útil. Este define etapas lógicas que cualquier proyecto de análisis o minería de datos debería recorrer para obtener los mejores resultados con una buena fundamentación con objetivos de negocios, así como su visión y misión.

Según Chapman et al. (2000), CRISP-DM describe el ciclo de vida de un proyecto de minería de datos a través de seis etapas que están muy relacionadas unas con otras que son: comprensión del negocio, comprensión de los datos, preparación de los datos, modelado, evaluación y despliegue. Estos pasos no son rígidos cada paso se puede ajustar como también retroalimentarse según cómo va el proceso y lo que se va descubriendo.

Gracias a su estructura esta metodología es una de las más utilizadas en proyectos de ciencia de datos a nivel mundial, como esta metodología no impone restricciones innecesarias ayuda mucho al momento de usarla en un proyecto con datos.

3.2 Características de la metodología

CRISP-DM se destaca de otras metodologías por tres características fundamentales que la hacen muy adecuada para proyectos de análisis de datos: su naturaleza iterativa, su carácter no lineal y su flexibilidad de aplicación (Chapman et al., 2000).

3.3 Fases de CRISP-DM

CRISP-DM organiza el proceso de la minería de datos en seis fases principales, las cuales permiten estructuras el desarrollo de proyecto analíticos de manera iterativa y flexible.

En la **Figura 2** se presenta la metodología CRISP-DM y la relación entre cada una de sus fases.

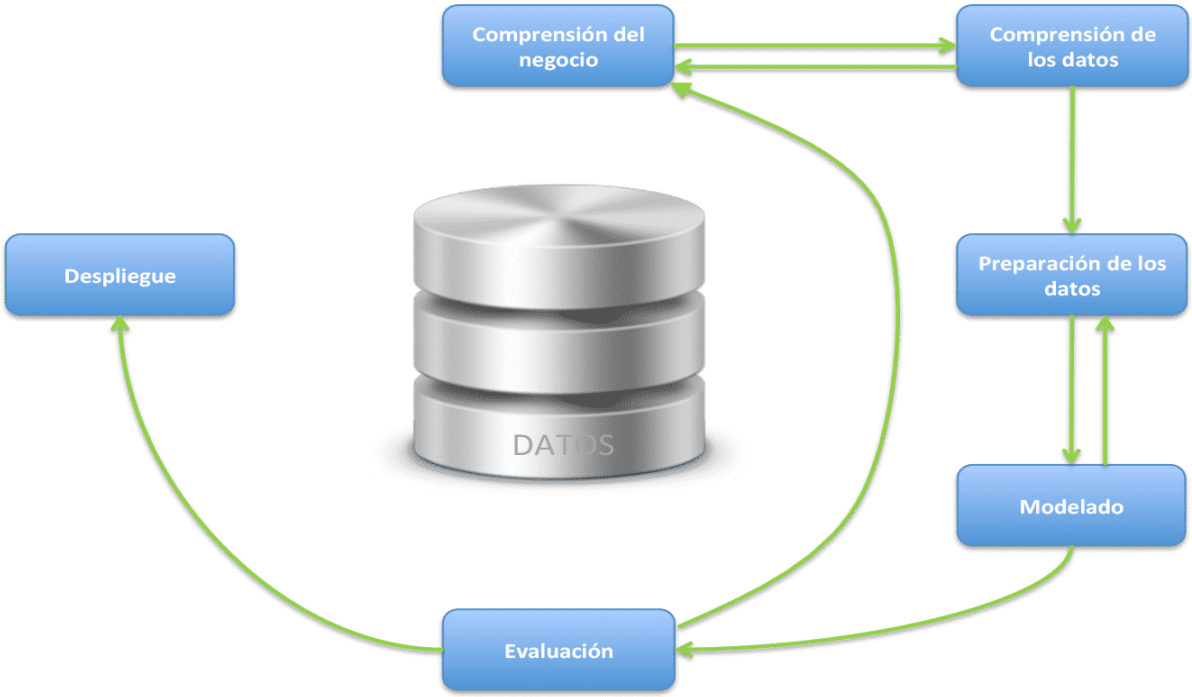


Figura 2. Metodología CRISP-DM.

Fuente: Chapman et al. (2000).

Comprensión del negocio: permite identificar la problemática, establecer los objetivos y definir los criterios con los que se evaluará el éxito del proyecto.

Comprensión de los datos: comprende la recopilación y exploración inicial de la información disponible, así como la identificación de su estructura, calidad, distribución e inconsistencias.

Preparación de los datos: incluye las actividades de limpieza, transformación, integración y selección necesarias para obtener un conjunto de datos adecuado para el modelado.

Modelado: consiste en seleccionar, entrenar y ajustar las técnicas de minería de datos o aprendizaje automático apropiadas para el problema planteado.

Evaluación: permite analizar el desempeño de los modelos mediante métricas pertinentes y determinar si los resultados responden a los objetivos establecidos.

Despliegue: corresponde a la presentación o integración de los resultados en un entorno que permita su utilización, ya sea mediante informes, aplicaciones o sistemas funcionales.

Las fases de CRISP-DM no constituyen una secuencia rígida. Su naturaleza iterativa permite regresar a etapas anteriores cuando los resultados obtenidos requieren ajustes en los datos, los modelos o los objetivos del proyecto (Chapman et al., 2000).

3.4 Importancia de CRISP-DM en proyectos de Machine Learning

En el campo del aprendizaje automático es común que los equipos de trabajo se enfoquen rápidamente en la parte técnica en el problema: seleccionar algoritmos, ajustar los parámetros y comparar las métricas. Sin embargo, este enfoque puede llevar a construir modelos que funcionan bien en términos estadísticos pero que no resuelven el problema real o que no logran ponerse en práctica de manera efectiva.

CRISP-DM aporta una estructura que evita este error se haga frecuente. Al comenzar con una comprensión profunda del negocio y terminar con un plan de despliegue concreto, la metodología hace tener un foco en el problema original durante todo el proceso, Cada decisión técnica, desde la selección de variables hasta elección del algoritmo correspondiente, se toma en función de los objetivos planteados al inicio y no de preferencias tecnológicas o modas en el campo (Chapman et al., 2000).

Un aporte importante es la gestión de la complejidad inherente a los proyectos de análisis de datos. Trabajar con grades volúmenes de información, múltiples variables y algoritmos complejos puede volverse difícil o caótico en muchos casos si no existe un marco ordenado. CRISP-DM proporciona ese orden sin quitar flexibilidad, permitiendo adaptarse a las particularidades de cada proyecto.

En proyectos relacionados con la detección de fraude, donde los datos generalmente están desbalanceados, los patrones son complejos y las consecuencias de los errores del modelo tienen un impacto económico directo, contar con una metodología bien estructurada es especialmente valioso. CRISP-DM ayuda a abordar muchas de estas dificultades de forma sistemática, asegurando que cada etapa del proceso reciba la atención que merece (Bhattacharyya et al., 2011).

Para finalizar, CRISP-DM ayuda en la comunicación entre el equipo y los responsables

del negocio, Al estructurar el proceso en fases comprensibles y con entregables claros. Esto facilita que personas sin formación técnica puedan entender en que parte se encuentra el proyecto y que se puede esperar de él. Está capacidad de comunicación es un activo fundamental en proyectos donde las decisiones técnicas deben estar alineadas con los objetivos organizacionales.

Con este marco establecido, los capítulos siguientes describen como cada una de las fases fue aplicada concretamente en el desarrollo del modelo de detección de fraude en transacciones financieras que constituye el objetivo principal del trabajo.

Capítulo 4. Implementación y Despliegue (Aplicación CRISP-DM)

En este capítulo se presenta la aplicación práctica de CRISP-DM en el desarrollo del sistema de detección de fraude. Se da una breve descripción de las fases realizadas durante el proyecto, desde la comprensión del problema, el análisis y la preparación de los datos, el modelado, la evaluación de resultados y el despliegue del sistema.

4.1 Enfoque metodológico

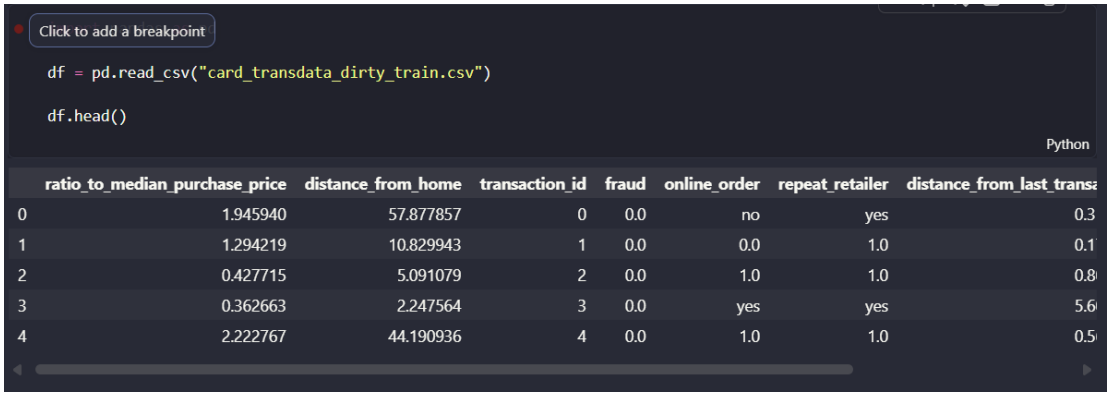
Con esta metodología se organiza las diferentes etapas del trabajo, desde la comprensión del problema hasta el despliegue del modelo con el prototipo, permitiendo un proceso iterativo y orientado a los resultados.

4.2 Fuente de datos

Para el desarrollo del presente proyecto se emplearon conjuntos de datos provenientes de plataformas especializadas en la ciencia de datos tales como **Kaggle,** donde dispone de datasets relacionados con transacciones financieras anonimizadas.

Estos conjuntos de datos contienen registros de transacciones reales, lo cual garantiza la confidencialidad de la información. Asimismo, incluyen variables numéricas que representan características relevantes de las transacciones, junto con una variable objetivo que ayuda a clasificar si cada una de las operaciones es fraudulenta o legitima.

De forma adicional con el fin de lograr simular condiciones cercanas a lo que sería un entorno real como un banco, se generó diferentes conjuntos de datos con un menor grado de preprocesamiento, así eliminando gran parte de la ingeniería aplicada y manteniendo inconsistencias comunes de los datos financieros o transaccionales. Esto ayuda en gran parte a evaluar el comportamiento del sistema frente a datos más cercanos a los que se reciben en instituciones financieras, donde la información no está tratada.



	ratio_to_median_purchase_price	distance_from_home	transaction_id	fraud	online_order	repeat_retailer	distance_from_last_transa
0	1.945940	57.877857	0	0.0	no	yes	0.3
1	1.294219	10.829943	1	0.0	0.0	1.0	0.1
2	0.427715	5.091079	2	0.0	1.0	1.0	0.8
3	0.362663	2.247564	3	0.0	yes	yes	5.6
4	2.222767	44.190936	4	0.0	1.0	1.0	0.5

distance_from_home	transaction_id	fraud	online_order	repeat_retailer	distance_from_last_transaction	used_chip	used_pin_number
57.877857	0	0.0	no	yes	0.311140	yes	no
10.829943	1	0.0	0.0	1.0	0.175592	0.0	0.0
5.091079	2	0.0	1.0	1.0	0.805153	0.0	0.0
2.247564	3	0.0	yes	yes	5.600044	yes	no
44.190936	4	0.0	1.0	1.0	0.566486	1.0	0.0

Figura 3. Vista del dataset con bajo nivel de preprocesamiento (datos sin tratar).

Como se observa en la **Figura 3**, el dataset presenta variables numéricas, binarias y categóricas representadas en distintos formatos, incluyendo valores como ’yes/no’ y ‘0/1’, lo cual evidencia inconsistencias típicas de datos sin preprocesamiento.

4.3 Fase 1 Comprensión del Negocio

La fase de comprensión del negocio debe definir claramente el problema a resolver desde una perspectiva organizacional, así como establece los criterios de éxito del proyecto. En el contexto de CRISP-DM, está etapa permite alinear los objetivos del negocio con el problema de minería de datos.

En este trabajo, el problema radica en la detección temprana de transacciones fraudulentas con tarjetas de crédito. Desde la perspectiva financiera, la identificación oportuna del fraude resulta critica desde cualquier ángulo debido a que un error en la clasificación puede traer consecuencias graves. En particular, los falsos negativos o sea fraudes no detectado, esto representa enormes pérdidas económicas directas, mientras que los falsos positivos pueden afectar a la experiencia del cliente al bloquear transacciones legítimas.

En este sentido, el objetivo del modelo es maximizar la capacidad de detección de fraude, priorizando métricas como el recall, sin descuidar la precisión del sistema en general.

Adicionalmente, es necesario que el modelo pueda integrarse a un sistema funcional donde se logre usar en entornos simulados, facilitando tanto la predicción de transacciones nuevas como la actualización del modelo con datos recientes.

4.4 Fase 2 Comprensión de los datos

En esta fase se realizó el análisis a un conjunto de datos explorando estos con el objetivo de entender su estructura, ver e identificar inconsistencias para poder observar patrones muy importantes para la detección del fraude en las transacciones financieras.

Los datos que se utilizaron tienen un aproximado de 1,000,000 de registros con 9 variables estas dan descripciones del comportamiento de cada transacción, como métricas de un monto, ubicación y métodos de pago.

Calidad y estructura de los datos

Se encontró valores nulos en las siguientes variables:

- ratio_to_median_purchase_price
- distance_from_home

Están son un aproximado del 2% total de los datos, lo cual simula condiciones o escenarios reales en los sistemas financieros donde generalmente la información puede estar incompleta.

Implicación:

Dado estas necesidades esto justifica el uso del proceso de imputación automática dentro del pipeline, esto ayuda a que el sistema haga los procesos con los datos incompletos sin la intervención manual evitando entrar al entorno por detrás y automatizando mucho las tareas.

Formato de los datos

Este dataset presenta las variables de distintos tipos:

- Numéricas (float64, int64)
- Categóricas binarias representadas como texto (str)

En el dataset se identificó algunas inconsistencias en estas variables, como:

- "yes", "no"
- "YES", "NO"
- "1.0", "0.0"

Implicación:

En este punto se requirió de un proceso de normalización de las variables binarias, con ello transformando estas en valores numéricos (0 y 1) para el uso correcto en los modelos de Machine Learning.

Desbalances de clases

En el análisis de la variable objetivo (fraud) se identificó la siguiente distribución:

- **91.26%** transacciones legítimas
- **8.74%** transacciones fraudulentas

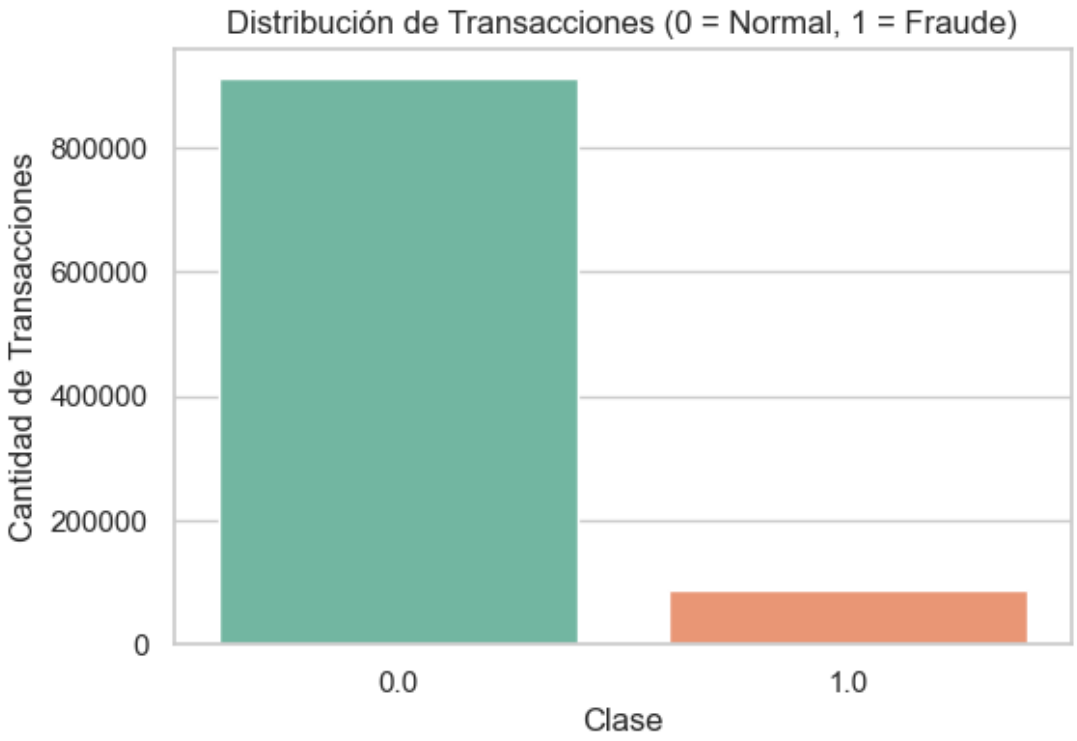


Figura 4. Distribución de las transacciones normales y fraudulentas.

Con se observa en la **Figura 4**, se identificó que hay un análisis moderado lo que da como resultado que existe fraude, pero es poco frecuente lo que lleva a aplicar técnicas de balanceo.

Análisis de variables numéricas

Ratio to Median Purchase Price

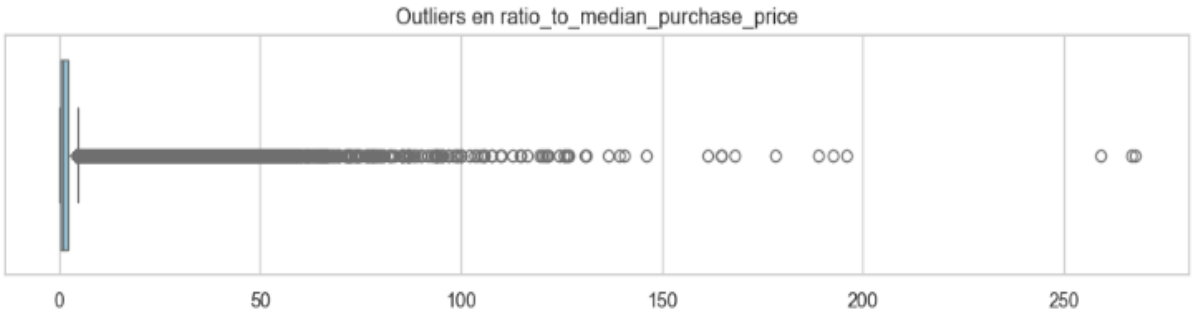


Figura 5. Valores atípicos en la variable ratio_to_median_purchase_price.

Esta variable representa la relación entre monto de la transacción y el comportamiento histórico del cliente. Como se observa en la **Figura 5,** los valores muestran extremos elevados, lo cual puede indicar comportamientos anómalos.

Distance from home

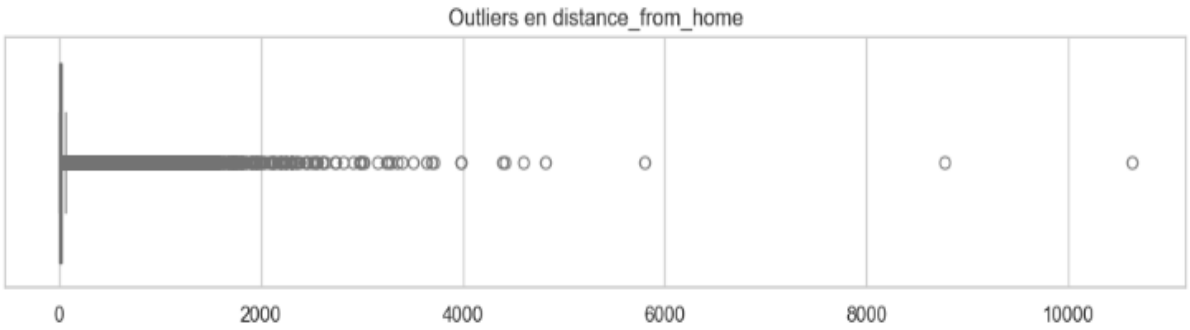


Figura 6. Valores atípicos en la variable distance_from_home.

Está variable representa la distancia entre el lugar donde se realizó la transacción y el domicilio habitual del usuario. Como se logra observar en la **Figura 6**, existen valores extremadamente altos, lo que puede indicar algunas posibles anomalías en el comportamiento de las transacciones.

Distance from Last Transaction

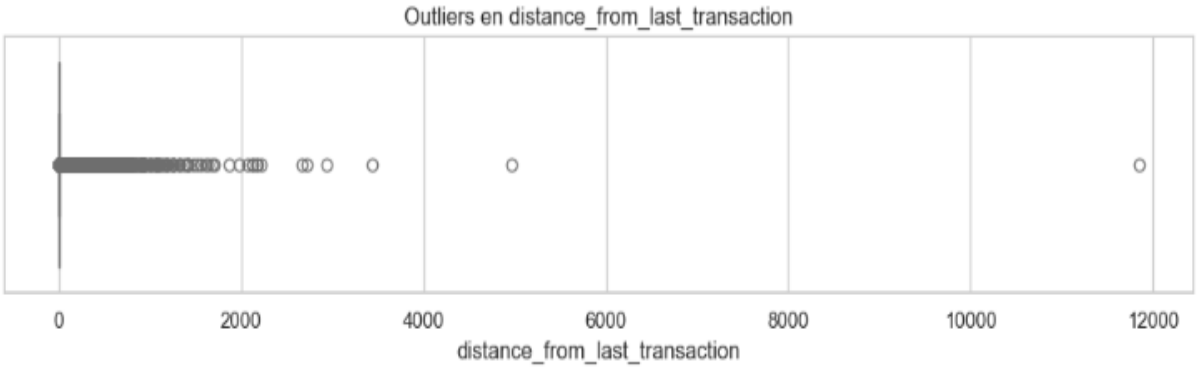


Figura 7. Valores atípicos en la variable distance_from_last_transaction.

Está variable permite analizar el desplazamiento entre transacciones consecutivas. Como se logra observar en la **Figura 7,** existen valores elevados que pueden dar a entender que hay actividades sospechosas en cortos periodos de tiempo.

Análisis de la correlación

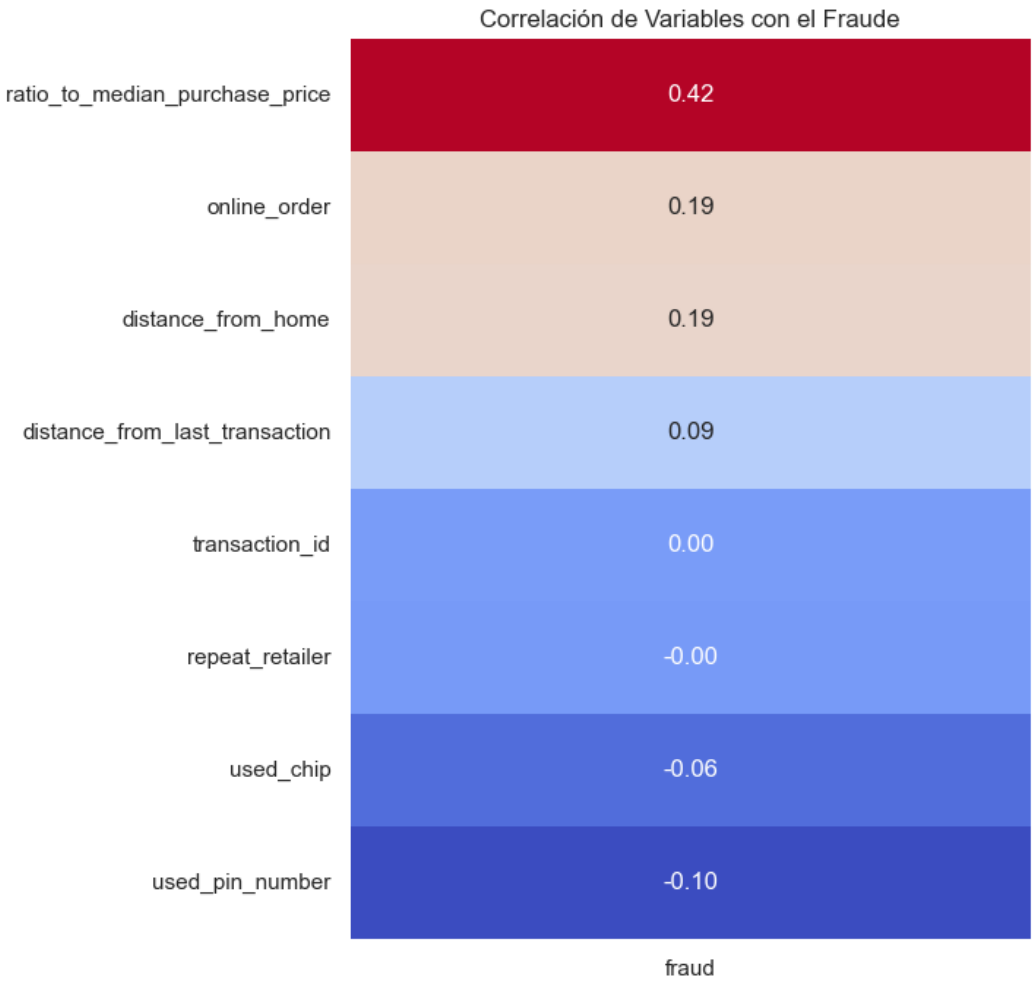


Figura 8. Correlación de variables con la variable objetivo fraude.

Como se observa en la **Figura 8,** la variable ratio**_to_median_purchase_price** presenta la mayor correlación positiva con el fraude. Así también variables como **online_order** y **distance_from_home** muestran correlaciones moderadas, mientras que otras como **used_chip** *y* **used_pin_number** muestran correlaciones negativas.

Hallazgos principales:

- ratio_to_median_purchase_price presenta la mayor correlación positiva (**0.42**)
- online_order y distance_from_home tienen correlaciones moderadas
- Variables como used_chip y used_pin_number presentan correlaciones negativas

Estas variables nos ayudan a ver varios patrones relevantes que está fuertemente vinculados al fraude y sirven como una base para el proyecto.

Simulación de condiciones reales

Se trabajó con un dataset que presentan desde valores nulos, inconsistencias en el formato y outliers esto a diferencia de un dataset limpio o que paso por ingeniería de datos, ayuda a simular condiciones reales en los sistemas financieros, donde generalmente los datos no estas completamente depurados.

El sistema requiere de un pipeline automatizado capaz de

- Limpiar datos
- Normalizar variables
- Imputar valores faltantes
- Preparar los datos para predicción

Enfoque del sistema

Con el análisis realizado se estableció que el sistema no depende de datos perfectamente estructurados, sino que es capaz de procesar información con características reales.

Esto da como valor agregado importante, ya que permite evaluar el sistema en escenarios cercanos a producción.

4.5 Fase 3 Preparación de los Datos

En esta fase se realizó el proceso para la limpieza de datos, transformar y estructurar el dataset, esto se hace con el objetivo de que el modelo de machine learning tenga datos adecuados para su entrenamiento.

Este sistema está diseñado para trabajar en condiciones reales por lo cual el pipeline contempla:

- Datos incompletos
- Variables inconsistentes
- Presencia de valores atípicos
- Formatos heterogéneos.

Creación de una copia de trabajo

Aquí se crea una copia del dataset original siendo esto algo bueno ya que no se toca el dataset original. Esto ayuda a conservar los datos originales y trabajar de forma segura durante la limpieza y transformación.

```python
print("--- Iniciando Fase 3: Preparación de Datos ---")

df_clean = df.copy()
df_clean.head()
```

Python

```
--- Iniciando Fase 3: Preparación de Datos ---
```

	ratio_to_median_purchase_price	distance_from_home	transaction_id	fraud	online_order	repeat_retailer	distance_from_last_tran
0	1.945940	57.877857	0	0.0	no	yes	0.
1	1.294219	10.829943	1	0.0	0.0	1.0	0.
2	0.427715	5.091079	2	0.0	1.0	1.0	0.
3	0.362663	2.247564	3	0.0	yes	yes	5.
4	2.222767	44.190936	4	0.0	1.0	1.0	0.

Figura 9. Creación de una copia del dataset de trabajo.

Para garantizar una buena integridad de los datos, se generó una copia del dataset, como se muestra en la **Figura 9**.

Esto permite a que se conserve la fuente original y facilita las pruebas, validaciones y ajustes posteriores.

Eliminación de variables independientes

Aquí se eliminó la columna **transaction_id** debido a que no aporta información útil para el aprendizaje del modelo ya que es un identificador único de cada transacción.

```python
df_clean = df_clean.drop(columns=['transaction_id'], errors='ignore')

print(df_clean.columns)
```

Python

```
Index(['ratio_to_median_purchase_price', 'distance_from_home', 'fraud',
       'online_order', 'repeat_retailer', 'distance_from_last_transaction',
       'used_chip', 'used_pin_number'],
      dtype='str')
```

Figura 10. Eliminación de la variable transaction_id.

Está variable no tiene un valor predictivo lo que puede introducir ruido y dar problemas con el modelado.

Normalización de variables binarias

En el dataset se identificó variables binarias que están representadas en diferentes formatos como: yes, no, 1, 0, 1.0, true y false. Esto ayuda a evitar inconsistencias, se definió una función que transforma estos valores a una representación numérica estándar.

```python
def limpiar_binarias(col):
    return col.astype(str).str.strip().str.lower().map({
        'yes': 1,
        'no': 0,
        '1': 1,
        '0': 0,
        '1.0': 1,
        '0.0': 0,
        'true': 1,
        'false': 0
    })
```

Python

Figura 11. Función de normalización de variables binarias.

Con este paso se logró que las variables categóricas binarias sean interpretadas de forma correcta por los modelos, ya que generalmente trabajan con datos numéricos.

Aplicación de la normalización a las columnas binarias

Ya definida la función de limpieza, se aplicó a las variables binarias del dataset:

online_order, repeat_retailer, used_chip y used_pin_number.

```python
binarias = ['online_order', 'repeat_retailer', 'used_chip', 'used_pin_number']

for col in binarias:
    df_clean[col] = limpiar_binarias(df_clean[col])

df_clean[binarias].head()
```

Python

Figura 12. Aplicación de la normalización a variables binarias.

Una vez definida la función, se aplicó a las variables binarias del dataset, como se observa en la **Figura 12.**

	online_order	repeat_retailer	used_chip	used_pin_number
0	0	1	1	0
1	0	1	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	0

Figura 13. Tipos de datos de las variables del dataset.

Como se observa en la **Figura 13** con este proceso se logró convertir las variables binarias a valores 0 y 1, esto garantiza que todas mantengan una misma convención. Los datos

pueden provenir de diferentes fuentes con formatos no homogéneos lo que hace esto especialmente importante para lo que son entornos reales.

Conversión de columnas a tipo numérico

Después del proceso de normalización con las variables binarias, se convirtió el dataset a formato numérico.

Los valores que no se pudieron ser interpretados correctamente fueron transformados en valores nulos mediante errors='coerce', tal como se presenta en la **Figura 14**.

```python
for col in df_clean.columns:
    df_clean[col] = pd.to_numeric(df_clean[col], errors='coerce')

print(df_clean.dtypes)
```

Python

Figura 14. Conversión de variables del dataset a formato numérico.

Este paso nos asegura que todas las variables tengan un formato compatible con los algoritmos de clasificación. Además, permite detectar registros problemáticos, debido a que cualquier valor no convertible queda marcado como NaN para posteriormente ser tratado mediante imputación.

```
ratio_to_median_purchase_price    float64
distance_from_home                float64
fraud                             float64
online_order                        int64
repeat_retailer                     int64
distance_from_last_transaction    float64
used_chip                           int64
used_pin_number                     int64
dtype: object
```

Figura 15. Tipos de datos del dataset después de la conversión numérica.

Separación de variables predictoras y variable objetivo

Posteriormente, se separó el dataset en las variables predictoras y la variable objetivo. Como se observa en la **Figura 16**, las variables predictoras se almaceno en x, las variables objetivo fraud se almaceno en y.

```python
X = df_clean.drop('fraud', axis=1)
y = df_clean['fraud']

print("Shape de X:", X.shape)
print("Shape de y:", y.shape)
```

Python

Figura 16. Separación de las variables predictoras y la variable objetivo del dataset.

La variable fraud es la que indica si tal transacción es legítima o fraudulenta, donde 0 representa una transacción normal y 1 una transacción fraudulenta. La separación es muy necesaria porque los modelos supervisados requieren distinguir entre los datos de entrada y la etiqueta que debe aprender a predecir.

En la **Figura 17** se verifica la forma (*shape*) de las matrices resultantes utilizadas en el modelo, confirmando la cantidad de registros y variables presentes en el conjunto de datos.

```
Shape de X: (1000000, 7)
Shape de y: (1000000,)
```

Figura 17. Verificación de la forma (*shape*) de las matrices de características y etiquetas utilizadas en el modelo.

División del dataset en entrenamiento y prueba

El dataset fue dividido en dos subconjuntos el de entrenamiento y el de prueba. Para entrenamiento se usó una proporción el 80% y para prueba un 20%, manteniendo la proporción original de clases mediante estratificación.

Como se observa en la **Figura 18**, el proceso de partición se realizó utilizando muestreo estratificado (*stratify*), lo cual permite conservar la proporción de transacciones fraudulentas y legítimas en ambos subconjuntos.

```python
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)

print("X_train:", X_train.shape)
print("X_test:", X_test.shape)
print("y_train:", y_train.shape)
print("y_test:", y_test.shape)
```

Python

Figura 18. Proceso de partición del conjunto de datos en entrenamiento y prueba utilizando muestreo estratificado (stratify) para conservar la distribución de clases.

Como se observa en la figura 19, la división ayuda en el entrenamiento del modelo con una parte de los datos y evaluar su rendimiento sobre los registros no vistos. El uso de stratify es muy necesario debido al considerable desbalance de las clases, ya que asegura que el conjunto de datos de entrenamiento como el de prueba conserven una distribución similar entre transacciones legítimas o fraudulentas.

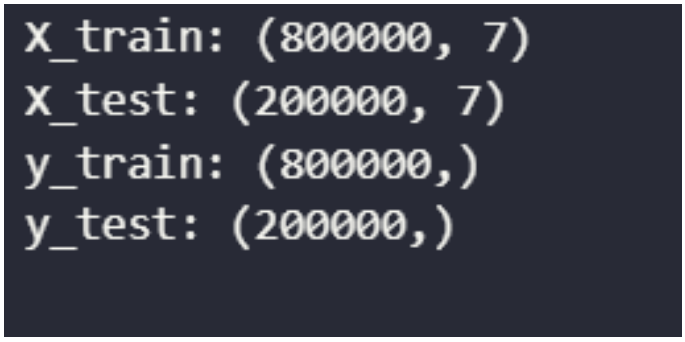


Figura 19. Dimensiones de los conjuntos de entrenamiento y prueba después de la partición del dataset.

Imputación de valores nulos

En esta etapa se aplicó imputación sobre los valores faltantes presentes en las variables numéricas. Para ello, se utilizó la mediana como estrategia de reemplazo.

La **Figura 20** muestra el proceso de tratamiento de datos faltantes mediante imputación por mediana aplicado a los conjuntos de entrenamiento y prueba.

La mediana fue seleccionada debido a que, en general, es más robusta frente a valores atípicos en comparación con la media. Esto resulta especialmente importante en detección de fraude, donde las variables pueden presentar valores extremos asociados a comportamientos anómalos.

```python
from sklearn.impute import SimpleImputer
import pandas as pd

imputer = SimpleImputer(strategy='median')

X_train_imputed = pd.DataFrame(
    imputer.fit_transform(X_train),
    columns=X_train.columns,
    index=X_train.index
)

X_test_imputed = pd.DataFrame(
    imputer.transform(X_test),
    columns=X_test.columns,
    index=X_test.index
)
```

Python

Figura 20. Imputación de valores faltantes mediante mediana.

La mediana fue seleccionada por que en general es más robusta a valores atípicos que la media. Esto en la detección de fraude resulta importante debido a que las variables pueden presentar valores extremos asociados a comportamientos anómalos.

Verificación posterior a la imputación

Como se observa en la **Figura 21**, se realizó una verificación posterior al proceso de imputación para comprobar la ausencia de valores nulos.

```python
print("Nulos en X_train_imputed:")
print(X_train_imputed.isnull().sum())

print("\nNulos en X_test_imputed:")
print(X_test_imputed.isnull().sum())
```

Python

Figura 21. Verificación de valores nulos después de la imputación.

Esto se hace por control de calidad antes de pasar a la fase 4 que es el modelado. La función es simple, es confirmar que los datos estén completos y listos debido a que algo mal puede hacer que el modelo falle.

La **Figura 22** muestra el resultado de la validación realizada sobre los conjuntos de entrenamiento y prueba después de la imputación.

```
Nulos en X_train_imputed:
ratio_to_median_purchase_price    0
distance_from_home                0
online_order                      0
repeat_retailer                   0
distance_from_last_transaction    0
used_chip                         0
used_pin_number                   0
dtype: int64

Nulos en X_test_imputed:
ratio_to_median_purchase_price    0
distance_from_home                0
online_order                      0
repeat_retailer                   0
distance_from_last_transaction    0
used_chip                         0
used_pin_number                   0
dtype: int64
```

Figura 22. Resultado de la validación de valores nulos.

En definitiva, la fase de preparación ayudó a tener el conjunto de datos limpios estructurado y listo para los modelos de clasificación supervisada. Aquí se trató valores nulos, se normalizo variables binarias se eliminó variables irrelevantes y se conservó los valores atípicos relevantes para el problema de fraude.

Finalmente, se visualiza una muestra del conjunto de entrenamiento ya preparado.

¿Por qué se realiza está revisión?

- Permite comprobar que las transformaciones se aplicaron correctamente.
- Ayuda a validar que las variables estén en formato numérico.
- Sirve como control previo antes de aplicar técnicas adicionales como escalado, balanceo y entrenamiento de modelos.

Como se observa en la **Figura 23**, el dataset queda listo para la fase de modelado y construcción del pipeline de Machine Learning.

	ratio_to_median_purchase_price	distance_from_home	online_order	repeat_retailer	distance_from_last_transaction	used_chip
950236	0.208383	3.426737	1.0	1.0	7.234169	1.0
358251	2.713531	15.106926	1.0	1.0	0.867626	1.0
597434	0.304588	31.797354	1.0	1.0	3.159440	0.0
296537	0.340111	10.214106	0.0	1.0	0.034442	0.0
235768	1.284013	81.804301	0.0	1.0	4.735865	0.0

edian_purchase_price	distance_from_home	online_order	repeat_retailer	distance_from_last_transaction	used_chip	used_pin_number
0.208383	3.426737	1.0	1.0	7.234169	1.0	0.0
2.713531	15.106926	1.0	1.0	0.867626	1.0	0.0
0.304588	31.797354	1.0	1.0	3.159440	0.0	0.0
0.340111	10.214106	0.0	1.0	0.034442	0.0	0.0
1.284013	81.804301	0.0	1.0	4.735865	0.0	0.0

Figura 23. Vista previa de los conjuntos de datos procesados y preparados para la etapa de modelado en Machine Learning.

4.6 Fase 4: Modelado

Naturaleza del problema

En este sistema se aplicó un enfoque de Aprendizaje Supervisado con un enfoque en la clasificación Dicotómica.

Por qué se escogió el Aprendizaje Supervisado

El aprendizaje no supervisado por ejemplo K-Means generalmente no usa etiquetas para detectar anomalías. Sin embargo, con este caso se dispone de una variable objetivo que es (fraud), que ayuda a indicar si es fraudulenta o no.

Esto ayuda a utilizar el aprendizaje supervisado, que en general es más preciso y eficiente ya que:

- Aprende directamente de ejemplos históricos etiquetados
- Identifica patrones reales de fraude en lugar de solo detectar comportamientos atípicos
- Permite optimizar métricas específicas como recall o precisión

En términos generales, no solo ayuda a detectar transacciones sospechosas, sino que aprende que es lo más característico de un fraude real.

El enfoque predictivo

El modelo no solo está diseñado para un análisis histórico de transacciones o movimientos financieros, sino que está listo para:

- Recibir nuevas transacciones (datos no etiquetados)
- Procesarlas automáticamente mediante un pipeline
- Generar una predicción en procesamiento por lotes en un entorno simulado

Lo principal es clasificar las transacciones como:

- 0 → Normal
- 1 → Fraudulenta

Esto hace que el sistema este alineado a entornos financieros reales.

2. Criterios de selección de los algoritmos

En este proyecto se buscó los mejores algoritmos que puedan resolver la problemática de forma óptima debido a que no existe el algoritmo perfecto para que resuelva todo. Por ello se seleccionó 3 formas que aumentan de manera progresiva la complejidad y capacidad predictiva. Dentro del sistema cada modelo cumple un rol importante para el desarrollo del proyecto.

A. Regresión logística (Modelo Base)

Un modelo lineal que estima la probabilidad de que una transacción sea fraudulenta

- Proporciona un punto de referencia inicial (baseline)
- Es altamente interpretable
- Permite entender el impacto de cada variable en la predicción

El objetivo del sistema es establecer una base mínima de rendimiento y servir como una comparación entre modelos más complejos.

B. Random Forest (Modelo Robusto)

Un algoritmo de ensamblado basado en múltiples árboles de decisión que normalmente trabajan en paralelo.

- Reduce el riesgo de sobreajuste (overfitting)
- Maneja bien relaciones no lineales
- Es menos sensible a ruido en los datos

El objetivo del sistema es el de controlar los falsos positivos y evitar clasificar de forma incorrecta transacciones legítimas como fraude.

C. XGBoost (Modelo Avanzado)

Un algoritmo de boosting que en general construye modelos en forma secuencial, corrigiendo errores de modelos anteriores.

- Es altamente preciso en datos tabulares
- Captura patrones complejos
- Optimiza el rendimiento mediante aprendizaje iterativo

El objetivo principal es minimizar los falsos negativos y ayudar a detectar la mayor cantidad de posibles fraudes.

Enfoque estratégico del modelo

Cada uno de los modelos será evaluado con las métricas más útiles para el problema del fraude.

- Recall (Sensibilidad) → Prioridad principal
- Precision
- AUC-ROC
- Accuracy (como referencia secundaria)

Escalado de Variables y Balanceo de Clases

Antes de entrenar la Regresión Logística, la aplicación de dos técnicas adicionales sobre el conjunto de entrenamiento es muy necesario:

- Escalado de variables

Escalado con RobustScaler

La Regresión Logística es un algoritmo sensible a la escala de las variables numéricas. En este dataset existen variables con rangos muy distintos, por ejemplo:

- ratio_to_median_purchase_price
- distance_from_home
- distance_from_last_transaction

¿Por qué se utiliza RobustScaler?

- Reduce el impacto de valores extremos
- Utiliza estadísticas robustas como la mediana y el rango intercuartílico
- Mejora la estabilidad del entrenamiento en presencia de outliers

El escalador se ajusta únicamente con los datos de entrenamiento para luego aplicar al conjunto de prueba. Esto hace que no haya una fuga de información.

Aplicación del escalado de variables

Después de este paso:

- Las variables fueron escaladas de forma robusta
- El conjunto de entrenamiento tendrá una distribución más equilibrada
- La Regresión Logística podrá entrenarse en condiciones más adecuadas para detectar fraude

Como se observa en la **Figura 24**, se aplicó un escalado robusto de variables utilizando RobustScaler para reducir el impacto de valores atípicos presentes en el dataset.

```python
from sklearn.preprocessing import RobustScaler

scaler = RobustScaler()

X_train_scaled = pd.DataFrame(
    scaler.fit_transform(X_train_imputed),
    columns=X_train_imputed.columns,
    index=X_train_imputed.index
)

X_test_scaled = pd.DataFrame(
    scaler.transform(X_test_imputed),
    columns=X_test_imputed.columns,
    index=X_test_imputed.index
)

print("Escalado completado.")
print(X_train_scaled.head())
```

```
Escalado completado.
        ratio_to_median_purchase_price  distance_from_home  online_order  \
950236                       -0.497076           -0.308110           0.0
358251                        1.058171            0.242409           0.0
597434                       -0.437349            1.029075           0.0
296537                       -0.415296            0.011797          -1.0
235768                        0.170697            3.386038          -1.0

        repeat_retailer  distance_from_last_transaction  used_chip  \
950236              0.0                        2.041111        1.0
358251              0.0                       -0.042518        1.0
597434              0.0                        0.707542        0.0
296537              0.0                       -0.315201        0.0
235768              0.0                        1.223471        0.0

        used_pin_number
950236              0.0
358251              0.0
597434              0.0
296537              0.0
235768              0.0
```

Figura 24. Escalado robusto de variables mediante RobustScaler.

Con el procedimiento hecho, el conjunto de entrenamiento está adecuadamente listo para algoritmos que son sensibles a los desbalances de clase, como por ejemplo la Regresión Logística. El conjunto de prueba se mantiene sin alteraciones estructurales, dando garantía de una evaluación más realista del modelo.

4.6.1 Modelo 1: Regresión Logística

Como punto referencial inicial en el rendimiento del sistema se usa la regresión logística que se lo utiliza como modelo base como punto de partida.

¿La regresión logística que es?

Es un modelo de clasificación lineal donde se estima la probabilidad de que una transacción pertenezca a la clase fraudulenta (fraud = 1), utilizando una función logística.

Justificación del modelo

- Es altamente interpretable

- Permite entender la influencia de cada variable en la predicción
- Sirve como referencia frente a modelos más complejos

Parámetros utilizados

max_iter = 1000

Justificación:

- Se requieren más iteraciones para asegurar la convergencia del modelo
- Evita errores de optimización durante el entrenamiento

Estrategia:

- Servir como modelo base
- Evaluar su desempeño en métricas clave
- Compararlo posteriormente con Random Forest y XGBoost

Si otros modelos no logran superarlo, no habría justificación en el incremento en complejidad y costo computacional.

Como se observa en la **Figura 25**, se realizó el entrenamiento del modelo de Regresión Logística utilizando el conjunto de datos preprocesado.

```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, roc_auc_score

print("--- Entrenando Modelo 1: Regresión Logística ---")

# Inicialización del modelo
modelo_lr = LogisticRegression(max_iter=1000, random_state=42)

# Entrenamiento
inicio = time.time()
modelo_lr.fit(X_train_resampled, y_train_resampled)
fin = time.time()

# Tiempo de entrenamiento
tiempo_lr = fin - inicio

# Predicciones
y_pred_lr = modelo_lr.predict(X_test_scaled)
y_prob_lr = modelo_lr.predict_proba(X_test_scaled)[:, 1]

# Métricas
acc_lr = accuracy_score(y_test, y_pred_lr)
prec_lr = precision_score(y_test, y_pred_lr)
rec_lr = recall_score(y_test, y_pred_lr)
auc_lr = roc_auc_score(y_test, y_prob_lr)
```

Figura 25. Entrenamiento del modelo de Regresión Logística.

La **Figura 26** muestra los resultados obtenidos durante la evaluación del modelo de Regresión Logística mediante métricas de rendimiento.

```python
# Mostrar resultados
print("\n--- Resultados Regresión Logística ---")
print(f"Accuracy:  {acc_lr:.4f}")
print(f"Precision: {prec_lr:.4f}")
print(f"Recall:    {rec_lr:.4f}")
print(f"AUC-ROC:   {auc_lr:.4f}")
print(f"Tiempo:    {tiempo_lr:.2f} segundos")

# Guardar resultados
resultados_modelos['Logistic Regression'] = {
    'Accuracy': acc_lr,
    'Precision': prec_lr,
    'Recall': rec_lr,
    'AUC': auc_lr,
    'Tiempo': tiempo_lr
}

predicciones['Logistic Regression'] = y_pred_lr
probabilidades['Logistic Regression'] = y_prob_lr
```

```
--- Entrenando Modelo 1: Regresión Logística ---

--- Resultados Regresión Logística ---
Accuracy:  0.9201
Precision: 0.5244
Recall:    0.9259
AUC-ROC:   0.9630
Tiempo:    0.27 segundos
```

Figura 26. Evaluación del desempeño del modelo de Regresión Logística mediante métricas de *Accuracy*, *Precision*, *Recall* y *AUC-ROC*.

4.6.2 Modelo 2: Random Forest

Random Forest se utiliza como un modelo de ensamblado robusto, este es capaz de capturar relaciones no lineales y así reducir el riesgo de sobreajuste.

Justificación técnica del modelo

- Combina múltiples árboles de decisión
- Mejora la estabilidad de la predicción
- Tolera mejor ruido y relaciones complejas en los datos
- Es menos sensible al sobreajuste que un árbol individual

Configuración utilizada

Se usan los siguientes parametros:

- n_estimators = 50
- n_jobs = -1

- random_state = 42

Justificación

- n_estimators = 50 permite un equilibrio adecuado entre rendimiento predictivo y tiempo de entrenamiento dentro del prototipo
- Un conjunto de 50 árboles es suficiente para capturar patrones relevantes sin incrementar excesivamente el costo computacional
- n_jobs = -1 permite utilizar todos los núcleos disponibles del procesador, acelerando el entrenamiento sobre un conjunto de datos de gran tamaño
- random_state = 42 asegura reproducibilidad en los resultados

Estrategia dentro del sistema

El modelo está como una alternativa robusta a la regresión logística, está tiene como objetivo el capturar patrones no lineales así para reducir los falsos positivos y mejorar la estabilidad en general del sistema frente a los datos complejos.

Evaluación

Este modelo será evaluado mediante las siguientes métricas que son: Accuracy, Precisión, Recall y AUC-ROC. Esto ayudara a comparar el desempeño frente a la regresión logística y así determinar si su mayor complejidad aporta mejoras significativas.

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, roc_auc_score

print("--- Entrenando Modelo 2: Random Forest ---")

# Inicialización del modelo
modelo_rf = RandomForestClassifier(
    n_estimators=50,
    n_jobs=-1,
    random_state=42
)

# Entrenamiento
inicio = time.time()
modelo_rf.fit(X_train_resampled, y_train_resampled)
fin = time.time()

# Tiempo de entrenamiento
tiempo_rf = fin - inicio

# Predicciones
y_pred_rf = modelo_rf.predict(X_test_scaled)
y_prob_rf = modelo_rf.predict_proba(X_test_scaled)[:, 1]

# Métricas
acc_rf = accuracy_score(y_test, y_pred_rf)
prec_rf = precision_score(y_test, y_pred_rf)
```

Figura 27. Entrenamiento del modelo Random Forest para la clasificación de transacciones fraudulentas.

```python
# Métricas
acc_rf = accuracy_score(y_test, y_pred_rf)
prec_rf = precision_score(y_test, y_pred_rf)
rec_rf = recall_score(y_test, y_pred_rf)
auc_rf = roc_auc_score(y_test, y_prob_rf)

# Mostrar resultados
print("\n--- Resultados Random Forest ---")
print(f"Accuracy:  {acc_rf:.4f}")
print(f"Precision: {prec_rf:.4f}")
print(f"Recall:    {rec_rf:.4f}")
print(f"AUC-ROC:   {auc_rf:.4f}")
print(f"Tiempo:    {tiempo_rf:.2f} segundos")

# Guardar resultados
resultados_modelos['Random Forest'] = {
    'Accuracy': acc_rf,
    'Precision': prec_rf,
    'Recall': rec_rf,
    'AUC': auc_rf,
    'Tiempo': tiempo_rf
}

predicciones['Random Forest'] = y_pred_rf
probabilidades['Random Forest'] = y_prob_rf
```

Python

```
--- Entrenando Modelo 2: Random Forest ---

--- Resultados Random Forest ---
Accuracy:  0.9725
Precision: 0.7685
Recall:    0.9811
AUC-ROC:   0.9910
Tiempo:    2.84 segundos
```

Figura 28. Resultados de evaluación del modelo Random Forest mediante métricas de clasificación.

4.6.3 Modelo 3: XGBoost

XGBoost se utiliza como el algoritmo más avanzado dentro del proyecto para la detección de fraude, gracias a su alta capacidad para capturar patrones complejos en datos tabulares.

¿Y por qué se utiliza el modelo?

En general XGBoost es considerado estándar en la industria para los problemas de clasificación con datos estructurados, destacándose por:

- Alta precisión en datasets tabulares
- Capacidad de modelar relaciones no lineales complejas
- Excelente rendimiento en problemas de clasificación desbalanceados

¿Cómo funciona?

- Construye árboles de decisión de forma secuencial

- Cada nuevo árbol se enfoca en corregir los errores del modelo anterior
- Optimiza una función de pérdida mediante gradient boosting

Con este enfoque se permite mejorar progresivamente el modelo en cada iteración, capturando patrones que otros algoritmos no detectan.

Configuración utilizada

Se usa los siguientes parametros:

- use_label_encoder = False
- eval_metric = logloss
- random_state = 42

Justificación

- use_label_encoder = False evita advertencias de versiones modernas de XGBoost y asegura compatibilidad con implementaciones actuales.
- eval_metric = logloss permite evaluar correctamente la función de pérdida durante el entrenamiento.
- random_state = 42 garantiza reproducibilidad de los resultados.

Estos parámetros representan configuraciones estándar en entornos profesionales y competiciones de Machine Learning.

Estrategia dentro del sistema

XGBoost cumple el rol de modelo avanzado dentro del sistema, debido a su capacidad para capturar patrones complejos en datos tabulares. En problemas de fraude, detectar la mayor cantidad posible de transacciones fraudulentas es la prioridad principal.

Evaluación

El modelo será evaluado mediante: Accuracy, Precision, Recall (métrica principal) y AUC-ROC. Esto permitirá comparar su rendimiento frente a los modelos anteriores y validar si su mayor complejidad se traduce en mejores resultados.

Como se observa en la **Figura 29**, se realizó el entrenamiento del modelo XGBoost utilizando el conjunto de datos previamente procesado

```python
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, roc_auc_score

print("--- Entrenando Modelo 3: XGBoost ---")

# Inicialización del modelo
modelo_xgb = XGBClassifier(
    use_label_encoder=False,
    eval_metric='logloss',
    random_state=42
)

# Entrenamiento
inicio = time.time()
modelo_xgb.fit(X_train_resampled, y_train_resampled)
fin = time.time()

# Tiempo de entrenamiento
tiempo_xgb = fin - inicio

# Predicciones
y_pred_xgb = modelo_xgb.predict(X_test_scaled)
y_prob_xgb = modelo_xgb.predict_proba(X_test_scaled)[:, 1]

# Métricas
acc_xgb = accuracy_score(y_test, y_pred_xgb)
prec_xgb = precision_score(y_test, y_pred_xgb)
rec_xgb = recall_score(y_test, y_pred_xgb)
auc_xgb = roc_auc_score(y_test, y_prob_xgb)
```

Python

Figura 29. Entrenamiento del modelo XGBoost para la detección de transacciones fraudulentas.

```python
# Mostrar resultados
print("\n--- Resultados XGBoost ---")
print(f"Accuracy:  {acc_xgb:.4f}")
print(f"Precision: {prec_xgb:.4f}")
print(f"Recall:    {rec_xgb:.4f}")
print(f"AUC-ROC:   {auc_xgb:.4f}")
print(f"Tiempo:    {tiempo_xgb:.2f} segundos")

# Guardar resultados
resultados_modelos['XGBoost'] = {
    'Accuracy': acc_xgb,
    'Precision': prec_xgb,
    'Recall': rec_xgb,
    'AUC': auc_xgb,
    'Tiempo': tiempo_xgb
}

predicciones['XGBoost'] = y_pred_xgb
probabilidades['XGBoost'] = y_prob_xgb
```

Python

```
--- Entrenando Modelo 3: XGBoost ---

--- Resultados XGBoost ---
Accuracy:  0.9711
Precision: 0.7620
Recall:    0.9740
AUC-ROC:   0.9869
Tiempo:    0.30 segundos
```

Figura 30. Resultados de evaluación del modelo XGBoost mediante métricas de clasificación.

Como conclusión, se observa que XGBoost presenta un desempeño alto y competitivo en la detección de fraude.

Sin embargo, Random Forest obtuvo un mejor equilibrio general entre precisión, recall, falsos positivos y falsos negativos.

Por lo tanto, el contexto operativo determina la elección del modelo: Random Forest resulta más adecuado cuando se busca el mejor equilibrio general entre detección y precisión, mientras que XGBoost representa una alternativa avanzada para capturar patrones complejos en los datos.

4.7 Fase 5: Evaluación

Una vez finalizado la fase de modelado y entrenamiento de los algoritmos de Machine Learning, se procede a realizar la evaluación comparativa con detalle de cada uno de los modelos desarrollados. El objetivo principal de esta fase es analizar el desempeño de cada algoritmo en la detección de transacciones fraudulentas, utilizando métricas especializadas que permitan medir no solamente la exactitud general del modelo, sino también su capacidad real de discriminación entre las transacciones legítimas y fraudulentas.

En problemas de detección de fraude financiero existe un fuerte desbalance entre clases, debido a que la cantidad de transacciones normales suelen ser significativamente superiores a las fraudulentas. Debido a esto, métricas tradicionales como el accuracy pueden resultar insuficientes para determinar la calidad real de un modelo predictivo.

Debido a esto, la evaluación se complementa mediante:

- Matrices de confusión
- Reportes de clasificación
- Curvas ROC-AUC
- Métricas de precision, recall y F1-score

Cada una de estas herramientas permiten analizar el comportamiento interno de cada algoritmo y determinar cuál es el que ofrece un mejor equilibrio entre la detección de fraude y reducción de errores.

Además, con esta fase se logra justificar técnicamente la selección de los modelos dentro del sistema desarrollado y sustentar las decisiones arquitectónicas implementadas en el pipeline inteligente de detección de fraude.

4.7.1 Matrices de Confusión

La matriz de confusión constituye una de las herramientas más importantes para evaluar modelos de clasificación, ya que permite observar de manera detallada como el modelo realiza las predicciones sobre cada una de las clases.

Con el contexto de detección de fraude, la matriz de confusión permite identificar:

- Verdaderos positivos (fraudes detectados correctamente)
- Verdaderos negativos (transacciones normales detectadas correctamente)
- Falsos positivos (transacciones normales clasificadas como fraude)
- Falsos negativos (fraudes no detectados)

Con los falsos negativos se tiene uno de los mayores riesgos más críticos de sistemas financieros, debido a que este implica tener transacciones fraudulentas sin ser detectados.

Por otro lado, un exceso de falsos positivos puede afectar a nivel usuarios y generar bloqueos innecesarios de operaciones legítimas.

A continuación, se presentará la evaluación de las matrices de confusión obtenida para cada uno de los modelos.

Regresión Logística

La regresión logística presentó una capacidad adecuada para detectar fraudes, logrando identificar una gran cantidad de transacciones fraudulentas reales. Sin embargo, el modelo generó una cantidad considerable de falsos positivos, clasificando múltiples transacciones legítimas como fraudulentas.

Esto da como evidencia que el modelo prioriza la sensibilidad en la detección de fraude, aunque sacrificando precisión en las predicciones.

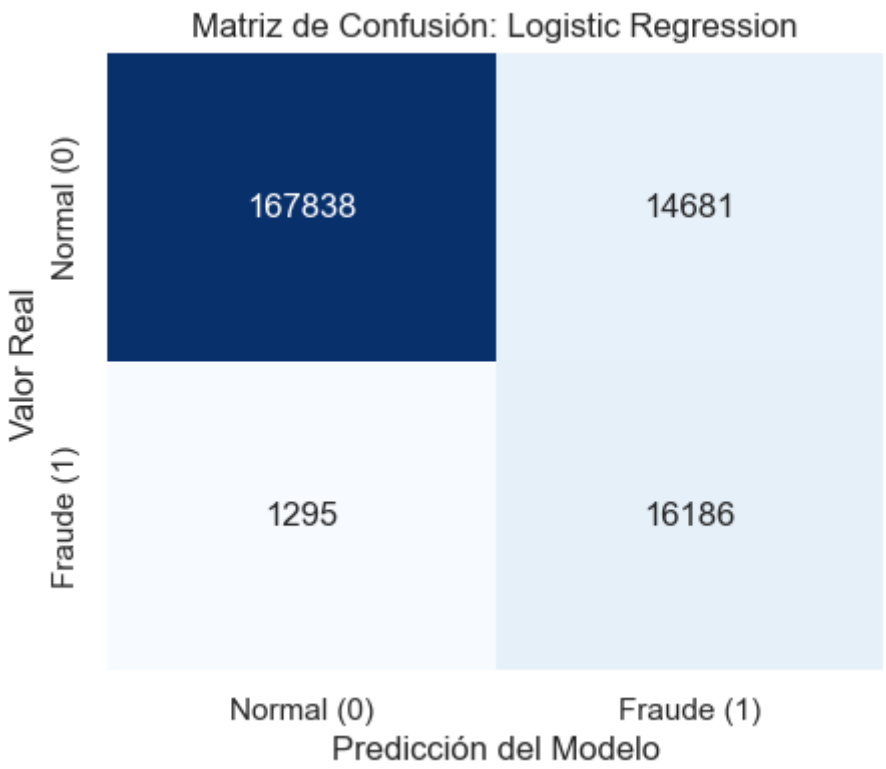


Figura 31. Matriz de confusión del modelo de Regresión Logística

En la **Figura 31** la matriz obtenida muestra:

- 167838 verdaderos negativos
- 16186 verdaderos positivos
- 16681 falsos positivos
- 1295 falsos negativos

Random Forest

El modelo Random Forest mostró un desempeño significativamente superior, logrando reducir drásticamente la cantidad de falsos positivos y falsos negativos.

La matriz de confusión evidencia una excelente capacidad de discriminación entre ambas clases, manteniendo un alto nivel de precisión y recall.

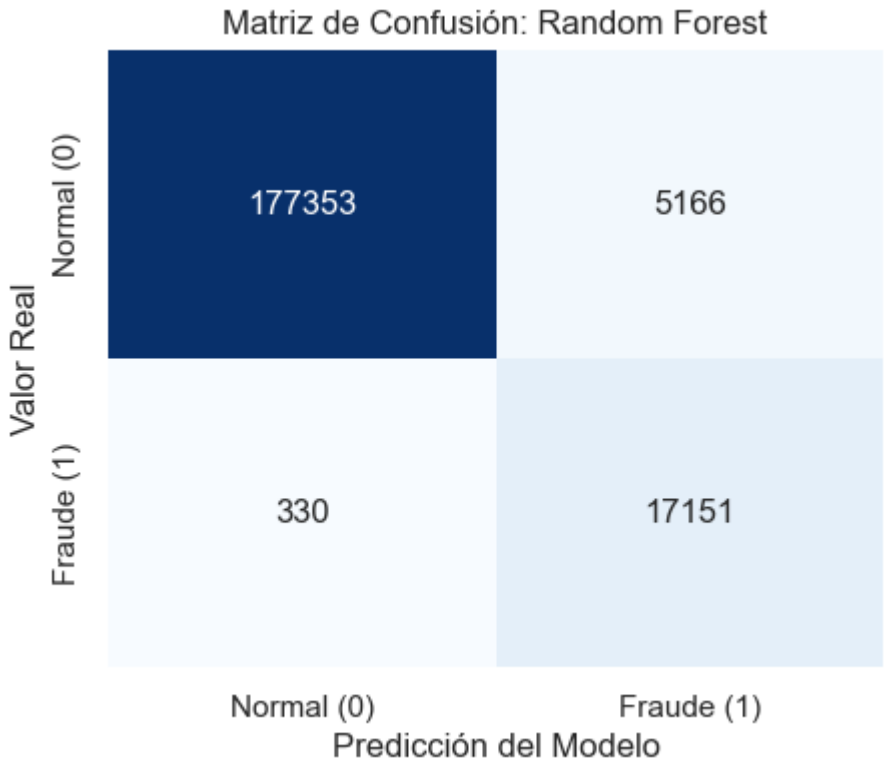


Figura 32. Matriz de confusión del modelo Random Forest

En la **Figura 32** se observa que los resultados obtenidos fueron:

- 177353 verdaderos negativos
- 17151 verdaderos positivos
- 5166 falsos positivos
- 330 falsos negativos

Esto demuestra que el algoritmo logra detectar correctamente la mayoría de las transacciones que son fraudulentas reales mientras mantiene una baja tasa de errores sobre las transacciones legítimas.

El comportamiento observado convierte a random forest en uno de los modelos más robustos, estable y altamente viable para sistemas reales de detección de fraude.

XGBoost

El modelo XGBoost fue un modelo con un equilibrio en su desempeño general dentro de la evaluación.

La matriz de confusión evidencia una alta capacidad para detectar transacciones fraudulentas. Sin embargo, al compararlo con Random Forest, XGBoost presentó una cantidad

ligeramente mayor de falsos positivos y falsos negativos, por lo que su desempeño fue competitivo, pero no superior en esta evaluación.

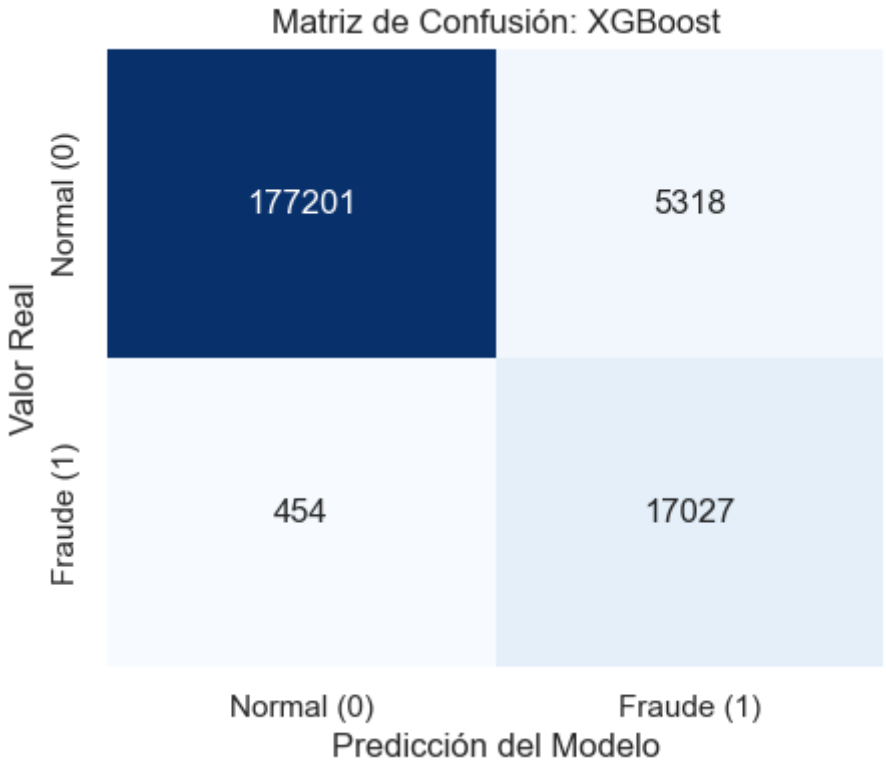


Figura 33. Matriz de confusión del modelo XGBoost

En la **Figura 33** se observa los resultados obtenidos en la matriz que fueron:

- 177201 verdaderos negativos
- 17027 verdaderos positivos
- 5318 falsos positivos
- 454 falsos negativos

Aunque XGBoost presenta un desempeño alto, obtuvo una cantidad ligeramente mayor de falsos positivos y falsos negativos en comparación con Random Forest. Por ello, su rendimiento se considera competitivo, aunque no superior dentro de esta evaluación.

Este comportamiento convierte a XGBoost en una alternativa sólida dentro del sistema, especialmente por su capacidad para capturar patrones complejos en datos tabulares.

4.7.2 Reportes de Clasificación

El reporte de clasificación permite analizar de manera cuantitativa el desempeño de los modelos mediante métricas fundamentales para la evaluación.

Las métricas que se usó fueron:

- Precision
- Recall
- F1-Score
- Accuracy

Cada una de estas métricas proporciona información distinta sobre el comportamiento del modelo.

La métrica precisión da a entender qué el porcentaje de las transacciones clasificadas como fraude realmente corresponden a fraude.

El recall mide la capacidad del modelo para detectar correctamente las transacciones fraudulentas existentes.

El F1-score representa el equilibrio entre precisión y recall.

Finalmente, el accuracy refleja el porcentaje global de predicciones correctas.

Resultados de Regresión Logística

La Regresión Logística alcanzó un recall de 0.93 para la clase fraudulenta demostrando una buena capacidad para detectar fraudes.

Pero obtuvo una precisión de 0.52, lo que da a entender que hay una elevada cantidad de falsos positivos.

```
--- Reporte de Clasificación: Logistic Regression ---
              precision    recall  f1-score   support

         0.0       0.99      0.92      0.95    182519
         1.0       0.52      0.93      0.67     17481

    accuracy                           0.92    200000
   macro avg       0.76      0.92      0.81    200000
weighted avg       0.95      0.92      0.93    200000
```

Figura 34. Reporte de clasificación de Regresión Logística

En la **Figura 34** se logra observar que el modelo alcanzó

- Accuracy: 0.92
- Recall fraude: 0.93

- Precision fraude: 0.52
- F1-score fraude: 0.67

Aunque tiene un recall alto, el bajo nivel de precisión limita su implementación práctica en escenarios reales.

Resultados Random Forest

Random Forest dio un desempeño altamente equilibrado entre la detección y precisión.

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

         0.0       1.00      0.97      0.98    182519
         1.0       0.77      0.98      0.86     17481

    accuracy                           0.97    200000
   macro avg       0.88      0.98      0.92    200000
weighted avg       0.98      0.97      0.97    200000

------------------------------------------------------------
```

Figura 35. Reporte de clasificación de Random Forest

Los resultados que se observan en la **Figura 35** fueron los siguientes:

- Accuracy: 0.97
- Recall fraude: 0.98
- Precision fraude: 0.77
- F1-score fraude: 0.86

El modelo logró detectar casi la totalidad de transacciones fraudulentas manteniendo simultáneamente una baja tasa de falsas alarmas. Este desempeño puede explicarse por la capacidad de Random Forest para manejar relaciones no lineales y reducir el sobreajuste mediante múltiples árboles de decisión.

Sin embargo, aunque las métricas obtenidas son altas, en entornos financieros reales el comportamiento del modelo puede variar debido a cambios constantes en los patrones de fraude y en la distribución de los datos. Por ello, es importante complementar el sistema con procesos de monitoreo y actualización periódica del modelo.

Resultados de XGBoost

XGBoost presentó un desempeño altamente competitivo en la detección de fraude, alcanzando métricas similares a las obtenidas por Random Forest.

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

         0.0       1.00      0.97      0.98    182519
         1.0       0.76      0.97      0.86     17481

    accuracy                           0.97    200000
   macro avg       0.88      0.97      0.92    200000
weighted avg       0.98      0.97      0.97    200000
```

Figura 36. Reporte de clasificación de XGBoost

En la **Figura 36** se observan los siguientes resultados:

- Accuracy: 0.97
- Recall fraude: 0.97
- Precision fraude: 0.76
- F1-score fraude: 0.86

El modelo presentó un recall extremadamente alto, minimizando la cantidad de fraudes no detectados.

Aunque la precisión es ligeramente menor a Random Forest, mantiene su desempeño general sobresaliente y una gran capacidad de discriminación entre transacciones legítimas y fraudulentas.

4.7.3 Curvas ROC-AUC

Las curvas ROC permiten evaluar la capacidad de discriminación de un modelo de clasificación bajo distintos umbrales de decisión.

Estás curvas comparan:

- Tasa de verdaderos positivos
- Tasa de falsos positivos

El área bajo la curva ROC (AUC) representa la capacidad global del modelo para diferenciar correctamente entre transacciones normales y fraudulentas.

El valor cercano a 1 indica un excelente desempeño predictivo.

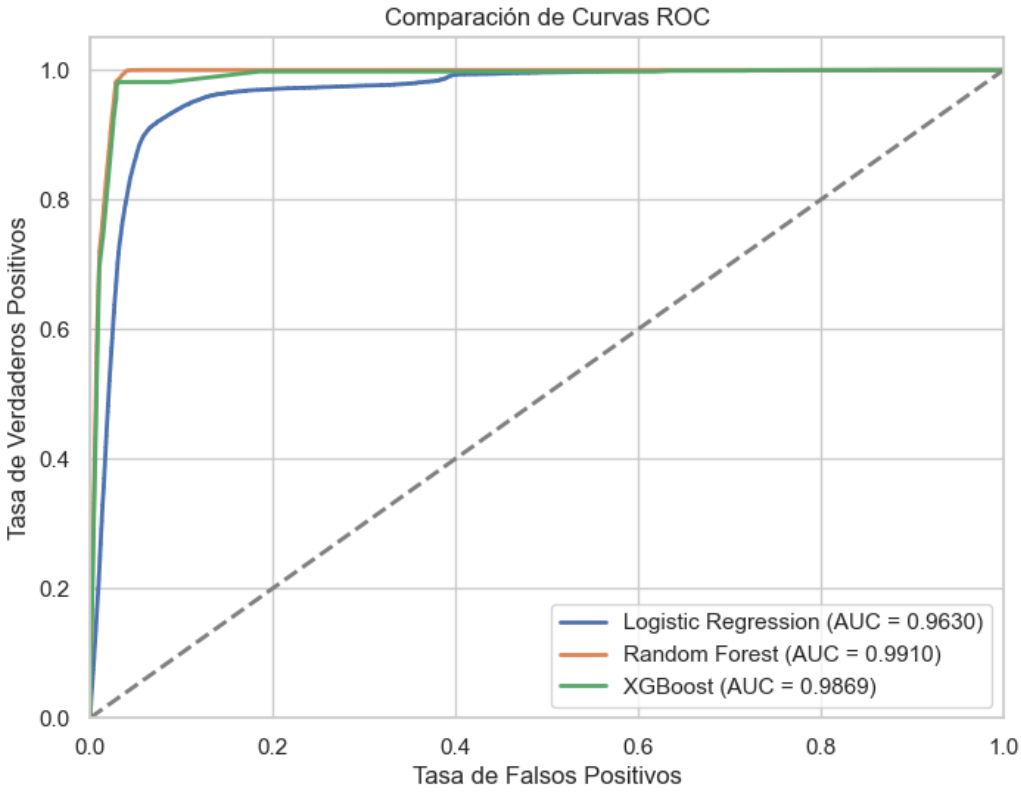


Figura 37. Comparación de curvas ROC-AUC entre modelos

En la **Figura 37** se puede observar los siguientes resultados:

- Regresión Logística: AUC = 0.9630
- Random Forest: AUC = 0.9910
- XGBoost: AUC = 0.9869

La grafica comparativa muestra que tanto Random Forest como XGBoost presentan curvas muy cercanas al valor óptimo, reflejando una gran capacidad de discriminación.

Random Forest alcanzó el mayor valor AUC, confirmando su superioridad global en el problema de detección de fraude.

Interpretación de Resultados

A partir de la evaluación comparativa realizada, se observa que todos los modelos presentan un desempeño elevado en la detección de fraude financiero. Sin embargo, existen diferencias importantes en el comportamiento de cada algoritmo.

La Regresión Logística funciona adecuadamente como modelo baseline, permitiendo establecer una referencia inicial de comparación. Aunque presenta un buen recall, genera una cantidad considerable de falsos positivos, lo que podría afectar negativamente la experiencia de usuarios legítimos.

Random Forest mostró el mejor equilibrio entre precisión y recall, manteniendo simultáneamente una alta capacidad de detección y una baja tasa de falsas alarmas. Además, obtuvo el mayor valor AUC-ROC y la menor cantidad de falsos negativos dentro de los modelos evaluados, convirtiéndose en una opción robusta y estable para escenarios financieros reales.

XGBoost presentó un desempeño altamente competitivo, alcanzando métricas muy cercanas a las obtenidas por Random Forest. Su capacidad para capturar patrones complejos lo convierte en una alternativa sólida para sistemas orientados a la detección de fraude.

No obstante, XGBoost presenta un mayor costo computacional respecto a Random Forest y Regresión Logística debido a la complejidad de su proceso iterativo de entrenamiento. A pesar de ello, mantiene un desempeño sobresaliente en tareas de clasificación sobre datos financieros estructurados.

Selección Estratégica del Modelo

En lugar de seleccionar un único modelo como única solución, con el sistema relacionado se implementa una arquitectura flexible basada en un pipeline configurable.

Este enfoque permite:

- Seleccionar dinámicamente el algoritmo de Machine Learning
- Ajustar estrategias de preparación de datos
- Configurar escalado, imputación y balanceo
- Adaptar el sistema según necesidades operativas específicas

De esta manera, el sistema puede utilizar:

- Random Forest cuando se requiera maximizar la detección de fraude manteniendo un equilibrio entre precisión y recall.
- XGBoost cuando se necesite un modelo con alta capacidad para capturar patrones complejos en los datos.
- Regresión Logística como modelo base o referencia comparativa.

Con este enfoque se da un escenario real en las instituciones financieras, donde la selección de los modelos depende directamente del riesgo que la organización esté dispuesta a asumir.

La arquitectura propuesta da una gran flexibilidad, escalabilidad y capacidad de adaptación ante los cambios futuros en los datos o condiciones operativas.

4.8 Fase 6: Despliegue

En la fase de despliegue tiene el objetivo de integrar el modelo predictivo desarrollado dentro de un entorno funcional e interactivo, ayudando a utilizar el sistema en escenarios cercanos a la realidad con un contexto dentro de una operación financiera.

A diferencia de un enfoque limitado únicamente el entrenamiento de modelos dentro de entornos experimentales como puede ser el Jupyter Notebook, con este proyecto se implementó un prototipo funcional mediante una interfaz web interactiva desarrollada con Streamlit, permitiendo la integración de procesos automatizados en la carga de datos, preprocesamiento, entrenamiento, predicción y visualización de resultados.

El sistema fue diseñado bajo una arquitectura modular orientada a facilitar tanto la actualización dinámica de modelos como a la ejecución de predicciones sobre nuevos conjuntos de datos transaccionales.

De esta manera, el despliegue no solamente representa la ejecución y monitoreo del modelo entrenado, sino la implementación completa de un entorno funcional capaz de:

- Recibir nuevos datasets históricos.
- Ejecutar pipelines automáticos de limpieza y transformación.
- Entrenar modelos de Machine Learning.
- Generar predicciones sobre nuevas transacciones.
- Visualizar métricas e indicadores mediante dashboards interactivos.

4.8.1 Arquitectura general del sistema

El sistema fue desarrollado utilizando una arquitectura modular organizada por componentes independientes, facilitando la mantenibilidad, escalabilidad y separación de funciones dentro del sistema.

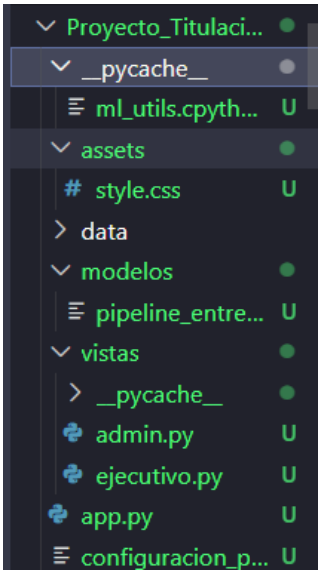


Figura 38. Arquitectura modular y estructura de directorios del sistema desarrollado.

Como se observa en la **Figura 38** la estructura del proyecto es de la siguiente manera:

Proyecto_Titulacion/

│

├── app.py

├── ml_utils.py

├── configuracion_pipeline.pkl

│

├── assets/

│ └── style.css

│

├── modelos/

│ └── pipeline_entrenado.pkl

│

├── vistas/

│ ├── admin.py

│ └── ejecutivo.py

│

├── data/

Descripción de los componentes

- **app.py**
  Archivo principal encargado de la ejecución general de la aplicación, la navegación entre módulos y la simulación de acceso al prototipo
- **ml_utils.py**
  Contiene la lógica relacionada con el procesamiento de datos, entrenamiento, predicción y gestión del pipeline de Machine Learning.
- **modelos/**
  Directorio destinado al almacenamiento persistente de modelos entrenados mediante serialización con Joblib.
- **vistas/**
  Contiene los módulos visuales independientes del sistema:
- Panel Administrador
- Panel Ejecutivo
- **assets/style.css**
  Archivo encargado de la personalización visual de la interfaz gráfica.
- **configuracion_pipeline.pkl**
  Archivo utilizado para almacenar configuraciones dinámicas del pipeline implementado.

4.8.2 Tecnologías utilizadas en el despliegue

Para la implementación y despliegue del sistema se utilizaron distintas tecnologías que están enfocadas tanto en el desarrollo del modelo predictivo como a la construcción de la interfaz con un entorno interactivo.

Tecnología	Función
Python	Lenguaje principal del sistema
Streamlit	Desarrollo de la interfaz web

Tecnología	Función
Pandas	Procesamiento de datos
Scikit-learn	Modelos y pipelines de ML
XGBoost	
Random Forest	Modelos avanzados de clasificación
Regresión Logística	
Joblib	Persistencia de modelos
CSS	Personalización visual

Con estas herramientas se logró integrar correctamente el procesamiento automático de los datos, el entrenamiento dinámico de los modelos y la visualización interactiva de los resultados obtenidos dentro de la aplicación.

4.8.3 Simulación de autenticación y control de acceso

El sistema implementa un mecanismo básico de Login mediante usuarios y contraseñas predefinidas, con fines demostrativos. Este mecanismo permite separar la navegación entre los módulos Administrador y Ejecutivo, pero no constituye un sistema de autenticación empresarial complejo ni una gestión de seguridad de identidades.

Se definieron dos tipos de usuarios:

- Administrador
- Ejecutivo

Cada rol posee permisos específicos dentro del sistema.

El módulo administrador tiene acceso a:

- Reentrenamiento del modelo.
- Configuración del pipeline.
- Gestión de datasets históricos.

Mientras que el módulo ejecutivo se enfoca en:

- Visualización de resultados.
- Análisis de predicciones.
- Monitoreo de indicadores.

Como se observa en la **Figura 39** el control de sesión fue implementado mediante st.session_state, ayudando a mantener la autenticación activa durante el uso de la aplicación.

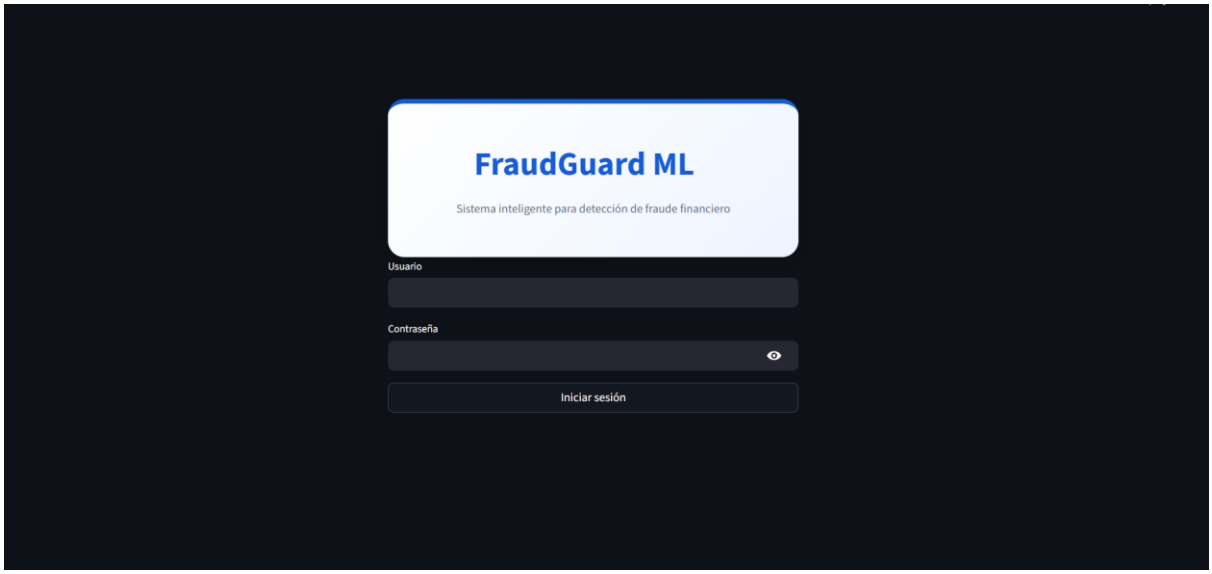


Figura 39. Interfaz de autenticación del sistema FraudGuard ML.

Implementación del Panel Administrador

El panel del Administrador constituye el núcleo operativo del sistema, este permite configurar y ejecutar el proceso del entrenamiento y actualización de los modelos predictivos.

Dentro del módulo se implementó las siguientes funcionalidades:

- Carga de datasets históricos en formato CSV.
- Selección dinámica del algoritmo de Machine Learning.

Activación opcional de:

- Imputación,
- Escalado robusto,
- Ejecución automática del pipeline de entrenamiento.
- Visualización previa del dataset cargado.
- Generación de métricas e indicadores.

El sistema fue diseñado para el reentrenamiento dinámico sin la necesidad de modificar directamente el código fuente o también acceder a el entorno de desarrollo del mismo.

Esto permite aproximar el funcionamiento del prototipo a los escenarios reales de actualización continua de modelos.

La **Figura 40** muestra el Panel Administrador del sistema FraudGuard ML, desde donde se gestionan las configuraciones de entrenamiento y actualización de modelos.

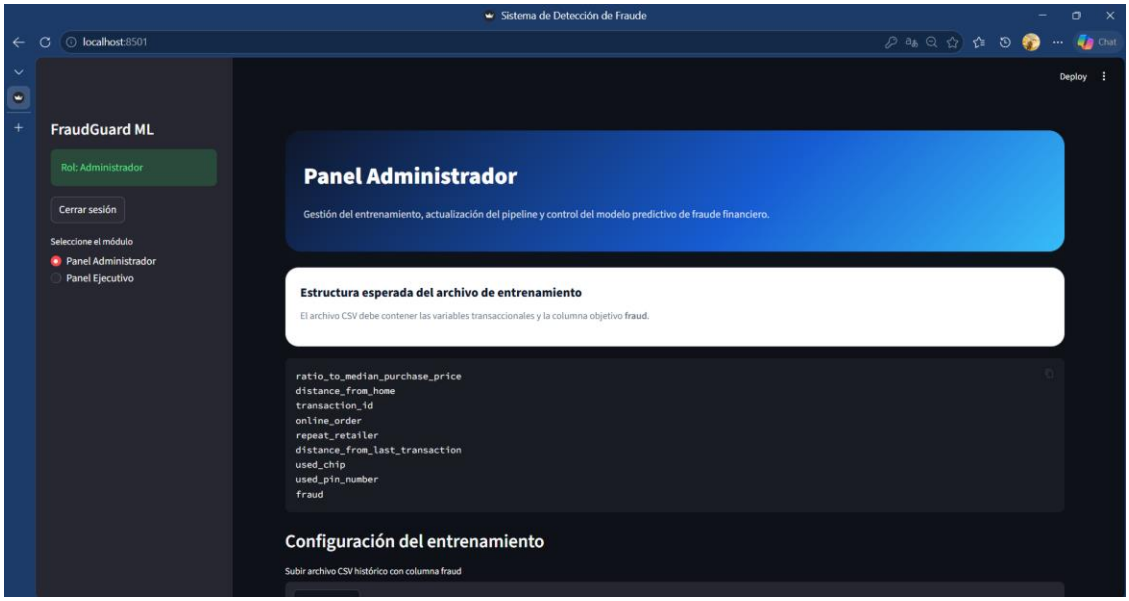


Figura 40. Panel Administrador del sistema FraudGuard ML.

Como se observa en la **Figura 41**, el sistema permite configurar dinámicamente distintos parámetros del pipeline de entrenamiento.

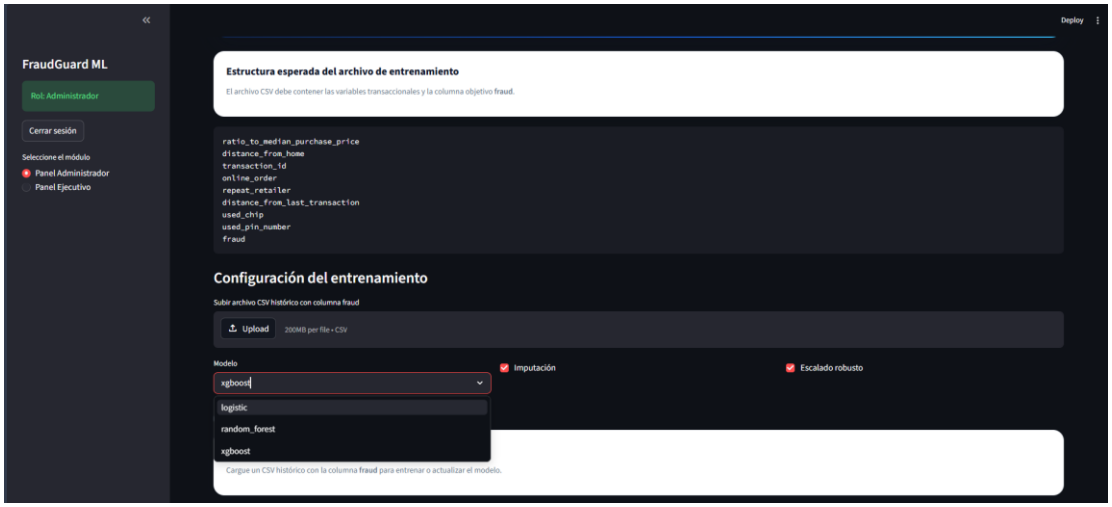


Figura 41. Configuración dinámica del pipeline de entrenamiento.

La **Figura 42** presenta la vista previa del dataset cargado en el sistema antes de ejecutar el proceso de entrenamiento



Figura 42. Vista previa del dataset cargado en el sistema.

Pipeline automatizado de procesamiento de datos

Uno de los principales componentes del sistema es el pipeline automatizado de procesamiento de datos.

- Este pipeline permite ejecutar de manera secuencial:
- Limpieza de variables.
- Normalización de datos binarios. Conversión de formatos.
- Imputación de valores faltantes.
- Escalado robusto.
- Entrenamiento del modelo seleccionado.

El pipeline fue creado para trabajar con datasets parcialmente inconsistentes o incompletos en muchos casos, así simulando escenarios reales presentes en sistemas financieros.

La automatización de este proceso permite reducir de forma considerable la intervención manual y facilita la reutilización del sistema con nuevos conjuntos de datos históricos.

Persistencia y actualización de modelos

Con el objetivo de mantener la reutilización del modelo entrenado dentro del sistema, se integró un mecanismo de persistencia mediante serialización utilizando Joblib.

Los modelos entrenados son almacenados en archivos. pkl, permitiendo:

- reutilizar modelos previamente entrenados,
- evitar reentrenamientos innecesarios,
- actualizar versiones del pipeline.

Entre los archivos generados están:

- pipeline_entrenado.pkl
- configuracion_pipeline.pkl

Con este enfoque se logra que el sistema mantenga una arquitectura adaptable y preparada para las futuras actualizaciones.

Implementación del Panel Ejecutivo

Como se observa en la **Figura 43** el panel Ejecutivo fue diseñado para dar una visualización simplificada y orientada al análisis de resultados del sistema de detección de fraude.

Con este módulo se logra:

- Cargar nuevas transacciones.
- Generar predicciones automáticas.
- Visualizar indicadores clave.
- Analizar resultados mediante tablas y dashboards interactivos.

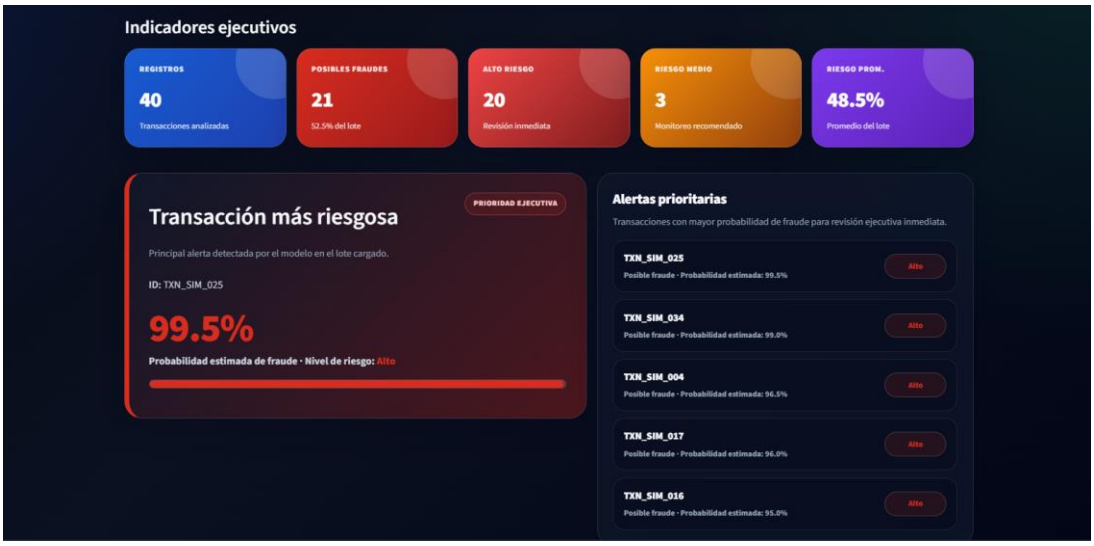


Figura 43. Panel ejecutivo con indicadores de riesgo y alertas prioritarias de fraude financiero.

La interfaz fue desarrollada utilizando componentes visuales modernos y personalizados mediante como se puede observar en la **Figura 44** CSS, priorizando:

- Claridad visual,
- Facilidad de uso,
- Interpretación rápida de resultados.

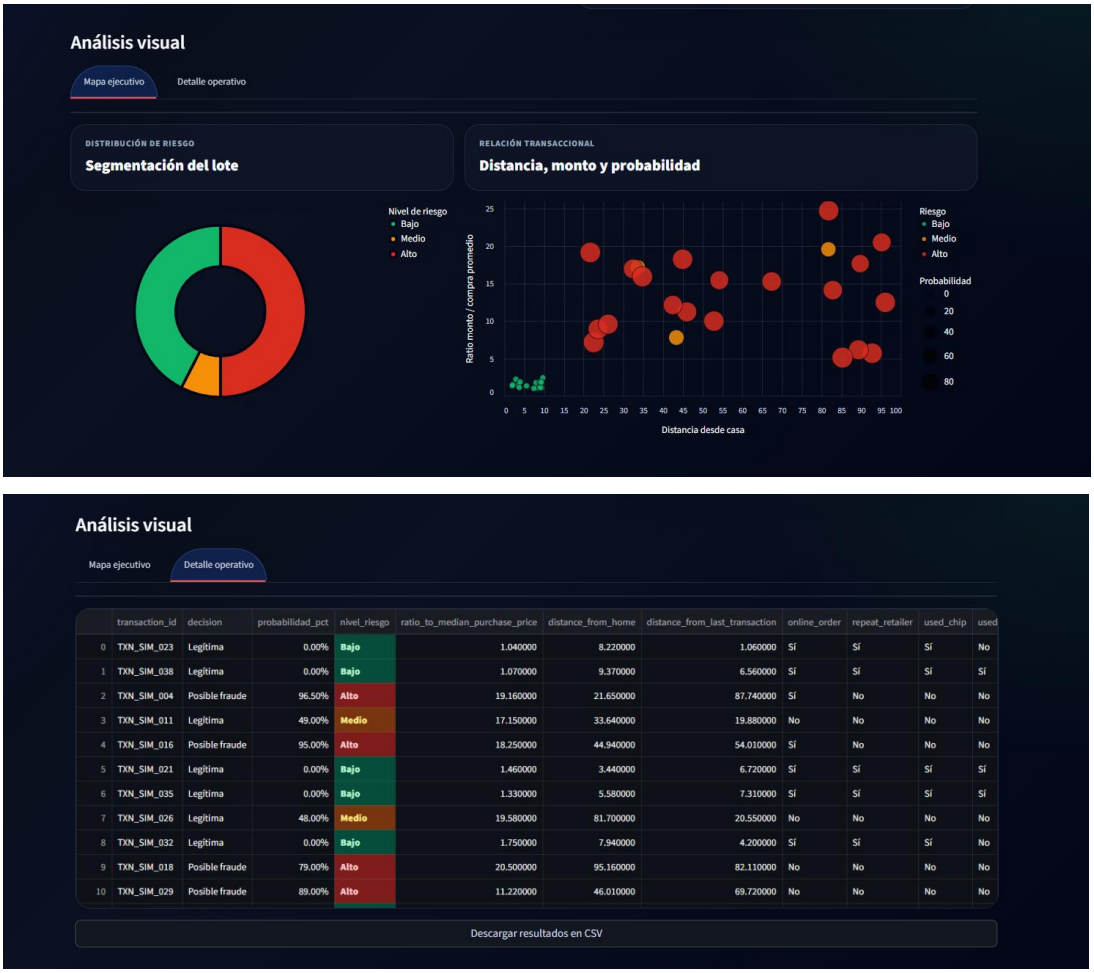


Figura 44. Módulo de análisis visual del sistema ejecutivo: segmentación de riesgo y detalle operativo de transacciones.

Flujo operativo general del sistema

Como se observa en la **Figura 45**, el funcionamiento general del sistema sigue el siguiente flujo:

- El usuario carga un archivo CSV.
- El pipeline procesa automáticamente los datos.
- El sistema aplica transformaciones y limpieza.

- El modelo seleccionado genera predicciones.
- Los resultados son visualizados mediante dashboards e indicadores.

Con este flujo se permite automatizar gran parte del ciclo operativo de detección de fraude dentro del entorno desarrollado

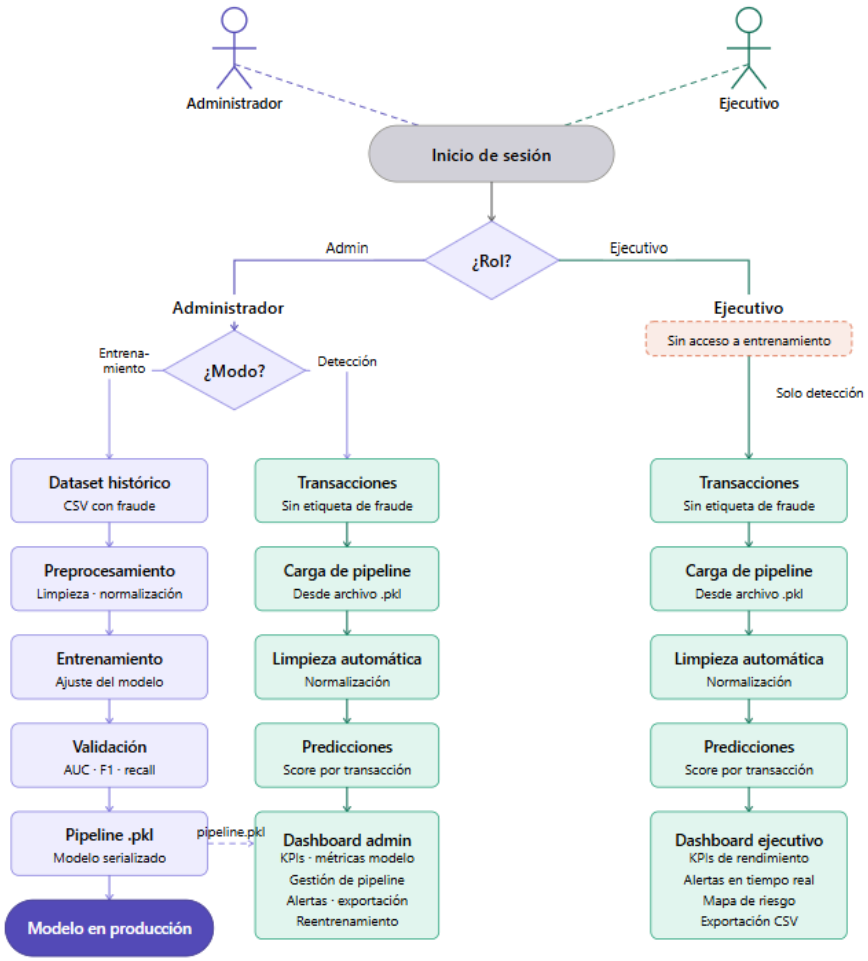


Figura 45. Flujo operativo y arquitectura funcional del sistema de detección de fraude financiero.

El despliegue del sistema permitió validar la integración funcional entre:

- procesamiento automático de datos,
- modelos predictivos,
- persistencia de pipelines,
- visualización interactiva.

El prototipo desarrollado logró:

- entrenar modelos dinámicamente,
- procesar datasets de gran volumen,
- generar predicciones automáticas,
- mantener una arquitectura modular y adaptable.

Además, el sistema permitió actualizar modelos utilizando nuevos conjuntos de datos sin necesidad de modificar directamente el entorno interno de desarrollo.

Con esto se demuestra la viabilidad del enfoque propuesto para los escenarios financieros orientados a la detección de fraude.

En definitiva, la fase de despliegue permitió que el modelo predictivo sea funcional e interactivo orientado a la detección de fraude financiero.

La implementación mediante Streamlit facilitó la construcción de un entorno accesible para los usuarios administrativos y ejecutivos, integrando procesos automatizados de entrenamiento, predicción y visualización de los resultados.

El sistema desarrollado no solamente demuestra la viabilidad técnica del uso de modelos de Machine Learning para la detección de fraude, sino también la importancia de integrar estos modelos dentro de arquitecturas funcionales capaces de operar en escenarios cercanos a los entornos reales.

Con ello, se logra completar el ciclo completo propuesto por la metodología CRISP-DM desde la comprensión del problema hasta el despliegue operativo del sistema.

4.9 Módulo complementario para detección de lavado de activos

Como complemento al módulo principal de la detección de fraude en transacciones con tarjeta de crédito, la incorporó un segundo módulo orientado a la identificación de posibles operaciones relacionadas con lavado de activos. Esta ampliación permitió demostrar que la arquitectura desarrollada no se limita a un único tipo de fraude financiero, sino que adaptarse a diferentes escenarios de análisis de riesgo mediante el uso de pipelines de aprendizaje automático reutilizables.

El módulo complementario fue implementado utilizando un dataset IBM AML, compuesto por registro transaccionales asociados a operaciones bancarias. Entre sus variables se encuentra el banco de origen, banco de destino, cuentas involucradas, montos recibidos y

pagados, moneda utilizada, formato de pago y variables objetivo Is Laundering, la cual indica si una transacción está asociada o no a lavado de activos.

A diferencia del módulo principal, este apartado no replica nuevamente todas las fases CRISP-DM, sino que presenta los aspectos más relevantes de adaptación del sistema: preparación del dataset, entrenamiento de modelos, evaluación comparativa e integración de resultados en el dashboard ejecutivo.

4.9.1 Contexto del módulo AML

El dataset original presentó un fuerte desbalance de clases, debido a que la mayoría de las transacciones correspondían a operaciones normales, mientras que los casos asociados a lavado de activos representaban una proporción reducida. Esta característica es común en problemas de detección de riesgo financiero, donde los eventos sospechosos suelen ser minoritarios frente al volumen total de operaciones legítimas.

En este caso para evitar que los modelos aprendieran a clasificar la mayoría de las transacciones como normales, se usó una estrategia de muestreo controlado. Como se observa en la Figura 46 la muestra experimental utilizada quedó distribuida en aproximadamente 83,33% de transacciones normales y 16,67% de transacciones asociadas a lavado de activos. Esta proporción logro mejorar el aprendizaje de la clase minoritaria por completo en el contexto de desbalance propio del problema.

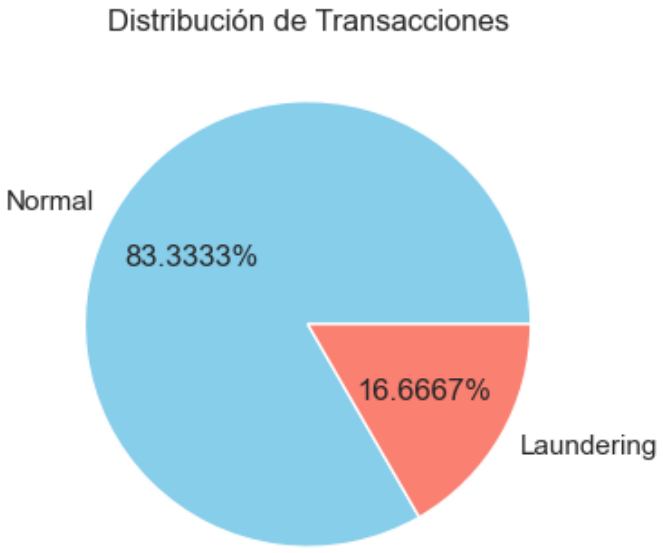


Figura 46. Distribución de transacciones normales y transacciones asociadas a lavado de activos.

4.9.2 Preparación resumida del dataset IBM AML

Para lograr adaptar el dataset AML al pipeline del sistema, se realizaron varias transformaciones previas. En primer lugar, se procesó la variable temporal para extraer componentes como hora, día y mes, permitiendo incorporar patrones temporales al modelo. Posteriormente, se eliminó columnas de alta cardinalidad, como por ejemplo identificadores de cuentas, ya que estas podían aumentar significativamente el costo computacional sin importar una generalización adecuada para nuevas transacciones.

También se aplicó técnicas de imputación para el trato de los valores nulos, escalado robusto para variables numéricas y codificación One-Hot para variables categóricas. Estas operaciones fueron integradas dentro de un pipeline de Scikit-learn, permitiendo que el mismo flujo de preprocesamiento utilizado durante el entrenamiento se aplique posteriormente en la etapa de predicción.

4.9.3 Evaluación comparativa de modelos

Para el módulo AML se evaluaron tres modelos de clasificación supervisada: Regresión Logística, Random Forest y XGBoost. La evaluación se realizó mediante matriz de confusión, precision, recall, F1-score y curva ROC-AUC, con énfasis en la clase asociada a lavado de activos.

En la **Figura 47** se presenta la matriz de confusión correspondiente a la Regresión Logística. Este modelo obtuvo 900 verdaderos positivos para la clase laundering y 135 falsos negativos, lo que evidencia una buena capacidad para detectar operaciones sospechosas. Sin embargo, también presentó 627 falsos positivos, lo cual indica que marcó como sospechosas varias transacciones normales.

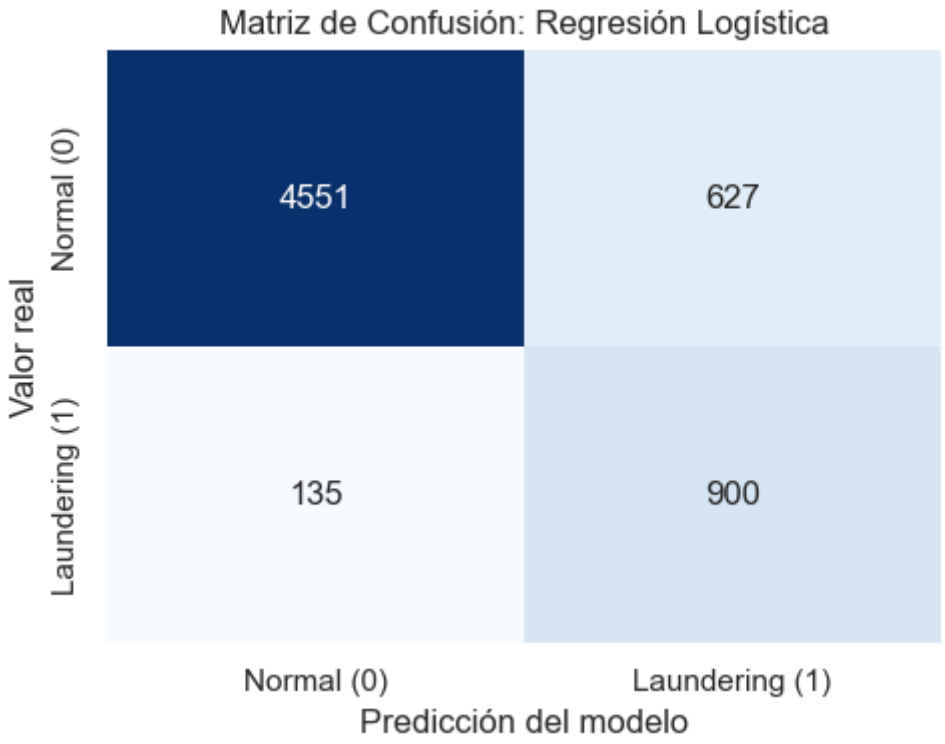


Figura 47. Matriz de confusión del modelo de Regresión Logística para el módulo AML.

El reporte de clasificación de la Regresión Logística, mostrado en la **Figura 48**, evidencia una exactitud de 0,88 y un recall de 0,87 para la clase laundering. Esto indica que el modelo logró identificar una proporción considerable de transacciones sospechosas, aunque con una precisión menor frente a los modelos de ensamble.

```
--- Reporte de Clasificación: Regresión Logística ---
              precision    recall  f1-score   support

      Normal       0.97      0.88      0.92      5178
  Laundering       0.59      0.87      0.70      1035

    accuracy                           0.88      6213
   macro avg       0.78      0.87      0.81      6213
weighted avg       0.91      0.88      0.89      6213

------------------------------------------------------------
```

Figura 48. Reporte de clasificación del modelo de Regresión Logística.

En la **Figura 49** se observa la matriz de confusión del modelo Random Forest. Este modelo obtuvo 701 verdaderos positivos y redujo considerablemente los falsos positivos a 138,

en comparación con la Regresión Logística y XGBoost. No obstante, presentó 334 falsos negativos, lo cual significa que dejó pasar una mayor cantidad de casos de lavado de activos.

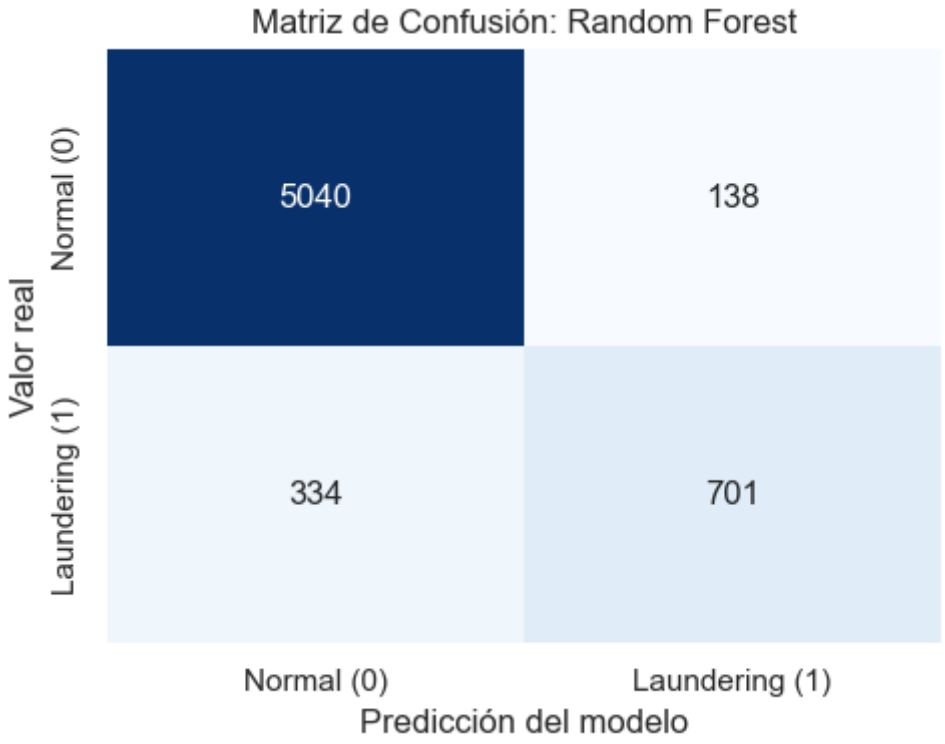


Figura 49. Matriz de confusión del modelo Random Forest para el módulo AML.

El reporte de clasificación del Random Forest se presenta en la **Figura 50**. Este modelo alcanzó una exactitud de 0,92, una precisión de 0,84 para la clase laundering y un F1-score de 0,75. Estos resultados evidencian que Random Forest obtuvo el mejor equilibrio operativo, especialmente cuando se busca reducir la cantidad de falsos positivos.

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

      Normal       0.94      0.97      0.96      5178
  Laundering       0.84      0.68      0.75      1035

    accuracy                           0.92      6213
   macro avg       0.89      0.83      0.85      6213
weighted avg       0.92      0.92      0.92      6213

------------------------------------------------------------
```

Figura 50. Reporte de clasificación del modelo Random Forest.

En la **Figura 51** se muestra la matriz de confusión del modelo XGBoost. Este modelo obtuvo 974 verdaderos positivos y únicamente 61 falsos negativos, lo que representa la mayor capacidad de detección de transacciones asociadas a lavado de activos entre los modelos evaluados. Sin embargo, también presentó 750 falsos positivos, por lo que tiende a generar más alertas sobre transacciones normales.

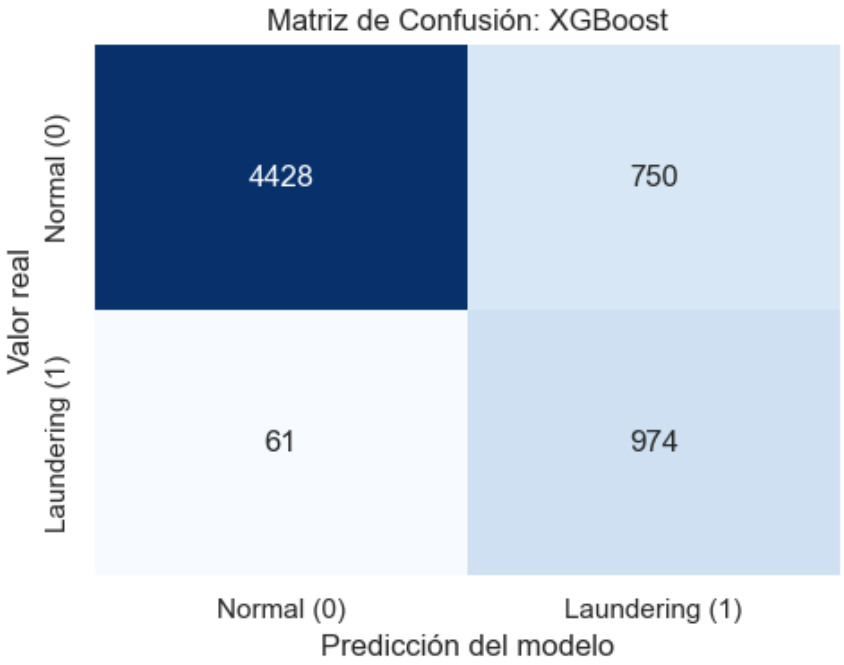


Figura 51. Matriz de confusión del modelo XGBoost para el módulo AML.

El reporte de clasificación de XGBoost, presentado en la **Figura 52**, muestra un recall de 0,94 para la clase laundering. Este resultado es especialmente relevante en contextos de prevención de lavado de activos, ya que reducir los falsos negativos permite disminuir el riesgo de omitir operaciones sospechosas.

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

      Normal       0.99      0.86      0.92      5178
  Laundering       0.56      0.94      0.71      1035

    accuracy                           0.87      6213
   macro avg       0.78      0.90      0.81      6213
weighted avg       0.92      0.87      0.88      6213

------------------------------------------------------------
```

Figura 52. Reporte de clasificación del modelo XGBoost.

La comparación de las curvas ROC-AUC se presenta en la **Figura 53**. En este análisis, XGBoost obtuvo el mayor valor AUC-ROC, con 0,9672, seguido de Random Forest con 0,9597 y Regresión Logística con 0,9003. Esto indica que XGBoost presentó la mejor capacidad discriminativa global para diferenciar entre transacciones normales y transacciones asociadas a lavado de activos.

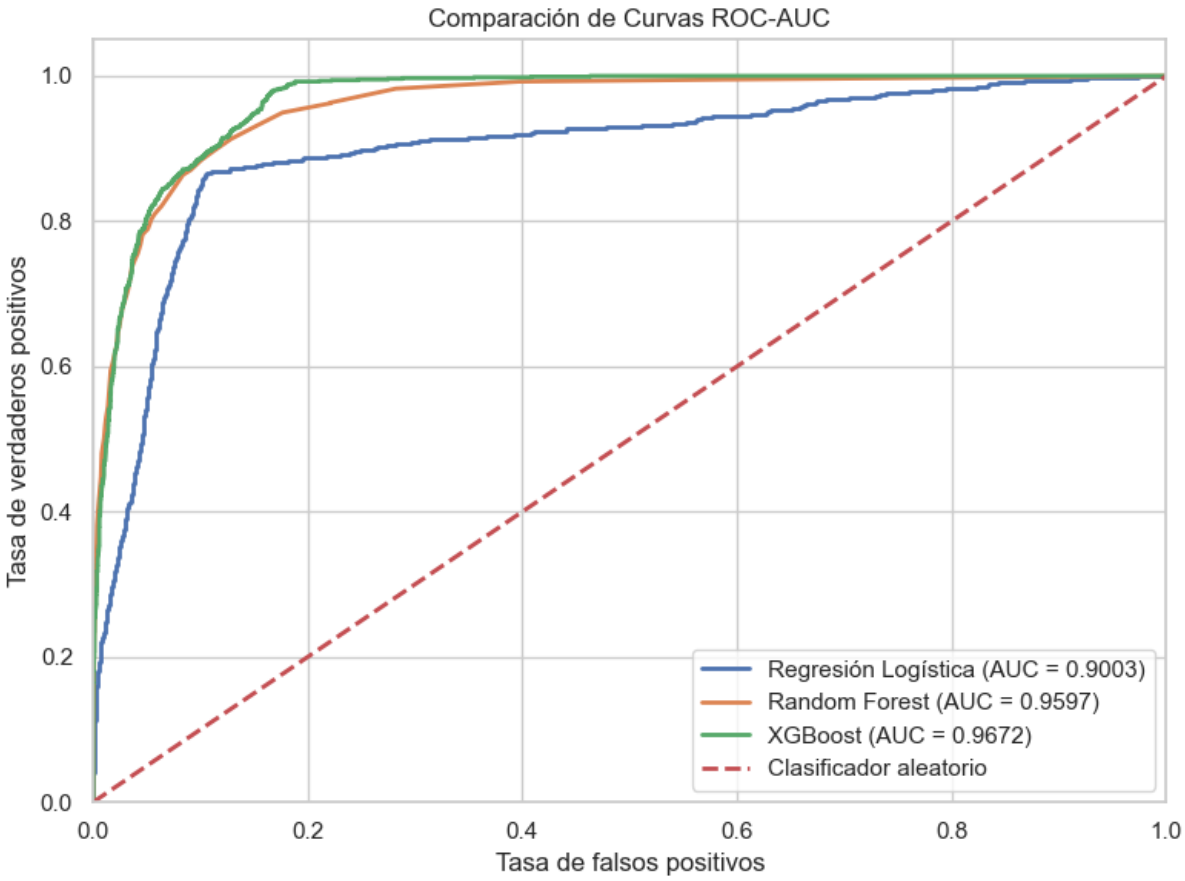


Figura 53. Comparación de curvas ROC-AUC de los modelos evaluados en el módulo AML.

La comparación general de métricas se resume en la **Figura 54**.

Modelo	Accuracy	Precision Laundering	Recall Laundering	F1-score Laundering	AUC-ROC
Regresión Logística	0,88	0,59	0,87	0,7	0,9003
Random Forest	0,92	0,84	0,68	0,75	0,9597
XGBoost	0,87	0,56	0,94	0,71	0,9672

Figura 54. Comparación de métricas de los modelos evaluados en el módulo AML.

Con estos resultados, se logra observar que en este caso Random Forest ofrece el mejor balance general entre la precisión y la recuperación de los casos sospechosos. Sin embargo, XGBoost destaca como un modelo más sensible para la detección de lavado de activo, debido a su recall que es mayor y su mejor capacidad discriminativa global media mediante el AUC-ROC. Por esos motivos, XGBoost resulta adecuado cuando es de prioridad máxima la detección de operaciones sospechosas, mientras que Random Forest representa una alternativa sólida cuando se busca reducir el número de alertas falsas.

4.9.4 Análisis visual de variables relevantes

Además de la evaluación de modelos, se realizó un análisis visual de las principales variables del dataset AML. En la **Figura 55** se presentan los outliers de la variable Amount Received, mientras que en la **Figura 56** se muestran los outliers de la variable Amount Paid. En ambos casos se evidencia la existencia de transacciones con montos significativamente superiores al comportamiento general.

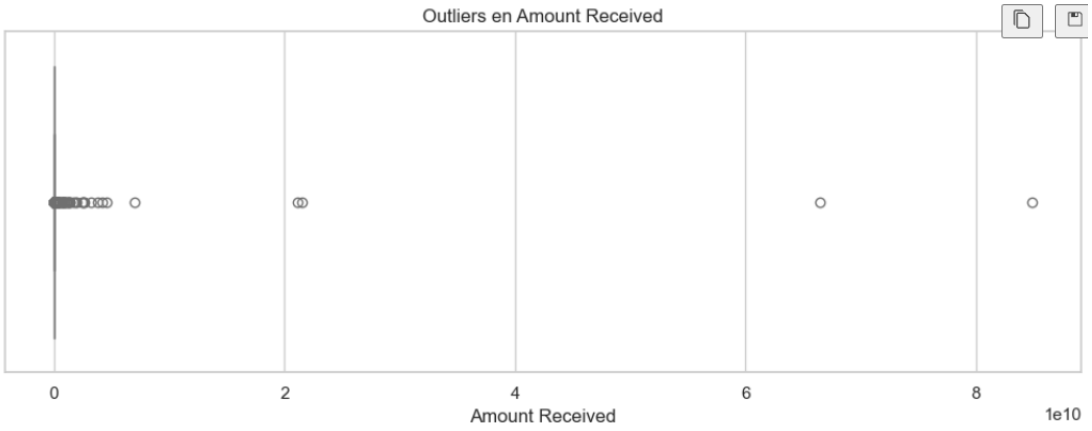


Figura 55. Detección de outliers en la variable Amount Received.

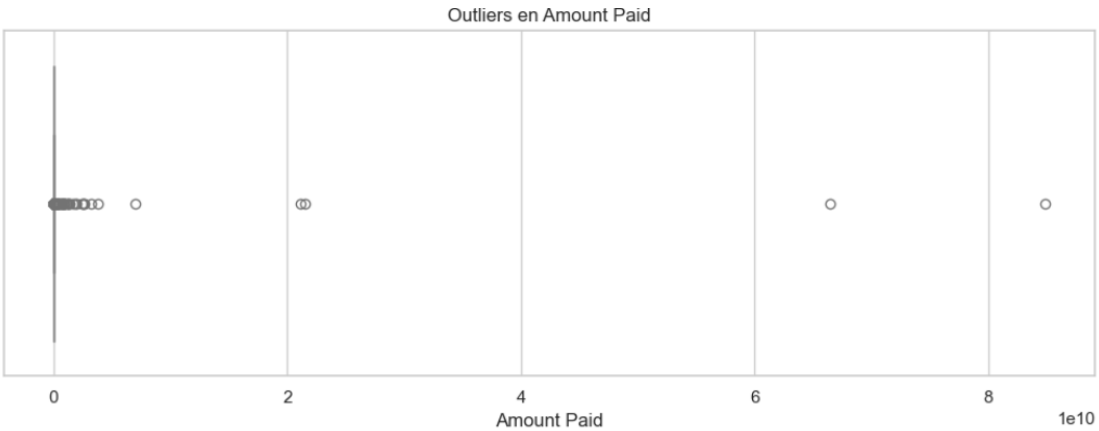


Figura 56. Detección de outliers en la variable Amount Paid.

Estos valores extremos son relevantes en el análisis de riesgo financiero, ya que las operaciones con montos atípicos pueden requerir una revisión adicional. No obstante, la existencia de un monto elevado no implica por sí solo lavado de activos; por ello, el modelo considera múltiples variables en conjunto para generar una predicción.

En la **Figura 57** se muestra el mapa de correlación de las variables numéricas. Se observa una correlación alta entre Amount Received y Amount Paid, lo cual es esperable debido a que ambas variables representan valores monetarios relacionados dentro de la transacción. Sin embargo, la variable objetivo Is Laundering no presenta una correlación lineal fuerte con una única variable numérica, lo que sugiere que la detección de lavado de activos depende de patrones combinados entre distintos atributos.

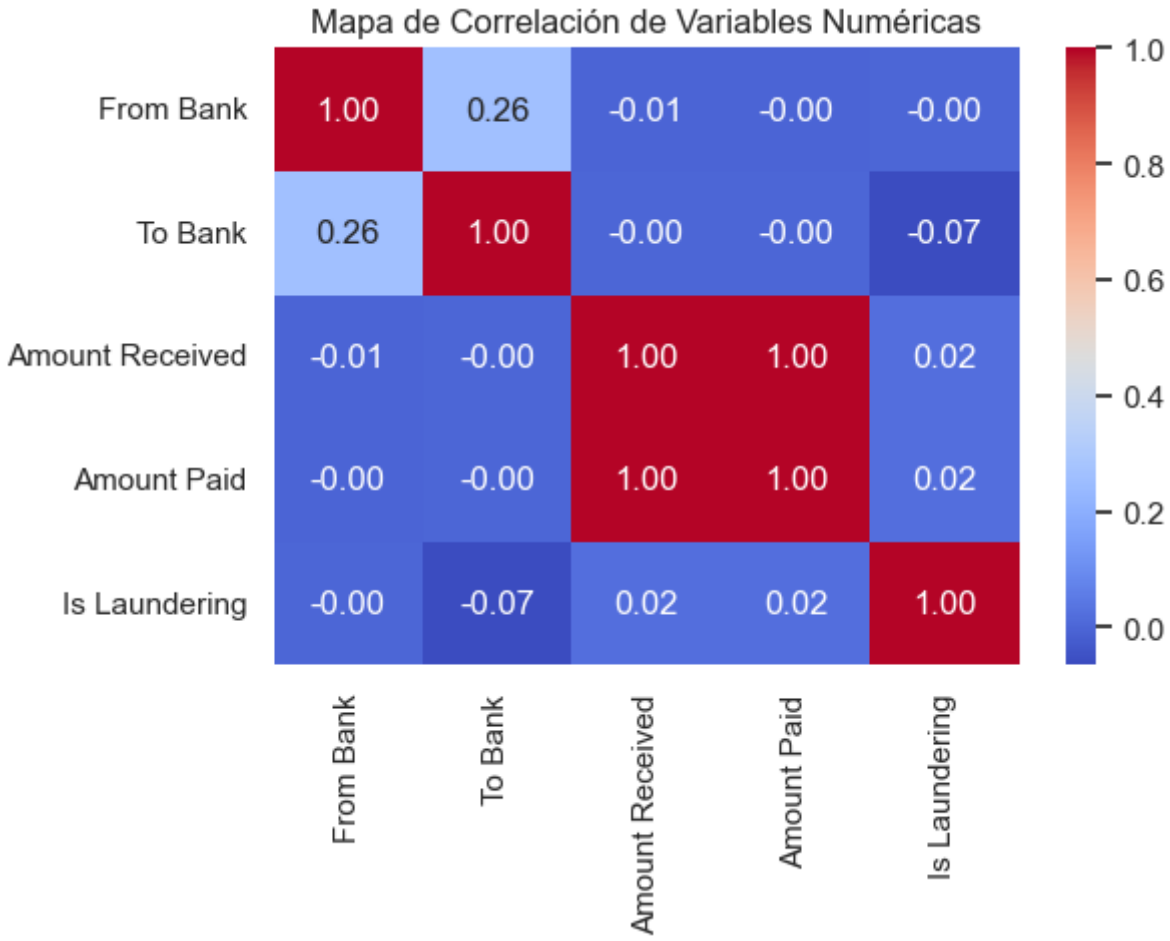


Figura 57. Mapa de correlación de variables numéricas del dataset AML.

Finalmente, en la **Figura 58** se presenta la distribución de transacciones por hora. Este análisis permite observar la concentración temporal de operaciones y puede aportar

información adicional para identificar comportamientos inusuales o periodos de mayor actividad transaccional dentro del conjunto de datos.

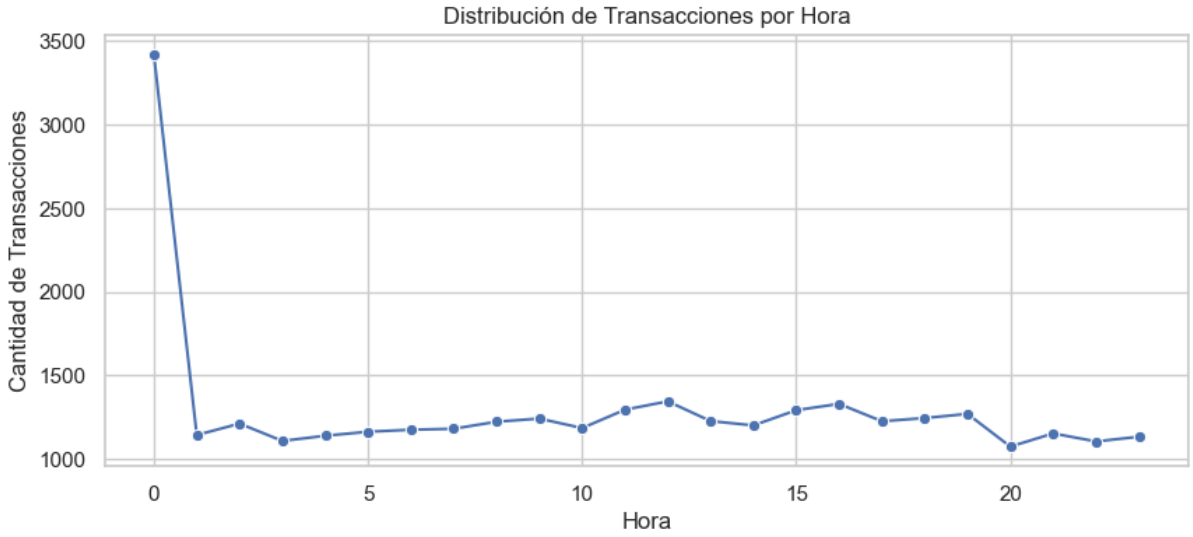


Figura 58. Distribución de transacciones por hora en el dataset AML.

4.9.5 Integración del módulo AML en el dashboard ejecutivo

El módulo AML fue integrado dentro de los dashboard ejecutivo del sistema, permitiendo cargar lotes de transacciones sin la columna objetivo y generar las predicciones automáticas mediante el pipeline entrenado. La interfaz presenta como en el módulo de tarjetas indicadores ejecutivos como total de registros analizados, cantidad de casos sospechosos, transacciones con un alto riesgo, transacciones de riesgo medio y de riesgo bajo del lote.

Como se observa en la **Figura 59**, el dashboard ejecutivo resume los principales indicadores de riesgo del lote analizado. En este caso, se procesaron 151.000 transacciones y se identificaron 101.155 casos sospechosos, lo cual se explica por el uso de un dataset de prueba construido con una proporción elevada de casos asociados a lavado de activos para validar la capacidad predictiva del sistema.

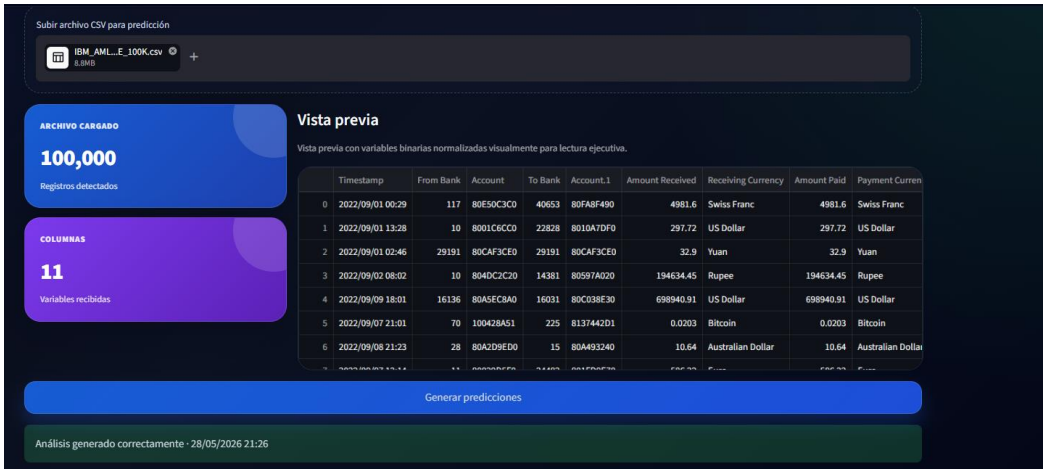
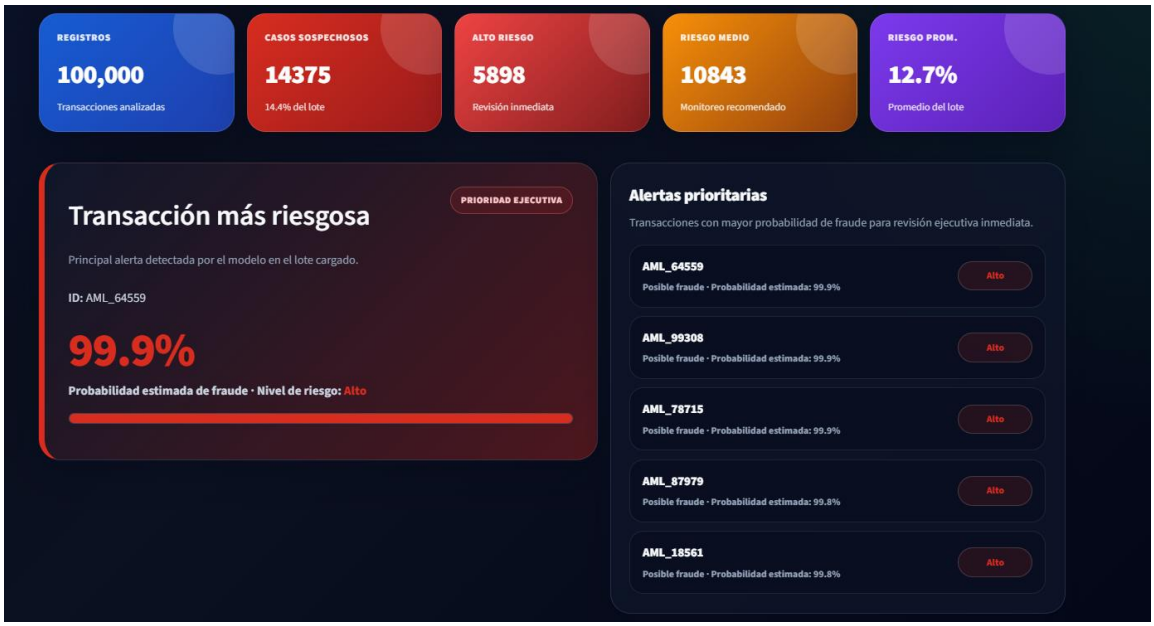
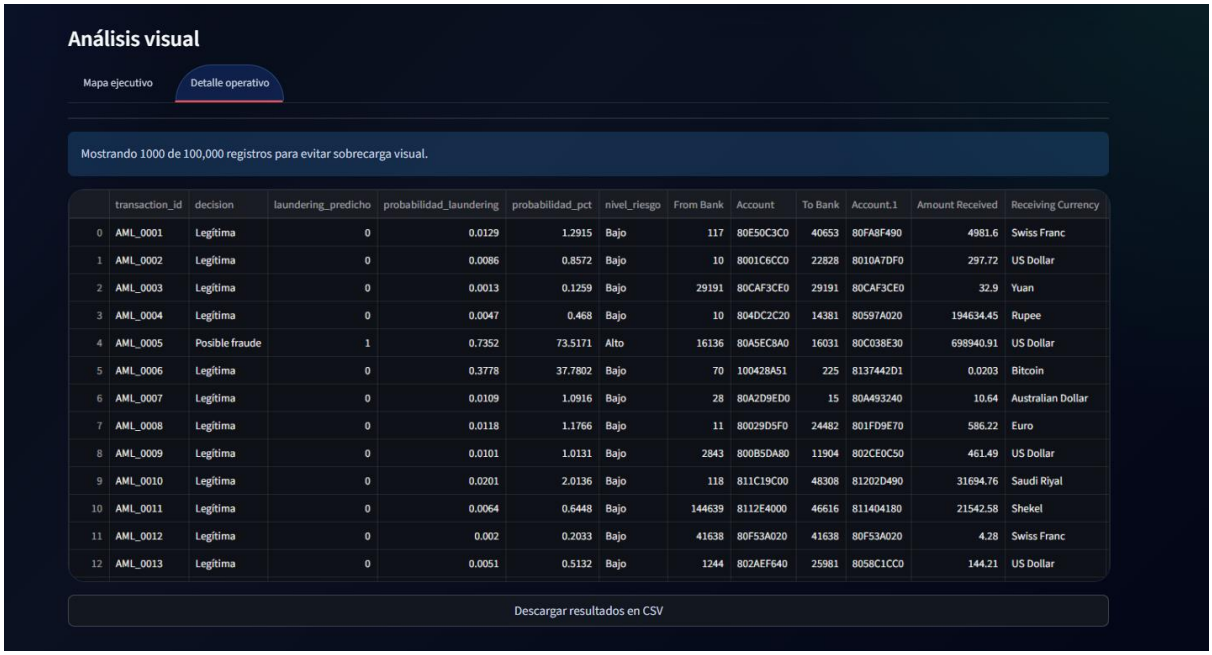


Figura 59. Indicadores ejecutivos del módulo AML en el dashboard del sistema.

Además, el sistema logra identificar la transacción con mayor probabilidad estimada de riesgo y muestra una lista de alertas prioritarias para la revisión similar al módulo de tarjetas. Como se aprecia en la **Figura 60**, la interfaz permite visualizar la transacción más riesgosa, su probabilidad estimada y las principales alertas generadas por el modelo. Con esta funcionalidad el modelo predictivo se convierte en una herramienta de apoyo a la toma de decisiones, ya que no solamente clasifica las transacciones, sino que también prioriza aquellas que requieren una atención urgente.





	transaction_id	decision	laundering_predicho	probabilidad_laundering	probabilidad_pct	nivel_riesgo	From Bank	Account	To Bank	Account.1	Amount Received	Receiving Currency
0	AML_0001	Legítima	0	0.0129	1.2915	Bajo	117	80E50C3C0	40653	80FA8F490	4981.6	Swiss Franc
1	AML_0002	Legítima	0	0.0086	0.8572	Bajo	10	8001C6CC0	22828	8010A7DF0	297.72	US Dollar
2	AML_0003	Legítima	0	0.0013	0.1259	Bajo	29191	80CAF3CE0	29191	80CAF3CE0	32.9	Yuan
3	AML_0004	Legítima	0	0.0047	0.468	Bajo	10	804DC2C20	14381	80597A020	194634.45	Rupee
4	AML_0005	Posible fraude	1	0.7352	73.5171	Alto	16136	80A5EC8A0	16031	80C038E30	698940.91	US Dollar
5	AML_0006	Legítima	0	0.3778	37.7802	Bajo	70	100428A51	225	8137442D1	0.0203	Bitcoin
6	AML_0007	Legítima	0	0.0109	1.0916	Bajo	28	80A2D9ED0	15	80A493240	10.64	Australian Dollar
7	AML_0008	Legítima	0	0.0118	1.1766	Bajo	11	80029D5F0	24482	801FD9E70	586.22	Euro
8	AML_0009	Legítima	0	0.0101	1.0131	Bajo	2843	800B5DA80	11904	802CE0C50	461.49	US Dollar
9	AML_0010	Legítima	0	0.0201	2.0136	Bajo	118	811C19C00	48308	81202D490	31694.76	Saudi Riyal
10	AML_0011	Legítima	0	0.0064	0.6448	Bajo	144639	8112E4000	46616	811404180	21542.58	Shekel
11	AML_0012	Legítima	0	0.002	0.2033	Bajo	41638	80F53A020	41638	80F53A020	4.28	Swiss Franc
12	AML_0013	Legítima	0	0.0051	0.5132	Bajo	1244	802AEF640	25981	8058C1CC0	144.21	US Dollar

Figura 60. Transacción más riesgosa y alertas prioritarias generadas por el módulo AML.

En general, este módulo complementario demuestra la flexibilidad del sistema propuesto para adaptarse a distintos problemas de riesgo financiero. Mientras que el módulo principal se enfoca en fraude con tarjetas, el módulo AML amplía el alcance del prototipo hacia la detección de posibles operaciones de lavado de activos, reutilizando la misma lógica general de carga de datos, preprocesamiento, predicción, visualización y la priorización ejecutiva.

Capítulo 5. Conclusiones y Recomendaciones

5.1 Conclusiones

El presente trabajo permitió desarrollar un sistema predictivo basado en aprendizaje automático para la detección de fraude en transacciones con tarjetas de crédito, aplicando la metodología CRISP-DM como guía principal del proceso. Con esta metodología se logró organizar el proyecto de una forma estructurada desde la comprensión de la problemática hasta el despliegue de un prototipo funcional, manteniendo una relación directa entre los objetivos del negocio y las decisiones técnicas tomadas durante el desarrollo.

Durante la preparación de los datos se identificaron valores nulos, variables binarias con distintos formatos, valores atípicos y el desbalance de las clases. Estos elementos justificaron la implementación de un pipeline automatizado capaz de limpiar, transformar, imputar y preparar los datos antes del entrenamiento de los modelos. De esta manera, el sistema puede trabajar con datos que simulan condiciones más cercanas a un entorno real.

La comparación entre Regresión Logística, Random Forest y XGBoost permitió determinar que Random Forest presentó el mejor desempeño global para la detección de fraude en transacciones con tarjeta de crédito, al alcanzar un AUC-ROC de 0,9910. XGBoost obtuvo un AUC-ROC de 0,9869, mientras que la Regresión Logística alcanzó un valor de 0,9630. Estos resultados evidencian que los modelos basados en ensamble ofrecen una mayor capacidad de discriminación entre transacciones legítimas y fraudulentas, siendo Random Forest el modelo más adecuado para el sistema principal.

En el módulo complementario de detección de lavado de activos, XGBoost presentó el mejor desempeño al alcanzar un AUC-ROC de 0,9672, seguido de Random Forest con 0,9597 y Regresión Logística con 0,9003. Esto demuestra que el enfoque propuesto también puede aplicarse a un segundo escenario de riesgo financiero, aunque con resultados y características distintas al problema principal de fraude con tarjetas de crédito.

El prototipo desarrollado mediante Streamlit permitió integrar el modelo predictivo en una interfaz funcional, facilitando la carga de archivos CSV, la ejecución del pipeline, la generación de predicciones y la visualización de resultados mediante tablas e indicadores.

Respecto a los objetivos específicos planteados, se logró analizar y preprocesar los conjuntos de datos seleccionados, identificando variables relevantes, valores atípicos e inconsistencias que fueron tratadas mediante un pipeline automatizado. Asimismo, se implementaron y compararon modelos supervisados de clasificación utilizando métricas como

precisión, recall, F1-score y AUC-ROC, lo que permitió seleccionar el algoritmo con mejor desempeño para cada escenario evaluado. En relación con los objetivos de implementación, se desarrolló un prototipo funcional en Streamlit que permite realizar predicciones individuales y por lotes, así como reentrenar y actualizar el modelo con nuevos datos. Finalmente, se evaluó el comportamiento del sistema durante la fase de despliegue mediante indicadores y comparaciones entre versiones, incorporando además un módulo complementario para analizar la aplicabilidad del aprendizaje automático en la detección de posibles operaciones de lavado de activos.

5.2 Recomendaciones

Se recomienda que, en futuras versiones del sistema, se utilicen datos financieros actualizados y provenientes de entornos reales, siempre que se garantice la anonimización y protección de la información sensible. Esto permitiría evaluar el comportamiento del modelo frente a patrones de fraude más recientes y mejorar su capacidad de generalización.

Se recomienda implementar mecanismos de monitoreo continuo para identificar fenómenos de Data Drift y Concept Drift, debido a que la distribución de los datos y los patrones asociados al fraude pueden cambiar con el tiempo, afectando el desempeño del modelo. Para este propósito, pueden emplearse herramientas como Evidently AI o MLflow, las cuales permiten supervisar métricas, comparar versiones del modelo y detectar variaciones significativas que justifiquen su reentrenamiento.

Se recomienda fortalecer los mecanismos de autenticación y control de acceso del prototipo mediante la adopción de estándares ampliamente utilizados en la industria, como OAuth 2.0 y JSON Web Tokens (JWT). Su implementación permitiría gestionar sesiones de manera más segura, controlar permisos de acceso y proteger la información procesada por el sistema en futuros entornos productivos.

Se recomienda ampliar el sistema para permitir la integración con bases de datos, APIs o servicios externos, en lugar de depender únicamente de archivos CSV. Esto facilitaría su adaptación a entornos empresariales donde la información transaccional se almacena en sistemas internos.

Finalmente, se sugiere explorar herramientas orientadas a la escalabilidad y mejora del rendimiento como Apache Spark o PySpark para el procesamiento de grandes volúmenes de datos, así como frameworks como PyTorch o TensorFlow para evaluar modelos de aprendizaje

profundo. Estas tecnologías podrían complementar el sistema desarrollado y permitir futuras comparaciones con modelos más avanzados en escenarios financieros complejos.

Referencias bibliográficas

Mitchell, T. M. (1997). Machine Learning. McGraw-Hill.

Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow. O'Reilly.

Narayanan R, D. (s. f.). *Credit Card Fraud*. Kaggle. Recuperado el 18 de mayo de 2026, de Kaggle.

Han, J., Kamber, M., & Pei, J. (2012). Data Mining: Concepts and Techniques. Morgan Kaufmann.

Provost, F., & Fawcett, T. (2013). Data Science for Business. O'Reilly.

IBM. (s. f.). *CRISP-DM en SPSS Modeler*. Documentación de IBM. https://www.ibm.com/docs/es/spss-modeler/19.0.0?topic=overview-crisp-dm-in-spss-modeler

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.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). CRISP-DM 1.0: Step-by-step data mining guide. SPSS.

Fawcett, T. (2006). An introduction to ROC analysis. Pattern Recognition Letters, 27(8), 861–874.

Delamaire, L., Abdou, H., & Pointon, J. (2009). Credit card fraud and detection techniques: A review. Banks and Bank Systems, 4(2), 57–68.

Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). Applied Logistic Regression. Wiley.

Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5–32.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

Treveil, M., & Omont, N. (2020). Introducing MLOps. O'Reilly Media.

Altman, E. (2023). *IBM transactions for anti money laundering (AML)* [Data set]. Kaggle. https://www.kaggle.com/datasets/ealtman2019/ibm-transactions-for-anti-money-laundering-aml