Universidad Siglo 21

Trabajo Final de Grado
Licenciatura en Ciencia de Datos
Prototipado Tecnológico

Detección de Fraudes en Pagos con Tarjetas de Crédito

Federici, Juan Maximiliano
Legajo VLDC000741
DNI 22.520.713

Tutor: Palazzesi, Victor Omar

Mendoza, noviembre de 2025

Indice

1 Introducción 6
2 Antecedentes 6
3 Descripción del Área Problemática 7
4 Justificación 9
5 Objetivos 10
5.1 Objetivo General 10
5.2 Objetivos Específicos 11
6 Marco Teórico 11
6.1 Dominio del Problema 11
6.1.1 Tipos de Fraude Electrónico 11
6.1.2 Modelos de Machine learning para la Detección de Fraude 12
6.1.3 Métricas de Evaluación 13
6.2 Técnicas para Datos Desbalanceados 14
6.3 Competencia 15
6.3.1 Soluciones Comerciales 15
6.3.2 Soluciones de Código Abierto 15
6.3.3 Justificación de la Propuesta 16
7 Diseño Metodológico 16
7.1 Metodología Analítica 16
7.2 Herramientas Metodológicas 18
7.3 Herramientas de Software 18
7.4 Recolección de Datos 19
7.5 Planificación del Proyecto 19
8 Relevamiento 20
8.1 Relevamiento Estructural 20
8.2 Relevamiento Funcional 21
9 Proceso de Negocio 22
10 Diagnóstico y Propuesta 24
10.1 Diagnóstico 24
10.2 Propuesta 25
11 Objetivos, Límites y Alcance del Prototipo 26
11.1 Objetivos del Prototipo 26
11.2 Límites del Prototipo 26

11.3 Alcance del Prototipo 26
12 Descripción del Proyecto 27
12.1 Product Backlog 27
12.2 Historias de Usuario 28
12.3 Sprint Backlog 30
12.4 Estructura de Datos 32
12.5 Prototipo de Interfaces de Pantalla 34
12.6 Diagrama de Arquitectura 36
13 Seguridad 38
13.1 Acceso a la Aplicación 38
13.2 Política de Respaldo de Información 38
13.3 Disponibilidad de la Información 39
14 Análisis de Costos 40
15 Análisis de Riesgos 42
16 Demo 45
16.1 Código Fuente del Prototipo 45
16.2 Salida que Arroja el Código 45
16.3 Descripción del Código 45
16.4 Documentación de Instalación 52
16.5 Alojamiento en la Nube 52
17 Conclusiones 52
18 Referencias Bibliográficas 55
A. Anexo 1 – Código del Prototipo en Python 58
B. Anexo 2 – Salida del Código 69

Agradecimientos

Quiero agradecer a quienes me acompañaron incondicionalmente a lo largo de toda la carrera: Chiqui, Luna y Bigote

Y especialmente a Milo, que aunque ya no estés físicamente, me seguís acompañando

Resumen

El presente trabajo final de grado tiene como propósito desarrollar un prototipo tecnológico basado en técnicas de *Machine learning* para la detección de fraudes en pagos con tarjetas de crédito. A partir del crecimiento exponencial de los medios electrónicos de pago, la detección temprana de operaciones anómalas se ha convertido en un desafío prioritario para el sector financiero.

Se utilizaron herramientas libres de ciencia de datos, como *Python*, *Pandas* y *Scikit-learn*, para analizar un conjunto de transacciones y entrenar modelos predictivos supervisados y no supervisados. El prototipo diseñado permite identificar patrones inusuales en el comportamiento de las operaciones, ofreciendo una base sólida para la prevención de fraudes y la toma de decisiones en tiempo real. Los resultados demuestran que la aplicación de técnicas analíticas en entornos financieros puede mejorar significativamente la detección de actividades fraudulentas y fortalecer la seguridad en los medios de pago digitales.

Palabras clave: fraude con tarjetas; detección de fraude; Machine learning; datos desbalanceados; optimización de modelo

Abstract

This final degree project aims to develop a technological prototype based on *Machine learning* techniques for detecting fraud in credit card payments. Due to the exponential growth of electronic payment systems, early detection of anomalous operations has become a major challenge for the financial sector.

Open-source data science tools such as *Python, Pandas*, and *Scikit-learn* were used to analyze transaction datasets and train both supervised and unsupervised predictive models. The designed prototype identifies unusual patterns in transaction behavior, providing a solid basis for fraud prevention and real-time decision-making. Results show that applying analytical techniques to financial environments can significantly enhance fraud detection and strengthen the security of digital payment systems

Keywords: credit card fraud; fraud detection; Machine learning; imbalanced data; model optimization.

Detección de Fraudes en Pagos con Tarjetas de Crédito

1 Introducción

Para el presente *Trabajo Final de Grado* se ha optado por el enfoque de *Prototipo Tecnológico*, y la temática de *Inteligencia de Negocios y Analítica de Datos*, ya que se buscar resolver un problema puntual mediante la implementación de herramientas de *Machine learning*.

En los últimos años el incremento de los medios electrónicos de pago ha tenido un crecimiento exponencial, sobre todo a partir de la pandemia del Covid, la cual obligó a muchas empresas a volcarse a comercializar sus productos o servicios a través de internet. Esto ha generado una gran cantidad de nuevos medios electrónicos de pago como las billeteras virtuales o las pasarelas de pago, las cuales permiten vincular rápidamente a los distintos actores económicos, haciendo que cualquier persona con una computadora, notebook o teléfono celular, pueda realizar sus pagos desde cualquier lugar y en cualquier momento.

Pero todos estos beneficios tienen una contracara, que es la proliferación de todo tipo de delitos electrónicos, pero principalmente en lo que se refiere al fraude en los pagos *on line*, es por lo que en el presente trabajo se pretende desarrollar el prototipo de un proceso de *Machine learning* para la detección de fraude en los pagos con tarjetas de crédito.

2 Antecedentes

Desde sus orígenes, las tarjetas de crédito han sufrido todo tipo de fraudes. Al principio era más difícil de llevar a cabo estas actividades, ya que se debía contar en algún momento con la tarjeta físicamente, pero a medida que el comercio electrónico fue avanzando, se fue complejizando la detección de este tipo de actividades ilícitas.

En sus inicios, la detección de fraude se basaba en reglas fijas y sistemas de expertos. Estos sistemas utilizaban un conjunto predefinido de reglas lógicas para identificar comportamientos sospechosos. Si bien estos métodos eran sencillos de implementar, tenían una limitación significativa: eran estáticos y no podían adaptarse a los nuevos patrones de fraude. Los criminales aprendían rápidamente a eludir estas reglas, y los falsos positivos (transacciones legítimas marcadas como fraudulentas) eran muy comunes, afectando la experiencia del usuario.

Estos problemas se solucionaron con la llegada de los algoritmos de *Machine Learning* (ML). A diferencia de las reglas fijas, los modelos de ML aprenden automáticamente de grandes volúmenes de datos históricos para identificar patrones complejos asociados con el fraude. Estos modelos no necesitan ser programados con reglas explícitas; en su lugar, analizan miles o millones de transacciones pasadas, tanto legítimas como fraudulentas, y construyen un "modelo" capaz de predecir si una nueva transacción es fraudulenta o no.

Actualmente, el campo se está moviendo hacia el *Aprendizaje Profundo* (*Deep Learning*), especialmente con el uso de *Redes Neuronales Recurrentes* (RNN) y *Redes Neuronales de Gráfico* (GNN). Las RNN son excelentes para analizar secuencias de datos, lo que es perfecto para detectar patrones en el historial de transacciones de una persona. Las GNN, por otro lado, pueden analizar la red de conexiones entre usuarios, comercios y transacciones, identificando grupos de usuarios fraudulentos. Estos modelos son extremadamente potentes, pero requieren una gran cantidad de datos y recursos computacionales para entrenarse.

3 Descripción del Área Problemática

El fraude electrónico asociado al uso de tarjetas de crédito se ha consolidado en los últimos años como uno de los principales desafíos para el sistema financiero. La masificación de los pagos digitales, el comercio electrónico y las transacciones *Card-Not-Present* (CNP) han generado nuevas

oportunidades para los delincuentes, que emplean técnicas cada vez más sofisticadas, como el *phishing*, el robo de identidades, el *skimming* y el uso de datos filtrados en la *dark web*.

Este problema afecta a varios actores:

- ✓ *Consumidores*: que enfrentan la pérdida temporal de su dinero y la exposición de sus datos personales.
- ✓ *Bancos y Entidades Emisoras*: que absorben gran parte de las pérdidas económicas y enfrentan costos crecientes de seguridad y reembolsos.
- ✓ *Comercios*: especialmente en el canal online, que deben afrontar contracargos y pérdida de confianza de los clientes.

La magnitud del problema puede observarse en las cifras reportadas por la industria. Según *The Nilson Report*[1], los montos anuales de pérdidas por fraude con tarjetas de crédito a nivel mundial se han mantenido en valores superiores a los 27 mil millones de dólares en los últimos años, con una tendencia creciente hasta 2023. Esto significa que, a pesar de la implementación de nuevas medidas de seguridad (como la autenticación reforzada en pagos electrónicos), el fraude continúa adaptándose y representa un costo económico y social significativo.

Pérdidas Globales Estimadas por Fraude con Tarjetas
(en miles de millones de dólares)

Año	***Pérdidas***
2018	27,85
2019	28,65
2020	28,58
2021	32,34
2022	33,45
2023	33,83

Fuente: The Nilson Report (2018–2023).

En términos de tendencia, se observa que las pérdidas globales crecieron aproximadamente un 21% en el período 2018–2023, impulsadas principalmente por el auge del fraude en transacciones CNP, donde la ausencia de presencia física favorece el uso de datos robados y técnicas de automatización; y la

[1] The Nilson Report. *Card Fraud Losses Worldwide in 2023*

concentración de casos en mercados de alto volumen como Estados Unidos, que representa alrededor del 40% de las pérdidas mundiales.

La detección y medición del fraude con tarjetas de crédito es un desafío. A menudo, el fraude no se detecta de inmediato, lo que permite a los estafadores realizar múltiples transacciones antes de ser detectados. Para medir el problema, se utilizan métricas como la tasa de fraude, que representa el porcentaje de transacciones fraudulentas sobre el total de transacciones procesadas. Un desafío clave es la naturaleza desequilibrada de los datos, ya que las transacciones fraudulentas representan un porcentaje minúsculo (típicamente menos del 1%) del total de transacciones. Esto dificulta el desarrollo de modelos predictivos efectivos, además de que los ciberdelincuentes van incorporando las nuevas tecnologías disponibles para llevar a cabo sus actividades ilegales.

4 Justificación

El presente trabajo se centra en la creación de un prototipo para la detección de fraude en pagos con tarjetas de crédito mediante técnicas de ciencia de datos y aprendizaje automático. Esta propuesta responde a una necesidad crítica y creciente en el panorama financiero digital actual, donde el volumen de transacciones electrónicas continúa en constante aumento. La lucha contra el fraude no solo es un imperativo para proteger a los consumidores, sino también para salvaguardar la confianza en el sistema bancario y de pagos.

El fraude electrónico asociado al uso de tarjetas de crédito constituye una de las principales problemáticas en el ámbito financiero global. Se estima que, a nivel mundial, las pérdidas por fraude con tarjetas superaron los 33 mil millones de dólares en 2023 y continúan en aumento conforme se expanden los pagos digitales (Nilson Report, 2023).

En este contexto, resulta necesario desarrollar herramientas analíticas basadas en ciencia de datos que permitan detectar de manera temprana y precisa patrones de comportamiento anómalos que puedan estar vinculados con

operaciones fraudulentas. Los enfoques tradicionales, como las reglas estáticas predefinidas, suelen ser insuficientes para adaptarse a la evolución de las técnicas utilizadas por los ciberdelincuentes (Bhattacharyya et al., 2011). Por ello, el uso de algoritmos de *machine learning*, capaces de aprender de grandes volúmenes de datos históricos y de identificar dinámicamente anomalías en transacciones financieras, representa una solución innovadora y altamente pertinente.

El valor esperado del proyecto radica en demostrar la aplicabilidad de técnicas avanzadas de análisis predictivo en un problema real y crítico. Mediante la construcción de un prototipo funcional, se busca evidenciar cómo los algoritmos de aprendizaje automático, aplicados a un dataset realista disponible en plataformas abiertas, pueden contribuir a reducir riesgos operativos y económicos.

Finalmente, la viabilidad del proyecto está garantizada por la disponibilidad de datasets públicos de calidad, como los que se encuentran en *Kaggle*[2], y por el acceso a herramientas libres y robustas de procesamiento de datos y modelado predictivo (por ejemplo, Python, Pandas, Scikit-learn). Esto asegura que los resultados obtenidos sean reproducibles y accesibles, reforzando el valor académico del trabajo y su aplicabilidad en distintos escenarios.

5 Objetivos

5.1 Objetivo General

Desarrollar un prototipo basado en técnicas de *machine learning* que permita detectar transacciones potencialmente fraudulentas en pagos con tarjetas de crédito, con el propósito de fortalecer la seguridad en los medios de pago electrónico y minimizar las pérdidas económicas asociadas.

[2] https://www.kaggle.com/

5.2 Objetivos Específicos

- Analizar y preparar el dataset de transacciones de tarjetas de crédito mediante la detección de valores atípicos, tratamiento de datos faltantes y generación de variables relevantes que permitan una mejor representación del problema.
- Entrenar y evaluar diferentes modelos de *machine learning* (supervisados y no supervisados) para la detección de fraude, comparando su rendimiento con métricas como precisión, recall, F1-score y AUC.
- Diseñar un prototipo funcional que integre el modelo seleccionado y permita simular la detección de operaciones fraudulentas, mostrando su utilidad práctica y potencial de implementación en escenarios reales.

6 Marco Teórico

6.1 Dominio del Problema

La detección de fraude electrónico en transacciones con tarjetas de crédito es un problema crítico para las instituciones financieras y comercios. Se busca identificar transacciones fraudulentas en tiempo real o casi real, diferenciándolas de las legítimas, para minimizar las pérdidas económicas. Este es un desafío complejo debido a la naturaleza volátil y evolutiva de los patrones de fraude.

6.1.1 Tipos de Fraude Electrónico

El fraude con tarjetas de crédito puede manifestarse de diversas maneras. A continuación, se describen tres de los tipos más comunes:

- *Fraude de Apropiación de Cuenta (Account Takeover - ATO)*: En este tipo de fraude, los estafadores obtienen acceso a la información de una cuenta legítima para realizar transacciones no autorizadas. Esto puede ocurrir a través de phishing, malware o la compra de datos robados en la dark web. Los delincuentes cambian la dirección de envío, el número de teléfono o las contraseñas, lo que les permite realizar compras a gran escala.

- *Fraude de Tarjeta No Presente (Card-Not-Present - CNP)*: Este es uno de los tipos de fraude más comunes en el comercio electrónico, donde la transacción se realiza sin la presencia física de la tarjeta. Los estafadores utilizan números de tarjeta robados para comprar bienes o servicios en línea, por teléfono o correo. La validación del Código de Verificación de la Tarjeta (CVV) y las direcciones de facturación son mecanismos de seguridad que intentan mitigar este tipo de fraude, aunque no son infalibles.
- *Fraude de Tarjeta Perdida o Robada*: Ocurre cuando un delincuente roba una tarjeta física y la utiliza para hacer compras. Aunque los chips de seguridad (EMV) y los PIN han reducido este tipo de fraude, sigue siendo un riesgo. Las transacciones de menor valor o las que no requieren PIN son particularmente vulnerables.

6.1.2 Modelos de Machine learning para la Detección de Fraude

Para abordar este problema, se utilizan algoritmos de *aprendizaje supervisado* y *no supervisado*. La elección del modelo depende del tipo de datos disponibles y del enfoque que se quiera dar a la detección[3].

- *Aprendizaje Supervisado*: utiliza datos previamente etiquetados, donde cada transacción está marcada como "legítima" o "fraudulenta". El algoritmo aprende a reconocer los patrones que distinguen a una clase de la otra. Ejemplos de modelos supervisados para esta tarea incluyen regresión logística, árboles de decisión, *Random Forest*, *Gradient Boosting* y redes neuronales. La principal ventaja es la precisión en la clasificación, pero su rendimiento depende de la calidad y la cantidad de datos etiquetados.
- *Aprendizaje No Supervisado*: se aplica a datos sin etiquetas. El objetivo es que el modelo descubra estructuras o patrones anómalos en los datos por sí mismo. En el contexto de la detección de fraude, se busca identificar transacciones que se desvían significativamente del comportamiento habitual del titular de la tarjeta. Ejemplos de algoritmos no supervisados son *Isolation Forest* y *Local Outlier Factor* (LOF). Son especialmente útiles para

[3] Google AI. (2025). Gemini (versión 11 de setiembre). [Modelo de lenguaje de gran tamaño], https://gemini.google.com

detectar nuevos tipos de fraude, ya que no requieren ejemplos previos para identificar patrones anómalos.

6.1.3 Métricas de Evaluación

La evaluación del rendimiento de un modelo de detección de fraude es crucial. Las métricas de precisión y exactitud pueden ser engañosas en un contexto de datos desbalanceados, donde las transacciones legítimas superan en gran medida a las fraudulentas. Por eso, se emplean métricas específicas que ofrecen una visión más clara del rendimiento del modelo.

- *Matriz de Confusión*: Es una tabla que resume el rendimiento de un modelo de clasificación, mostrando el número de predicciones correctas e incorrectas. Se compone de cuatro valores:
  - *Verdaderos Positivos (VP)*: transacciones fraudulentas correctamente identificadas.
  - *Verdaderos Negativos (VN)*: transacciones legítimas correctamente identificadas.
  - *Falsos Positivos (FP)*: transacciones legítimas erróneamente clasificadas como fraudulentas (también conocido como error de tipo I).
  - *Falsos Negativos (FN)*: transacciones fraudulentas erróneamente clasificadas como legítimas (también conocido como error de tipo II).
- *Precisión (Precision) y Sensibilidad (Recall)*: Estas métricas son más informativas que la exactitud.
  - *Precisión*: se enfoca en los resultados positivos y responde a la pregunta: "De todas las transacciones que mi modelo clasificó como fraudulentas, ¿cuántas lo fueron realmente?". Se calcula como VP/(VP+FP). Una alta precisión es importante para minimizar las falsas alarmas que pueden molestar a los clientes.
  - *Sensibilidad (Recall)*: responde a la pregunta: "De todas las transacciones fraudulentas que existían, ¿cuántas encontró mi modelo?". Se calcula como VP/(VP+FN). Una alta exhaustividad es vital para detectar el mayor número de fraudes posible.

- *Área bajo la Curva ROC (AUC-ROC)*: La curva ROC (Receiver Operating Characteristic) es una gráfica que muestra el rendimiento de un modelo de clasificación en todos los umbrales. El área bajo esta curva (AUC) proporciona una medida del rendimiento general del modelo. Un valor de 1.0 indica un clasificador perfecto, mientras que 0.5 indica un clasificador aleatorio. El AUC-ROC es particularmente útil en problemas de clases desbalanceadas, ya que no se ve afectado por la distribución de las clases.

6.2 Técnicas para Datos Desbalanceados

El problema de la detección de fraude se caracteriza por tener un conjunto de datos altamente desbalanceado, donde la cantidad de transacciones legítimas supera ampliamente a las fraudulentas. Entrenar un modelo con datos tan sesgados puede llevar a que el modelo simplemente aprenda a predecir la clase mayoritaria (legítima), lo que resulta en un mal rendimiento al detectar la clase minoritaria (fraudulenta).

Para abordar este desafío, se utilizan diversas técnicas de muestreo:

- *Muestreo por Sobremuestreo (Oversampling)*: consiste en aumentar la cantidad de ejemplos de la clase minoritaria para equilibrar el conjunto de datos. El método *SMOTE* (Synthetic Minority Oversampling Technique) es muy popular. Crea nuevos ejemplos sintéticos de la clase minoritaria que no son copias exactas de los existentes, sino que se generan a partir de los vecinos más cercanos.
- *Muestreo por Submuestreo (Undersampling)*: se basa en reducir el número de ejemplos de la clase mayoritaria. Esto puede ser aleatorio o basado en algún criterio (por ejemplo, eliminando ejemplos redundantes o que están cerca del límite de decisión). Si bien esta técnica puede ser útil, existe el riesgo de perder información valiosa de la clase mayoritaria.
- *Combinaciones de Sobremuestreo y Submuestreo*: algunas estrategias combinan ambos enfoques. Por ejemplo, se puede realizar un submuestreo de la clase mayoritaria y un sobremuestreo de la minoritaria para obtener un conjunto de datos más equilibrado.

6.3 Competencia

El mercado de soluciones para la detección de fraude electrónico es vasto y dinámico, impulsado por el constante avance de las técnicas de ciberdelincuencia. La competencia se puede dividir en dos grandes categorías: soluciones comerciales (o propietarias) y soluciones de código abierto. Ambas aprovechan la ciencia de datos y el aprendizaje automático para identificar patrones anómalos en las transacciones. El análisis de estas soluciones es crucial para entender el estado del arte y posicionar la propuesta de este proyecto.

6.3.1 Soluciones Comerciales

Las soluciones comerciales, ofrecidas por empresas especializadas, suelen ser plataformas robustas, escalables y con un soporte técnico completo. Un ejemplo destacado es *ThreatMetrix*, ahora parte de LexisNexis Risk Solutions (LexisNexis Risk Solutions, 2024). Esta plataforma utiliza una combinación de inteligencia de dispositivos, datos de identidad digital y análisis de comportamiento para crear perfiles de riesgo en tiempo real. Otro líder en el mercado es la *FICO Falcon Platform*, que ha sido un pilar en la industria por décadas (FICO, 2024). Su enfoque se basa en modelos de comportamiento predictivos que aprenden de billones de transacciones para detectar anomalías con una alta precisión. Estas plataformas suelen integrar múltiples fuentes de datos, como geolocalización, información de la red y datos biométricos, para enriquecer el análisis. A pesar de su robustez, estas soluciones presentan un alto costo de licenciamiento y menor flexibilidad para la personalización.

6.3.2 Soluciones de Código Abierto

Las soluciones de código abierto ofrecen una alternativa flexible y a menudo más económica. Se basan en librerías y frameworks de ciencia de datos que permiten a los desarrolladores construir sus propios sistemas de detección de fraude. El ecosistema *Python* es particularmente relevante en este ámbito. Librerías como *Scikit-learn* (Pedregosa et al., 2011) y *TensorFlow* (Abadi et al., 2016) son fundamentales para implementar algoritmos de clasificación y

detección de anomalías. Por ejemplo, los modelos basados en árboles de decisión como Random Forest o algoritmos más complejos como las Redes Neuronales Profundas son ampliamente utilizados. Estas herramientas brindan un control total sobre el modelo, permitiendo una adaptación precisa a las necesidades específicas de la institución financiera.

6.3.3 Justificación de la Propuesta

El análisis de la competencia revela que las soluciones comerciales, si bien son efectivas, presentan limitaciones en cuanto a su costo y flexibilidad. Este proyecto busca abordar esta brecha al desarrollar un prototipo que resalta la personalización y optimización del modelo. Al utilizar técnicas de ciencia de datos y bibliotecas de código abierto, se demuestra la viabilidad técnica y económica de construir una solución a medida. El valor añadido de la presente propuesta reside en la implementación práctica y optimización de un modelo analítico de código abierto para un conjunto de datos específico. Este enfoque pretende alcanzar un rendimiento comparable al de las soluciones comerciales, pero con la ventaja de poder adaptar y perfeccionar el modelo según las necesidades puntuales del entorno, justificando así la originalidad y necesidad de la propuesta.

7 Diseño Metodológico

7.1 Metodología Analítica

Para abordar el proyecto de detección de fraude, se adoptará la metodología *CRISP-DM* (Cross-Industry Standard Process for Data Mining). Este marco de trabajo, reconocido por su enfoque estructurado e iterativo, es ideal para proyectos de ciencia de datos, ya que proporciona una guía clara desde la conceptualización hasta la implementación. Las fases se aplicarán de la siguiente manera[4]:

[4] Google AI. (2025). Gemini (versión 13 de setiembre). [Modelo de lenguaje de gran tamaño], https://gemini.google.com

1. *Comprensión del negocio*: el objetivo principal es desarrollar un modelo predictivo capaz de identificar transacciones fraudulentas con tarjetas de crédito, lo cual permitirá mitigar pérdidas financieras y mejorar la seguridad. Se definirá el problema de manera precisa y se establecerán los criterios de éxito del modelo, como la precisión, el recall y la puntuación F1, que son métricas clave en problemas de detección de desequilibrio de clases (Ferreira, 2017).
2. *Comprensión de los datos*: se analizará el dataset de transacciones de tarjetas de crédito obtenido de *Kaggle*. Esta fase incluirá un análisis exploratorio de datos (EDA), donde se examinarán las características de las variables, se detectarán valores atípicos y se estudiará el desequilibrio de clases, un aspecto crítico en la detección de fraude (Chen et al., 2018).
3. *Preparación de los datos*: se aplicarán técnicas de preprocesamiento de datos, como la limpieza, la normalización, la codificación de variables categóricas y el manejo de valores faltantes. Para abordar el desequilibrio de clases, se evaluará el uso de técnicas de submuestreo (undersampling) y sobremuestreo (oversampling), como SMOTE (Synthetic Minority Over-sampling Technique), que generan datos sintéticos para la clase minoritaria (Chawla et al., 2002).
4. *Modelado*: se entrenarán y evaluarán varios algoritmos de *machine learning* para la clasificación binaria. Entre los candidatos se encuentran modelos como Regresión Logística, Árboles de Decisión, Bosque Aleatorio (Random Forest) y XGBoost. Se utilizará la validación cruzada para asegurar la robustez de los modelos y se ajustarán los hiperparámetros para optimizar su rendimiento.
5. *Evaluación*: los modelos se evaluarán en un conjunto de datos de prueba independiente. Las métricas clave incluirán la matriz de confusión, la curva ROC (Receiver Operating Characteristic) y la puntuación F1, ya que proporcionan una visión más completa del rendimiento del modelo en un problema con clases desequilibradas (Srivastava & Srivastava, 2018).
6. *Implementación*: aunque el prototipo no se implementará en un entorno de producción real, esta fase consistirá en la presentación del modelo final, su

rendimiento y las conclusiones del estudio. El prototipo desarrollado servirá como una prueba de concepto del potencial de los modelos de *machine learning* para la detección de fraude.

7.2 Herramientas Metodológicas

Para gestionar el proyecto de manera ágil y adaptativa, se utilizará el marco de trabajo *Scrum*. Este enfoque se ajusta perfectamente a la naturaleza exploratoria de los proyectos de ciencia de datos, donde los hallazgos durante el análisis de datos pueden llevar a la redefinición de tareas.

El proyecto se dividirá en *sprints* de aproximadamente dos semanas. Al inicio de cada *sprint*, se planificarán las tareas a realizar, como la preparación del dataset, la exploración de datos, el entrenamiento de modelos específicos o la evaluación de métricas. Al final de cada *sprint*, se realizará una revisión (Sprint Review) para evaluar los resultados obtenidos y una retrospectiva (Sprint Retrospective) para identificar oportunidades de mejora en el proceso (Rubin, 2012). Este enfoque iterativo y de mejora continua permitirá ajustar el trabajo del proyecto de manera flexible.

7.3 Herramientas de Software

Se utilizarán las siguientes herramientas:

- *Lenguaje de programación*: se empleará *Python* como lenguaje principal. Su versatilidad, la amplia gama de bibliotecas disponibles y su popularidad en el campo de la ciencia de datos lo convierten en la opción ideal.
- *Entorno de desarrollo***:** se trabajará con *Visual Studio Code*.
- *Bibliotecas de Python:*
    - *Pandas*: para la manipulación y análisis de datos.
    - *NumPy*: para operaciones numéricas eficientes.
    - *Scikit-learn*: para la implementación de los algoritmos de *machine learning,* el preprocesamiento de datos y la evaluación de modelos.
    - *Matplotlib* y *Seaborn*: para la creación de visualizaciones estadísticas y la exploración de datos.

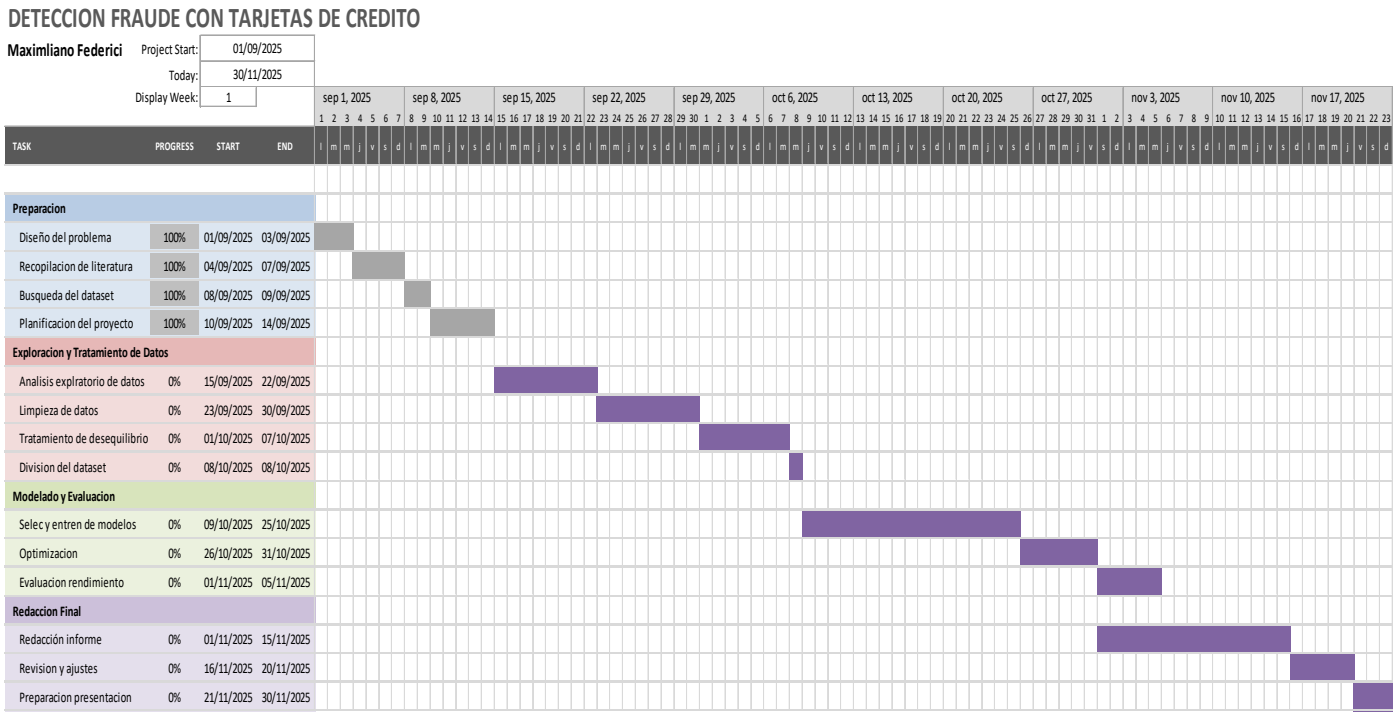
- *Control de versiones*: Se utilizará *Git* y se alojará el repositorio en *GitHub* para el control de versiones del código, lo cual facilita el seguimiento de los cambios y la colaboración.

7.4 Recolección de Datos

El principal insumo de datos para el modelo será un conjunto de datos público de transacciones de tarjetas de crédito de la plataforma *Kaggle*. Este dataset (*Credit Card Fraud Detection*) ya ha sido anonimizado y preprocesado en cierta medida para proteger la privacidad. Se consultarán las descripciones y la documentación del dataset para comprender la estructura de las variables, el tipo de datos y la distribución de las clases.

7.5 Planificación del Proyecto

La planificación del proyecto se ha estructurado para guiar el desarrollo de principio a fin, dividiendo el trabajo en fases bien definidas y fechas estimadas. La gestión del tiempo se visualizará a través de un *Diagrama de Gantt*, que detallará las tareas, su duración y las dependencias entre ellas.



8 Relevamiento

8.1 Relevamiento Estructural

Al no contar con una empresa real con la cual trabajar, el desarrollo del prototipo se desarrollará sobre el dataset "*Credit Card Fraud Detection*" disponible en *Kaggle*, el cual contiene transacciones simuladas y etiquetadas como *fraudulentas* o *legítimas*. En consecuencia, lo que se realizará es una descripción hipotética, inspirada en entornos propios de empresas financieras y *fintechs* que implementan soluciones de análisis de datos y prevención de fraude.

Se asume que el procesamiento de la información se realizará en un entorno computacional estándar, con recursos de hardware y software accesibles para el ámbito académico, como equipos de procesamiento de datos con una capacidad estándar, y librerías de análisis estadístico y de *machine learning* ampliamente utilizadas, entre ellas *Pandas*, *Numpy* y *Scikit-learn* en el lenguaje *Python*.

Componentes Estructurales de Datos

Componentes	***Descripción del Contexto***	***Implicaciones para el Diseño del Prototipo***
Fuente de Datos	Se dispone de un *dataset* representativo de transacciones con tarjeta de crédito, etiquetadas como "normales" o "fraudulentas".	El prototipo se centrará en el preprocesamiento y modelado del conjunto de datos específico de Kaggle, sin la necesidad de integrar múltiples fuentes.
Infraestructura de Datos	Se asume un entorno de desarrollo local (ej. IDE en laptop/computadora personal) para la carga y el procesamiento del *dataset* (modelado). No se requiere una arquitectura de *data warehouse* o *data lake* compleja.	La capacidad computacional se limitará a la disponible en el entorno de desarrollo. Se seleccionarán algoritmos eficientes que no requieran *clusters* de procesamiento a gran escala.

Volumen y Tipo de Datos	El *dataset* es de tamaño fijo (acotado), compuesto principalmente por variables numéricas desidentificadas. Se asume un problema de desbalance de clases (más transacciones normales que fraudulentas).	Será crucial aplicar técnicas para manejar el desbalance de clases (ej., *oversampling*, *undersampling* o ajuste de umbrales) y realizar una ingeniería de características a partir de las variables disponibles (m*onto*, h*ora, etc*).
Dispositivos/Conectividad	Se asume conectividad estándar a internet solo para la descarga inicial del *dataset* y de librerías (*Python*, *Scikit-learn*, *Pandas*, etc.). El resto del procesamiento será local.	No se considerarán restricciones de latencia de red o infraestructura de transmisión para el diseño del prototipo, ya que no operará en tiempo real en un entorno productivo.
Entorno de Uso	El prototipo operará en un entorno de prueba y desarrollo, cuyo objetivo es la validación del algoritmo de *machine learning* y la obtención de métricas de rendimiento (precisión, *recall*, F1-score).	El foco estará en la precisión predictiva y la robustez del modelo, en lugar de la integración con sistemas heredados o la interfaz de usuario final.

Fuente: Google AI. (2025). Gemini

El prototipo será pensado para ser utilizado por usuarios con formación en análisis de datos o en áreas vinculadas al riesgo financiero (complience), lo que implica que los resultados obtenidos deben ser interpretables y estar orientados a la toma de decisiones. En este sentido, el prototipo no se limitará a ejecutar modelos predictivos, sino que busca establecer un marco de referencia sobre cómo, en una organización real, podrían integrarse estos resultados a los procesos de gestión de fraude.

8.2 Relevamiento Funcional

Al tratarse de una organización ficticia, no es posible describir un organigrama jerárquico real ni áreas formalmente constituidas, pero sí resulta pertinente caracterizar los roles y funciones que estarían involucrados en la gestión del sistema.

Roles Funcionales Claves

Rol Funcional	***Responsabilidad Central***	***Interacción con el Prototipo***
Analista de Fraude	Monitorear transacciones, investigar alertas de fraude y tomar decisiones de bloqueo o verificación.	Consumidor principal de la salida del modelo de ML (alerta de fraude o puntaje de riesgo). Proporciona *feedback* para el reentrenamiento del modelo.
Data Scientist	Desarrollar, implementar y monitorear modelos analíticos y de ML.	Responsable del entrenamiento y despliegue del prototipo. Monitorea métricas de rendimiento y propone mejoras.
Gerente de Riesgos	Definir políticas de fraude, gestionar el riesgo financiero y supervisar la efectividad de las soluciones de prevención.	Utiliza los reportes agregados y las métricas de negocio generadas por el modelo para evaluar la exposición al riesgo.

Fuente: Rayo Mondragon, C. A. (2020). *Prototipo de detección de fraudes con tarjetas de crédito basado en inteligencia artificial aplicado a un banco peruano*.

De esta manera, se describe un flujo de trabajo en el cual los datos son generados y validados, posteriormente analizados por algoritmos de *machine learning* y, finalmente, convertidos en insumos para la toma de decisiones. Aunque no se trata de una organización real, esta descripción permite comprender cómo los roles y procesos interactúan en un escenario posible de prevención de fraude en transacciones electrónicas.

9 Proceso de Negocio

De acuerdo con Dumas et al. (2013), un proceso de negocios puede entenderse como una secuencia estructurada y medible de actividades que buscan alcanzar un resultado concreto, aportando valor dentro de un contexto organizacional. En este trabajo, el proceso de negocios se encuentra estrechamente vinculado con el objetivo del proyecto: la detección de fraude electrónico en operaciones con tarjetas de crédito mediante técnicas de aprendizaje automático.

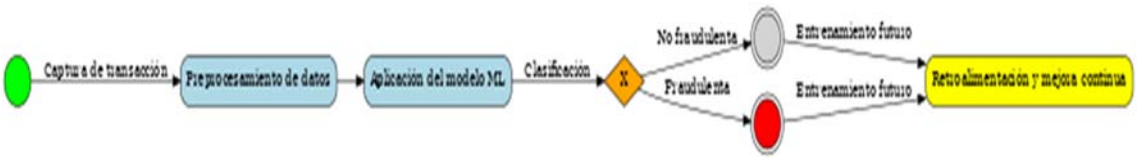
Si bien el proyecto no se enmarca en una empresa específica, es posible modelar un proceso genérico que refleje cómo se gestionaría la detección de fraude en un entorno transaccional digital. Este proceso se desarrolla a partir de las prácticas habituales en el sector financiero o *fintechs*, donde la detección temprana y automatizada de anomalías es fundamental para reducir pérdidas y mitigar riesgos.

El flujo central del proceso de negocio propuesto comprende las siguientes etapas[5]:

1. *Captura de datos transaccionales*: cada operación con tarjeta de crédito genera un registro con atributos tales como monto, hora, ubicación, comercio y datos del titular. En este caso, la información será provista por el dataset obtenido en la plataforma *Kaggle*.
2. *Preprocesamiento y preparación de datos*: las transacciones son depuradas y transformadas para que puedan ser analizadas por el modelo. Esto incluye la normalización de valores, la detección de outliers y la generación de variables derivadas relevantes para la clasificación (Han, Kamber & Pei, 2012).
3. *Aplicación del modelo de machine learning*: el prototipo implementa un algoritmo supervisado o no supervisado de detección de fraude. Este modelo analiza patrones de comportamiento y asigna una probabilidad de fraude a cada transacción.
4. *Clasificación de transacciones*: el sistema distingue entre operaciones legítimas y potencialmente fraudulentas. Las transacciones de alto riesgo se marcan para su revisión, lo que permitiría en un entorno real la intervención manual o el bloqueo automático.
5. *Retroalimentación y mejora continua*: los resultados obtenidos permiten refinar el modelo a través del aprendizaje iterativo, mejorando su precisión y reduciendo falsos positivos.

[5] OpenAI. (2025). ChatGPT (versión 30 de setiembre). [Modelo de lenguaje amplio], https://chat.openai.com/chat

Este proceso puede representarse mediante un diagrama BPMN, donde los eventos de inicio corresponden a la captura de una transacción y los eventos de fin corresponden a la clasificación de la operación como “legítima” o “fraudulenta”. Entre ambos puntos se articulan las tareas de preprocesamiento, modelado y decisión automatizada.



Fuente: elaboración propia

El proceso de negocio propuesto no busca replicar el funcionamiento total de una entidad financiera, sino simular de manera acotada la secuencia operativa esencial de un sistema de detección de fraude. De esta forma, se garantiza la coherencia con el alcance del proyecto y se sientan las bases para la implementación del prototipo.

10 Diagnóstico y Propuesta

10.1 Diagnóstico

El fraude electrónico asociado al uso de tarjetas de crédito representa un desafío creciente para los sistemas financieros y los medios de pago digitales, debido al aumento de las operaciones en línea y a la sofisticación de las técnicas de los defraudadores (Ngai et al., 2011). Entre las principales problemáticas identificadas se encuentran:

- *La dificultad para distinguir transacciones legítimas de fraudulentas*, dado que ambas pueden compartir características similares.
- *El desequilibrio en la proporción de fraudes frente al total de operaciones*, lo cual genera limitaciones para los mecanismos de detección tradicionales.
- *La adaptabilidad de los atacantes*, que ajustan constantemente sus patrones para evadir sistemas de control estáticos.

Estas situaciones evidencian una brecha entre la necesidad de protección de los usuarios y la capacidad de los sistemas actuales para anticiparse a los intentos de fraude. En consecuencia, se requiere un enfoque flexible y basado en datos que permita identificar anomalías y patrones ocultos de comportamiento.

10.2 Propuesta

La propuesta consiste en el diseño de un prototipo de detección de fraude mediante *machine learning* que aporte una solución efectiva al problema planteado. Este prototipo no busca reemplazar los sistemas existentes, sino demostrar cómo un modelo predictivo avanzado puede complementar y fortalecer los mecanismos de seguridad.

La solución estará orientada a los siguientes objetivos concretos:

- *Incrementar la capacidad de identificación temprana de fraudes*: el prototipo permitirá reconocer transacciones sospechosas antes de que se concreten pérdidas económicas.
- *Reducir la cantidad de falsos positivos*: a diferencia de los sistemas basados en reglas rígidas, el modelo podrá distinguir mejor entre operaciones legítimas poco comunes y operaciones efectivamente fraudulentas (Bhattacharyya et al., 2011).
- *Adaptarse a patrones cambiantes*: gracias al aprendizaje a partir de los datos, el sistema podrá ajustarse a nuevas estrategias de fraude sin requerir una redefinición manual de reglas.
- *Ofrecer una herramienta replicable y escalable*: aunque se trate de un prototipo académico, la lógica puede aplicarse en escenarios reales de entidades financieras y billeteras virtuales.

En términos prácticos, el prototipo funcionará como una herramienta de alerta inteligente. Cada transacción será evaluada y se asignará un nivel de riesgo; aquellas con mayor probabilidad de fraude serán marcadas para revisión inmediata. De este modo, se brinda un soporte a la toma de decisiones que mejora la eficiencia de los controles humanos y reduce la exposición al riesgo.

El valor agregado de esta propuesta radica en que transforma el problema del fraude en una oportunidad de innovación en la seguridad financiera, mostrando cómo las técnicas de ciencia de datos pueden fortalecer la confianza de los usuarios y mejorar la sostenibilidad de los sistemas de pago electrónico (West & Bhattacharya, 2016).

11 Objetivos, Límites y Alcance del Prototipo

11.1 Objetivos del Prototipo

Diseñar y construir un prototipo basado en algoritmos de *machine learning* que permita la identificación temprana de transacciones fraudulentas en pagos con tarjeta de crédito, con el fin de analizar su efectividad predictiva en un entorno simulado de datos históricos.

11.2 Límites del Prototipo

El prototipo tecnológico comienza con la ingesta y el análisis exploratorio (EDA) del dataset, abarcando la ingeniería de características y la aplicación de técnicas de tratamiento del desequilibrio de clases, y concluye con la validación del modelo de *machine learning* y la generación de un informe de resultados que contenga la cuantificación de la capacidad de detección de fraude a través de las métricas de performance seleccionadas.

11.3 Alcance del Prototipo

El prototipo abarcará todas las etapas necesarias para crear, probar y evaluar el modelo de detección de fraude:

- ✓ *Carga*: del dataset obtenido de *Kaggle*.
- ✓ *Preparación*: limpieza de datos, transformación de variables y la aplicación de técnicas para el manejo del desequilibrio de clases.
- ✓ *Modelado*: implementación y entrenamiento de los algoritmos de *machine learning* seleccionados, incluyendo la optimización de sus parámetros para maximizar su eficacia.

- ✓ *Evaluación*: prueba del modelo con datos de prueba, y cálculo de las métricas de rendimiento (Precision, Recall, ROC-AUC, etc.).
- ✓ *Generación del informe final*: análisis de los resultados obtenidos y recomendaciones de posibles mejoras.

Esto excluye:

- ✓ La integración en tiempo real con cualquier sistema o ERP.
- ✓ El desarrollo de una interfaz de usuario final.
- ✓ La gestión automática para la toma de decisiones basadas en las predicciones.

12 Descripción del Proyecto

12.1 Product Backlog

El *product backlog* constituye el eje de planificación del proyecto y reúne los elementos funcionales necesarios para desarrollar el prototipo de detección de fraude con tarjetas de crédito. Cada elemento representa una *historia de usuario*, expresada de forma breve y priorizada según su valor técnico y analítico.

De acuerdo con Rubin (2012) y Kerzner (2017), el *backlog* en *Scrum* es un artefacto dinámico que se actualiza de manera iterativa, permitiendo adaptar el trabajo a medida que se adquiere nuevo conocimiento. En este caso, los ítems reflejan las etapas del proceso de ciencia de datos aplicado al desarrollo del modelo predictivo, desde la exploración del dataset hasta la presentación de los resultados.

Product Backlog del Proyecto

ID	***Historia de Usuario***	***Prioridad***	***Puntos de historia***	***Dependencias***
DF-1	Análisis exploratorio del dataset de Kaggle	Alta	8	—
DF-2	Preprocesamiento y balanceo de datos mediante técnicas de muestreo	Alta	13	DF-1
DF-3	Entrenamiento inicial de modelos de *machine learning*	Alta	13	DF-2

DF-4	Optimización y validación de modelos predictivos	Media	8	DF-3
DF-5	Evaluación de desempeño y comparación de resultados entre modelos	Media	5	DF-4
DF-6	Elaboración del informe técnico y presentación de resultados	Media	3	DF-5

Fuente: elaboración propia

Cada historia será desarrollada de forma iterativa mediante *sprints*, aplicando el principio de refinamiento progresivo (Kerzner, 2017), donde las tareas de mayor prioridad reciben un mayor nivel de detalle y se abordan primero.

Este enfoque promueve la entrega incremental de valor y la mejora continua, asegurando que el desarrollo del modelo se mantenga alineado con los objetivos del proyecto.

En conjunto, este *product backlog* representa una planificación ágil, clara y adaptable, que guía la construcción del prototipo de detección de fraude basado en *machine learning*, garantizando la coherencia metodológica y la trazabilidad de las etapas del trabajo.

12.2 Historias de Usuario

Las *historias de usuario* constituyen una herramienta fundamental dentro de las metodologías ágiles, ya que permiten expresar los requerimientos del producto desde la perspectiva del usuario o beneficiario final. Representan una descripción breve de una funcionalidad o tarea que aporta valor, redactada en lenguaje no técnico y orientada a resultados (Cohn, 2004).

A continuación, se detallan las *historias de usuario* correspondientes al *product backlog* del proyecto, siguiendo la estructura “como <rol de usuario>, quiero <función> para lograr <valor de negocio>”, junto con sus criterios de aceptación expresados mediante el formato “dado – cuando – entonces”

Historias de Usuarios del Proyecto

ID	***Historia de Usuario***	***Descripción***	***Criterios de Aceptación***
DF-01	Análisis exploratorio del dataset de Kaggle	Como *analista de datos*, quiero realizar un análisis exploratorio del dataset de Kaggle para comprender la estructura, distribución y posibles anomalías de los datos.	*Dado* un dataset descargado de Kaggle, *cuando* se realiza el análisis exploratorio inicial, *entonces* se identifican las variables relevantes, su distribución estadística y la proporción de transacciones fraudulentas frente a las legítimas.
DF-02	Preprocesamiento y balanceo de datos	Como *científico de datos*, quiero preprocesar y balancear los datos mediante técnicas de muestreo, para obtener un conjunto equilibrado y listo para el entrenamiento del modelo.	*Dado* el dataset original con clases desbalanceadas, *cuando* se aplican técnicas de limpieza, normalización y muestreo (por ejemplo, SMOTE), *entonces* se obtiene un conjunto de datos balanceado y preparado para el modelado.
DF-03	Entrenamiento inicial de modelos de *machine learning*	Como *científico de datos*, quiero entrenar distintos modelos de machine learning, para evaluar su desempeño preliminar en la detección de transacciones fraudulentas.	*Dado* el conjunto de datos preprocesado, *cuando* se entrenan modelos supervisados y no supervisados (por ejemplo, Regresión Logística, Random Forest e Isolation Forest), *entonces* se obtienen métricas iniciales de precisión, recall y F1-score.
DF-04	Optimización y validación de modelos predictivos	Como *científico de datos*, quiero ajustar y validar los modelos seleccionados, para identificar el algoritmo con mejor rendimiento predictivo.	*Dado* el conjunto de modelos entrenados, *cuando* se realiza la optimización de hiperparámetros y la validación cruzada, *entonces* se determina el modelo más preciso y robusto para detectar fraude.

DF-05	Evaluación de desempeño y comparación de resultados	Como *investigador*, quiero evaluar el desempeño final y comparar los resultados obtenidos, para demostrar la efectividad del modelo elegido frente a otros enfoques.	*Dado* el modelo validado y sus métricas de evaluación, *cuando* se comparan los resultados entre distintos algoritmos, *entonces* se justifica la selección del modelo final con base en indicadores cuantitativos y cualitativos.
DF-06	Elaboración del informe técnico y presentación de resultados	Como *responsable del proyecto*, quiero documentar los resultados del modelo y generar visualizaciones de apoyo, para comunicar de forma clara los hallazgos y conclusiones del prototipo.	*Dado* el modelo final y los resultados obtenidos, *cuando* se elabora el informe y se crean las visualizaciones (gráficos, métricas, comparaciones), *entonces* se obtiene un documento técnico que sintetiza el proceso y evidencia la validez del prototipo.

Fuente: elaboración propia

Estas historias de usuario constituyen la base operativa del desarrollo ágil del proyecto, permitiendo organizar el trabajo de manera incremental, mantener la trazabilidad de cada requerimiento y garantizar que el resultado final —el prototipo funcional de detección de fraude— responda a los objetivos definidos.

12.3 Sprint Backlog

El *sprint backlog* es un artefacto fundamental dentro del marco de trabajo *Scrum*, que representa el plan detallado del equipo para completar las tareas seleccionadas de un *sprint* específico. A diferencia del *product backlog*, que contiene todas las funcionalidades planificadas para el desarrollo del prototipo, el *sprint backlog* se enfoca exclusivamente en el trabajo comprometido para un ciclo determinado, con el propósito de generar un incremento tangible del producto (Menzinsky et al., 2018).

En el marco del presente proyecto, el objetivo del *sprint* es avanzar en la construcción del modelo predictivo de detección de fraude a partir del dataset público de *Kaggle*, logrando un incremento funcional que permita validar los resultados intermedios del modelo de *machine learning*.

El *sprint backlog* se ha organizado en torno a los seis elementos del *product backlog* definidos previamente. Para cada historia de usuario se identifican las tareas específicas, su prioridad y el esfuerzo estimado, expresado en puntos de historia.

Sprint Backlog del Proyecto

Sprint	***Historia de Usuario***	***Tareas del Sprint***	***Prioridad***	***Días Estim.***	***Estado***
SP-1	Análisis exploratorio del dataset de Kaggle	- Importar dataset y revisar estructura de variables.	Alta	4	Por hacer
		- Realizar análisis descriptivo y detección de outliers.			
		- Generar visualizaciones para interpretar distribución de datos.			
	Preprocesamiento y balanceo de datos	- Limpiar datos faltantes o inconsistentes.	Alta	6	Por hacer
		- Normalizar variables numéricas.			
		- Aplicar técnica SMOTE para balancear clases.			
		- Guardar dataset procesado para modelado.			
	Entrenamiento inicial de modelos de *machine learning*	- Entrenar modelos iniciales (Regresión Logística, Random Forest, Isolation Forest).	Alta	6	Por hacer
		- Obtener métricas iniciales de desempeño.			
SP-2	Optimización y validación de modelos predictivos	- Ajustar hiperparámetros mediante validación cruzada.	Media	5	Por hacer
		- Comparar métricas y seleccionar el mejor modelo.			
	Evaluación de desempeño y comparación de resultados	- Evaluar modelo final sobre conjunto de prueba.	Media	4	Por hacer
		- Analizar errores de clasificación y falsos positivos.			
	Elabor. informe técnico y present. de resultados	- Elaborar resumen de resultados del modelo.	Media	3	Por hacer
		- Crear gráficos de rendimiento y curva ROC.			
		- Redactar conclusiones y recomendaciones finales.			

Fuente: elaboración propia

Los *sprint backlogs* de este proyecto, por lo tanto, representan la manifestación táctica del plan de desarrollo del prototipo, donde cada historia de usuario se convierte en un conjunto de acciones concretas que avanzan progresivamente hacia la entrega del modelo final de detección de fraude.

12.4 Estructura de Datos

Para el desarrollo del presente prototipo se emplea el dataset público "*Credit Card Fraud Detection*"[6] disponible en la plataforma *Kaggle* (Bhadouria, 2023). Este conjunto de datos ofrece una representación realista de transacciones con tarjetas de crédito, etiquetadas como legítimas o fraudulentas, y se orienta a la experimentación con técnicas de *machine learning* aplicadas a la detección de fraude financiero.

El dataset contiene 1.852.394 registros y 23 atributos, con información asociada a la transacción, al titular de la tarjeta y al comercio donde se realizó la operación.

Cada registro representa una transacción individual, identificada de forma única y clasificada según el campo *is_fraud*, que actúa como variable objetivo:

1 → transacción fraudulenta

0 → transacción legítima

El conjunto de datos está anonimizado para proteger la identidad de los titulares, pero mantiene características representativas del comportamiento financiero real, lo que lo convierte en un recurso idóneo para la construcción de modelos predictivos en un entorno académico y experimental.

Estructura de las Variables

Categoría	***Variable***	***Descripción***	***Tipo de dato***
Información de la transacción	trans_date_trans_time	Fecha y hora de la transacción	Texto/Datetime
	cc_num	Identificador único (anonimizado) de la tarjeta	Numérico
	merchant	Nombre comercio donde se realizó transacción	Texto

[6] https://www.kaggle.com/datasets/tusharbhadouria/credit-card-fraud-detection

	category	Categoría del comercio (viajes, comida, etc.)	Texto
	amt	Monto de la transacción	Numérico
Datos del titular	first, last	Nombre y apellido del titular (anonimizados)	Texto
	gender	Género del titular	Texto
	street, city, state, zip	Dirección postal	Texto / Numérico
	lat, long	Coordenadas geográficas del domicilio del titular	Numérico
	city_pop	Población de la ciudad del titular	Numérico
	job	Profesión del titular	Texto
	dob	Fecha de nacimiento del titular	Fecha
Identificadores y tiempo	trans_num	Identificador único de la transacción	Texto
	unix_time	Marca temporal en formato Unix	Numérico
Datos del comercio	merch_lat, merch_long	Coordenadas del comercio	Numérico
Variable objetivo	is_fraud	Indicador de fraude (1 = Sí, 0 = No)	Binario

Fuente: Kaggle

Esta organización tabular permite un acceso eficiente a los datos y facilita las operaciones de preprocesamiento, normalización y codificación necesarias para el modelado predictivo. Las variables geográficas y demográficas aportan información contextual valiosa para identificar patrones de comportamiento inusual, mientras que las variables temporales y categóricas permiten detectar desviaciones en frecuencia o ubicación de las operaciones (Ngai et al., 2011)

El dataset ya se encuentra particionado en las tablas *fraud_train* y *fraud_test* (ambas con la misma estructura de variables), por lo que no es necesario realizar una división básica train/test sobre el archivo original. Esta partición preexistente debe aprovecharse como sigue:

- *Uso de fraud_train*: se utiliza como el conjunto principal para entrenamiento y validación interna. Dentro de *fraud_train* se recomienda aplicar técnicas de validación como *k-fold cross-validation* estratificada para seleccionar modelos y ajustar hiperparámetros, garantizando que la proporción de casos

is_fraud se mantenga en cada pliegue. Esto permite estimar rendimiento de forma robusta sin tocar el conjunto de prueba.

- *Uso de fraud_test*: se reserva exclusivamente como conjunto de evaluación final para medir el desempeño real e imparcial del modelo seleccionado. *Fraud_test* no debe influir en decisiones de ajuste de hiperparámetros ni en la ingeniería de características salvo para análisis post-hoc.
- *Consistencia en el preprocesado*: cualquier transformación debe entrenarse/ajustarse únicamente sobre *fraud_train* y luego aplicarse a *fraud_test* usando los parámetros aprendidos.
- *Manejo del desbalance*: las técnicas de re-muestreo (SMOTE, undersampling, etc.) deben aplicarse solo sobre los datos de entrenamiento y después de la partición/estratificación interna; no deben aplicarse al conjunto *fraud_test*.
- *Evaluación final*: emplear *fraud_test* para calcular las métricas definitivas (Precision, Recall, F1, AUC-PR, AUC-ROC) y para análisis de negocio (costes asociados a FP/FN).

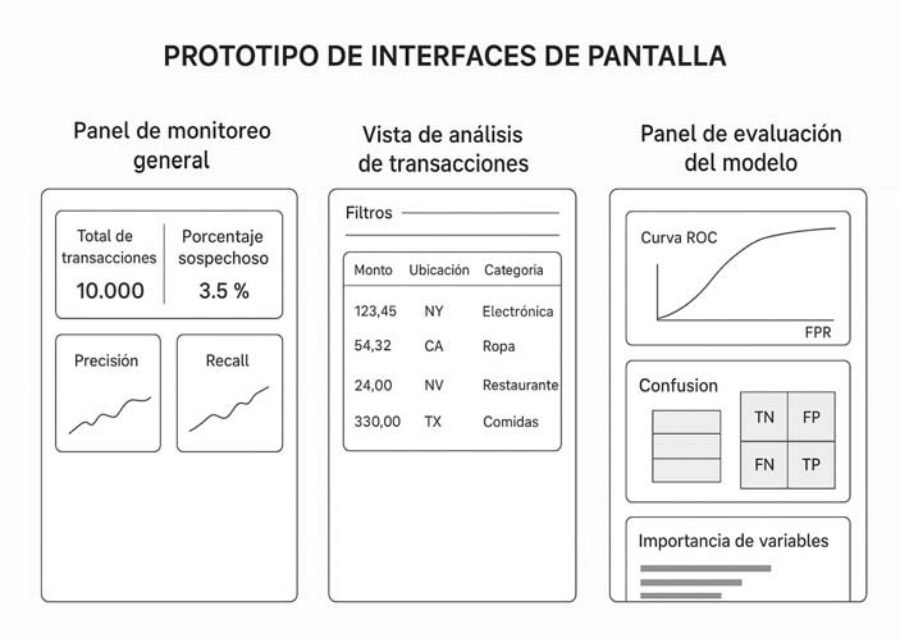
12.5 Prototipo de Interfaces de Pantalla

El diseño de interfaces constituye un elemento esencial en el desarrollo de sistemas analíticos, ya que permite transformar los resultados de los modelos predictivos en información comprensible y útil para los usuarios finales (De Liddo & Buckingham, 2010). Sin embargo, en los límites de este trabajo se ha definido que no se desarrollarán interfaces de usuarios finales, dado que el objetivo central es la validación del modelo analítico de detección de fraude y no la implementación de una herramienta de uso operativo.

No obstante, resulta pertinente describir de manera conceptual cómo se visualizarían los resultados del prototipo en un entorno real. En un sistema de detección de fraude con tarjetas de crédito, las interfaces de usuario cumplirían un rol mediador entre el modelo de *machine learning* y los analistas de fraude o responsables de riesgo, facilitando la interpretación y priorización de las alertas generadas por el sistema (Ngai et al., 2011).

En un escenario productivo, podrían contemplarse tres tipos de pantallas principales[7]:

- *Panel de monitoreo general*: presentaría indicadores clave como la cantidad total de transacciones procesadas, el porcentaje detectado como sospechoso, y las métricas de desempeño del modelo (precisión, recall, F1-score). Este panel permitiría una visión global del comportamiento del sistema en tiempo real.
- *Vista de análisis de transacciones*: permitiría filtrar y examinar transacciones específicas marcadas como fraudulentas, mostrando atributos relevantes como el monto, la ubicación geográfica, la categoría del comercio y el puntaje de riesgo asignado por el modelo. Esta visualización facilitaría la labor de los analistas de fraude para verificar casos concretos.
- *Panel de evaluación del modelo*: orientado a los científicos de datos o analistas técnicos, incluiría visualizaciones como la curva ROC, la matriz de confusión y gráficos de importancia de variables, lo que posibilitaría evaluar el rendimiento del modelo y ajustar parámetros en futuros entrenamientos.



Fuente: Chat GPT

[7] OpenAI. (2025). ChatGPT (versión 19 de octubre). [Modelo de lenguaje amplio], https://chat.openai.com/chat

De este modo, aunque el prototipo no cuenta con una interfaz operativa, se mantiene la correspondencia entre el diseño conceptual de pantallas y los requerimientos funcionales del proceso de detección de fraude, fortaleciendo la comprensión del modelo y su potencial de aplicación práctica.

12.6 Diagrama de Arquitectura

La arquitectura propuesta para el prototipo de detección de fraude con tarjetas de crédito se basa en un entorno de desarrollo local, utilizando una notebook estándar como único nodo físico de procesamiento. Dado que el trabajo tiene fines académicos y experimentales, no se requiere una infraestructura distribuida ni servicios en la nube, sino un entorno reproducible y accesible mediante herramientas de software libre.

El entorno *físico* se compone de un único equipo de cómputo personal (notebook) con sistema operativo Windows, sobre el cual se ejecutan los procesos de análisis, modelado y validación. Este equipo actúa como servidor y cliente al mismo tiempo, albergando tanto los componentes de procesamiento como los artefactos de software. La conectividad a internet se limita a la descarga inicial del dataset de Kaggle y las dependencias de librerías necesarias.

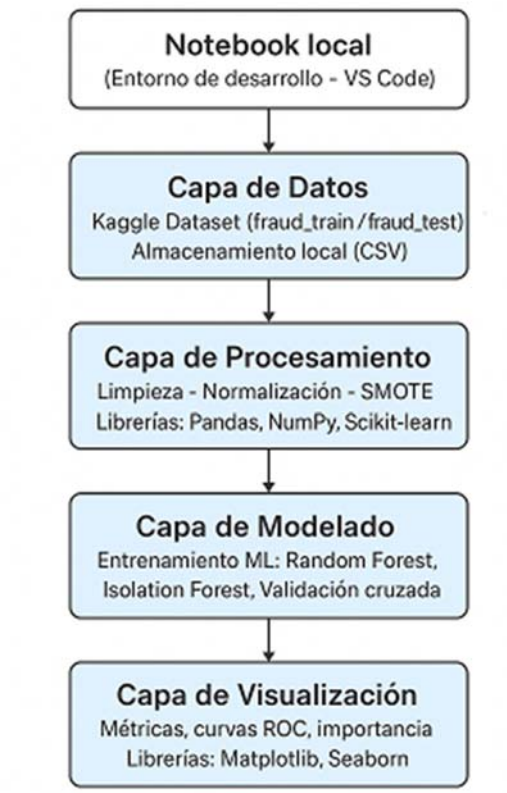
Desde un punto de vista *lógico*, el sistema se estructura en cuatro capas principales que reflejan el flujo de trabajo de un proyecto de ciencia de datos siguiendo la metodología CRISP-DM (Ferreira, 2017):

1) *Capa de datos*: incluye el dataset "*Credit Card Fraud Detection*" obtenido de Kaggle, compuesto por las tablas *fraud_train* y *fraud_test*. Los datos se almacenan localmente en formato CSV y son accedidos mediante la librería *Pandas*.
2) *Capa de procesamiento*: en esta etapa se realiza el preprocesamiento, limpieza, normalización y balanceo de clases mediante las librerías *NumPy* y *Scikit-learn*, aplicando técnicas como *SMOTE* para tratar el desbalance de datos (Chawla et al., 2002).
3) *Capa de modelado*: aquí se implementan los algoritmos de *machine learning* seleccionados (*LogisticRegression*, *Random Forest* e *Isolation Forest*) para

entrenar, validar y evaluar los modelos predictivos. Se utilizan métricas de rendimiento tales como *Precision*, *Recall*, *F1-score* y *AUC-ROC*.

4) *Capa de visualización y resultados*: comprende la presentación de resultados mediante gráficos estadísticos generados con *Matplotlib* y *Seaborn*, incluyendo la matriz de confusión, curvas ROC y gráficos de importancia de variables. Esta capa cumple una función analítica y de comunicación de hallazgos, no operativa (De Liddo & Buckingham, 2010).

Diagrama de Arquitectura Conceptual



Fuente: Chat GPT

Esta arquitectura, simple pero funcional, permite ejecutar de forma secuencial todo el flujo analítico del proyecto —desde la carga de datos hasta la interpretación de resultados— dentro de un mismo entorno de desarrollo. Su estructura modular garantiza la reproducibilidad de los experimentos y la trazabilidad del modelo predictivo, cumpliendo los criterios de transparencia y replicabilidad exigidos en la investigación académica.

13 Seguridad

13.1 Acceso a la Aplicación

En el marco del presente trabajo, el acceso al sistema se limita al entorno de desarrollo del modelo de detección de fraude, dado que se trata de un proyecto académico basado en un dataset público obtenido de la plataforma *Kaggle*. Por este motivo, las medidas de seguridad adoptadas se orientan principalmente a garantizar el uso controlado y responsable de los recursos del entorno de ciencia de datos, en lugar de implementar políticas complejas de acceso corporativo.

El acceso al entorno se gestiona mediante un sistema de autenticación simple, que requiere usuario y contraseña. Esta modalidad resulta adecuada para un contexto académico, en el que no se manipula información confidencial ni datos personales de usuarios reales. Sin embargo, se aplican buenas prácticas básicas de gestión de credenciales, como la definición de contraseñas seguras (mínimo de ocho caracteres, combinación de letras, números y símbolos). De esta forma, se logra un equilibrio entre protección y facilidad de uso para los participantes autorizados.

Si este sistema se implementara en un entorno multiusuario o de aplicación real, sería necesario incorporar mecanismos adicionales de seguridad. Entre ellos se incluirían la autenticación multifactor (MFA) —combinando contraseña, dispositivo verificado o dato biométrico—, la asignación de roles y permisos diferenciados para controlar el acceso a datos y modelos según el perfil de usuario, y la aplicación de protocolos seguros de comunicación entre los módulos del sistema (Mell & Grance, 2011). Estas medidas fortalecerían la protección ante intentos de acceso indebido y garantizarían la trazabilidad de las acciones dentro del sistema.

13.2 Política de Respaldo de Información

El desarrollo del prototipo de detección de fraude se llevará a cabo en una notebook local, utilizando el lenguaje *Python* y el entorno de desarrollo *Visual Studio Code*. En este contexto, la política de respaldo de información se orienta

a proteger los archivos del proyecto frente a posibles pérdidas por errores humanos, fallas del sistema o daños en el dispositivo de almacenamiento.

La estrategia de respaldo se basa en una combinación de prácticas de seguridad y copias programadas que garanticen la preservación de los avances del proyecto. En primer lugar, se implementarán respaldos completos del directorio principal del proyecto —incluyendo scripts de *Python*, datasets originales y procesados, modelos entrenados y documentación—. Estos respaldos se realizarán de forma semanal, asegurando una copia íntegra de todos los componentes críticos del entorno de trabajo.

Los respaldos se almacenarán en dos ubicaciones complementarias: una unidad externa física (como un disco duro portátil o pendrive cifrado) y un servicio de almacenamiento en la nube (por ejemplo, *Google Drive* o *GitHub*). Esta redundancia minimiza los riesgos asociados a fallas locales y garantiza la disponibilidad de las copias en caso de desperfectos del equipo principal (Marinescu, 2017).

Los elementos protegidos por esta política incluyen los datasets originales y transformados, los modelos de *machine learning* entrenados junto con sus hiperparámetros, los scripts de procesamiento y visualización, y la documentación técnica. Se pretende además realizar verificaciones periódicas de integridad y restauración, para confirmar que los respaldos sean válidos y funcionales ante eventuales contingencias.

Si este modelo se implementara en un entorno productivo o multiusuario, sería necesario complementar esta política con medidas adicionales, tales como almacenamiento geográficamente distribuido, control de acceso basado en roles y protocolos de recuperación ante desastres con responsabilidades y tiempos de respuesta definidos. Estas prácticas garantizan la continuidad operativa y la protección de los datos en sistemas analíticos de mayor complejidad.

13.3 Disponibilidad de la Información

La disponibilidad de la información en este proyecto se refiere a garantizar el acceso continuo a los datasets, scripts y modelos durante todo el proceso de

desarrollo. Dado que el prototipo se ejecuta en una notebook personal y en un entorno local de *Python* mediante *Visual Studio Code*, las medidas adoptadas se enfocan en mantener la continuidad del trabajo y evitar pérdidas de datos.

Si este sistema se implementara en un entorno productivo o multiusuario, sería necesario disponer de una arquitectura de alta disponibilidad, con servidores redundantes, balanceadores de carga y monitoreo automatizado de rendimiento, así como definir acuerdos de nivel de servicio (SLA) que aseguren porcentajes de disponibilidad cercanos al 99%. Estas prácticas, junto con mecanismos de escalabilidad automática y planes de continuidad operativa, permitirían mantener la accesibilidad y estabilidad del sistema incluso ante picos de demanda o contingencias críticas (ISO/IEC 27031:2011).

14 Análisis de Costos

El análisis de costos del presente proyecto tiene como objetivo estimar lo que serían los recursos económicos necesarios para el desarrollo, implementación y mantenimiento del prototipo propuesto para la detección de fraude en pagos con tarjetas de crédito[8].

Se distinguen tres categorías principales de costos:

1. *Costos de desarrollo*: asociados al trabajo del equipo técnico en las distintas etapas del proyecto (análisis, modelado, entrenamiento, evaluación e implementación del prototipo).
2. *Costos de software e infraestructura*: vinculados al uso de herramientas, servidores y servicios en la nube necesarios para el procesamiento y entrenamiento de los modelos.
3. *Costos operativos recurrentes*: que incluyen servicios de soporte y mantenimiento del entorno de trabajo analítico.

[8] OpenAI. (2025). ChatGPT (versión 01 de noviembre). [Modelo de lenguaje amplio], https://chat.openai.com/chat

Para el desarrollo del prototipo se consideran cuatro perfiles de trabajo, ajustados a la magnitud del proyecto, los cuales se detallan a continuación;

Costos de Desarrollo

Rol	*Honorarios Mensuales*	*Meses*	*Monto (ARS)*
Científico de Datos	1.600.000	2	3.200.000
Ingeniero de Datos	1.400.000	2	2.800.000
Analista de Datos	1.200.000	2	2.400.000
Scrum Master / Coordinador	1.550.000	2	3.100.000
Total			**11.500.000**

Fuente: Chat GPT con base en los valores de referencia del Consejo Profesional de Ciencias Informáticas de Córdoba (CPCIPC, 2025)

El desarrollo del prototipo se llevará a cabo utilizando software de código abierto y recursos accesibles en la nube para el entrenamiento y prueba de los modelos. Los costos se estiman considerando el uso de plataformas escalables y licencias individuales.

Inversión en Software e Infraestructura

Recurso	*Cantidad*	*Fuente*	*Monto (ARS)*
Servicio Cloud (Google Colab Pro+ o AWS SageMaker, uso 2 meses)	1	https://aws.amazon.com/sagemaker/pricing	480.000
Almacenamiento en la nube (Google Cloud Storage, 1 TB)	1	https://cloud.google.com/storage/pricing	60.000
Licencia Tableau Public / Power BI (versión gratuita)	1	tableau.com / powerbi.microsoft.com	0
Software open source (Python, Scikit-learn, Pandas, etc.)	-	open source	0
Total			**540.000**

Fuente: Chat GPT, noviembre 2025

También se incluyen los gastos vinculados al mantenimiento del entorno de desarrollo, soporte técnico y servicios asociados al almacenamiento y ejecución de tareas analíticas durante el ciclo de vida del proyecto.

Costos Operativos Recurrentes

Recurso	*Cantidad*	*Fuente*	*Monto Mensual (ARS)*
Conectividad y servicios asociados	1	proveedor local	100.000
Soporte técnico eventual	1	dataconsulting.com.ar	200.000
Total mensual estimado			**300.000**

Fuente: Chat GPT, noviembre 2025

Resumiendo los cuados anteriores tenemos el siguiente detalle sobre la inversión total requerida:

Resumen General de Inversión

Categoría	*Montos Iniciales (ARS)*	*Montos Recurrentes (ARS/mes)*
Capital humano	11.500.000	-
Software e infraestructura	540.000	-
Operativos	-	300.000
Totales	**12.040.000**	**300.000**

Fuente: elaboración propia con base a Chat GPT

El desarrollo del prototipo requiere una inversión inicial estimada de $12.040.000, correspondiente principalmente a los costos de capital humano y a la infraestructura mínima de cómputo en la nube. Los costos mensuales de operación se estiman en $300.000, que incluyen servicios de mantenimiento, ejecución de modelos y almacenamiento en la nube.

El uso de herramientas open source y datasets públicos permite minimizar significativamente los gastos en licencias y software propietario, garantizando la viabilidad económica del proyecto sin comprometer la calidad técnica del modelo analítico.

15 Análisis de Riesgos

El análisis de riesgos permite identificar y gestionar los factores que podrían afectar el correcto desarrollo y la calidad del prototipo de detección de

fraude en pagos con tarjetas de crédito. En proyectos de ciencia de datos, los riesgos pueden provenir de la calidad de los datos, el comportamiento de los modelos, la disponibilidad de recursos tecnológicos o la planificación del proyecto (Kloppenborg et al., 2022).

La anticipación y control de estos riesgos son fundamentales para garantizar la validez de los resultados y la continuidad del proceso analítico.

Riesgos Identificados del Proyecto

Tipo	***Riesgo***	***Causa Probable***
Datos	Calidad insuficiente del dataset	Presencia de valores faltantes, ruido o inconsistencias que afectan el aprendizaje del modelo
Técnico-analítico	Sobreajuste (overfitting) del modelo	Dataset poco representativo o ajuste excesivo de hiperparámetros
Datos	Sesgos en los datos	Distribución desigual entre clases o sesgos en la recolección de información
Infraestructura	Limitaciones de procesamiento	Recursos computacionales limitados o incompatibilidad de librerías
Gestión	Retrasos en la ejecución del proyecto	Reprocesamientos o ajustes derivados de resultados no satisfactorios

Fuente: elaboración propia

Cada riesgo se evalúa considerando su *probabilidad de ocurrencia* (P) y su *impacto* (I) en una escala del 1 al 5, donde 1 representa un nivel muy bajo y 5 un nivel muy alto.

El grado de exposición se obtiene multiplicando ambas variables (P × I), permitiendo estimar la severidad de cada riesgo.

Evaluación de Riesgos

Riesgo	***Probabilidad (P)***	***Impacto (I)***	***Grado de Exposición***
Calidad insuficiente del dataset	4	5	20
Sesgos en los datos	3	5	15
Overfitting del modelo	3	4	12
Limitaciones de procesamiento	3	4	12
Retrasos en la ejecución del proyecto	2	3	6

Fuente: elaboración propia

Los valores más elevados indican riesgos de mayor relevancia, que requieren seguimiento continuo y estrategias de mitigación tempranas.

Plan de Contingencia

Riesgo	***Estrategia preventiva / correctiva***
Calidad insuficiente del dataset	Implementar controles automáticos de validación y limpieza de datos; aplicar técnicas de imputación y normalización antes del modelado.
Sesgos en los datos	Evaluar la representatividad de las clases; aplicar métodos de balanceo como SMOTE o undersampling; revisar métricas de equidad.
Overfitting del modelo	Utilizar validación cruzada, regularización y ajuste prudente de hiperparámetros; comparar resultados entre entrenamiento y prueba.
Limitaciones de procesamiento	Optimizar código, reducir dimensionalidad y aprovechar entornos en la nube o hardware acelerado en fases críticas.
Retrasos en la ejecución	Establecer cronogramas ajustados, documentar avances y aplicar gestión ágil mediante sprints cortos y entregables intermedios.

Fuente: elaboración propia

El análisis evidencia que los riesgos más significativos se concentran en la calidad y representatividad de los datos, elementos esenciales para la precisión de los modelos predictivos.

La implementación de controles de calidad, junto con buenas prácticas de gestión y documentación, asegura la estabilidad del proyecto y la confiabilidad de los resultados obtenidos.

16 Demo

16.1 Código Fuente del Prototipo

Ver Anexo 1.

16.2 Salida que Arroja el Código

Ver Anexo 2.

16.3 Descripción del Código

A continuación, se hace una breve reseña (para mayor detalle ver los anexos) de sobre cómo se implementa el código en *Python*:

A – Análisis Exploratorio de Datos (EDA)

El prototipo inicia importando las librerías necesarias para la carga, manipulación, visualización y modelado de datos. Se incluyen herramientas de análisis (*Pandas y NumPy*), gráficos (*Matplotlib* y *Seaborn*) y funciones de *Scikit-learn* para entrenar modelos supervisados y no supervisados, además de módulos específicos para curvas ROC, Precision, Recall y curvas de aprendizaje. También se incorpora *SMOTE*, una técnica clave para corregir el fuerte desbalance de clases presente en los datos de fraude.

Posteriormente se cargan los datasets *Train* y *Test* provistos por *Kaggle*[9]. En ambos se elimina la columna auxiliar *Unnamed: 0*, y se verifican dimensiones para confirmar la integridad de la carga.

Estructura de los Datasets

	Train	***Test***
Registros	1.296.675	555.719
Campos	22	22

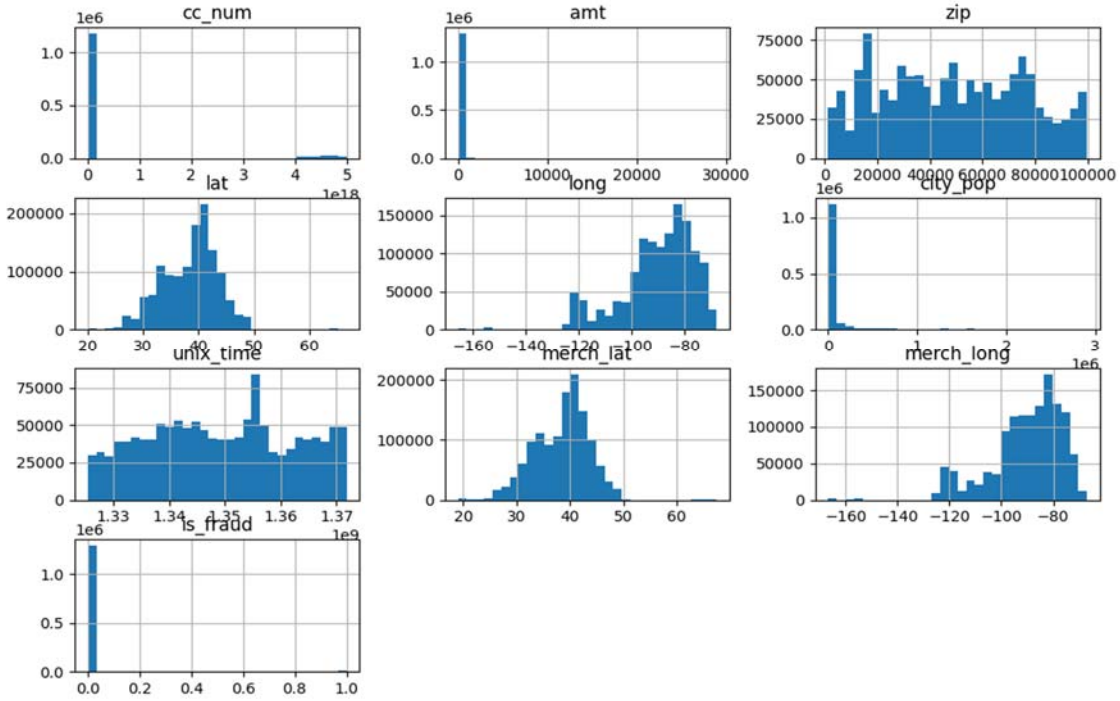
Fuente: elaboración propia

Luego se muestran las primeras filas del conjunto de entrenamiento para analizar la estructura general: características temporales, montos, ubicación

[9] https://www.kaggle.com/datasets/tusharbhadouria/credit-card-fraud-detection

geográfica, categoría del comercio y variable objetivo *is_fraud*. Este paso se complementa con estadísticas descriptivas de las variables numéricas y la verificación de valores faltantes, confirmando que no existen nulos y que la calidad de los datos permite avanzar sin necesidad de realizar imputaciones.

Histogramas de Variables Numéricas



Fuente: elaboración propia

Se continúa examinando la distribución de la variable objetivo en ambos conjuntos, constatándose un desbalance severo: menos del 1% de las operaciones son fraudulentas. Este hallazgo justifica la futura aplicación de técnicas de *oversampling* para evitar que los modelos ignoren la clase minoritaria.

Distribución de 'is_fraud'

(0 = legitima, 1 = fraudulenta)

Clases	***Train***		***Test***	
	#	**%**	**#**	**%**
0	1.289.169	99,4211	553.574	99,6140
1	7.506	0,5789	2.145	0,3860

Fuente: elaboración propia

Para completar el análisis, se calculan valores atípicos en las variables numéricas mediante el método IQR. Aunque se identifican *outliers*, especialmente en montos, estos no se eliminan, ya que en problemas de fraude pueden representar justamente los patrones más relevantes.

B – Preparación de Datos e Ingeniería de Características

Tras el análisis inicial, se seleccionan las variables numéricas más relevantes para el proceso de modelado: *amt, lat, long, city_pop, merch_lat, merch_long, zip, unix_time* e *is_fraud*. Esta selección reduce la complejidad del dataset y se enfoca en atributos con influencia directa en el comportamiento de las transacciones.

Además, se generan nuevas variables como: *avg_amt_24h_cc_nu* (monto promedio por tarjeta en las últimas 24 hs), *count_24h_cc_num* (cantidad de operaciones por tarjeta en las últimas 24 hs) y *count_1h_cc_num* (cantidad de operaciones por tarjeta en la última hora).

Luego se separan las características predictoras de la variable objetivo y se aplica *SMOTE* sobre el conjunto de *Train*. Esta técnica genera ejemplos sintéticos únicamente para la clase minoritaria, equilibrando la representación de ambos grupos y favoreciendo el aprendizaje de patrones asociados al fraude. El *Test* permanece intacto para preservar su rol de evaluación real.

Distribución Aplicando SMOTE

Clases	***Train***	
	Sin SMOTE	***Con SMOTE***
0	1.289.169	1.289.169
1	7.506	128.916

Fuente: elaboración propia

Para asegurar que las variables operen en escalas comparables, se aplica *StandardScaler* tanto en *Train* como sobre *Test.* De esta manera, ambas particiones quedan normalizadas de forma coherente, evitando distorsiones en el entrenamiento.

C – Entrenamiento de Modelos

El prototipo entrena tres modelos distintos, cada uno con un propósito específico:

1. *Regresión Logística: p*ermite establecer un rendimiento mínimo esperado y comparar mejoras obtenidas con modelos más complejos.
2. *Random Forest: e*s el modelo central del prototipo debido a su robustez ante ruido, outliers y relaciones no lineales. Durante las primeras ejecuciones presentó sobreajuste, evidenciado por una brecha muy amplia entre el rendimiento en *Train* y *Test*. Para corregirlo, se ajustaron hiperparámetros clave (max_depth, min_samples_split, n_estimators) y se incorporaron técnicas de validación cruzada. También se redujo al 10% la proporción en el balanceo de clases. Estos ajustes mejoraron la capacidad de generalización del modelo y redujeron la varianza.
3. *Isolation Forest: s*e utiliza como método *no supervisado* de detección de anomalías. Su inclusión permite complementar el enfoque *supervisado* (ya que tiene un bajo nivel predictivo), otorgando una capa adicional de análisis útil para escenarios donde puedan surgir comportamientos inusuales no reflejados en la variable objetivo.

Estos son los resultados obtenidos por cada modelo:

Métricas de los Modelos

Modelo	***Precision (Clase 1)***	***Recall (Clase 1***	***F1-Score (Clase 1)***	***AUC***
Regresion Logistica	0,33	0,62	0,43	0,9573
Random Forest	0,80	0,86	0,83	0,9831
Isolation Forest	0,08	0,15	0,10	0,5708

Fuente: elaboración propia

D – Optimización y Validación

Dada la falta de capacidad de cómputos, se decide optimizar solo uno de los modelos, es por ello que esta sección se aplica solamente a *Random Forest*, ya que presenta indicadores muy superiores a los otros 2 modelos. Se emplea validación cruzada estratificada para preservar la proporción de clases en cada

fold, lo que asegura estimaciones de rendimiento más realistas en datos desbalanceados. Para dicho modelo se evaluaron varias combinaciones de hiperparámetros mediante *GridSearchCV/RandomizedSearchCV*, priorizando configuraciones que reduzcan la varianza sin sacrificar *recall* sobre la clase fraudulenta.

Los principales criterios considerados durante la optimización fueron:

- estabilizar la diferencia entre desempeño en *Train* y en *Test*;
- maximizar *recall* para la clase fraudulenta sin aumentar de manera desproporcionada los falsos positivos;
- mantener tiempos de entrenamiento razonables.

Mejores hiperparámetros encontrados: '

- n_estimators: 200,
- min_samples_split: 5,
- min_samples_leaf: 2,
- max_features: 'sqrt',
- max_depth: 15

E – Evaluación de Desempeño

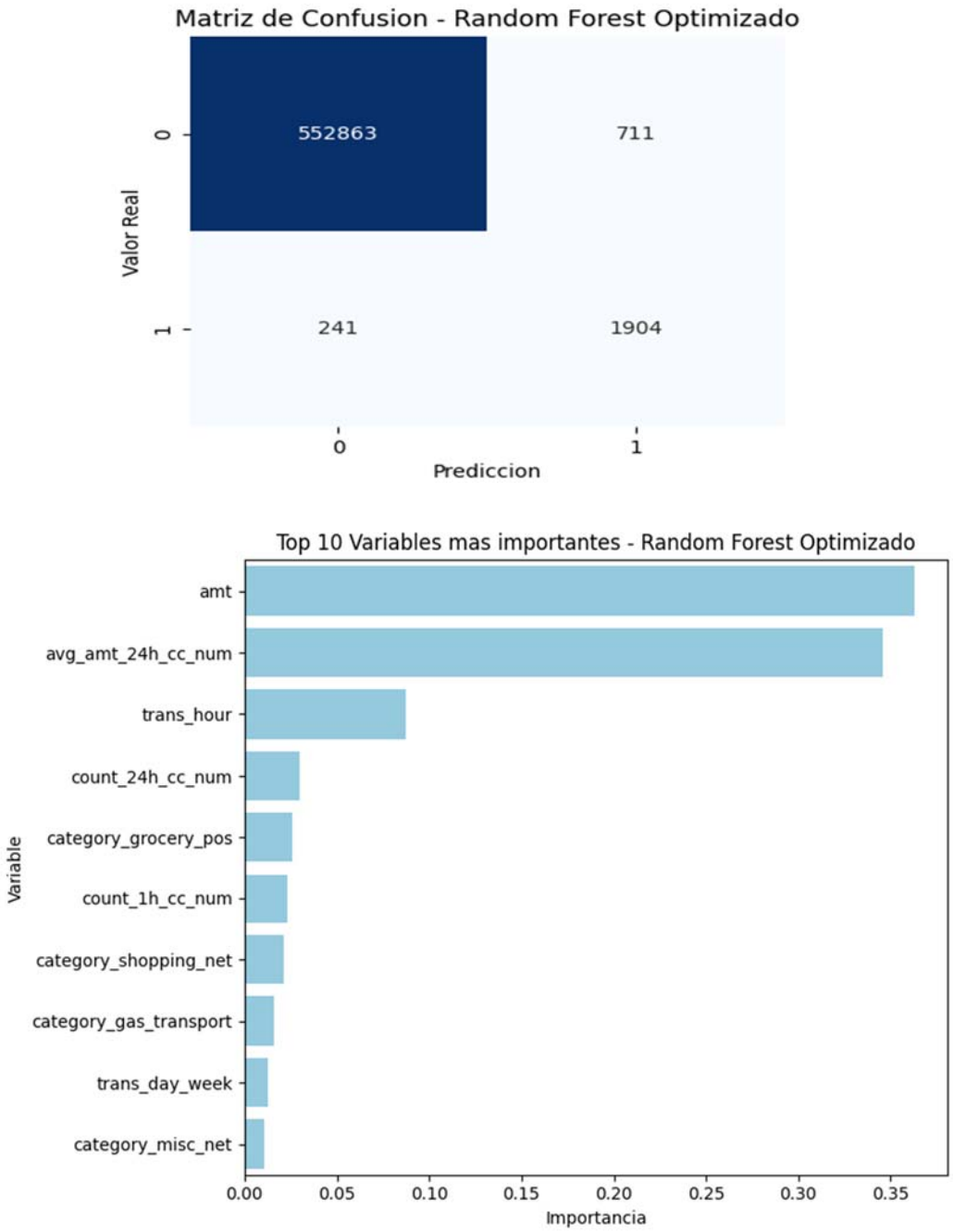
Con el modelo *Random Forest* optimizado, se realizan las evaluaciones finales sobre el conjunto de prueba sin procesar (*Test*). Las métricas calculadas incluyen: *precisión, recall, F1-score, AUC-ROC* y la *matriz de confusión*. Se enfatiza el análisis de *recall* para la clase fraudulenta, dado que el costo asociado a falsos negativos es sustancial en términos económicos y operativos.

Resumen Comparativo del Modelo

Modelo	***Precision (Clase 1)***	***Recall (Clase 1***	***F1-Score (Clase 1)***	***AUC***
Random Forest Base	0,8350	0,8615	0,8315	0,9831
Random Forest Optimizado	0,7281	0,8876	0,8000	0,9848

Fuente: elaboración propia

Adicionalmente, se presenta la *matriz confusión* y un gráfico con las principales variables que explican el comportamiento del modelo optimizado:

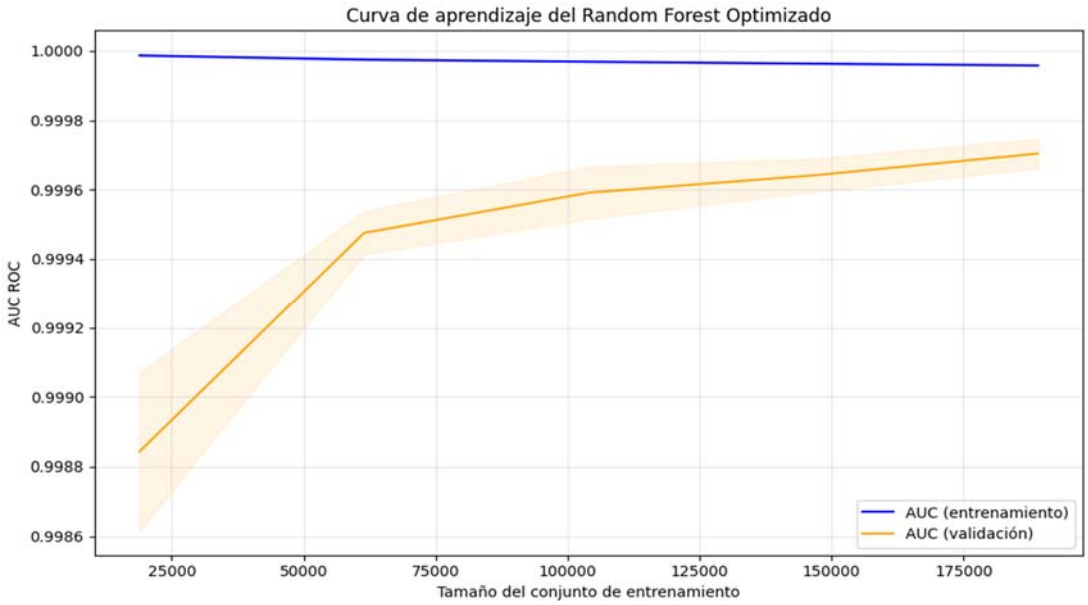


Fuente: elaboración propia

Finalmente, se construye una *curva de aprendizaje* para el *Random Forest* con el objetivo de verificar la evolución del desempeño en función del tamaño del conjunto de entrenamiento. Inicialmente la curva mostró un gap entre *Train* y *Test* (indicador de sobreajuste). Tras la optimización de hiperparámetros

mencionada en *D*, la brecha se redujo, indicando mayor capacidad de generalización.

La estabilidad observada en la curva final y la consistencia entre métricas de validación y métricas sobre *Test* respaldan la robustez del prototipo para su aplicación en entornos reales.



Fuente: elaboración propia

Por último, los procesos descritos permiten concluir que el modelo final presenta un comportamiento sólido y estable. La combinación de métricas favorables, baja diferencia entre desempeño en entrenamiento y prueba, y la coherencia observada en la curva de aprendizaje demuestran que el sistema generaliza adecuadamente frente a información nueva. En síntesis, el prototipo resultante no sólo alcanza un rendimiento alto en la identificación de transacciones fraudulentas, sino que lo hace cumpliendo criterios de robustez, interpretabilidad y consistencia metodológica, por lo que se encuentra en condiciones de integrarse a un proceso real de monitoreo y análisis de fraude, tal como se enuncia en los objetivos del presente trabajo.

16.4 Documentación de Instalación

Para poder ejecutar el prototipo funcional el principal requisito es contar con un IDE que incluya Python versión 3.12.4, y tener instaladas las siguientes librerías:

- ✓ pandas
- ✓ numpy
- ✓ seabon
- ✓ matplotlib
- ✓ sklearn
- ✓ imblearn
- ✓ tqdm

Además, se requieren los dataset (Train y Test), y el archivo con el código fuente en Python; ambos se encuentran alojados en la nube, cuyo link figura debajo.

16.5 Alojamiento en la Nube

En el siguiente enlace se podrá descargar el prototipo de ciencia de datos desarrollado, y los datasets necesarios para su ejecución:

https://drive.google.com/drive/folders/1YNePLIIsDupNgGmcG7keT7vne6hcNd_k?usp=drive_link

17 Conclusiones

El proyecto permitió demostrar que la aplicación de técnicas de ciencia de datos y modelos de *machine learning* mejora de manera significativa la capacidad de detección temprana de fraude en operaciones con tarjetas de crédito. Los objetivos definidos al inicio del trabajo se cumplieron con éxito, logrando un sistema analítico capaz de identificar patrones de riesgo y reducir la exposición financiera ante transacciones fraudulentas.

El modelo final desarrollado —un Random Forest optimizado— alcanzó un *Recall* de 0.8876 (de cada 100 operaciones fraudulentas el sistema detecta

casi 89 de ellas), desempeño alineado con la prioridad estratégica del negocio: minimizar la cantidad de fraudes no detectados. La *Precision* de 0.7281 (cada 100 operaciones detectadas como fraudulentas, solo 73 lo son realmente), aunque menor, se mantuvo dentro de niveles aceptables para un entorno donde un falso positivo implica un costo menor que un falso negativo. Esta relación costo-riesgo justificó los ajustes realizados en los hiperparámetros y en el umbral de decisión para favorecer la sensibilidad del modelo. La decisión fue coherente con las prácticas estándar del sector financiero, donde la mitigación del fraude prima sobre el impacto operativo que pueden generar los bloqueos preventivos.

La experiencia técnica del proyecto dejó en evidencia tanto fortalezas como limitaciones. Entre las primeras, destaca la capacidad del modelo para generalizar adecuadamente, la estabilidad obtenida tras el proceso de optimización y el establecimiento de un proceso reproducible que integra limpieza, balanceo, entrenamiento, validación y métricas de desempeño.

Entre las limitaciones, el tiempo total de procesamiento fue uno de los principales desafíos. La ejecución completa del proyecto demandó más de 1 hora, con la fase de optimización como el componente que más tiempo demoraba. Este aspecto obligó a reducir el espacio de búsqueda de hiperparámetros y ajustar la complejidad del entrenamiento para mantener la viabilidad operativa en un entorno local. Adicionalmente, las primeras versiones del modelo mostraron señales claras de sobreajuste, lo que requirió introducir controles específicos, como la regulación de profundidad, el ajuste de *min_samples_split* y el uso estricto de validación cruzada para estabilizar el rendimiento.

La calidad y estructura del dataset —altamente desbalanceado y basado en datos simulados— también impusieron restricciones. Aunque el uso de *SMOTE* permitió entrenar el modelo con una representación más equilibrada de clases, persisten limitaciones propias de los datos artificiales: ausencia de ruido real, patrones temporales incompletos y baja diversidad de fraudes sofisticados. Estas condiciones deben ser consideradas al evaluar la aplicabilidad del modelo a entornos productivos.

A pesar de estos desafíos, el sistema desarrollado permite visualizar con claridad el valor potencial de incorporar modelos analíticos en procesos de gestión de fraude. La solución también establece bases sólidas para evolucionar hacia modelos operativos que funcionen en tiempo real e integren múltiples fuentes de información.

A partir del análisis realizado, se identifican tres líneas de evolución futuras que podrían potenciar el rendimiento del sistema:

- *Integración de nuevas fuentes de datos*: incorporar información transaccional en tiempo real, dispositivos, localización y patrones conductuales permitiría enriquecer el modelo y reducir falsos positivos.
- *Exploración de algoritmos más avanzados*: técnicas como *LightGBM*, *XGBoost* o modelos profundos podrían ofrecer mejoras adicionales en *Recall* y en capacidad de detección de patrones complejos.
- *Escalado hacia entornos productivos*: migración a infraestructuras de procesamiento distribuido, APIs de scoring y arquitectura en streaming para ejecutar el modelo en tiempo real y con volúmenes masivos.

Desde una perspectiva profesional, el proyecto consolidó habilidades fundamentales para el ejercicio de la ciencia de datos: análisis exploratorio, *machine learning*, manejo de desbalance, validación rigurosa y comunicación de hallazgos técnicos a actores no técnicos. La necesidad de iterar, resolver problemas de rendimiento y justificar decisiones metodológicas fortaleció la capacidad de análisis crítico y la comprensión integral del ciclo de vida analítico.

En conclusión, el prototipo desarrollado constituye una demostración concreta del valor que la analítica avanzada puede aportar a la prevención de fraude en *fintechs,* bancos y comercios. Los resultados son robustos para un entorno académico, las conclusiones son extrapolables a escenarios reales y la solución deja una base sólida para futuras extensiones, tanto técnicas como estratégicas, orientadas a la toma de decisiones basada en datos.

18 Referencias Bibliográficas

Bhattacharyya, S., Jha, S., Tharakunnel, K., & Westland, J. C. (2011). *Data mining for credit card fraud: A comparative study. Decision Support Systems*, 602–613.

Bobadilla, J. (2020). *Machine learning y Deep Learning, Usando Python, Scikit y Keras*

Buczak, A. L., & Guven, E. (2016). *A survey of data mining and machine learning methods for fraud detection. Journal of the American Medical Informatics Association*, 12-21.

Chawla, N. V. et al. (2002). SMOTE: *Synthetic Minority Over-sampling Technique*. Journal of Artificial Intelligence Research, 16, 321–357.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). *SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research*, 16, 321-357.

Chen, L., Wu, T., & Xu, Y. (2018). *Imbalanced data classification using resampling methods and ensemble learning*. IEEE International Conference on Big Data.

Cohn, M. (2004). *User Stories Applied: For Agile Software Development*. Addison-Wesley

Dal Pozzolo, A., Caelen, O., Johnson, R. A., & Bontempi, G. (2015). *Calibrating probability with undersampling for unbalanced classification*. IEEE Symposium on Computational Intelligence and Data Mining (CIDM).

De Liddo, A., & Buckingham, S. (2010). *Cohere: A Prototype for Contested Collective Intelligence. ACM Computer Supported Cooperative Work* (CSCW 2010) - Workshop:CollectiveIntelligence In Organizations - Toward a Research Agenda

Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. A. (2013). *Fundamentals of Business Process Management*. Springer.

Fawcett, T., & Provost, F. (1996). *Robust Classification for Imbalanced Datasets. Proceedings of the Twelfth International Conference on Machine learning*, 201-208.

Ferreira, A. (2017). *Data Mining with R: Learning with Case Studies*. CRC Press.

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining: Concepts and Techniques* (3rd ed.). Morgan Kaufmann

ISO/IEC. (2011). ISO/IEC 27031*: Information technology — Security techniques — Guidelines for information and communication technology readiness for business continuity*. International Organization for Standardization.

Kerzner, H. (2017). *Project Management: A Systems Approach to Planning, Scheduling, and Controlling.* Wiley.

Kloppenborg, T., Anantatmula, V., & Wells, K. (2022). *Contemporary Project Management.Cengage*.

Marinescu, D. C. (2017). *Cloud computing: Theory and practice* (2nd ed.). Morgan Kaufmann.

Mell, P., & Grance, T. (2011). *The NIST definition of cloud computing* (NIST Special Publication 800-145). National Institute of Standards and Technology.

Menzinsky, D., Perin, C., & Pérez, M. (2018). *Gestión ágil de requerimientos en proyectos de desarrollo de software*. Universidad Nacional de La Plata.

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

Pineda Pertuz, C. (2022). *Aprendizaje automático y profundo en Python, una mirada hacia la inteligencia artificial*

Pyle, D. (2003). *Business Modeling and Data Mining*. Morgan Kaufmann Publishers.

Rayo Mondragon, C. A. (2020). *Prototipo de detección de fraudes con tarjetas de crédito basado en inteligencia artificial aplicado a un banco peruano*. Universidad de Lima.

Rubin, K. S. (2012). *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Addison-Wesley Professional.

Srivastava, P., & Srivastava, B. (2018). *Fraud Detection in Credit Card Transactions using Machine learning*. International Journal of Computer Applications, 180(5), 18-22.

The Nilson Report. (2021). *Global Credit Card Fraud Losses*. Recuperado de: https://nilsonreport.com

West, J., & Bhattacharya, M. (2016). *Intelligent financial fraud detection: A comprehensive review*. Computers & Security,

Wysocki, R. K. (2013). *Effective Project Management: Traditional, Agile, Extreme*. John Wiley & Sons.

A. Anexo 1 – Código del Prototipo en Python

```
# ============================================================
# TRABAJO FINAL DE GRADO - UNIVERSIDAD SIGLO 21
# Autor: Maximiliano Federici
# Proyecto: Deteccion de fraude en pagos con tarjetas de credito
# Dataset: Credit Card Fraud Detection (Kaggle)
# ============================================================

# ----------------------------------------
# A - Analisis Exploratorio de Datos (EDA)
# ----------------------------------------

print("--------------------------------------")
print("A-Analisis Exploratorio de Datos (EDA)")
print("--------------------------------------")

# A.1. Importacion de librerías

import matplotlib
matplotlib.use('Qt5Agg') # usa Qt en lugar de Tkinter (soluciona el
       error con graficos)
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, IsolationForest
from sklearn.metrics import classification_report, confusion_matrix,
       roc_auc_score, roc_curve, precision_recall_curve,
       accuracy_score, precision_score, recall_score, f1_score,
       average_precision_score, classification_report
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import GridSearchCV, StratifiedKFold
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import learning_curve
from tqdm import tqdm

# A.2. Carga de los datasets

train = pd.read_csv('d:\\SIGLO 21\\2025\\SEM311 - SEMINARIO FINAL EN
       CIENCIA DE DATOS\\Prototipo\\Credit Card Fraud
       Detection\\fraudTrain.csv',
                   usecols=lambda x: x != 'Unnamed: 0')

test = pd.read_csv('d:\\SIGLO 21\\2025\\SEM311 - SEMINARIO FINAL EN
       CIENCIA DE DATOS\\Prototipo\\Credit Card Fraud
       Detection\\fraudTest.csv',
                   usecols=lambda x: x != 'Unnamed: 0')

print("Datasets cargados correctamente")
print(f"Filas y columnas Train: {train.shape}\n")
print(f"Filas y columnas Test: {test.shape}\n")

# A.3. Revision de la estructura Train

print("Primeras filas del dataset Train:")
```

```
print(train.head(),"\n")

print("Descripcion estadistica Train:")
print(train.describe(),"\n")

print("Valores nulos por columnas Train:")
print(train.isnull().sum(),"\n")
```

 # A.4. Analisis descriptivo y deteccion de outliers Train

```
# Distribucion de la variable objetivo
print("Distribucion de 'is_fraud' Train (0 = legitima, 1 =
       fraudulenta):")
print(train['is_fraud'].value_counts(normalize=True) * 100)
print("Distribucion de 'is_fraud' Test (0 = legitima, 1 =
       fraudulenta):")
print(test['is_fraud'].value_counts(normalize=True) * 100)

# Deteccion de outliers mediante el metodo IQR Train
train_numerica = train.select_dtypes(include=['number'])
Q1 = train_numerica.quantile(0.25)
Q3 = train_numerica.quantile(0.75)
IQR = Q3 - Q1
outliers = ((train_numerica < (Q1 - 1.5 * IQR)) | (train_numerica >
       (Q3 + 1.5 * IQR))).sum()
print("____________")
print("Cantidad de valores atípicos detectados por variable numerica
       Train:")
print(outliers[outliers > 0], "\n")

# Histogramas Train
train.hist(figsize=(12,8), bins=30)
plt.title("Distribucion gral de las variables Train", fontsize=14)
plt.show()

# Boxplt Train
plt.figure(figsize=(12,8))
sns.boxplot(x='category', y='amt', data=train)
plt.title("Boxplot de 'category' y 'amt' Train")
plt.xticks(rotation=45)
plt.show()

# Correlaciones Train
plt.figure(figsize=(12,8))
corr = train_numerica.corr()
sns.heatmap(corr, cmap="coolwarm", center=0)
plt.title("Mapa de correclacion entre variables Train")
plt.show()

# Distribucion del monto de transaccion por clase Train
plt.figure(figsize=(8,6))
sns.boxplot(x='is_fraud', y='amt', data=train)
plt.title('Distribucion del monto por tipo de transaccion Train')
plt.show()
```

 # --
 # B - Preprocesamiento y balanceo de datos
 # --

```
print("--------------------------------------")
print("B-Preprocesamiento y balanceo de datos")
print("--------------------------------------")
```

 # B.1. Eliminacion de datos duplicados

```
train=train.drop_duplicates()
print(f"Train luego de eliminar duplicados: {train.shape}\n")
test=test.drop_duplicates()
print(f"Test luego de eliminar duplicados: {test.shape}\n")
```

 # B.2. Convertir las columnas de fecha/hora

```
train['trans_date_trans_time'] =
       pd.to_datetime(train['trans_date_trans_time'])
test['trans_date_trans_time'] =
       pd.to_datetime(test['trans_date_trans_time'])

train['trans_month'] = train['trans_date_trans_time'].dt.month
train['trans_day'] = train['trans_date_trans_time'].dt.day
train['trans_day_week'] = train['trans_date_trans_time'].dt.dayofweek
train['trans_hour'] = train['trans_date_trans_time'].dt.hour
train['trans_minute'] = train['trans_date_trans_time'].dt.minute

test['trans_month'] = test['trans_date_trans_time'].dt.month
test['trans_day'] = test['trans_date_trans_time'].dt.day
test['trans_day_week'] = test['trans_date_trans_time'].dt.dayofweek
test['trans_hour'] = test['trans_date_trans_time'].dt.hour
test['trans_minute'] = test['trans_date_trans_time'].dt.minute
```

 # B.3. Creacion de variables de tiempo

```
# Ordenar por tarjeta y tiempo
train = train.sort_values(by=['cc_num', 'trans_date_trans_time'])
test = test.sort_values(by=['cc_num', 'trans_date_trans_time'])

# Crear una nueva variable para el monto promedio en 24h por tarjeta
train['avg_amt_24h_cc_num'] = 0.0
test['avg_amt_24h_cc_num'] = 0.0
for cliente, data in tqdm(train.groupby('cc_num'), desc='Calculando
       monto promedio 24h por tarjeta'):
    train.loc[data.index, 'avg_amt_24h_cc_num'] = (
        data.set_index('trans_date_trans_time')['amt']
        .rolling('1D', closed='left')  # ventana movil de 1 dia
        .mean()
        .values
    )
for cliente, data in tqdm(test.groupby('cc_num'), desc='Calculando
       monto promedio 24h por tarjeta'):
    test.loc[data.index, 'avg_amt_24h_cc_num'] = (
        data.set_index('trans_date_trans_time')['amt']
        .rolling('1D', closed='left')  # ventana movil de 1 dia
        .mean()
        .values
    )

# Crear una nueva variable para cantidad de operaciones en 24h por
       tarjeta
train['count_24h_cc_num'] = 0.0
```

```
test['count_24h_cc_num'] = 0.0
for cliente, data in tqdm(train.groupby('cc_num'), desc='Calculando
       cantidad 24h por tarjeta'):
    train.loc[data.index, 'count_24h_cc_num'] = (
        data.set_index('trans_date_trans_time')['amt']
        .rolling('1D', closed='left')  # ventana movil de 1 dia
        .count()
        .values
    )
for cliente, data in tqdm(test.groupby('cc_num'), desc='Calculando
       cantidad 24h por tarjeta'):
    test.loc[data.index, 'count_24h_cc_num'] = (
        data.set_index('trans_date_trans_time')['amt']
        .rolling('1D', closed='left')  # ventana movil de 1 dia
        .count()
        .values
    )

# Crear una nueva variable para cantidad de operaciones en 1h por
       tarjeta
train['count_1h_cc_num'] = 0.0
test['count_1h_cc_num'] = 0.0
for cliente, data in tqdm(train.groupby('cc_num'), desc='Calculando
       cantidad 1h por tarjeta'):
    train.loc[data.index, 'count_1h_cc_num'] = (
        data.set_index('trans_date_trans_time')['amt']
        .rolling('1h', closed='left')  # ventana movil de 1 hora
        .count()
        .values
    )
for cliente, data in tqdm(test.groupby('cc_num'), desc='Calculando
       cantidad 1h por tarjeta'):
    test.loc[data.index, 'count_1h_cc_num'] = (
        data.set_index('trans_date_trans_time')['amt']
        .rolling('1h', closed='left')  # ventana movil de 1 dia
        .count()
        .values
    )

# Rellenar valores NaN iniciales (con pocas operaciones)
train['avg_amt_24h_cc_num'] =
       train['avg_amt_24h_cc_num'].fillna(train['amt'].median())
train['count_24h_cc_num'] =
       train['count_24h_cc_num'].fillna(train['amt'].median())
train['count_1h_cc_num'] =
       train['count_1h_cc_num'].fillna(train['amt'].median())
test['avg_amt_24h_cc_num'] =
       test['avg_amt_24h_cc_num'].fillna(test['amt'].median())
test['count_24h_cc_num'] =
       test['count_24h_cc_num'].fillna(test['amt'].median())
test['count_1h_cc_num'] =
       test['count_1h_cc_num'].fillna(test['amt'].median())

# Elimina variable de tiempo original
train = train.drop('trans_date_trans_time', axis=1)
test = test.drop('trans_date_trans_time', axis=1)
```

 # B.4. Codificar variables categoricas

```
train_encoded = pd.get_dummies(train, columns=['category'],
       drop_first=True)
train = train_encoded

test_encoded = pd.get_dummies(test, columns=['category'],
       drop_first=True)
test = test_encoded

# Alinear columnas de test con las de train
missing_cols = set(train.columns) - set(test.columns)
for c in missing_cols:
    test[c] = 0
test = test[train.columns]

# B.5. Escalado de columnas numericas

scaler = StandardScaler()
numericas =
       train.select_dtypes(include=['number']).columns.drop('is_fraud'
       )
scaler.fit(train[numericas])
train[numericas] = scaler.transform(train[numericas])
test[numericas]  = scaler.transform(test[numericas])

# B.6. Balanceo de clases

print("Distribucion original de clases en el conjunto de
       entrenamiento:")
print(train['is_fraud'].value_counts())

x_train = train.drop('is_fraud', axis=1)
y_train = train['is_fraud']
x_test = test.drop('is_fraud', axis=1)
y_test = test['is_fraud']

# Eliminar columnas con texto
x_train.drop(columns=['cc_num','merchant','trans_num','first','last',
       'gender', 'street', 'state', 'city', 'dob', 'job'],
       errors='ignore', inplace=True)
x_test.drop(columns= ['cc_num','merchant','trans_num','first','last',
       'gender', 'street', 'state', 'city', 'dob', 'job'],
       errors='ignore', inplace=True)

# Balanceo de clases con SMOTE (reduccion para evitar sobreajuste)
smote = SMOTE(sampling_strategy=0.1, random_state=42)
x_train_balan, y_train_balan = smote.fit_resample(x_train, y_train)

print("Distribucion despues de aplicar SMOTE:")
print(y_train_balan.value_counts())

# ----------------------------------------
# C - Entrenamiento inicial de modelos
# ----------------------------------------

print("-------------------------------------")
print("C-Entrenamiento inicial de modelos")
print("-------------------------------------")

# C.1. Regresion Logistica
```

```
print("----------------------------")
print("Modelo 1: Regresion Logistica")
lr = LogisticRegression(max_iter=1000, random_state=42)
lr.fit(x_train_balan, y_train_balan)
y_pred_lr = lr.predict(x_test)

print("Metricas de Regresion Logistica")
print(classification_report(y_test, y_pred_lr))
print("AUC:", roc_auc_score(y_test, lr.predict_proba(x_test)[:,1]),
      "\n")
```

 # C.2. Random Forest

```
print("----------------------------")
print("Modelo 2: Random Forest")
rf = RandomForestClassifier(n_estimators=100, random_state=42,
      n_jobs=-1)
rf.fit(x_train_balan, y_train_balan)
y_pred_rf = rf.predict(x_test)

print("Metricas de Random Forest")
print(classification_report(y_test, y_pred_rf))
print("AUC:", roc_auc_score(y_test, rf.predict_proba(x_test)[:,1]),
      "\n")
```

 # C.3. Isolation Forest

```
print("----------------------------")
print("Modelo 3 - Isolation Forest (No supervisado)")
iso = IsolationForest(contamination=0.005, random_state=42)
iso.fit(x_train)
pred_iso = iso.predict(x_test)
pred_iso = np.where(pred_iso == -1,1,0)

print("Metricas Isolation Forest")
print(classification_report(y_test, pred_iso))
print("AUC (aprox.):", roc_auc_score(y_test, pred_iso), "\n")
```

 # C.4. Comparacion de resultados

```
resultados = pd.DataFrame({
    "Modelo": ["Regresion Logistica", "Random Forest", "Isolation
       Forest"],
    "AUC": [
        roc_auc_score(y_test, lr.predict_proba(x_test)[:,1]),
        roc_auc_score(y_test, rf.predict_proba(x_test)[:,1]),
        roc_auc_score(y_test, pred_iso)
        ]
})
print("Resumen comparativo de desempeño de modelos")
print(resultados.sort_values(by="AUC", ascending=False))
```

 # --
 # D - Optimizacion y validacian de modelos predictivos
 # --

```
print("--------------------------------------")
print("D-Optimizacion de hiperparametros - Random Forest")
```

```
print("-------------------------------------")

# D.1 - Muestreo para reducir tiempo de computo

x_train_sample = x_train_balan.sample(frac=0.3, random_state=42)
y_train_sample = y_train_balan.loc[x_train_sample.index]

print(f"Tamaño de muestra para optimizacion: {x_train_sample.shape}")

# D.2 - Definicion del modelo base y espacio de busqueda

rf_base = RandomForestClassifier(random_state=42, n_jobs=-1)

param_distributions = {
    'n_estimators': [100, 200],
    'max_depth': [10, 15],
    'min_samples_split': [5, 10],
    'min_samples_leaf': [2, 5],
    'max_features': ['sqrt', 'log2'],
 }

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

random_search = RandomizedSearchCV(
    estimator=rf_base,
    param_distributions=param_distributions,
    n_iter=10,
    cv=cv,
    scoring='roc_auc',
    n_jobs=-1,
    random_state=42,
    verbose=2
)

# Entrenamiento con búsqueda aleatoria
random_search.fit(x_train_sample, y_train_sample)

print("\nMejores hiperparametros encontrados:")
print(random_search.best_params_)
print("Mejor AUC promedio en validacion cruzada:",
       round(random_search.best_score_, 4))

# D.3 - Validacion rapida del modelo optimizado

print("----------------------------")
print("Validacion rapida del modelo optimizado (Random Forest)")
best_rf = random_search.best_estimator_

# Probabilidades y AUC preliminar
y_proba_best = best_rf.predict_proba(x_test)[:, 1]
auc_best = roc_auc_score(y_test, y_proba_best)
print(f"AUC preliminar del modelo optimizado en test: {auc_best:.4f}")

# D.4 - Comparacion resumida: Modelo Base vs Optimizado

y_pred_rf = rf.predict(x_test)
y_proba_rf_base = rf.predict_proba(x_test)[:, 1]
```

```
report_base = classification_report(y_test, y_pred_rf,
       output_dict=True)
report_best = classification_report(y_test, best_rf.predict(x_test),
       output_dict=True)
auc_base = roc_auc_score(y_test, y_proba_rf_base)

comparacion = pd.DataFrame({
    'Modelo': ['Random Forest Base', 'Random Forest Optimizado'],
    'Precision (Clase 1)': [
        round(report_base['1']['precision'], 4),
        round(report_best['1']['precision'], 4)
    ],
    'Recall (Clase 1)': [
        round(report_base['1']['recall'], 4),
        round(report_best['1']['recall'], 4)
    ],
    'F1-Score (Clase 1)': [
        round(report_base['1']['f1-score'], 4),
        round(report_best['1']['f1-score'], 4)
    ],
    'AUC': [round(auc_base, 4), round(auc_best, 4)]
})

print("\nResumen comparativo entre modelo base y optimizado:")
print(comparacion)

# ------------------------------------------------
# E - Evaluacion de desempeño y comparación de resultados
# ------------------------------------------------

print("--------------------------------------")
print("Evaluacion Final del Modelo Random Forest Optimizado")
print("--------------------------------------")

# E.1 - Predicciones finales y metricas

y_pred_best = best_rf.predict(x_test)
y_proba_best = best_rf.predict_proba(x_test)[:, 1]

accuracy = accuracy_score(y_test, y_pred_best)
precision = precision_score(y_test, y_pred_best)
recall = recall_score(y_test, y_pred_best)
f1 = f1_score(y_test, y_pred_best)
roc_auc = roc_auc_score(y_test, y_proba_best)
pr_auc = average_precision_score(y_test, y_proba_best)

print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-Score: {f1:.4f}")
print(f"ROC-AUC: {roc_auc:.4f}")
print(f"PR-AUC: {pr_auc:.4f}")

print("\nReporte detallado de clasificacion:")
print(classification_report(y_test, y_pred_best, digits=4))

# E.2 - Matriz de confusion

plt.figure(figsize=(5,4))
```

```
sns.heatmap(confusion_matrix(y_test, y_pred_best), annot=True,
       fmt='d', cmap='Blues', cbar=False)
plt.title("Matriz de Confusion - Random Forest Optimizado")
plt.xlabel("Prediccion")
plt.ylabel("Valor Real")
plt.show()
plt.close('all')
```

 # E.3 - Curva ROC

```
fpr, tpr, _ = roc_curve(y_test, y_proba_best)
plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, label=f"AUC = {roc_auc:.3f}", color="darkorange")
plt.plot([0,1],[0,1],'--', color='gray')
plt.title("Curva ROC - Random Forest Optimizado")
plt.xlabel("Tasa de Falsos Positivos (FPR)")
plt.ylabel("Tasa de Verdaderos Positivos (TPR)")
plt.legend()
plt.show()
plt.close('all')
```

 # E.4 - Curva Precision-Recall

```
precision_vals, recall_vals, _ = precision_recall_curve(y_test,
       y_proba_best)
plt.figure(figsize=(6,5))
plt.plot(recall_vals, precision_vals, color="blue", label=f"AP =
       {pr_auc:.3f}")
plt.title("Curva Precision-Recall - Random Forest Optimizado")
plt.xlabel("Recall (Sensibilidad)")
plt.ylabel("Precision")
plt.legend(loc='lower left')
plt.show()
plt.close('all')
```

 # E.5 - Importancia de variables

```
importances = pd.Series(best_rf.feature_importances_,
       index=x_train_balan.columns).sort_values(ascending=False)
plt.figure(figsize=(8,6))
sns.barplot(x=importances[:10], y=importances.index[:10],
       color="skyblue")
plt.title("Top 10 Variables mas importantes - Random Forest
       Optimizado")
plt.xlabel("Importancia")
plt.ylabel("Variable")
plt.tight_layout()
plt.show()
plt.close('all')
```

 # E.6 - Curva de aprendizaje

```
sample_frac = 0.2
X_sample = x_train_balan.sample(frac=sample_frac, random_state=42)
y_sample = y_train_balan.loc[X_sample.index]

train_sizes, train_scores, val_scores = learning_curve(
    estimator=best_rf,
    X=X_sample,
```

```
    y=y_sample,
    cv=3,
    scoring='roc_auc',
    n_jobs=-1,
    train_sizes=np.linspace(0.1, 1.0, 5),
    shuffle=True,
    random_state=42
)

train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
val_mean = np.mean(val_scores, axis=1)
val_std = np.std(val_scores, axis=1)

plt.figure(figsize=(10, 6))
plt.plot(train_sizes, train_mean, label="AUC (entrenamiento)",
       color="blue")
plt.plot(train_sizes, val_mean, label="AUC (validación)",
       color="orange")
plt.fill_between(train_sizes, train_mean - train_std, train_mean +
       train_std, color="blue", alpha=0.1)
plt.fill_between(train_sizes, val_mean - val_std, val_mean + val_std,
       color="orange", alpha=0.1)
plt.title("Curva de aprendizaje del Random Forest Optimizado")
plt.xlabel("Tamaño del conjunto de entrenamiento")
plt.ylabel("AUC ROC")
plt.legend()
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
plt.close('all')
```

 # E.7 - Guardar metricas finales

```
metricas_finales = pd.DataFrame({
    'Accuracy': [accuracy],
    'Precision': [precision],
    'Recall': [recall],
    'F1_Score': [f1],
    'ROC_AUC': [roc_auc],
    'PR_AUC': [pr_auc]
})
metricas_finales.to_csv("d:\\SIGLO 21\\2025\\SEM311 - SEMINARIO FINAL
       EN CIENCIA DE
       DATOS\\Prototipo\\metricas_random_forest_final.csv",
       index=False)
print("\nMetricas finales guardadas en
       'metricas_random_forest_final.csv'")
```

 # E.8 - Interpretacion general

```
print("\nCONCLUSIONES E INTERPRETACION GENERAL:")
print("El modelo Random Forest optimizado muestra un rendimiento
       sólido, con un equilibrio adecuado entre precisión y
       sensibilidad.")
print("El alto valor de ROC-AUC confirma su capacidad para distinguir
       operaciones legítimas de las fraudulentas.")
```

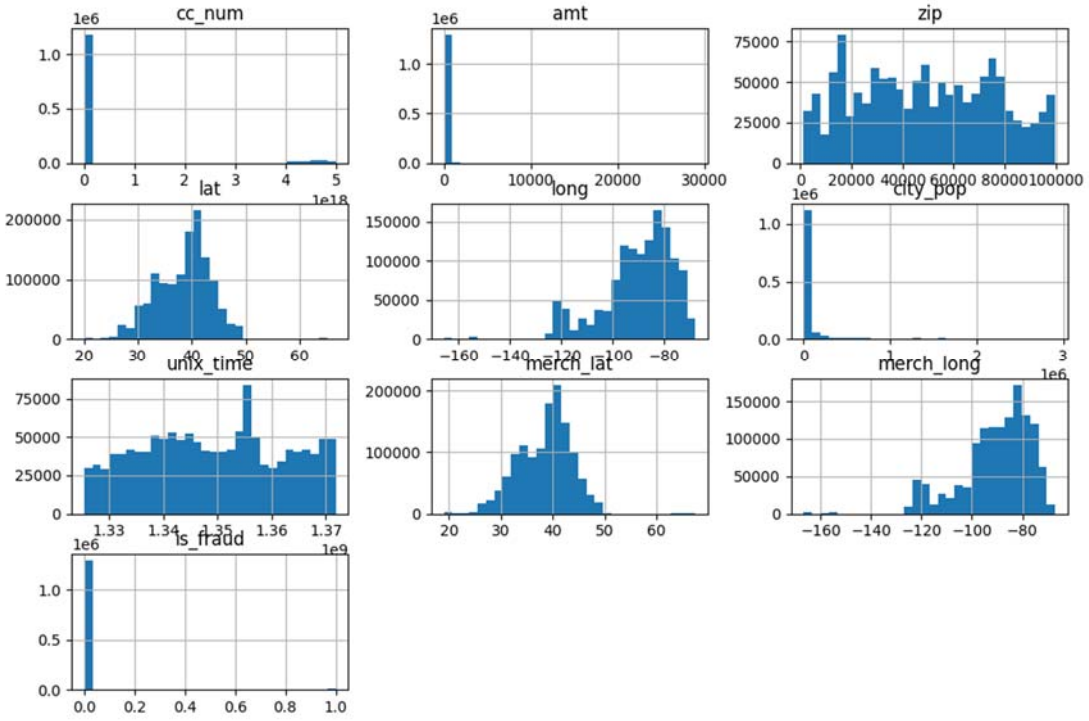
```
print("La curva Precision-Recall demuestra que el modelo mantiene
       buena precisión incluso en un contexto de clases
       desbalanceadas.")
print("Las variables con mayor importancia se relacionan
       principalmente con el monto, promedio diario de montos, hora
       del dia y cantidad de operaciones diarias, coherentes con
       patrones típicos de fraude.")
```

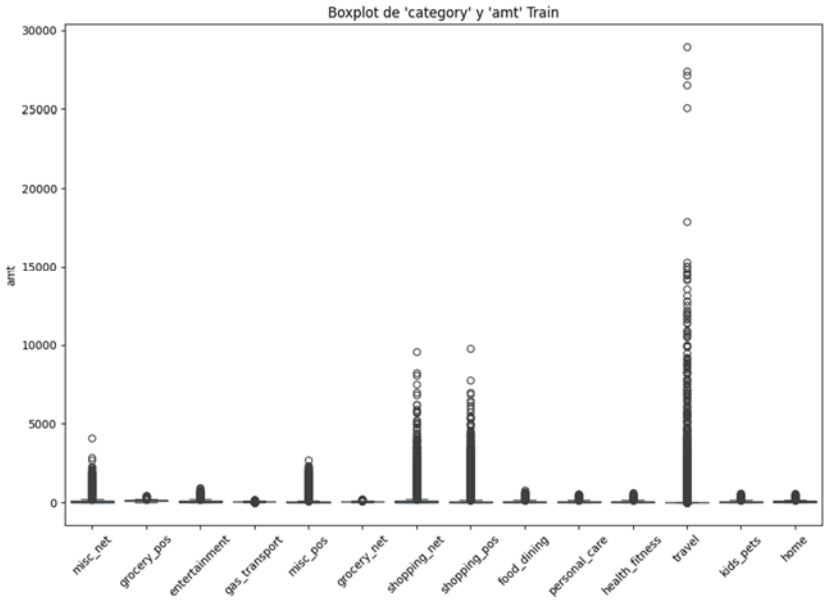
B. Anexo 2 – Salida del Código

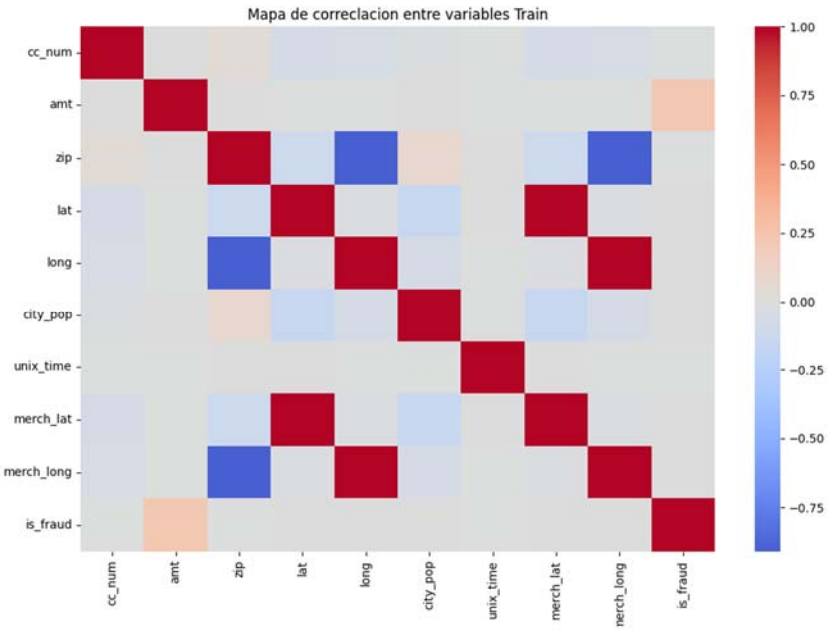
```
--------------------------------------
A-Analisis Exploratorio de Datos (EDA)
--------------------------------------
Datasets cargados correctamente
Filas y columnas Train: (1296675, 22)
Filas y columnas Test: (555719, 22)
Primeras filas del dataset Train:
  trans_date_trans_time          cc_num                            merchant       category  ...   unix_time  merch_lat  merch_long is_fraud
0   2019-01-01 00:00:18  2703186189652095          fraud_Rippin, Kub and Mann       misc_net  ...  1325376018  36.011293  -82.048315        0
1   2019-01-01 00:00:44      630423337322      fraud_Heller, Gutmann and Zieme    grocery_pos  ...  1325376044  49.159047 -118.186462        0
2   2019-01-01 00:00:51    38859492057661                fraud_Lind-Buckridge  entertainment  ...  1325376051  43.150704 -112.154481        0
3   2019-01-01 00:01:16  3534093764340240  fraud_Kutch, Hermiston and Farrell  gas_transport  ...  1325376076  47.034331 -112.561071        0
4   2019-01-01 00:03:06   375534208663984                fraud_Keeling-Crist       misc_pos  ...  1325376186  38.674999  -78.632459        0
[5 rows x 22 columns]
Descripcion estadistica Train:
             cc_num          amt          zip          lat  ...     unix_time     merch_lat    merch_long      is_fraud
count  1.296675e+06  1.296675e+06  1.296675e+06  1.296675e+06  ...  1.296675e+06  1.296675e+06  1.296675e+06  1.296675e+06
mean   4.171920e+17  7.035104e+01  4.880067e+04  3.853762e+01  ...  1.349244e+09  3.853734e+01 -9.022646e+01  5.788652e-03
std    1.308806e+18  1.603160e+02  2.689322e+04  5.075808e+00  ...  1.284128e+07  5.109788e+00  1.377109e+01  7.586269e-02
min    6.041621e+10  1.000000e+00  1.257000e+03  2.002710e+01  ...  1.325376e+09  1.902779e+01 -1.666712e+02  0.000000e+00
25%    1.800429e+14  9.650000e+00  2.623700e+04  3.462050e+01  ...  1.338751e+09  3.473357e+01 -9.689728e+01  0.000000e+00
50%    3.521417e+15  4.752000e+01  4.817400e+04  3.935430e+01  ...  1.349250e+09  3.936568e+01 -8.743839e+01  0.000000e+00
75%    4.642255e+15  8.314000e+01  7.204200e+04  4.194040e+01  ...  1.359385e+09  4.195716e+01 -8.023680e+01  0.000000e+00
max    4.992346e+18  2.894890e+04  9.978300e+04  6.669330e+01  ...  1.371817e+09  6.751027e+01 -6.695090e+01  1.000000e+00
[8 rows x 10 columns]
Valores nulos por columnas Train:
trans_date_trans_time    0
cc_num                   0
merchant                 0
category                 0
amt                      0
first                    0
last                     0
gender                   0
street                   0
city                     0
state                    0
zip                      0
lat                      0
long                     0
city_pop                 0
job                      0
dob                      0
trans_num                0
unix_time                0
merch_lat                0
merch_long               0
is_fraud                 0
dtype: int64
Distribucion de 'is fraud' Train (0 = legitima, 1 = fraudulenta):
is_fraud
0    99.421135
1     0.578865
Name: proportion, dtype: float64
Distribucion de 'is_fraud' Test (0 = legitima, 1 = fraudulenta):
is_fraud
0    99.614014
1     0.385986
Name: proportion, dtype: float64

___________
Cantidad de valores atípicos detectados por variable numerica Train:
cc_num        118789
```

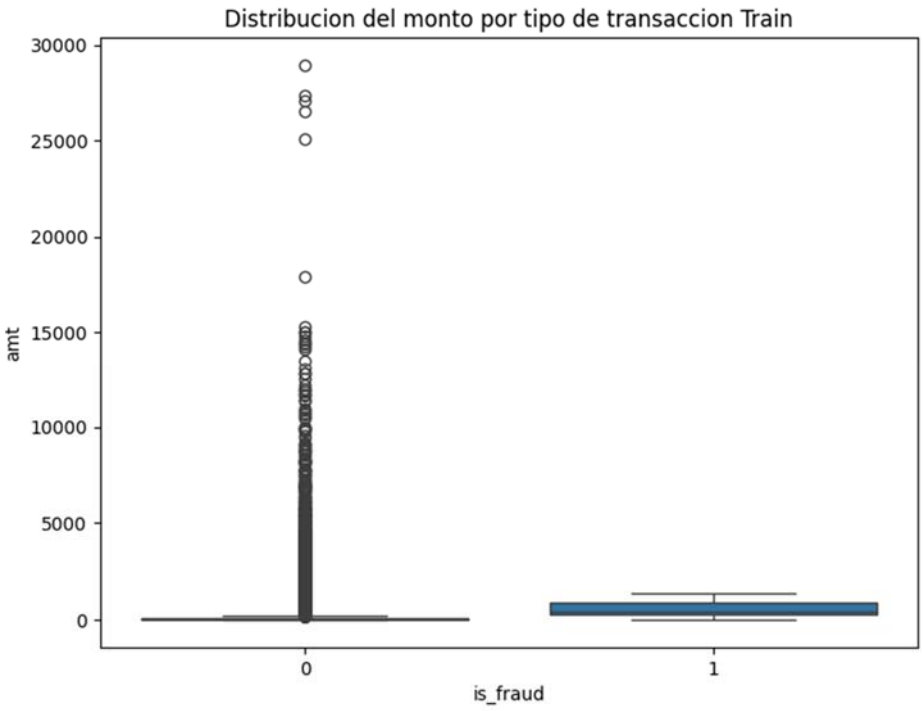
```
amt            67290
lat             4679
long           49922
city_pop      242674
merch_lat       4967
merch_long     41994
is_fraud        7506
dtype: int64
```

Distribucion gral de las variables Train









B-Preprocesamiento y balanceo de datos

```
Train luego de eliminar duplicados: (1296675, 22)
Test luego de eliminar duplicados: (555719, 22)
Calculando monto promedio 24h por tarjeta:
100%|██████████████████████████████████████████████████████████| 
983/983 [00:02<00:00, 395.70it/s]
```

```
Calculando monto promedio 24h por tarjeta:
100%|██████████████████████████████████████████████████████████|
924/924 [00:01<00:00, 495.05it/s]
Calculando cantidad 24h por tarjeta:
100%|██████████████████████████████████████████████████████████|
983/983 [00:04<00:00, 216.35it/s]
Calculando cantidad 24h por tarjeta:
100%|██████████████████████████████████████████████████████████|
924/924 [00:03<00:00, 255.54it/s]
Calculando cantidad 1h por tarjeta:
100%|██████████████████████████████████████████████████████████
| 983/983 [00:03<00:00, 322.62it/s]
Calculando cantidad 1h por tarjeta:
100%|██████████████████████████████████████████████████████████
| 924/924 [00:02<00:00, 322.54it/s]
Distribucion original de clases en el conjunto de entrenamiento:
is_fraud
0    1289169
1       7506
Name: count, dtype: int64
Distribucion despues de aplicar SMOTE:
is_fraud
0    1289169
1     128916
Name: count, dtype: int64

--------------------------------------
C-Entrenamiento inicial de modelos
--------------------------------------
Modelo 1: Regresion Logistica
Metricas de Regresion Logistica
              precision    recall  f1-score   support
           0       1.00      1.00      1.00    553574
           1       0.33      0.62      0.43      2145
    accuracy                           0.99    555719
   macro avg       0.66      0.81      0.71    555719
weighted avg       1.00      0.99      0.99    555719
AUC: 0.957345360691255
---------------------------
Modelo 2: Random Forest
Metricas de Random Forest
              precision    recall  f1-score   support
           0       1.00      1.00      1.00    553574
           1       0.80      0.86      0.83      2145
    accuracy                           1.00    555719
   macro avg       0.90      0.93      0.92    555719
weighted avg       1.00      1.00      1.00    555719
AUC: 0.9831450872117522
---------------------------
Modelo 3 - Isolation Forest (No supervisado)
Metricas Isolation Forest
              precision    recall  f1-score   support
           0       1.00      0.99      1.00    553574
           1       0.08      0.15      0.10      2145
    accuracy                           0.99    555719
   macro avg       0.54      0.57      0.55    555719
weighted avg       0.99      0.99      0.99    555719
AUC (aprox.): 0.570820083030194
Resumen comparativo de desempeño de modelos
```

```
                Modelo       AUC
1        Random Forest  0.983145
0  Regresion Logistica  0.957345
2     Isolation Forest  0.570820
```

D-Optimizacion de hiperparametros - Random Forest

Tamaño de muestra para optimizacion: (425426, 29)

Fitting 5 folds for each of 10 candidates, totalling 50 fits ***(por razon de espacio se muestran solo los primeros 10)***

```
[CV] END max_depth=10, max_features=log2, min_samples_leaf=5, min_samples_split=10, n_estimators=200; total time= 8.0min
[CV] END max_depth=10, max_features=log2, min_samples_leaf=5, min_samples_split=10, n_estimators=200; total time= 8.1min
[CV] END max_depth=10, max_features=log2, min_samples_leaf=5, min_samples_split=10, n_estimators=200; total time= 8.1min
[CV] END max_depth=15, max_features=log2, min_samples_leaf=5, min_samples_split=5, n_estimators=200; total time= 9.7min
[CV] END max_depth=15, max_features=log2, min_samples_leaf=5, min_samples_split=5, n_estimators=200; total time= 9.8min
[CV] END max_depth=15, max_features=log2, min_samples_leaf=5, min_samples_split=5, n_estimators=200; total time= 9.8min
[CV] END max_depth=15, max_features=log2, min_samples_leaf=5, min_samples_split=5, n_estimators=200; total time= 9.9min
[CV] END max_depth=15, max_features=log2, min_samples_leaf=5, min_samples_split=5, n_estimators=200; total time=10.3min
[CV] END max_depth=15, max_features=log2, min_samples_leaf=2, min_samples_split=5, n_estimators=100; total time= 5.5min
[CV] END max_depth=15, max_features=log2, min_samples_leaf=2, min_samples_split=5, n_estimators=100; total time= 5.5min
```

Mejores hiperparametros encontrados:

{'n_estimators': 200, 'min_samples_split': 5, 'min_samples_leaf': 2, 'max_features': 'sqrt', 'max_depth': 15}

Mejor AUC promedio en validacion cruzada: 0.9998

Validacion rapida del modelo optimizado (Random Forest)

AUC preliminar del modelo optimizado en test: 0.9848

Resumen comparativo entre modelo base y optimizado:

	Modelo	Precision (Clase 1)	Recall (Clase 1)	F1-Score (Clase 1)	AUC
0	Random Forest Base	0.8035	0.8615	0.8315	0.9831
1	Random Forest Optimizado	0.7281	0.8876	0.8000	0.9848

E-Evaluacion Final del Modelo Random Forest Optimizado

Accuracy: 0.9983

Precision: 0.7281

Recall: 0.8876

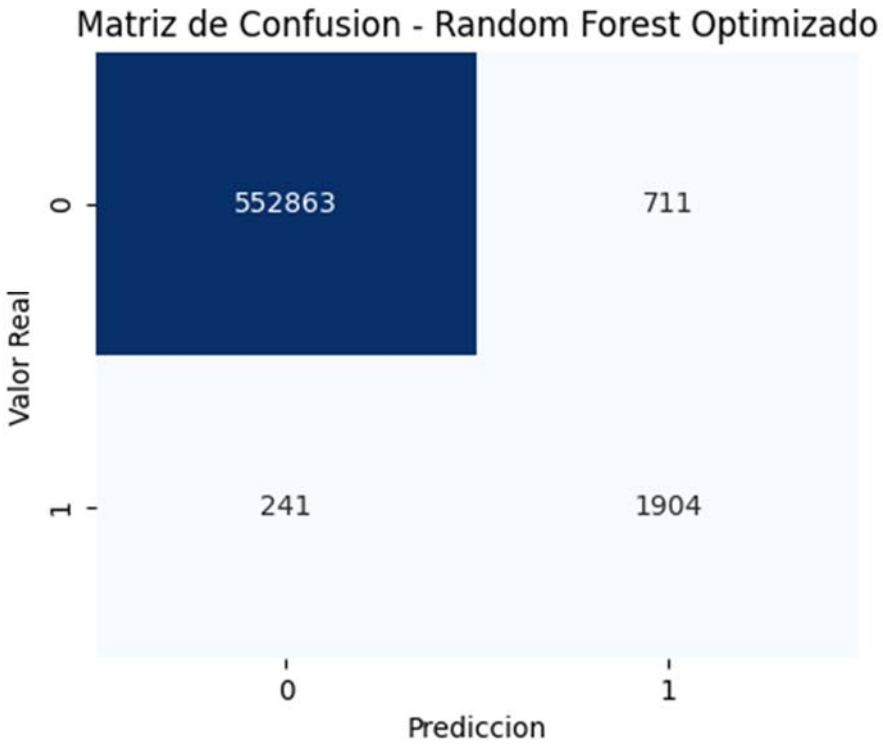
F1-Score: 0.8000

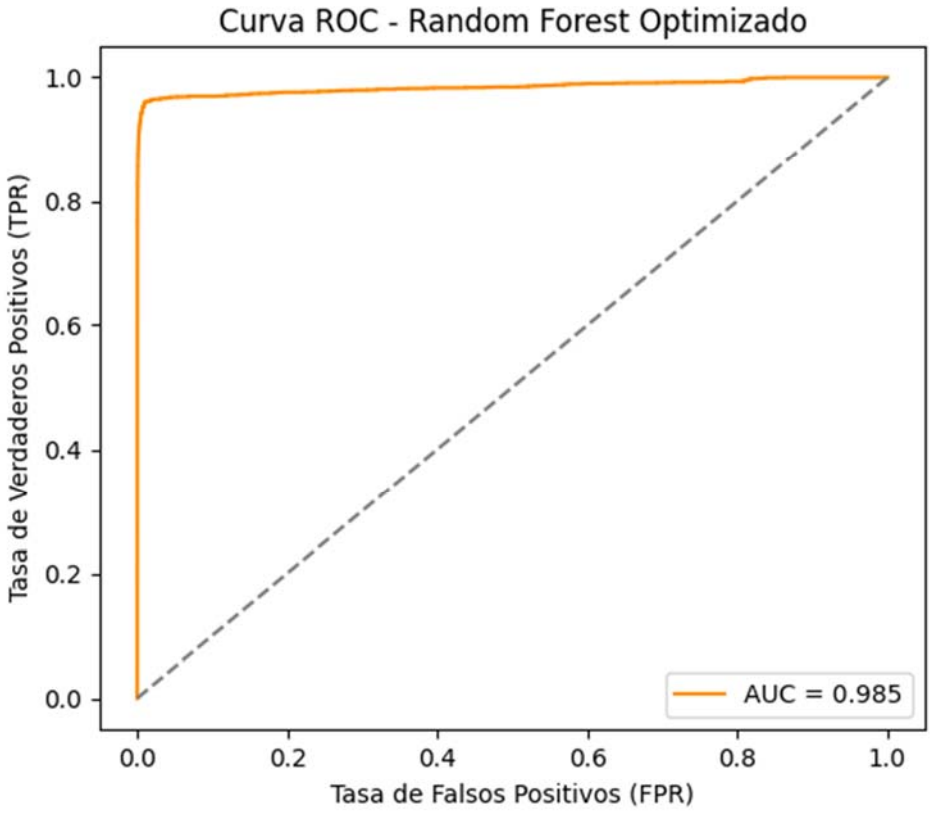
ROC-AUC: 0.9848

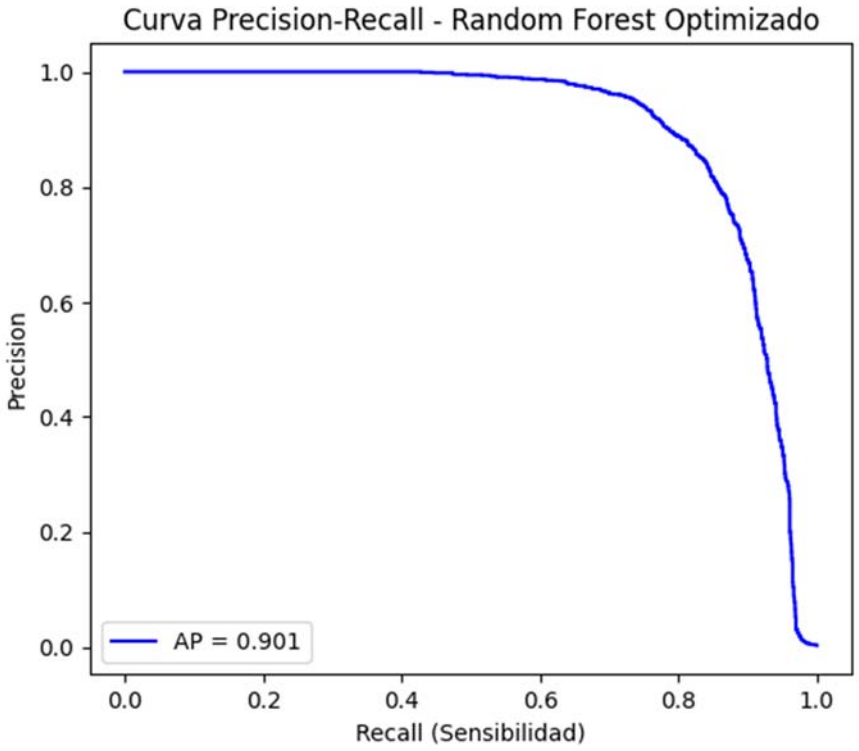
PR-AUC: 0.9010

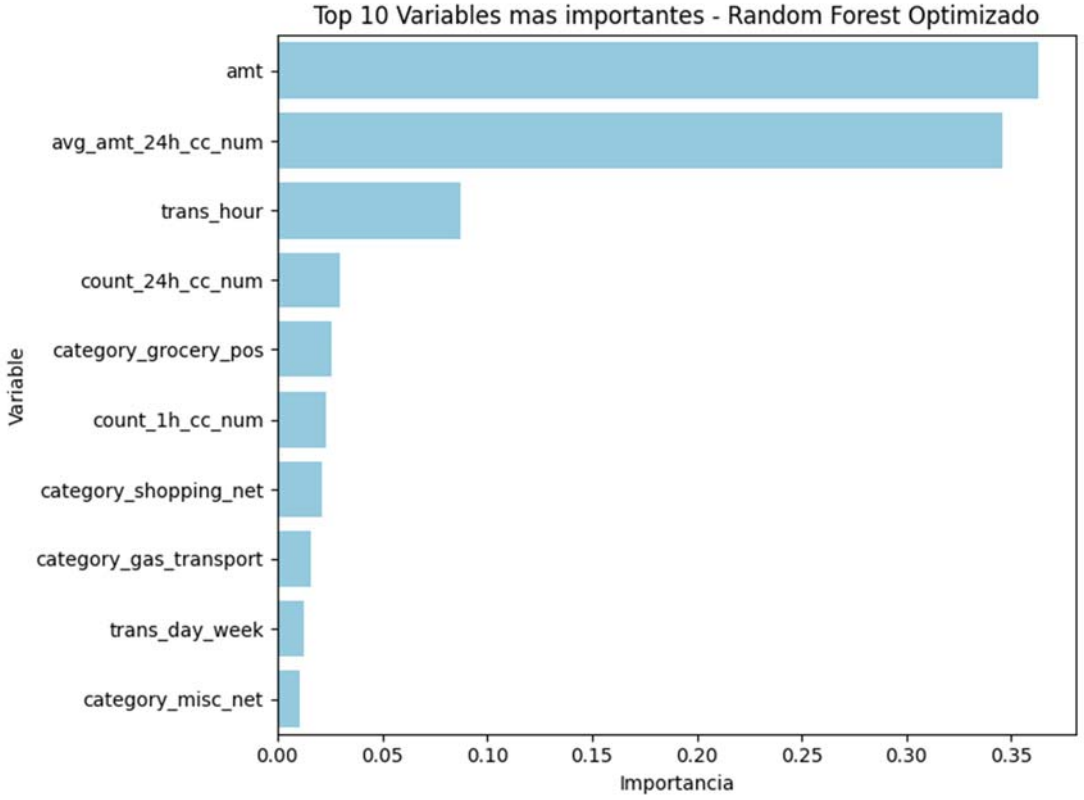
Reporte detallado de clasificacion:

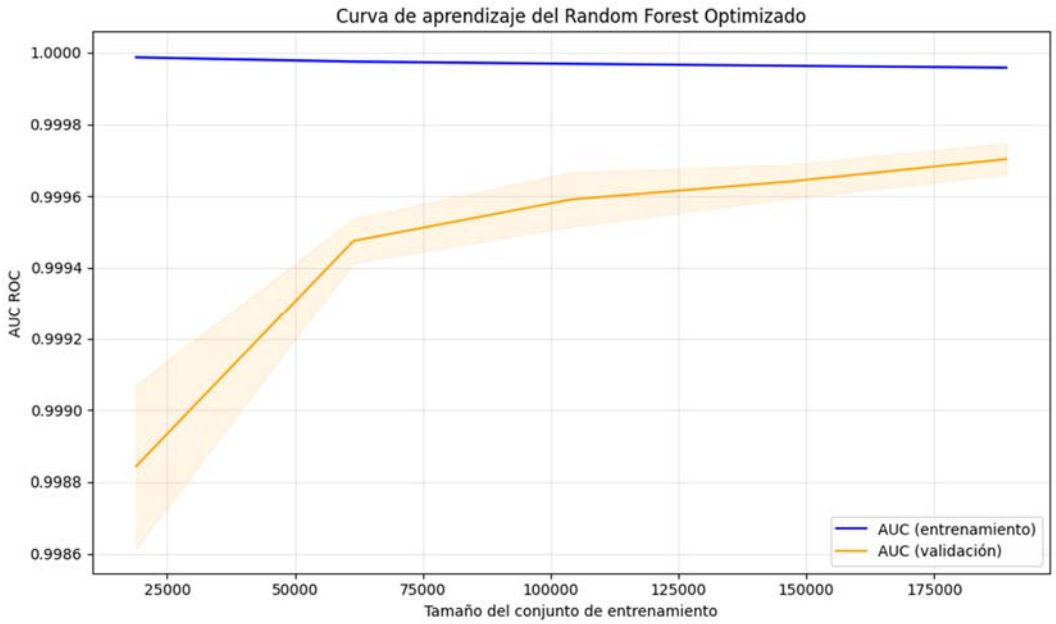
	precision	recall	f1-score	support
0	0.9996	0.9987	0.9991	553574
1	0.7281	0.8876	0.8000	2145
accuracy			0.9983	555719
macro avg	0.8638	0.9432	0.8996	555719
weighted avg	0.9985	0.9983	0.9984	555719


Matriz de Confusion - Random Forest Optimizado
552863
711
241
1904
Valor Real
0
1
0
1
Prediccion


Curva ROC - Random Forest Optimizado
Tasa de Verdaderos Positivos (TPR)
Tasa de Falsos Positivos (FPR)
AUC = 0.985
0.0
0.2
0.4
0.6
0.8
1.0


Curva Precision-Recall - Random Forest Optimizado
Precision
Recall (Sensibilidad)
AP = 0.901
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Top 10 Variables mas importantes - Random Forest Optimizado
Variable
amt
avg_amt_24h_cc_num
trans_hour
count_24h_cc_num
category_grocery_pos
count_1h_cc_num
category_shopping_net
category_gas_transport
trans_day_week
category_misc_net
Importancia
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35



Metricas finales guardadas en 'metricas_random_forest_final.csv'

CONCLUSIONES E INTERPRETACION GENERAL:
El modelo Random Forest optimizado muestra un rendimiento sólido, con un equilibrio adecuado entre precisión y sensibilidad.
El alto valor de ROC-AUC confirma su capacidad para distinguir operaciones legítimas de las fraudulentas.
La curva Precision-Recall demuestra que el modelo mantiene buena precisión incluso en un contexto de clases desbalanceadas.
Las variables con mayor importancia se relacionan principalmente con el monto, promedio diario de montos, hora del dia y cantidad de operaciones diarias, coherentes con patrones típicos de fraude.