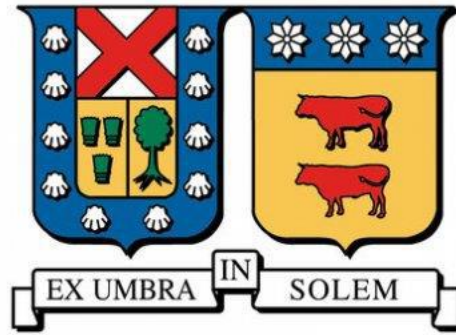




UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE INDUSTRIAS

**EVALUACIÓN COMPARATIVA DE MODELOS DE INTELIGENCIA
ARTIFICIAL PARA DETECCIÓN DE FRAUDE EN TRANSACCIONES
DIGITALES**

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL INDUSTRIAL

AUTOR

VICENTE ACUÑA

PROFESOR GUÍA

ESTEBAN SALGADO

PROFESOR CORREFERENTE

GONZALO AMÉSTICA

SANTIAGO DE CHILE, 2026



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo: Evaluación comparativa de modelos de inteligencia artificial para detección de fraude en transacciones digitales

Nombre del candidato(a): Vicente Patricio Acuña Olguin

Carrera / Grado: Ingeniería Civil Industrial

Campus: Vitacura

Departamento: Industrias

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Esteban Salgado, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por (**marcar una opción**):

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 21-09-2026

Firma:

Estudiante o Candidato(a):

Fecha: 15-09-2026

Firma:

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Tabla de Contenido

1. Resumen	4
2. Problema de investigación	6
3. Objetivos.....	9
3.1. Objetivo general.....	9
3.2. Objetivos Específicos.....	9
4. Alcance.....	10
5. Marco Teórico	11
5.1. Introducción a la inteligencia artificial.....	11
5.1.1. Inteligencia artificial en instituciones bancarias	13
5.2. Fraude financiero en transacciones digitales	16
5.2.1. Deepfake o suplantación de identidad con IA.....	19
5.3. Fraude en transacciones en Chile	20
5.4. Aportes de la inteligencia artificial en ciberseguridad y su aporte en la detección de fraudes	22
5.5. Modelos de Machine Learning	24
5.5.1. Aprendizaje Supervisado	25
5.5.2. Aprendizaje No supervisado.....	47
5.5.3. Redes neuronales y deep learning	50
5.6. Limitaciones de la inteligencia artificial en la detección de fraudes por reforzar	51
6. Metodología.....	54
6.1. Fases de la Metodología	57
6.2. Herramientas de desarrollo	59
7. Resultados	61
7.1. Análisis Exploratorio de Datos (EDA).....	61
7.1.1. Reducción de dimensionalidad por medio de PCA	66
7.1.2. Relaciones mediante gráficos y matriz de correlaciones de las principales variables numéricas	69
7.1.3. Relaciones mediante gráficos y matriz de correlaciones de las principales variables categóricas.....	78
7.2. Evaluación con modelos de Regresión Lineal	90

7.3.	Análisis de Clusters con K means	98
7.4.	Modelo de clasificación supervisada: Entrenamiento, testeo y resultados..	107
7.4.1.	Regresión Logística.....	107
7.4.2.	K-Nearest Neighbors (KNN).....	113
7.4.3.	Support Vector Machine (SVM).....	120
7.4.4.	Árboles de decisión	123
7.4.5.	Random Forest.....	129
7.4.6.	Naive Bayes	133
7.4.7.	Redes Neuronales (RRNN)	136
7.5.	Comparación y selección final de los modelos supervisados	144
8.	Conclusiones.....	153
9.	Discusión.....	159
10.	Limitaciones	162
11.	Anexos.....	165
12.	Referencias	181

1. Resumen

La presente investigación evalúa y compara distintos modelos de inteligencia artificial aplicados a la detección de fraude en transacciones digitales, utilizando el *dataset* **IEEE-CIS Fraud Detection**. Este conjunto de datos, construido a partir de transacciones reales de comercio electrónico proporcionadas por *Vesta Corporation*, permitió trabajar con un problema de alta complejidad, caracterizado por una gran cantidad de variables, presencia de valores faltantes, atributos enmascarados y un fuerte desbalance entre transacciones fraudulentas y no fraudulentas. A partir de este contexto, el estudio busca determinar qué enfoques de *machine learning* presentan mejor desempeño para identificar posibles fraudes y apoyar la prevención de riesgos en entornos transaccionales digitales.

La metodología se estructuró bajo el enfoque **CRISP-DM**, considerando etapas de comprensión del problema, exploración y preparación de datos, reducción de dimensionalidad, análisis no supervisado y entrenamiento de modelos predictivos. Durante el proceso se realizó limpieza del *dataset*, tratamiento de variables numéricas y categóricas, escalamiento, codificación *dummy*, aplicación de PCA y segmentación mediante *K-Means*. Posteriormente, se implementaron y compararon modelos lineales, distintos modelos de clasificación supervisada y una red neuronal desarrollada con *TensorFlow* y *Keras* en lenguaje de programación *Python*. La evaluación se realizó matrices de confusión y métricas como *accuracy*, *recall*, *precision*, *specificity*, *F1-score* y **ROC-AUC**, priorizando aquellas más adecuadas para bases desbalanceadas.

Los resultados muestran que los modelos más flexibles fueron los más efectivos para detectar fraude. En particular, la red neuronal obtuvo el mejor desempeño global sobre la

base completa, destacando por su mayor *recall*, **ROC-AUC** y equilibrio general entre métricas. *Random Forest* se posicionó como el mejor modelo tradicional, combinando buen rendimiento predictivo con mayor interpretabilidad a través de la importancia de variables. La investigación demuestra que la inteligencia artificial puede ser una herramienta útil para apoyar la detección temprana de fraudes digitales, siempre que los modelos sean evaluados con métricas adecuadas, datos representativos y una validación previa antes de su aplicación en contextos reales.

2. Problema de investigación

El aumento exponencial de las transacciones digitales desde la pandemia ha traído consigo un incremento en los intentos de fraude financiero, especialmente en pagos electrónicos y uso de tarjetas de crédito. Este fenómeno genera pérdidas millonarias en instituciones financieras y afecta la confianza de los usuarios en los sistemas digitales (Rayo, 2020). El *Nilsson Report* expone que: “En 2021, las pérdidas globales por fraude con tarjetas (crédito, débito, prepago) fueron de aproximadamente US\$32,34 mil millones.” (2022, pág.1).

Chile tampoco se queda atrás en esta problemática. Un ejemplo claro es el caso del Banco Estado, que sufrió una pérdida superior a los \$6.100 millones de pesos a causa de ex trabajadores del organismo entre 2021 y 2024. Este tipo de situaciones evidencia que el fraude financiero no solo proviene de agentes externos, sino que también puede originarse dentro de las propias instituciones. Debido a ello, la Comisión para el Mercado Financiero (CMF) ha advertido reiteradamente sobre el alza sostenida de fraudes digitales y la urgencia de implementar mecanismos más eficaces para identificarlos y prevenirlos a tiempo (Sanhueza, 2024).

En este escenario, la inteligencia artificial emerge como una alternativa relevante y necesaria en el ámbito financiero. Tal como señala la tesis de la investigadora Sheyla Nieves, “contribuye en gran medida a proteger la identidad de los usuarios y evitar la realización de transacciones fraudulentas recurriendo al empleo de técnicas de reconocimiento biométricas” (2019). De este modo, la IA se perfila como una herramienta prometedora para reforzar la seguridad en los movimientos bancarios y aplicaciones digitales, fortaleciendo la confianza de los usuarios en un entorno cada vez más expuesto a riesgos cibernéticos.

Así es como la mayoría de los bancos o instituciones financieras han incorporado IA en sus sistemas de gestión, donde son utilizadas en comercio algorítmico, automatización del flujo de trabajo, evaluación de desempeño de vendedores, entre otras (Downie y McGrath, 2025). Ahora bien, su uso se está enfocando en la ciberseguridad, específicamente en la detección de fraudes financieros como suplantación de identidad de un empleado bancario o la captura de información de tarjetas de pago (Alvarez, 2020).

Una forma de contrarrestar esto, es el uso de diferentes enfoques de entrenamiento en *machine learning* y programación de sistemas base para el desarrollo de IA, entre ellos el aprendizaje supervisado, no supervisado o por refuerzo, que permiten construir redes neuronales capaces de determinar si una transacción debe clasificarse como fraudulenta o legítima (Valor Añadido Fundaciones y Servicios, 2023). En esta línea, según diversos estudios recopilados en la tesis de Rayo, “la combinación de técnicas de aprendizaje automático y minería de datos pudo identificar las transacciones genuinas y no genuinas al aprender los patrones de los datos” (2020). Esto evidencia el impacto positivo de la automatización en la capacidad de detectar fraudes en las operaciones financieras.

Para el desarrollo de esta investigación se utilizará la base de datos **IEEE-CIS Fraud Detection**, publicada en la página de bases de datos *Kaggle* en el marco de una competencia organizada por la *IEEE Computational Intelligence Society* y construida con información aportada por *Vesta Corporation*, empresa norteamericana reconocida por su trabajo en soluciones de pago garantizado para comercio electrónico. La relevancia de esta base para el presente estudio radica en que proviene de transacciones reales de *e-commerce* y reúne información transaccional e identitaria en archivos complementarios, los cuales pueden integrarse mediante el identificador de transacción. Además, incorpora un amplio conjunto

de variables asociadas al monto, medio de pago, dispositivo, producto y otros atributos enmascarados por razones de privacidad, lo que la convierte en una fuente especialmente robusta para analizar distintos modelos orientados a clasificar operaciones legítimas y fraudulentas. En este sentido, su amplitud, complejidad y desbalance entre clases la vuelven particularmente pertinente para el problema abordado (IEEE Computational Intelligence Society (IEEE-CIS), 2019).

Ante lo señalado, cabe preguntarse ¿Cómo la inteligencia artificial puede adecuarse a las transacciones volátiles que realizan los usuarios por distintos medios de pago cada día? A partir de esto, ¿Es posible que un modelo de *machine learning* logre identificar un fraude o robo digital, considerando lo sutiles que pueden resultar? Y si es así, ¿Qué tan eficaces son los distintos algoritmos de estos modelos disponibles para detectar fraudes cuando se aplican a grandes volúmenes de datos como el presentado en **IEEE-CIS Fraud Detection**?

3. Objetivos

3.1. Objetivo general

Evaluar y comparar la aplicación de distintos modelos de inteligencia artificial en el procesamiento de transacciones digitales, a través del *dataset* **IEEE-CIS Fraud Detection** con el propósito de identificar posibles fraudes financieros y apoyar su prevención.

3.2. Objetivos Específicos

Identificar los modelos actuales existentes de *machine learning* en las transacciones financieras, a través de la información bancaria disponible.

Explorar, limpiar y preparar el *dataset* **IEEE-CIS Fraud Detection**, generando variables relevantes a nivel de cliente y banco, aplicando técnicas adecuadas para encontrar datos faltantes, codificación de variables categóricas y tratamiento del desbalance de clases.

Implementar y comparar una variedad de algoritmos de clasificación binaria y redes neuronales, utilizando modelos de aprendizaje no supervisado que agrupen los datos relevantes para el estudio y modelos de regresión como línea base comparativa.

Comparar el desempeño obtenido mediante índices de desempeño o métricas como *AUC*, *accuracy*, *specificity*, *precision*, *recall*, *F1-score* y matriz de confusión que midan la eficiencia de los algoritmos y las predicciones válidas.

Escoger los mecanismos que sean capaces de encontrar la mayor cantidad de transacciones clasificadas como fraudulentas, validando que las fallas encontradas se alejen del patrón normal y no sean errores propios de la ejecución del programa.

4. Alcance

El alcance de esta investigación radica en su pertinencia práctica a un problema vigente y de creciente relevancia social y económica. Al combinar técnicas de inteligencia artificial con bases de datos públicas y accesibles, el estudio se presenta como un proyecto viable de desarrollar en un tiempo razonable, sin depender de información confidencial ni de empresas específicas. Además, los resultados pueden orientar a escoger variables críticas en la selección de modelos más eficientes para reducir riesgos y fortalecer la confianza en los sistemas digitales.

5. Marco Teórico

Para entender el problema en profundidad y cómo se abordarán los objetivos, es necesario entender el impacto que ha tenido la inteligencia artificial en la actualidad y cómo se instala en el ámbito económico, específicamente temas de ciberseguridad.

5.1. Introducción a la inteligencia artificial

Primero que todo, se debe entender el concepto de inteligencia artificial (IA). De forma resumida, es un programa informático que busca desarrollar sistemas o máquinas capaces de realizar tareas que normalmente requieren inteligencia humana, como aprender, razonar, percibir, tomar decisiones y resolver problemas (Baroni, 2019, pág.7). Según un artículo de la revista médica *Elsevier*, Ávila et al. : “Se trata, por lo tanto, de diseñar herramientas informáticas que simulen procesos de inteligencia humana que incluyen el aprendizaje, el razonamiento y la autocorrección” (2020, pág.779). Por tanto, la IA funciona como un campo o conjunto de técnicas computacionales capaces de razonar y entregar resultados por medio del procesamiento inmediato.

Su expansión masiva sucede después de la pandemia, cuando se masifica en 2022 la IA generativa llamada *ChatGPT* disponible en el mercado, desarrollada por la empresa *OpenAI*. A pesar de las dudas iniciales de los usuarios y los errores que tenía el bot en preguntas o prompts más elaborados, la organización indica que existen 700 millones de usuarios activos semanales, donde sólo el 30% se utiliza para el uso de parte de empresas y el 70% personas que lo usan para resolver problemas no necesariamente relacionados con su trabajo, dado que lo usa una población más joven (OpenIA, 2025). La competencia no se podía quedar atrás, entonces se han ido desarrollando más modelos de IA por parte de otras organizaciones de software, como *Gemini* de *Google*, *Copilot* de *Microsoft* y más recientemente *DeepSeek*

desarrollado en China, que pueden ser igual o más eficientes que *ChatGPT*, dependiendo del *prompt* ejecutado.

Actualmente, bastantes industrias de diferentes sectores la utilizan generalmente en la automatización de procesos que antes eran más lentos de hacer de forma manual y más costosos al requerir a una persona que se encargaba de ellos. La siguiente tabla, realizada por la página de estadísticas *Vention*, presenta la tasa de adopción de IA por diferentes Industrias en este año:

Tabla 1

Tasa de adopción de IA por industria en 2024

	Recursos Humanos	Fabricación	Marketing y Ventas	Riesgo	Operaciones de servicio	Estrategia y finanzas corporativas
Servicios empresariales, legales y profesionales	11%	10%	9%	16%	20%	19%
Bienes de consumo/venta minorista	14%	4%	3%	15%	31%	29%
Servicios financieros	1%	8%	7%	17%	24%	23%
Atención sanitaria/farmacéutica	15%	7%	2%	22%	12%	8%
Alta tecnología/telecomunicaciones	6%	6%	4%	38%	21%	25%
Todas las industrias	11%	8%	5%	19%	19%	21%

Tomado de Vention, 2025.

Como se puede observar, los sectores con mayor tasa de adopción de implementar IA, son las telecomunicaciones en gestión de riesgo con un 38%, los bienes de consumo minoristas en operaciones con un 31% y los servicios financieros con un 23% en adopción. Esto refleja el impacto de este programa en servicios de tecnología y en la automatización de

procesos para empresas medianas o pequeñas que les ha servido para ahorrarse tiempo y dinero con la rapidez con la que trabaja el programa.

Por otro lado, se expone que en el ámbito financiero, sirve para generar pronósticos de los riesgos inmersos en las inversiones y proyectos, además de analizar resultados económicos del período en informes detallados para apoyar la toma de decisiones de largo plazo. Para la ciberseguridad, según *Vention*: “ Se prevé que el mercado de ciberseguridad con IA crezca de aproximadamente 24.000 millones de dólares en 2023 a unos 134.000 millones de dólares para 2030” (2025), dando indicios de una nueva era de protección de datos y detección de amenazas que resulta ser significativo en el sistema bancario.

5.1.1. Inteligencia artificial en instituciones bancarias

La inteligencia artificial puede resultar muy útil en el sector de las finanzas, pues aquí se caracteriza por procesos que dependen de usos intensivos de datos y de la toma de decisiones en tiempo real. Esta puede procesar grandes volúmenes de datos de forma clara y precisa, apoyando en áreas clave como eficiencia operativa, atención al cliente o gestión de riesgos (Downie y McGrath, 2025).

Existen muchas aplicaciones de este sistema dentro de los bancos, generando beneficios de distinta índole. A continuación, se enumeran los principales según la empresa de software IBM (Downie y Mcgrath, 2025):

- Trading algorítmico: Los algoritmos impulsados por IA son parte de los mercados financieros actuales, lo que permite el *trading* de alta frecuencia (HFT) y estrategias de inversión complejas. Estos sistemas pueden realizar análisis de

datos más complejos y revisar enormes conjuntos de datos para ejecutar operaciones a velocidades muy altas.

- Automatización de flujos de trabajo financieros: Las herramientas con IA permiten a las organizaciones manejar volúmenes de transacciones crecientes manteniendo la precisión y la consistencia. Estos avances mejoran la productividad y respaldan la escalabilidad de las operaciones financieras.
- Calificación crediticia y evaluación de riesgos: Se amplía el alcance de las decisiones crediticias al incorporar fuentes de datos alternativas, como los pagos de servicios públicos o la actividad en redes sociales. Este enfoque puede mejorar el acceso al crédito de las personas que carecen de historiales crediticios tradicionales.
- Servicio de atención al cliente: Los *chatbots* o asistentes virtuales programados con IA gestionan las consultas rutinarias de los clientes, proporcionando respuestas instantáneas y liberando a los agentes humanos para que se centren en cuestiones más complejas.
- Detección y prevención de fraudes: Los esfuerzos de detección del fraude utilizan algoritmos de *deep learning* y análisis predictivo para rastrear patrones de transacciones en tiempo real e identificar anomalías que puedan indicar actividades sospechosas. Los modelos de *machine learning* pueden adaptarse a nuevas tácticas de fraude, mejorando la precisión de la detección y reduciendo los falsos positivos.

También, se exponen los beneficios de usar IA generativa a partir de lo expuesto por EY Chile que utiliza un enfoque más sistémico, unificando funciones que involucran distintas soluciones de *software* (2025):

- Asociaciones para crear soluciones colaborativas: En IA generativa, asociarse para crear soluciones de forma colaborativa puede ser una forma más segura y rápida de garantizar un rendimiento y una protección eficaces contra las vulnerabilidades. Dos o más empresas pueden tener alianzas para compartir herramientas que les ha dado la IA y poder ayudarse mutuamente.
- Herramientas computacionales eficientes: Existen muchas herramientas que al utilizar este lenguaje mejoran el desarrollo de procesos y controlan la calidad de estos. La empresa EY ha desarrollado algunos, como el *Talk to data* que permite a los ejecutivos interactuar con *dashboards* utilizando lenguaje natural, el *EY Negocia* que analiza la situación financiera del cliente y crea planes de pago personalizados, ajustando la deuda en el tiempo, o el *Code Jumper Assessment* que analiza, comprende e interpreta información, acelerando el proceso de modernización del sistema a uno más completo y profundo.

En resumen, la inteligencia artificial está transformando la industria financiera al automatizar procesos y personalizar servicios. Sin embargo, su implementación requiere una gobernanza sólida y ética, respaldada por una gestión capacitada que pueda manejar adecuadamente la tecnología, minimizando el riesgo de errores del sistema o la exposición de información confidencial que podría constituir un delito (EY Chile, 2025). Además, es esencial contar con un presupuesto adecuado, ya que, aunque los programas de IA pueden

ser costosos tanto en su implementación como en su mantenimiento, su tasa de fallos es considerablemente menor en comparación con los sistemas tradicionales obsoletos.

5.2. Fraude financiero en transacciones digitales

El fraude es un delito que trata de engañar a las personas, haciéndoles creer que recibieron algo a cambio de un trabajo o trueque, pero en realidad no obtuvieron nada o algo muy distinto a lo que merecían. Similar a la estafa, según la RAE es una acción “contraria a la verdad y a la rectitud contra quien se comete”, ya que se perjudica a la persona o institución contra la que se realiza y se diferencia porque es un método complejo y planeado donde se obtiene una ventaja económica a través de manipulaciones que distorsionan la verdad y se saltan los controles establecidos para robar dinero de una contraparte (Rodríguez et al., 2020, pág. 260).

El impacto del fraude puede ser distinto para cada persona afectada. Según un estudio realizado por el periodista Vicente Rodríguez y otros de la revista *Elsevier* en España, que entrevistó a 32 personas afectadas por este delito, la mayoría de consecuencias son psicológicas, provocando ansiedad, alteraciones de hábitos saludables, consumo de drogas, depresión e intentos suicidas en los casos más graves en los que las víctimas quedaron casi en la ruina. Rodríguez expone que: “El fraude financiero no solo implica una pérdida económica, sino que también tiene consecuencias graves en la salud y el bienestar integral de las personas afectadas” (2020, pág. 272).

Esto demuestra la importancia de que los bancos inviertan en seguridad, para detectar el acto a tiempo y prevenirlo, antes de que lo ejecuten con éxito los culpables. No obstante, debido a los crecientes cambios tecnológicos y la astucia de los delincuentes, la

ciberseguridad resulta más importante en este tema, pues los sistemas computacionales se vuelven vulnerables y debe existir un *staff* de apoyo en caso de un ataque externo.

Es muy importante tener en cuenta el “Triángulo del fraude” que es un modelo desarrollado por el economista Donald Cressey que explica la razón de por qué una persona cometería un acto tan ruin como lo es el fraude. Existen tres vértices del triángulo que son: las oportunidades recibidas por los conocimientos del perpetrador, el motivo o estímulo para cometerlo (suele suceder cuando se tiene experiencia trabajando en el área económica) y la racionalización donde el sujeto se autojustifica a sí mismo que su actuar es válido y generará una recompensa al hacerlo con una gran falta de empatía (Rayo, 2020). La siguiente figura representa de forma más explícita la unión de estos componentes:

Figura 1

Triángulo del fraude



Nota: La figura representa los tres elementos que conforman el triángulo del fraude en cada uno de sus vértices. Tomado de Rayo, 2020.

Existen muchos individuos o bandas criminales en el mundo que son capaces de hackear o entrar a los sistemas bancarios para robar lo que más puedan para beneficio personal y

existen al día de hoy tipos de fraude con métodos sofisticados que son difíciles de encontrar para las autoridades (Alvarez, 2020).

Dentro de los fraudes más comunes y relevantes se encuentran (Rayo, 2020):

- **Phishing**

Consiste en el envío de una gran cantidad de correos electrónicos no deseados, a través de versiones falsas de sitios web, con la finalidad de convencer a las personas de que hagan clic en un enlace que los enviará a una página web falsa que robará su información personal. El autor se basa en una conferencia expuesta por Hussain Mahdi, M.D. et al. que menciona que: “Un tipo usual es la instalación de un software espía tipo *ransomware* que se instala al momento de hacer clic en el enlace, que cifra o bloquea el sistema del usuario y registrará todos sus datos bancarios cuando hackea la cuenta” (2010, pág. 234).

- **Fraude por teléfono**

Esta modalidad es utilizada por estafadores, sirve para conseguir información sensible a través de llamada telefónicas, haciéndose pasar por colaboradores o empleados de una institución financiera. Ellos le hacen creer al usuario que ganó u obtuvo un beneficio como un crédito, solicitando sus datos personales y de sus tarjetas de débito o crédito.

- **Regleta**

Es una modalidad de fraude que consiste en manipular los cajeros automáticos y colocar objetos similares a una regla en la zona donde se dispensa el dinero, a fin de que se impida la salida de este y la persona piense que se trata de una falla,

a raíz de obtener el dinero cuando esta se vaya del lugar. Algunos ladrones más valientes insertan una cámara escondida en el cajero para visualizar la contraseña de la tarjeta bancaria para crear una réplica con la misma y sacar el dinero de la cuenta asociada.

- **Cambiao**

Aquí se busca distraer la atención de la víctima con cualquier situación que capte su atención, a propósito de cambiar la tarjeta de crédito o débito por una de características similares. Técnica muy poco utilizada, ya que para esto tienen que sacar la billetera del usuario que es más probable de que se percate de la situación.

5.2.1. Deepfake o suplantación de identidad con IA

Como se puede observar, cada vez existen más maneras de robar o engañar a las personas para obtener la información de sus cuentas personales. Incluso, así como la tecnología ha ido en función de ayudar a la sociedad, los delincuentes le siguen la corriente, pero a su propio beneficio a costa del perjuicio de los demás.

Por ejemplo, se ha usado la inteligencia artificial en casos de robo y estafas organizadas, esto le sucedió al CEO de WPP, Mark Read, que detalló un intento de fraude en el que los criminales se hicieron pasar por él a partir de un número de *WhatsApp* con una imagen realista creada con IA. Ellos enviaban audios a los socios de la empresa y los clientes para organizar una reunión por *Microsoft Teams* en el que pedirían fondos para un nuevo proyecto que supuestamente iban a realizar pronto. Por suerte, lograron capturarlos a tiempo y descifrar el intento de *Deepfake* (suplantación de identidad) antes de que se llevara a cabo, Read dijo “Todos debemos estar atentos a las técnicas que van más allá del correo electrónico

para aprovecharse de las reuniones virtuales, la inteligencia artificial y los deepfakes” (Robins, 2024).

El *deepfake* es un problema muy importante al usar IA para hacerse pasar por otra persona en el que fácilmente se puede caer en un fraude si no se verifica la existencia de este que puede parecer muy real, pero es un personaje hecho con fines de manipular o obtener algún beneficio económico de la posible víctima inocente.

5.3. Fraude en transacciones en Chile

En Chile, a pesar de no tener delincuentes tan capacitados en este tipo de crímenes, igual existieron casos aislados de *Deepfakes* realizados con inteligencia artificial. El caso más reciente fue del entonces presidente, Gabriel Boric, en el que en un video falso recreado con IA incentiva a invertir en una supuesta plataforma digital de criptomonedas, asegurando “ganancias rápidas sin sustento real con poco riesgo” (Teletrece, 2025). Sin embargo, algunos usuarios que lo vieron encontraron falencias poco evidentes, como la gesticulación del rostro, tono de voz y pausas al hablar o inconsistencias en algunas capturas del video.

Así fue como descubrieron que se trataba de un *Deepfake* hecho con la intención de estafar a los chilenos que entraran a la página para quitarles su dinero o información personal, por lo que la brigada del cibercrimen se puso manos a la obra para encontrar a los implicados. Juan Valdenegro, comisario de la brigada comenta: “estos montajes buscan captar la atención de posibles víctimas que quieran invertir en sitios financieros fraudulentos” (2025).

La idea es diseñar un modelo de IA que sea capaz de evitar esta clase de delitos a través de un entrenamiento sistemático y progresivo que pueda detectar transacciones fraudulentas a tiempo y prevenirlas en su eliminación inmediata. Si se consideran los

deepfakes, se tendría que desarrollar un agente automatizado que pueda combatirlo al utilizar un lenguaje similar, acorde a los más altos estándares de ciberseguridad que se están desarrollando en los países desarrollados.

Otra situación similar sucedió en el BancoEstado, donde trabajadores de la propia compañía robaron más de \$6.100 millones de pesos chilenos provocados por empleados que se aprovecharon de una falla del sistema computacional. (Sanhueza, 2024). El hecho fue provocado por dos ex jefes del área de sistemas del banco y con apoyo de un funcionario de una empresa subcontratada “S2S”, lograron sobrepasar la ciberseguridad del banco con archivos dañados .txt para entrar en la banca principal donde se almacenaba el dinero.

Francisco del Pino, empleado que denunció el hecho y gracias al cual se descubrió a los culpables de esta pérdida menciona que ofrecieron: “Retirar dinero de cuentas (...) que estaban vulnerables” (2024). Así es como enviaban el dinero de forma automática con el archivo “inyector” que enviaba los fondos a una cuenta de la empresa subcontratada. Aunque intentaron remediar esto, tenían más cuentas por otros bancos por lo que se enviaban montos enormes, que llegaron hasta los \$500 millones en una transacción.

Estas incidencias evidencian las fallas del sistema bancario, especialmente cuando ocurren en instituciones como el BancoEstado, que deberían ser las más seguras del país. Por ello, es fundamental fortalecer la ciberseguridad en Chile, donde la inteligencia artificial puede desempeñar un papel clave: un gestor automatizado puede generar alertas con mayor rapidez que un analista, quien podría pasar horas revisando transacciones sin detectar fallas que un sistema bien entrenado identificaría de inmediato.

5.4. Aportes de la inteligencia artificial en ciberseguridad y su aporte en la detección de fraudes

La inteligencia artificial es crucial para la ciberseguridad. La ciberseguridad se define como la práctica de proteger a los aparatos informáticos o de cómputo como por ejemplo computadoras, servidores, teléfonos, etc. A través de técnicas que permitan resguardar la privacidad, integridad y disponibilidad de los ordenadores y los datos frente a vulnerabilidades de los sistemas en los que se puedan entrar agentes externos que los ataquen (Rayo, 2020).

Esto se vuelve sumamente relevante, pues cualquier persona con habilidades de informática o que sea capaz de hackear un sistema puede extraer información privada o perjudicar los servidores en los que se adentra a través de un virus para volverlos inservibles o tomar el control para su beneficio personal. En un banco los principales dispositivos afectados son los canales de transacción como cajeros automáticos, celulares, banca en línea y tarjetas de débito o crédito (Rayo, 2020). Al final perjudica a los agentes externos que serían los mismos usuarios del banco que son los que pierden su dinero, la privacidad de sus datos y la confianza que depositan en la institución, pues casi siempre le echan la culpa a esta entidad, terminando el caso en una demanda en la que la institución responde con una compensación monetaria a los afectados.

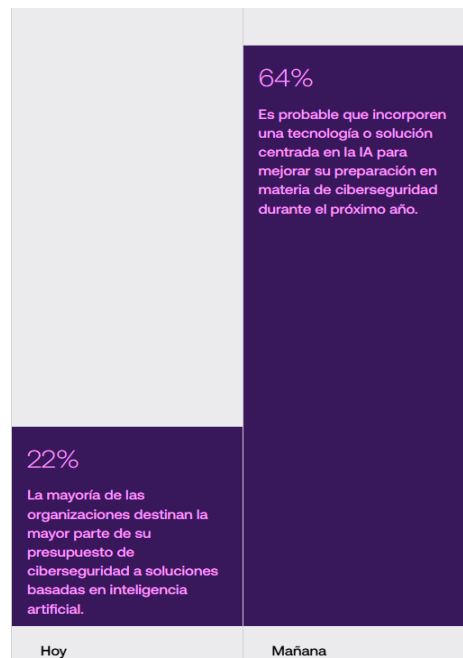
La inteligencia artificial, a pesar de ser utilizada a veces con fines delictivos, puede ser una herramienta crucial como método de defensa ante estos ataques, pues se implementa en los sistemas bancarios transaccionales y de confidencialidad de datos de los usuarios para contrarrestar los intentos de hackeo o dar alertas ante actitudes sospechosas en el sistema. Según la investigación de *Vention*:

Se espera que el mercado de la ciberseguridad basado en IA crezca de aproximadamente 24.000 millones de dólares en 2023 a cerca de 134.000 millones en 2030, lo que inaugura una nueva era de protección de datos, detección avanzada de amenazas y control de acceso de usuarios sin interrupciones (2025).

El siguiente gráfico hecho por la investigación, muestra cómo las empresas ya al día de hoy invierten en IA en materia de ciberseguridad para proteger sus núcleos operacionales, mientras que un mayor porcentaje tiene pensado proyectar un software de protección central con una IA como ejecutor fundamental.

Figura 2

Utilización de la IA en ciberseguridad al día de hoy y su proyección futura



Nota: El gráfico muestra cómo las empresas tienen una gran expectativa respecto a la IA en materia de soluciones de ciberseguridad del futuro, a raíz de su uso actual. Tomado de Vention, 2025.

Se prevé que un 64% de las empresas incorporen IA en sus sistemas de *software* más importantes de protección de bases y defensa contra virus o ataques de hackeo, ya que tiene

mayor capacidad de alerta y mejor predicción de posibles fraudes que se puedan generar. El 69% de las organizaciones lo considera una herramienta crucial en la prevención de delitos y un mejor arma al ser un refuerzo más concreto que los antivirus ya existentes, lo que provocará un cambio en los puestos de personal encargado de ciberseguridad (Vention, 2025).

5.5. Modelos de Machine Learning

Como se mencionó previamente, las inteligencias artificiales se desarrollan a partir de lenguajes de programación como *Python*, *R* o *MATLAB*, los cuales permiten implementar modelos matemáticos y econométricos orientados al aprendizaje automático (*Machine Learning*). Este campo se estructura principalmente en tres enfoques: aprendizaje supervisado, no supervisado y por refuerzo.

En el aprendizaje supervisado, los algoritmos se entrenan con datos previamente etiquetados, de modo que la máquina aprende a reconocer patrones y relaciones entre variables para predecir valores futuros con la menor desviación posible respecto al valor real observado. En cambio, el aprendizaje no supervisado no busca predecir una variable específica, sino identificar estructuras o agrupamientos ocultos dentro de los datos mediante técnicas como clustering o análisis de componentes principales, lo que permite descubrir nuevas relaciones y segmentaciones. Finalmente, el aprendizaje por refuerzo consiste en un proceso de entrenamiento basado en la retroalimentación continua: el sistema aprende a partir del ensayo y error, sin disponer de datos clasificados ni resultados esperados, optimizando su comportamiento mediante recompensas o penalizaciones hasta alcanzar una solución eficiente (Manrique, 2020).

El siguiente diagrama muestra las clasificaciones que existen entre los distintos tipos de entrenamientos, recalcar que el aprendizaje reforzado no se aborda porque no corresponde al tipo de problema supervisado y estático planteado en esta investigación.

Figura 3

Diagrama de los tipos de aprendizaje que existen y sus clasificaciones



Nota: El esquema expresa como en Machine Learning existen dos grupos importantes de aprendizaje y los modelos que tienen cada uno dependiendo de su función. Tomado de Rayo, 2020.

5.5.1. Aprendizaje Supervisado

Sirve para predecir una variable objetivo a partir del entrenamiento de variables predictoras relacionadas a esta. La idea de estos modelos matemáticos es utilizar el comportamiento pasado de las variables para que el resultado sea lo más semejante posible al valor real de la variable objetivo.

Existen dos tipos de variables a predecir, las numéricas que intentan predecir un número a partir de modelos econométricos de regresión o polinómicos, y las categóricas que predicen

variables de decisión, transformando las elecciones en dos valores, ceros y unos, para lo cuál se denominan variables dicotómicas o *dummies* (Rayo, 2020).

Primero, se describirán los modelos de regresión como los más simples de analizar por predecir a través de datos numéricos observables (Selvaraj, 2024).

- **Modelos de regresión lineal simple y múltiple**

Son modelos que analizan la tendencia de una variable dependiente Y a partir de una o varias variables predictoras X por medio de ecuaciones econométricas donde cada variable explicativa irá acompañada de un parámetro β que le otorga un peso especial a la variable sobre su capacidad predictiva. Una regresión simple utiliza sólo una variable exógena X, mientras que la múltiple utiliza dos o varias exógenas para obtener el valor de Y.

Regresión Simple:

$$Y = \beta_0 + \beta_1 \cdot X + \varepsilon$$

Regresión múltiple:

$$Y = \beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_2 + \dots + \varepsilon$$

El símbolo ε representa el error o incertidumbre que existe en el modelo respecto a la diferencia del valor predicho y el valor real medido. La idea es reducirlo al máximo, dado que un buen modelo es el que tiene mínima variabilidad de los datos entregados. Así es como nace la métrica del error cuadrático medio o MSE por sus siglas en inglés que promedia todos los errores obtenidos y los eleva al

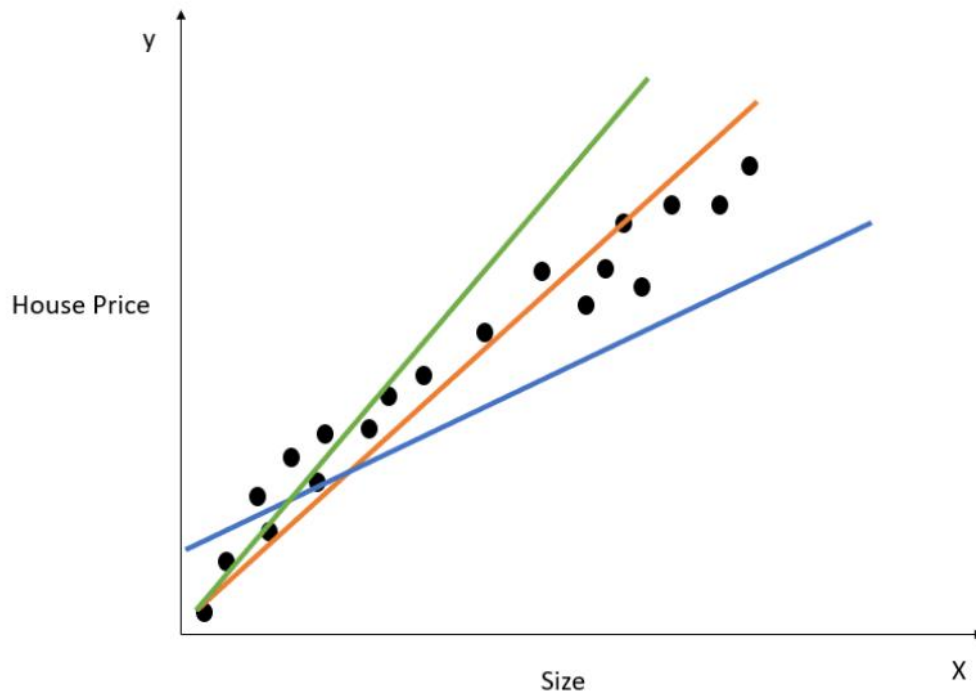
cuadrado para expresar qué tan grande es la varianza de las predicciones que se generan:

$$MSE = \frac{1}{N} \sum_{i=1}^N (Y_i - \hat{Y}_i)^2$$

De forma gráfica, una regresión lineal simple se representa así:

Figura 4

Ejemplo de una regresión lineal simple



Nota: El gráfico muestra tres regresiones lineales aplicadas al pronóstico del precio de una casa a partir de su tamaño. Tomado de Selvaraj, 2024.

Otra métrica muy interesante es el coeficiente de determinación o R^2 , ya que indica la proporción en que la variable Y es explicada por las variables X. Se le conoce como coeficiente de determinación entre la varianza de los datos estimados y la de

los datos observados que indica la porción de variabilidad determinada por el ajuste de la regresión. Se obtiene por softwares computacionales como la parte de análisis de datos de *Excel* o aplicando librerías de *Python* o *R* que lo calculan (Reguera, 2021).

- **Modelos de regresión no lineal y polinómicos. Regularización por Overfitting.**

Como su nombre lo indica, estos modelos utilizan ecuaciones con funciones más complejas como logaritmo, exponencial o son una transformación del modelo lineal al aplicar polinomios de distinto grado a las variables independientes. Los siguientes son ejemplos de cada una (Sarker, 2021):

- Regresión polinómica:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \dots + \beta_p X_1^n + \varepsilon$$

- Regresión logarítmica:

$$\log Y = \beta_0 + \beta_1 \log X_1 + \beta_2 \log X_2 + \dots + \varepsilon$$

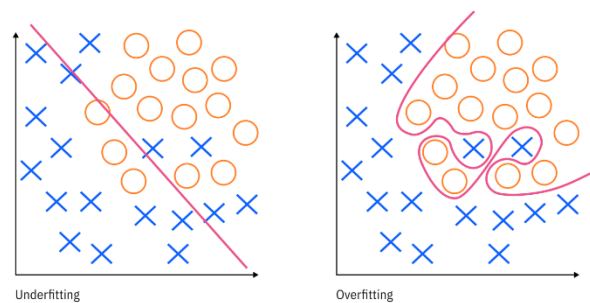
Pueden existir muchas funciones o combinaciones de estas dentro de una regresión no lineal, pero se pueden modificar a una lineal si se aplican ciertas transformaciones o cambios en las variables en las que se pueda trabajar con una regresión más simple y obtener los mismos resultados que si se hubiera trabajado sin estas.

Por otro lado, al predecir los modelos pueden suceder tres casos de análisis de resultados dependiendo del entrenamiento al que fueron sometidos: *underfitting* que es cuando los modelos tienen muy poco tratamiento de datos porque los resultados son demasiado sesgados, tienen alta variabilidad y mal ajuste en *training*, entregando

datos aleatorios; el *overfitting* que ocurre al sobre entrenar los datos y el modelo está tan sobrecargado de información que identifica patrones hacia donde va la mayoría de datos, no predice los valores atípicos que son los que realmente importan y existe una gran varianza por los errores que comete al no reconocer los comportamientos con los que debería trabajar; y el *right fitting* que sucede al existir un equilibrio entre el entrenamiento del modelo y el testeo con los valores que se quiere predecir, hay un límite de tratamiento de datos y la varianza es mínima al igual que el sesgo de los valores (Mucci, 2025).

Figura 5

Ejemplo gráfico de underfitting y overfitting en el entrenamiento de modelos



Nota: El gráfico de la izquierda muestra un modelo subajustado con underfitting y el de la derecha un modelo saturado con overfitting. Tomado de Mucci, 2025.

Debido al sobreajuste del *overfitting* y como es el aprendizaje más peligroso al que se puede entrar, existen métodos de regularización que agregan un límite al modelo a través de un hiper parámetro (parámetro definido por el mismo usuario) λ de penalización que reduce el tamaño de los estimadores del modelo β si generan un error muy grande dentro del modelo. Existen tres tipos de regresiones con este

agregado: *Ridge* que penaliza con los estimadores al cuadrado, *Lasso* que lo hace con su valor absoluto y *Elastic Net* que es una ponderación de las anteriores a partir de otro hiper parámetro α (Selvaraj, 2024). El siguiente ejemplo muestra el funcionamiento del estimador por *Elastic Net* que combina ambas regularizaciones:

$$\widehat{\beta}_{NET} = MIN\left\{\sum_{i=1}^n (Y - (\beta_0 + \sum_{j=1}^p \beta_j X_{ij}))^2 + \lambda\left(\alpha \sum_{j=1}^p |\beta_j| (Lasso) + (1 - \alpha) \sum_{j=1}^p \beta_j^2 (Ridge)\right)\right\}$$

▪ Modelos de clasificación

En este apartado está la otra sección del aprendizaje supervisado que es cuando se quiere predecir valores de decisión para dos categorías particulares en las que puede pertenecer un objeto, a partir de sus características y atributos. El modelo transforma las categorías en ceros y unos para poder trabajar la clasificación, es decir, variables *dummies*.

Una manera de medir el desempeño de los modelos es a partir de una matriz de confusión que indica a través de cuatro clases, que tan erradas fueron las predicciones de la clasificación respecto a si acertó en la respuesta el programa con el valor real o si se equivocó, dependiendo si era negativa o positiva la categoría a evaluar. Las clases son (Rayo, 2020):

- **Verdaderos Positivos:** Mide si un resultado fue etiquetado correctamente como positivo.

- **Verdadero Negativo:** Mide si un resultado fue etiquetado correctamente como negativo.
- **Falso Positivo:** Mide si el resultado fue etiquetado incorrectamente como positivo.
- **Falso Negativo:** Mide si el resultado fue etiquetado incorrectamente como negativo.

Tabla 2

Matriz de confusión de un modelo de clasificación

MATRIZ DE CONFUSIÓN		Predicción	
		Positivos	Negativos
Observación	Positivos	Verdadero Positivo	Falso Negativo
	Negativos	Falso Positivo	Verdadero Negativo

Nota: La matriz representa las cuatro categorías de resultados a partir del eje vertical de la predicción hecha por el computador y las observaciones que realmente se obtuvieron. Tomado de Rayo, 2020.

Por otro lado, aparecen 5 indicadores de desempeño que se logran por las distribuciones que se obtienen de la misma matriz. Las fórmulas de cada uno son las siguientes:

✓ **Accuracy (Exactitud):**

$$\frac{VP + VN}{P + N}$$

Mide el desempeño global del modelo al evaluar los casos que se clasificaron correctamente.

✓ **Sensitivity (Recall):**

$$\frac{VP}{VP + FN}$$

Mide porcentaje de positivos que se etiquetaron efectivamente como positivos.

✓ **Specificity (Especificidad):**

$$\frac{VN}{VN + FP}$$

Mide porcentaje de negativos que se etiquetaron efectivamente como negativos.

✓ **Precision (Precisión):**

$$\frac{VP}{VP + FP}$$

Mide el porcentaje de positivos que realmente fueron positivos del total clasificado.

✓ **F-Score:**

$$2 \cdot \left(\frac{Prec \cdot Recall}{Prec + Recall} \right)$$

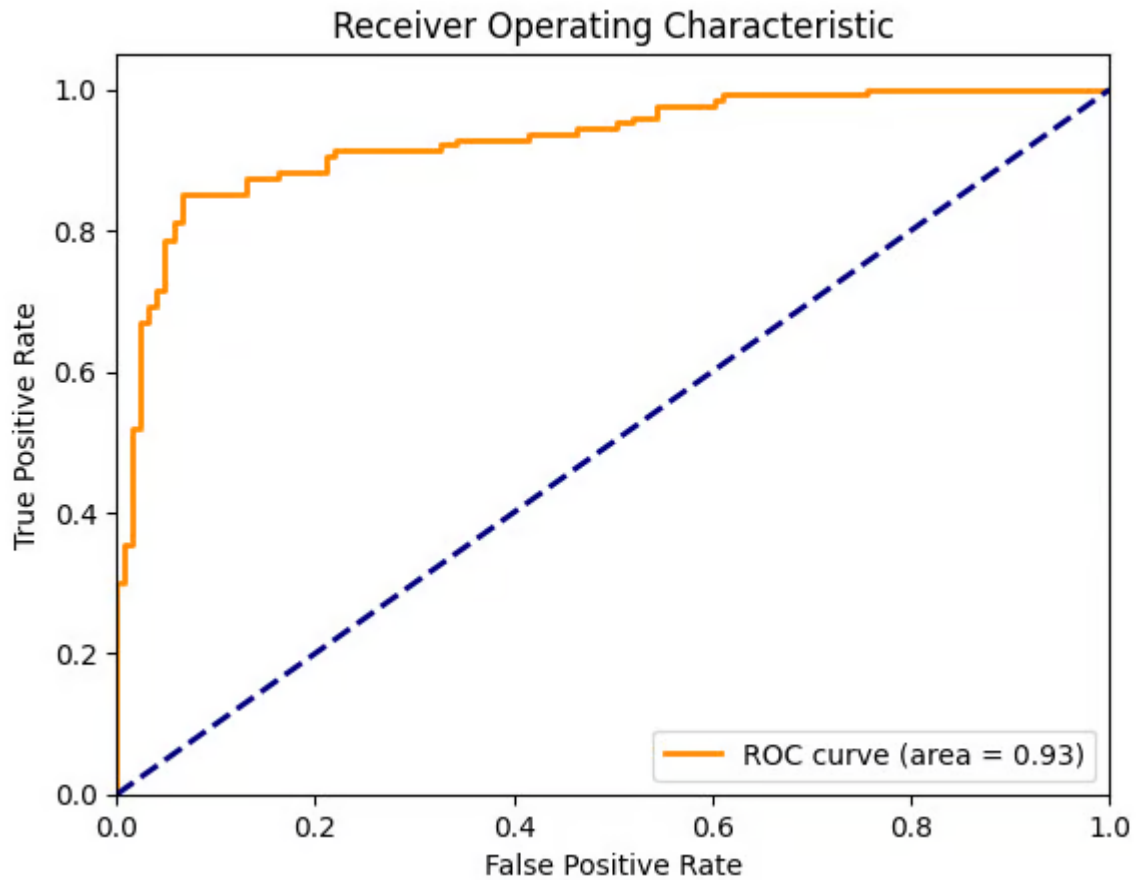
Utiliza los otros dos indicadores para comparar modelos distintos y revisar cuál es el que le atina mejor a la clase positiva.

Estos indicadores son muy relevantes, en especial en el tema que se está tratando, pues si se quieren observar transacciones fraudulentas se debe tener en cuenta que la tasa de verdaderos positivos sea la más grande posible para confirmar que el modelo efectivamente encontrará los movimientos sospechosos que se quiere alertar dentro de la base **IEEE-CIS Fraud Detection**. De esta forma, el *accuracy* no es el único indicador que se quiere tratar pues puede generalizar a todos los valores probados y dar interpretaciones que no serían las más óptimas en la elección del modelo.

Otra medida bien relevante para ver la eficiencia de la clasificación del modelo es utilizar la curva ROC y analizar su área bajo la curva (AUC), pues utilizan probabilidades de que la variable Y caiga en cierta categoría, a través de un gráfico de *Recall* vs $1 - \textit{Specificity}$ que identifica que tan bien predice el *software* tanto para verdaderos positivos como para verdaderos negativos (Chugh, 2024). Principalmente se utiliza un área del 50% con el fin de determinar la efectividad de la curva, de manera que si el AUC pasa esa barrera, el modelo está prediciendo de mejor manera y viceversa.

Figura 6

Curva ROC y medida del AUC



Nota: El gráfico muestra la curva ROC que tiene un modelo con un AUC del 93%, por lo que sobrepasa el umbral del 50% remarcado con la línea punteada. Tomado de Chugh, 2024.

▪ Regresión logística

El algoritmo de regresión logística es una función que utiliza una función logística y una función sigmoide para realizar una clasificación binaria basada en distintos factores del conjunto de datos. La función de manera matemática es la siguiente:

$$P(x) = \frac{1}{1 + e^{-zx}}$$

En el que z corresponde a una función de regresión lineal múltiple.

La función lo que intenta es encontrar la probabilidad de que la variable objetivo Y pertenezca a una categoría de las que se quiere evaluar y a partir de una función logarítmica poder generar el pronóstico de la categoría a evaluar.

A partir de la función, se obtiene un *odds* que es la manipulación de la función anteriormente descrita, que puede tomar cualquier valor en los R^+ , donde valores cercanos a cero y tendiendo a infinito indican muy baja y alta probabilidad de default, respectivamente. Se aplica una función de logaritmo a esta para poder tener una función *logit lineal* que será primordial para evaluar la función de costos del modelo y hacer la predicción adecuada a partir de la probabilidad.

Logit lineal:

$$Y = \ln\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 X$$

Así es como se obtiene una ecuación de regresión lineal en base a las probabilidades calculadas y se define una función de costos, llamada **Cross Entropy Loss Function**, que optimiza la regresión logística y en base a los resultados se define la predicción.

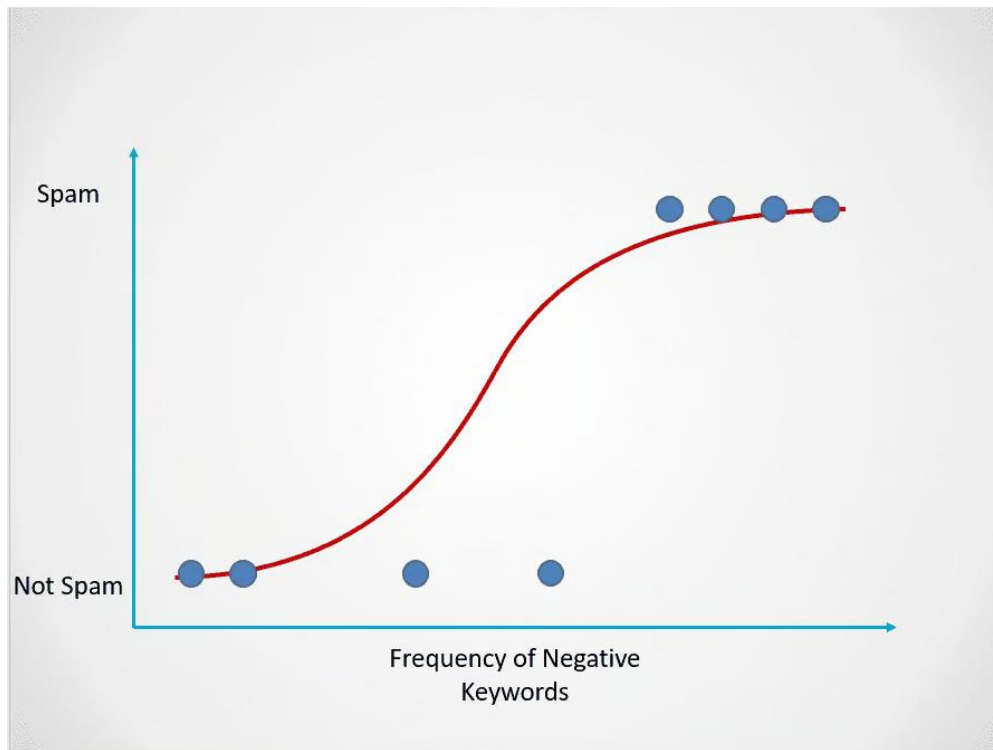
Cross Entropy Loss Function:

$$\text{Costo } (L) = -\frac{1}{k} \sum_{i=1}^k [Y_i \ln(\hat{Y}_i) + (1 - Y_i) \ln(1 - \hat{Y}_i)]$$

Con los resultados de la función va presentando si acertó o se equivocó el modelo. Por ejemplo, si una transacción no era fraude, pero el modelo indica fraude la función se irá al infinito positivo, pero en el caso de que fuera fraude resultaría cero, sucediendo algo similar si el modelo niega el fraude.

Figura 7

Función Sigmoide para la regresión logística



Tomado de Selvaraj, 2024.

- **KNN o K vecinos más cercanos**

KNN es un algoritmo que clasifica a partir de los valores de los datos más cercanos a él, es decir, supone que si la mayoría de puntos más próximos al que se quiere predecir poseen un valor, entonces, este también lo debería tener. Es un modelo no paramétrico, es decir, no asume la forma de la función, sino que la estima a partir de la distribución de los datos que la componen, por lo que necesita una gran cantidad de ellos y se va ajustando con mayor precisión dependiendo de su variabilidad.

El algoritmo funciona al tomar las distancias de todos los puntos K que estén más próximos a lo que se quiere clasificar y partir de ese radio de cobertura de datos revisa los

valores predominantes con el propósito de darle esa categoría al valor incógnito (Selvaraj, 2024).

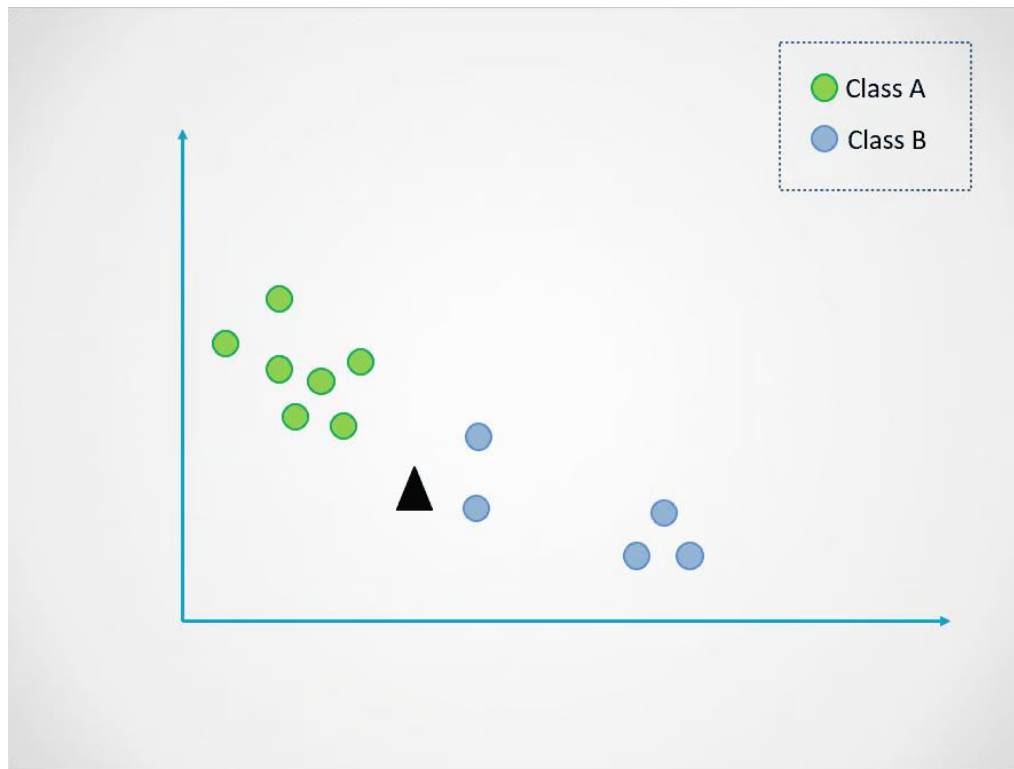
La función para definir distancias es la siguiente (Selvaraj, 2024):

$$p(y = j|X = x) = \frac{1}{K} \sum l(y_i = j)$$

Donde K es un hiperparámetro que define el usuario por la cantidad de “vecinos” a medir y l es la función de medir distancias a utilizar.

Figura 8

Funcionamiento de KNN



Nota: Se aprecia que el triángulo (valor a obtener) obtendrá la forma de las figuras más próximas a él. Tomado de Selvaraj, 2024.

El valor de K regula el *trade off* entre sesgo y varianza, ya que aumenta el sesgo al escoger un k más grande y limita el resultado hacia la mayoría, mientras que un menor K aumenta la varianza porque se toman menos datos y puede errar el modelo con la presencia de atípicos.

Para poder generar la predicción, se pueden tomar pesos uniformes de los puntos del vecindario, o bien, utilizar funciones de distancia entre los vecinos x al valor de y a predecir. La votación proporcional a la distancia se mide como (Beristain, 2025):

$$\hat{y} = \frac{1}{k} \sum_{i \in Vec(x)} \frac{y_i}{d(x, x_i)}$$

Existen tres tipos de distancias con las que se pueden generar las instancias: la Euclidiana, la *Manhattan* y la *Chebyshev*, las cuáles se definen en el código de *Python* al definir el modelo.

- **Distancia Euclidiana:**

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

- **Distancia Manhattan:**

$$d = \sum_{i=1}^m |x_i - y_i|$$

- **Distancia Chebyshev:**

$$d = \max_{i=1, \dots, m} \{|x_i - y_i|\}$$

Dentro de sus ventajas más importantes para el caso de estudio, es que sólo se necesita determinar el k y la medida de distancia, y es fácil de actualizar cuando el modelo está en producción. Sin embargo, dentro de sus limitaciones están que es sensible a valores atípicos,

datos mal etiquetados y atributos irrelevantes. Además contiene un alto costo computacional por los cálculos que necesita hacer de todos los datos, necesitando más tiempo de procesamiento.

▪ Árboles de decisión

Este modelo se basa en la ramificación de los puntos a evaluar, en el que en cada nodo se evalúa un criterio de desigualdad o separación para volver a ramificar el árbol y así segmentar los datos hasta llegar a un límite final que pueda utilizar ese grupo de ese nodo para hacer la predicción (Rayo, 2020).

Una desventaja es la “impureza” que es la mezcla de los grupos a evaluar en el que se tendrían que realizar infinitos cortes para poder separar los conjuntos. Para medirlo existe la entropía y el índice de *Gini* que la idea es minimizar estos valores para obtener un valor más óptimo (Rayo, 2020). Ambos utilizan la proporción de clase p_c que es la cantidad de puntos s que están dentro de una zona delimitada por el corte originado.

▪ Entropía:

$$Ent(s) = \sum_{c \in Clases(s)} -p_c \cdot \log_2(p_c)$$

Donde,

$$p_c = \frac{|\{x \in s, clase(x) = c\}|}{|s|}$$

▪ Índice de Gini:

$$Gini(s) = \sum_{c \in Clases(s)} p_c(1 - p_c) = 1 - \sum_{c \in Clases(s)} p_c^2$$

Ya con el valor de la impureza, se utiliza la ganancia de información o *Information Gain* por su traducción (Ganancia de *Gini* si se usa el índice) para medir qué tan grande es la entropía dentro del conjunto del corte. A más grande ganancia, significa una reducción significativa de entropía y el corte utilizado es más óptimo.

Si se define V como nodo padre con N datos, X al *feature*, s al corte del *feature* tal que $X < s$, entonces:

$$IG(X_j, s) = Ent(V) - \sum_{V_i} \frac{N_i}{N} \cdot Ent(V_i)$$

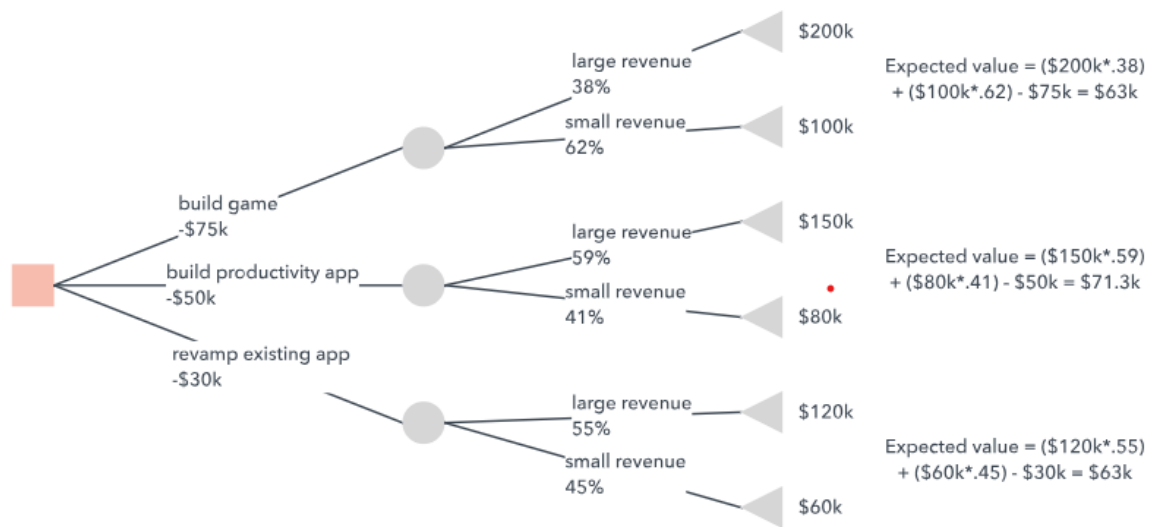
De donde,

$$V_i = \{X \in V | X_j \leq s\} \text{ o } \{X \in V | X_j > s\}$$

De igual manera, existen condiciones de *Stop* para generar un límite en las ramificaciones a realizar y evitar el *overfitting* por la gran cantidad de cortes que se realizan a los datos que aumentan la impureza. Estas se representan como hiperparámetros que se ajustan en el modelo de programación utilizado para ejecutar en la base de datos.

Figura 9

Diagrama esquemático de Árboles de decisión



Nota: El esquema muestra la separación de cada nodo en más a medida que se utilizan nuevos criterios para hacer la predicción. Tomado de LucidChart, s.f.

▪ Support Vector Machine (SVM)

Parecido a Árboles de decisión, pero ahora existe un hiperplano que separa a los distintos grupos que se quieren predecir, con la ayuda de vectores de soporte que delimitan al último punto más cercano al hiperplano con el mismo y así para el otro grupo (Rayo, 2020).

Hiperplano:

$$\beta_0 + \beta_1 \cdot x_1 + \beta_2 x_2 + \dots + \beta_p x_p = 0$$

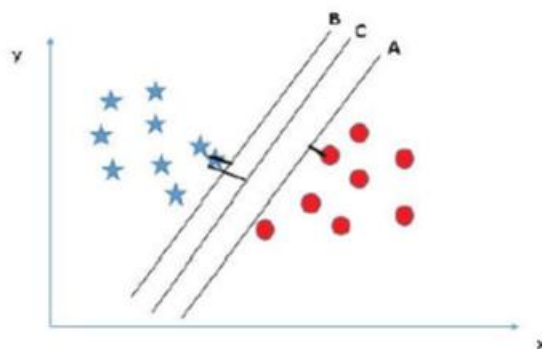
En el que los β son los coeficientes o pendientes de las rectas x y p es la dimensión en la que se encuentra el hiperplano.

Una desventaja es que tiene que aceptar cierto grado de error el modelo cuando están mezclados los valores de los grupos, pues se caería en *overfitting* al considerar *outliers* o valores atípicos del modelo, por tanto, se tienen que aceptar como ciertos en un grupo al que

no pertenecen con un **Support Vector Classifier**. Además, se tiene que cambiar la dimensión de un hiperplano en el que caso de que no se puedan separar los conjuntos de forma fácil, lo que se conoce como el *kernel trick* (Zhou y Kang, 2024).

Figura 10

Support Vector Machine de manera gráfica



Nota: El gráfico muestra como la separación de grupos se hace a distintos lados del hiperplano, con los vectores de soporte que los delimitan. Tomado de Rayo, 2024.

Este modelo es igual de útil que los anteriores al tener que definir pocos hiperparámetros para escoger los vectores que delimitarán el hiperplano. Genera una partición más simple cuando los grupos están bien definidos en la base de datos, por lo que tiene menor costo computacional y menos cálculos matemáticos. Además, el *kernel trick* sirve cuando se tiene muchas dimensiones el problema y puede almacenar outliers que en otra dimensión podían alterar resultados. Sin embargo, es sensible a la variabilidad y a puntos demasiado mezclados, por lo que debe existir un escalamiento previo de las variables para poder trabajarlo. Por otro lado, tener que usar el SVC puede alterar los resultados dependiendo del suavizamiento o *soft margin* con el que proceda el modelo, ya que muchos datos dispersos

en un grupo erróneo puede indicar una predicción equivocada que puede ser perjudicial en un tema tan delicado como transacciones bancarias.

▪ Naive Bayes

El modelo *Naive Bayes* corresponde a un método de clasificación supervisada sustentado en el teorema de *Bayes*. Su propósito es estimar la probabilidad posterior de que una observación pertenezca a una clase determinada, por ejemplo, fraude o no fraude, a partir de un conjunto de variables explicativas. Su formulación parte de la probabilidad condicional de la clase y dado el vector de atributos $(x_1, \dots, x_n)$, expresada como (Python, s.f.):

$$P(y | x_1, \dots, x_n) = \frac{P(y) P(x_1, \dots, x_n | y)}{P(x_1, \dots, x_n)}$$

La particularidad del modelo radica en el supuesto de independencia condicional entre las variables predictoras una vez conocida la clase, es decir, se asume que cada atributo aporta información de manera independiente al resto dentro de cada categoría. Bajo esa hipótesis, la expresión anterior se simplifica a (Universidad de Cambridge, 2009, pág. 258):

$$P(y | x_1, \dots, x_n) \propto P(y) \prod_{i=1}^n P(x_i | y)$$

En consecuencia, la regla de decisión del clasificador consiste en asignar la observación a la clase con mayor probabilidad posterior para el estimador de la variable predictora:

$$\hat{y} = \arg \max_y P(y) \prod_{i=1}^n P(x_i | y).$$

Esta estructura convierte a *Naive Bayes* en un modelo probabilístico simple, pero formalmente consistente, especialmente útil en tareas de clasificación binaria y multiclase cuando se requiere una solución computacionalmente eficiente y de implementación directa en entornos de análisis de datos.

Cuando las variables explicativas son continuas, una de las variantes más utilizadas es *Gaussian Naive Bayes*, la cual supone que cada atributo sigue una distribución normal dentro de cada clase. En ese caso, la probabilidad condicional de una variable x_i dado y se modela mediante la función de densidad gaussiana (Beristain, 2025):

$$P(x_i | y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right),$$

donde μ_y y σ_y^2 representan, respectivamente, la media y la varianza del atributo dentro de la clase y .

Esta formulación permite estimar probabilidades a partir de parámetros relativamente sencillos, lo que reduce el costo computacional del entrenamiento y facilita su aplicación sobre bases de datos amplias. Desde una perspectiva metodológica, ello constituye una ventaja relevante cuando se requiere establecer un modelo base para contrastar resultados con algoritmos más complejos. No obstante, su principal limitación proviene del propio supuesto de independencia condicional, ya que, en contextos reales, como las transacciones financieras, muchas variables suelen presentar relaciones entre sí.

▪ **Random Forest**

Es un modelo supervisado de tipo ensemble que construye un conjunto de árboles de decisión y combina sus predicciones con el fin de mejorar la capacidad predictiva y controlar el sobreajuste. La idea central del método consiste en generar múltiples árboles a partir de muestras aleatorias de los datos de entrenamiento y, adicionalmente, introducir aleatoriedad en la selección de variables candidatas en cada partición interna del árbol. Breiman define formalmente el bosque aleatorio como una “colección de clasificadores $\{h(x, \Theta_k)\}_{k=1}^K$,” donde cada árbol depende de un vector aleatorio Θ_k generado de manera independiente y con la misma distribución para todos los árboles del conjunto (Breiman, 2001). La predicción final para un problema de clasificación se obtiene mediante votación mayoritaria:

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

o de forma equivalente,

$$\hat{y} = \arg \max_c \sum_{k=1}^K I(h_k(x) = c),$$

donde $I(\cdot)$ es la función indicadora o indicatriz. Esta formulación permite que la decisión final no dependa de un único árbol, sino de una agregación de múltiples estructuras de decisión, lo cual aporta mayor estabilidad estadística al modelo.

En el proceso interno de construcción de cada árbol, las divisiones sucesivas se determinan a partir de medidas de impureza. Una de las más utilizadas en clasificación es el índice de *Gini* en el que el árbol selecciona la partición θ que minimiza la impureza ponderada de los nodos hijos (Python, s.f.):

$$G(Q_m, \theta) = \frac{n_m^{\text{left}}}{n_m} H(Q_m^{\text{left}}(\theta)) + \frac{n_m^{\text{right}}}{n_m} H(Q_m^{\text{right}}(\theta)).$$

De este modo, el algoritmo busca generar subconjuntos cada vez más homogéneos respecto de la variable objetivo. Adicionalmente, Breiman plantea el concepto de margen de clasificación como una forma de medir la solidez de la decisión del bosque (Breiman, 2001):

$$mg(X, Y) = \text{av}_k I(h_k(X) = Y) - \max_{j \neq Y} \text{av}_k I(h_k(X) = j),$$

de donde se deriva el error de generalización:

$$PE^* = P_{X,Y}(mg(X, Y) < 0).$$

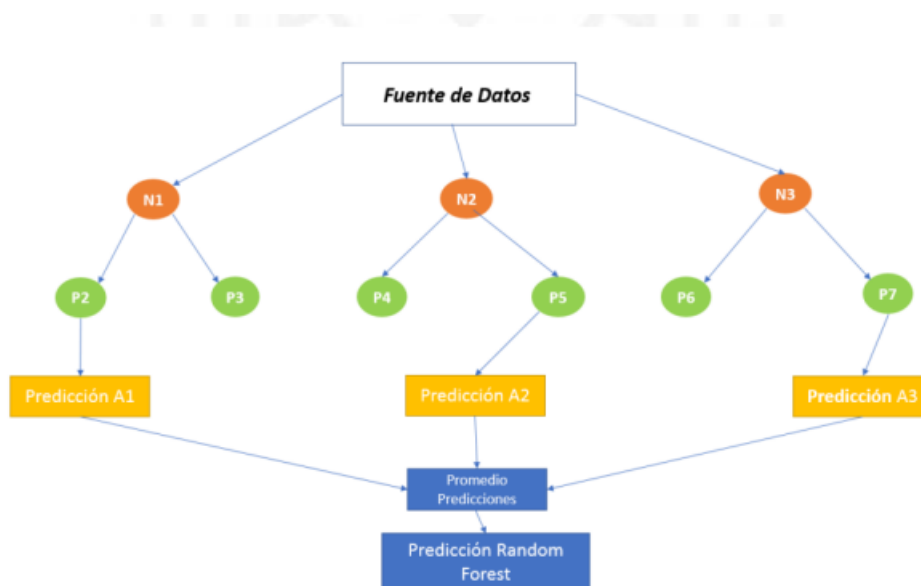
Estas expresiones reflejan que el método no solo persigue una clasificación correcta, sino también una clasificación robusta frente a nuevas observaciones no vistas durante el entrenamiento.

Desde el punto de vista aplicado, *Random Forest* presenta ventajas importantes para problemas de clasificación sobre datos tabulares, ya que admite relaciones no lineales, interacciones complejas entre variables y estructuras de datos donde un único árbol tendería a sobre ajustarse (Rayo, 2020). Asimismo, su desempeño suele ser sólido aun cuando el conjunto de datos contenga ruido o una gran cantidad de variables predictoras. Sin embargo, al tratarse de una agregación de numerosos árboles, su interpretación resulta menos

transparente que la de un árbol individual, lo que dificulta explicar de manera directa la lógica exacta de cada predicción. Además, el costo computacional y el uso de memoria aumentan a medida que se incrementa el número de árboles o la profundidad de estos (Beristain, 2025).

Figura 11

Diagrama de Random Forest



Nota: El diagrama explica el funcionamiento del modelo de Bosque aleatorio de forma gráfica. Tomado de Rayo, 2020.

5.5.2. Aprendizaje No supervisado

En esta sección ya no se trata de predecir variables o datos futuros, sino que el entrenamiento se genera para separar los grupos de datos en *clusters* por características que tienen en común o reducir la dimensionalidad de una base de datos demasiado grande que pueda tener un costo energético alto (Manrique, 2020).

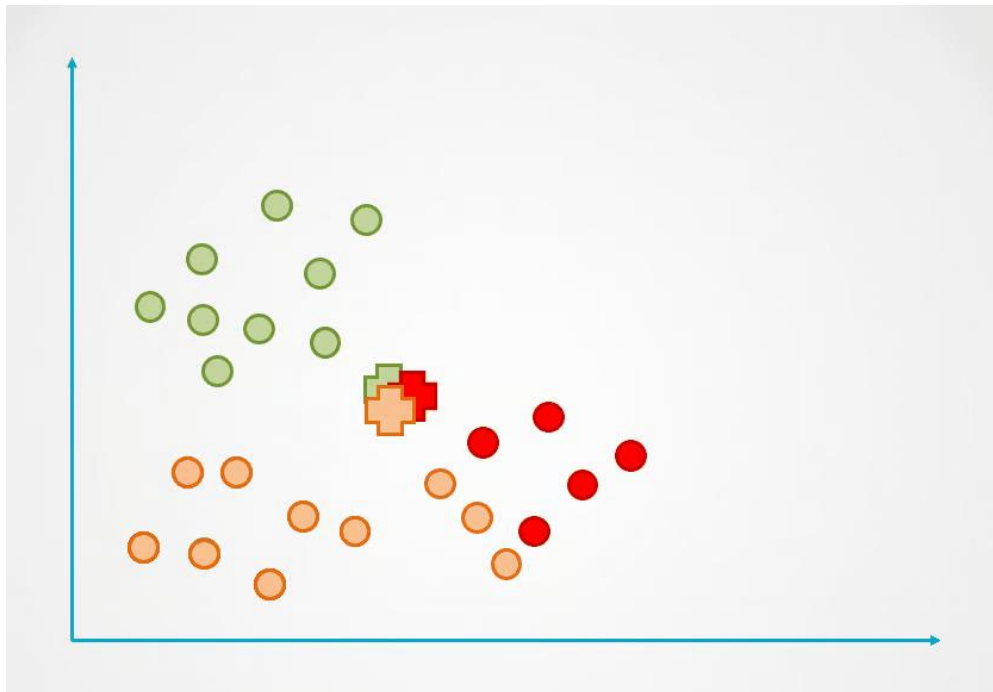
▪ Modelo de K Means

Este modelo agrupa las observaciones en un número predefinido de K *clusters* de forma que la suma de las varianzas sea la más baja posible. El método del codo o silueta promedio son aceptables para encontrar cuantos *clusters* o grupos tiene el modelo y así trabajar el algoritmo (Sarker, 2021, pág. 9).

El modelo lo que hace es medir los centroides de cada grupo con los datos que se van a posicionar de forma aleatoria hasta que la asignación sea correcta y exista una separación de grupos definida (Selvaraj, 2024).

Figura 12

Agrupación en K means



Nota: Se aprecia cómo los valores a asignar se distribuyen al centroide del grupo al que pertenecen. Tomado de Selvaraj, 2024.

▪ Reductores de dimensionalidad PCA

Este último modelo de aprendizaje no supervisado sirve para hacer más pequeño la complejidad de problemas que utilizan gigantes bases de datos y utilizan alto costo computacional.

Utiliza el método de Análisis de Componentes Principales (PCA por sus siglas en inglés) que reduce el número de variables que componen las dimensiones de un *DataFrame*. Esto a través de vectores propios que mantienen los resultados de una predicción, por ejemplo, pero reducen la dimensión del problema significativamente por la eliminación de información redundante o que tiene menos peso al momento de trabajar con los datos (Zhou y Kang, 2024).

Los componentes principales (Z_i) se tratan como combinaciones lineales de las variables exógenas con *loadings* (ϕ) que son las importancias de cada componente y ayudan a reconocer la información de cada uno, por ello, es de suma importancia que las variables estén escaladas para un mejor trabajo y conservar la varianza.

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \dots + \phi_{p1}X_p$$

Los *loadings* deben estar normalizados, por tanto:

$$\sum_{j=1}^p \phi_{j1}^2 = 1$$

No obstante, se necesita la varianza para saber los componentes a mantener y es muy sensible a los *outliers* de las tablas.

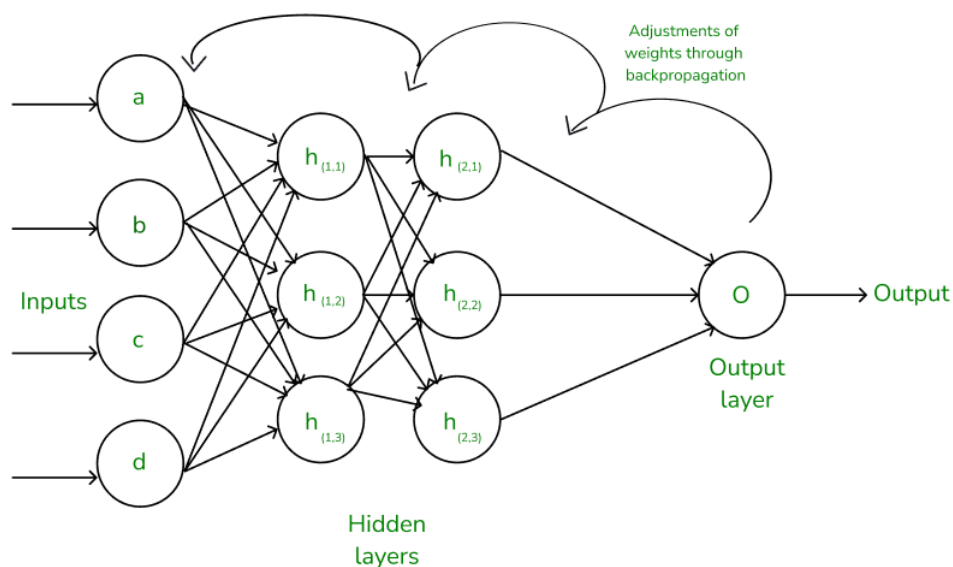
5.5.3. Redes neuronales y deep learning

Las redes neuronales artificiales imitan el comportamiento del cerebro humano mediante capas de neuronas interconectadas que transforman entradas en salidas. Cada neurona recibe valores ponderados, aplica una función de activación y transmite el resultado a la siguiente capa. Cuando se tiene una arquitectura de *deep learning*, es decir, múltiples capas ocultas, la red es capaz de capturar relaciones complejas y no lineales dentro de los datos (Schmidhuber, 2015, pág. 88). Este tipo de diseño estructural permite que los modelos aprendan representaciones jerárquicas de las características de los datos, lo que ha revolucionado aplicaciones en visión artificial, procesamiento del lenguaje y sistemas de recomendación (LeCun et al., 2015).

El proceso de entrenamiento de una red neuronal se basa en el método de *back-propagation*, que consiste en propagar hacia adelante los datos de entrada, calcular el error entre la salida predicha y el valor real, y luego retroceder ese error para ajustar los pesos y sesgos mediante un algoritmo de descenso del gradiente. Este ciclo se repite hasta que la función de pérdida se minimiza satisfactoriamente (Dávila, 2018). Las redes profundas tienden a exigir mayor poder de cómputo y estrategias de regularización para evitar el sobreajuste, lo que plantea tanto retos como oportunidades en su aplicación práctica (Goodfellow et al., 2016).

Figura 13

Funcionamiento de una red neuronal con backpropagation



Nota: El esquema expone cómo a través del descenso del gradiente, *backpropagation* propaga el error de una neurona a otra para minimizar las pérdidas. Tomado de Dávila, 2018.

En el ámbito de la detección de fraude financiero, las redes neuronales ofrecen ventajas significativas sobre los métodos tradicionales debido a su capacidad para procesar miles o millones de transacciones, identificar patrones atípicos y adaptarse a nuevas formas de delito que emergen con la evolución tecnológica. Sin embargo, también traen desafíos, tales como la replicabilidad del modelo (el “caja negra”), la necesidad de grandes volúmenes de datos etiquetados, y la posibilidad de sesgo en los resultados si los datos no están bien balanceados o representados (Shahrudnejad, 2021).

5.6. Limitaciones de la inteligencia artificial en la detección de fraudes por reforzar

A pesar de ser una herramienta poderosa para detectar fraudes, la IA aún tiene algunos obstáculos y fallas que debe mejorar para poder identificar al 100% todas las actividades sospechosas dentro del mercado financiero.

Primero que todo, se necesita trabajar con una gran cantidad de información o bases de datos para su correcto funcionamiento. Hay ocasiones en las que no se puede tener gran cantidad de información, dado que un banco pequeño puede trabajar con pocos clientes y esa información puede ser insuficiente para que la IA sea capaz de procesarla (Nieves, 2019).

Después, se necesita personal capacitado que sea capaz de trabajar con la IA, aprenda rápidamente su funcionamiento y sepa tomar acción ante las señales de alerta que genere el sistema (Alvarez, 2020). Actualmente, no existen muchas personas que conozcan estas tecnologías al ser avanzadas para la época, las universidades recién se están adecuando a ella y no existe el interés necesario de las instituciones o empleados de aplicar estos sistemas al ya estar acostumbrados a los obsoletos (Rayo, 2020).

Por otro lado, hay que saber escoger bien el modelo a utilizar, pues puede no ser el adecuado para detectar fraudes. Ya se tiene en cuenta que los mejores predictores son los de clasificación, pues una transacción fraudulenta se puede categorizar como una variable dicotómica transformada a números binarios con las repuestas ES o NO ES FRAUDE. Y como existen bastantes modelos capaces de trabajar con esta data, puede ser una tarea desafiante y compleja de llevar a cabo si no se realiza una evaluación antes (Alvarez, 2020). Por ello, el dataset **IEEE-CIS Fraud Detection** resulta ser un caso favorable para este análisis y representa variables confiables para generar una comparación adecuada.

Por último, se debe tener en cuenta que la implementación de soluciones de inteligencia artificial en las organizaciones en general acarrea costes significativos que van más allá del software y hardware inicial, incluyendo infraestructura especializada, adquisición y limpieza de datos, integración con sistemas heredados y mantenimiento continuo. Por ejemplo, las infraestructuras necesarias para entrenar modelos avanzados de

IA pueden representar entre el 47 % y el 67 % del presupuesto total de un proyecto (Vodworks, 2024). Este alto coste inicial y operativo puede hacer dudar a las instituciones bancarias sobre si aplicarla, ya que retrasa el retorno de la inversión, especialmente si la empresa aún no cuenta con capacidades de talento preparados, lo que puede traducirse en una ventaja competitiva reducida o en recursos financieros desviados hacia mantenimiento en lugar de innovación.

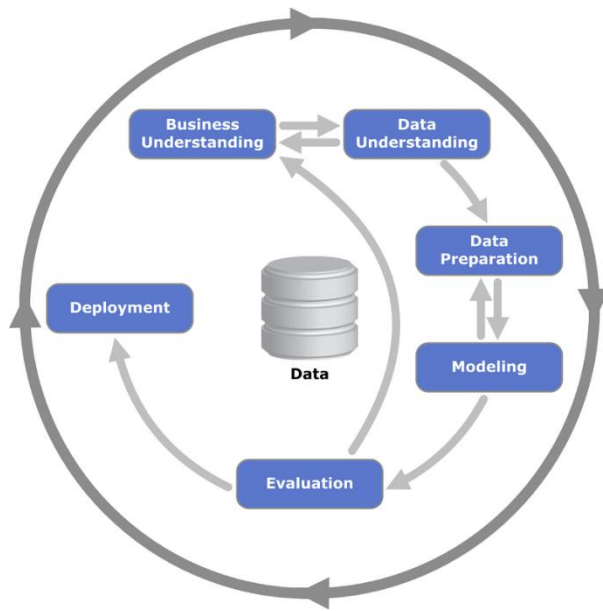
6. Metodología

Para estructurar el desarrollo metodológico de esta investigación se adopta el enfoque **CRISP-DM** (*Cross-Industry Standard Process for Data Mining*), ampliamente utilizado en estudios de ciencia de datos por su carácter sistemático y adaptable. Este marco propone un proceso iterativo compuesto por seis fases: comprensión del problema, comprensión de los datos, preparación de los datos, modelado, evaluación y despliegue. Su principal ventaja radica en que ofrece una guía clara para organizar el trabajo analítico, facilita la trazabilidad de las decisiones metodológicas y puede aplicarse en distintos dominios sin depender de una herramienta computacional específica (Beristain, 2025).

En la actualidad, CRISP-DM sigue siendo una de las metodologías más utilizadas tanto en el ámbito académico como en aplicaciones de la industria, debido a su capacidad de adaptarse a problemas contemporáneos de analítica avanzada, inteligencia artificial y aprendizaje automático. Entre sus principales aportes destacan la posibilidad de ordenar de manera sistemática cada etapa del análisis, facilitar la reproducción de los resultados obtenidos y mantener una conexión clara entre el desarrollo técnico del modelo y los objetivos prácticos del problema estudiado (Vallalta, s.f.).

Figura 14

Flujo del procesamiento de CRISP-DM



Nota: El diagrama representa como CRISP-DM es capaz de trabajar con bases de datos a través de sus distintas etapas para entregar una solución a problemas de negocios. Tomado de Vallalta s.f.

Este enfoque se utilizará para procesar los fraudes en la **IEEE-CIS Fraud Detection**, debido a su fácil adaptabilidad a diferentes conjuntos de datos y marca un protocolo estándar a trabajar en la aplicación de modelos, selección de los más adecuados y análisis de los resultados. Como se está trabajando con *Python* en un entorno *.ipynb* en el ambiente *Visual Code*, se pueden ir mostrando los pasos de a uno a medida que se analiza la fuente y se procesa la información.

El **IEEE-CIS Fraud Detection** es una gigantesca base de datos que se divide en dos archivos *.csv*, una de *Train_identity* para la información correspondiente al ID de las transacciones (no tan relevante por ser solo identificadores unidos), y una de *Train_Transaction* que muestra toda la información relevante respecto a las transacciones realizadas por los usuarios de los servicios de *Vesta* (IEEE Computational Intelligence Society (IEEE-CIS), 2019). Al unir las dos bases de datos, se tiene un conjunto de 590.000

filas con más de 400 columnas que presentan la información bancaria de la institución. Como estas fueron proporcionadas para una competencia del 2019, no todas las variables son identificables, pero dentro de las más relevantes que se pueden mencionar se encuentran (IEEE Computational Intelligence Society (IEEE-CIS), 2019):

- ✓ **TransactionDT:** Diferencia de tiempo a partir de una fecha y hora de referencia dada (no es una marca de tiempo real).
- ✓ **TransactionAmt:** Importe del pago de la transacción en USD.
- ✓ **ProductCD:** Código de producto, el producto para cada transacción.
- ✓ **card1 - card6:** Información de la tarjeta de pago, como tipo de tarjeta, categoría de tarjeta, banco emisor, país, etc.
- ✓ **Dirección:** Dirección de los usuarios de donde realizaron el pago.
- ✓ **Distancia:** Distancia del banco al usuario.
- ✓ **Dominio de correo electrónico P_ y (R__):** Dominio de correo electrónico del comprador y del destinatario
- ✓ **C1-C14:** Recuento, como por ejemplo cuántas direcciones se encuentran asociadas a la tarjeta de pago, etc. El significado real está enmascarado.
- ✓ **D1-D15:** Intervalo de tiempo, como días entre la transacción anterior, etc.
- ✓ **M1-M9:** Coincidencia, como nombres en la tarjeta y dirección, etc.
- ✓ **Vxxx:** Vesta desarrolló funciones avanzadas, que incluyen clasificación, conteo y otras relaciones entre entidades. Variables que sólo son numeradas y

por razones de confidencialidad de la empresa no se pueden saber su significado.

- ✓ **DeviceType:** Tipo de dispositivo.
- ✓ **DeviceInfo:** Información del dispositivo.
- ✓ **Id_12 – Id_38:** Identificador de atributos enmascarados por la compañía.

6.1. Fases de la Metodología

1. **Comprensión del negocio:** Contextualizar el negocio de las transacciones digitales en el sistema bancario actual internacional, resaltando la relevancia de la delincuencia a partir del fraude con métodos avanzados en tecnología. Definir los objetivos predictivos de la investigación como guía de la evaluación e identificar las necesidades y restricciones del negocio que condicionen el avance del trabajo como el tiempo de procesamiento de los códigos, errores en los resultados o complejidad de los datos de trabajo.
2. **Comprensión de los datos:** Aquí se empieza a trabajar con el *dataFrame* de **IEEE-CIS Fraud Detection** seleccionado para solucionar el problema del negocio, a través de la totalidad de las observaciones inmersas en él, se verifica el tipo y naturaleza de las variables en las columnas, junto con la distribución de la variable objetivo. Se realiza un Análisis Exploratorio de Datos (EDA) para limpiar el *dataset* por medio de eliminación de variables que contengan muchos valores nulos o constantes que no aporten significado al modelo, revisar relaciones entre conjuntos, encontrar *outliers* y analizar la dimensionalidad del problema.

- 3. Preparación de los datos:** Una vez depurado el *DataFrame*, se aplica un Análisis de Componentes Principales (PCA) con el objetivo de reducir la dimensionalidad del conjunto de datos y, con ello, disminuir el costo computacional asociado al procesamiento e interpretación de una base de gran tamaño. Para ello, las variables se separan en numéricas y categóricas, aplicando PCA únicamente sobre las variables numéricas y reteniendo los componentes necesarios para explicar al menos el 90% de la varianza acumulada. Posteriormente, las variables numéricas son escaladas y las variables categóricas se transforman mediante codificación adecuada, evitando la multicolinealidad asociada a la denominada “trampa de las variables *dummy*”. Finalmente, ambos grupos de variables se integran en un único conjunto de datos, el cual será utilizado en la etapa de entrenamiento y comparación de los modelos predictivos de fraude.
- 4. Entrenamiento y modelados:** Se entrenarán distintos modelos de aprendizaje supervisado, empezando con los de regresión para saber su impacto en los resultados de la predicción y evitar el *overfitting* con regularización. Por otro lado, Se realizará una división de *clusters* con *K-means* para encontrar los grupos con mayor varianza o sesgo que puedan alterar los resultados de los modelos. Luego, se entrenarán los modelos de clasificación para saber su destreza y rendimiento en la predicción del fraude a raíz de la información proporcionada. Los modelos utilizados en esta instancia serán: **Regresión Logística, Naive Bayes, KNN, Árboles de decisión, SVM y Random Forest** como modelo de ensamble que pueda resultar interesante su predicción. También se trabajará con redes neuronales para revisar la significancia del modelo y si los resultados son acordes a los parámetros dados.

5. **Evaluación del desempeño:** Se aplicará *train/test Split* para poder entrenar los distintos modelos de aprendizaje supervisado de clasificación y obtener estimaciones de rendimiento robustas de cada uno. Las métricas para utilizar serán especialmente sensibles al desbalance de clases: **accuracy, precisión, recall y F1-score, área bajo la curva ROC (AUC-ROC)** y matriz de confusión. Se priorizará la capacidad del modelo para detectar correctamente a las detecciones de fraude, maximizando el *recall* y observando a detalle los que tengan mayor AUC.
6. **Interpretabilidad de los resultados y despliegue:** Ya con los modelos entrenados y evaluados, se seleccionarán los que con mayor desempeño funcionaron y menos se demoraron en arrojar resultados consecuentes con la información proporcionada. Se cotejará si las predicciones son significativas con las variables seleccionadas y explicará el comportamiento del fraude a partir de los datos sospechosos que pueda tener una transacción, facilitando la decisión de detener un pago desde una perspectiva estratégica para prevenir una estafa.

6.2. Herramientas de desarrollo

El desarrollo del estudio se realizará en lenguaje de programación **Python**, debido a su amplia adopción en la comunidad de ciencia de datos y su ecosistema robusto de librerías orientadas a análisis estadístico, modelamiento predictivo y visualización de datos, facilitando el uso, incluso, de redes neuronales que son las más complejas en el desarrollo de su proceso interno.

El análisis será implementado en el entorno *Jupyter Notebook*, mediante *VS Code* utilizando las siguientes librerías:

- **Pandas y NumPy** para manipulación y análisis de datos.

- **Matplotlib y Seaborn** para visualización exploratoria.
- **Scikit-learn** para modelamiento predictivo, validación cruzada y métricas de evaluación.
- **Tensorflow + Keras** para el modelamiento con redes neuronales complejas que necesiten GPUs con *frameworks* especializados.

7. Resultados

7.1. Análisis Exploratorio de Datos (EDA)

Para comenzar con los primeros resultados de lo revisado en la metodología, primero se unieron ambas bases utilizadas para entrenamiento: *train_identity* y *train_transaction*, resultando en un gigantesco *dataset* de 590.540 filas y 434 columnas. A continuación, se presentan las primeras cinco filas del resultado, con algunas columnas del inicio y del final que representan las variables de la predicción:

Tabla 3

Primeras filas del dataset para entrenamiento y testeo

TransactionID	isFraud	TransactionDT	TransactionAmt	ProductCD	card1	card2	card3	card4	card5
2987000.0	0.0	86400.0	68.5	W	13926.0	7	150.0	discover	142.0
2987001.0	0.0	86401.0	29.0	W	2755.0	404.0	150.0	mastercard	102.0
2987002.0	0.0	86469.0	59.0	W	4663.0	490.0	150.0	visa	166.0
2987003.0	0.0	86499.0	50.0	W	18132.0	567.0	150.0	mastercard	117.0
2987004.0	0.0	86506.0	50.0	H	4497.0	514.0	150.0	mastercard	102.0

Nota: Se omitieron las demás filas y columnas, ya que la base es demasiado extensa y sólo se presentó la información inicial para exponer parte de la base. Elaboración propia.

En una primera evaluación de la base existen 403 variables del tipo numérico (sin considerar los nulos) y 31 del tipo categórica o texto. La columna con la variable a predecir sería la segunda denominada *isFraud* con sólo dos números en cada celda: el valor 0 significa que la transacción no es considerada fraudulenta y el 1 sí lo sería, por tanto, estas filas son las más importantes a analizar para saber las razones por la que el sistema las identificó así.

Sin embargo, muchas de las variables de la base tienen más del 99% de valores nulos, especialmente las de *train_identity*, acumulando 115.523.073 celdas en blanco que el sistema

identifica como vacío. La siguiente tabla ordena de mayor a menor estas columnas con más del 90% de columnas de este estilo:

Tabla 4

Variables con más del 90% de valores nulos ordenadas de mayor a menor

variable	nulos	porcentaje nulos	base
id_24	585793.0	99.20%	df_train
id_25	585408.0	99.13%	df_train
id_07	585385.0	99.13%	df_train
id_08	585385.0	99.13%	df_train
id_21	585381.0	99.13%	df_train
id_26	585377.0	99.13%	df_train
id_27	585371.0	99.12%	df_train
id_23	585371.0	99.12%	df_train
id_22	585371.0	99.12%	df_train
dist2	552913.0	93.63%	df_train
D7	551623.0	93.41%	df_train
id_18	545427.0	92.36%	df_train

Elaboración propia.

Estas variables fueron eliminadas del *dataset*, pues su aporte es muy insignificante para el entrenamiento de los modelos al tener solo ceros o espacios en blanco en casi todas las casillas. Se puede revisar que en su mayoría corresponden a datos de **ID** que sólo sirven para identificar a un usuario o transacción particular, por lo que su anexión al *dataset* sobra y sólo contribuye a hacer más grande la varianza y el procesamiento en *Python*. No obstante, existe un porcentaje del 24,42% de transacciones con al menos un dato de *identity*, por lo que no se podrían eliminar todas, sino las que realmente se puedan dar de baja.

Después se revisaron los valores constantes en columnas, es decir, números que puedan considerarse repetitivos en la base al ser todos iguales, sin considerar los nulos, e irrelevantes

para el análisis. Al revisarlo en el código, no existe ninguna variable constante ni fila duplicada, por consiguiente, no habría que eliminar ninguna variable por esta razón. La única más cercana es *isFraud* que tiene dos valores, pero no tendría sentido eliminarla, ya que es la variable predictora y con una muy baja variación. Ya después vienen tópicos que tienen 3 valores en sus filas como V88, *DeviceType* o M1 que a partir de este punto podrían tomar importancia en el modelo, sin la necesidad de eliminar ninguna.

La siguiente tabla resumen une los valores únicos con los nulos para tener mayor apreciación de las 403 variables numéricas y 31 categóricas que serían eliminadas en primera instancia:

Tabla 5

Tabla resumen inicial de la base de datos con las primeras columnas a eliminar

index	dtype	nulos	porcentaje_nulos	valores_unicos
id_24	float64	585793.0	99.20%	13.0
id_25	float64	585408.0	99.13%	342.0
id_07	float64	585385.0	99.13%	85.0
id_08	float64	585385.0	99.13%	95.0
id_21	float64	585381.0	99.13%	491.0
id_26	float64	585377.0	99.13%	96.0
id_27	object	585371.0	99.12%	3.0
id_23	object	585371.0	99.12%	4.0
id_22	float64	585371.0	99.12%	26.0
dist2	float64	552913.0	93.63%	1752.0
D7	float64	551623.0	93.41%	598.0
id_18	float64	545427.0	92.36%	19.0
D13	float64	528588.0	89.51%	578.0

Elaboración propia.

Como no era tanta la variación de la base al eliminar las columnas con un 90% de valores nulos, se decidió hacerlo con las de 80% para tener una limpieza más profunda del

DataFrame, logrando una base final de 360 variables de las cuales 334 son con números y 26 con formato texto u *object* según el intérprete.

Tabla 6

Variables finales del dataset con un umbral del 80% de valores nulos

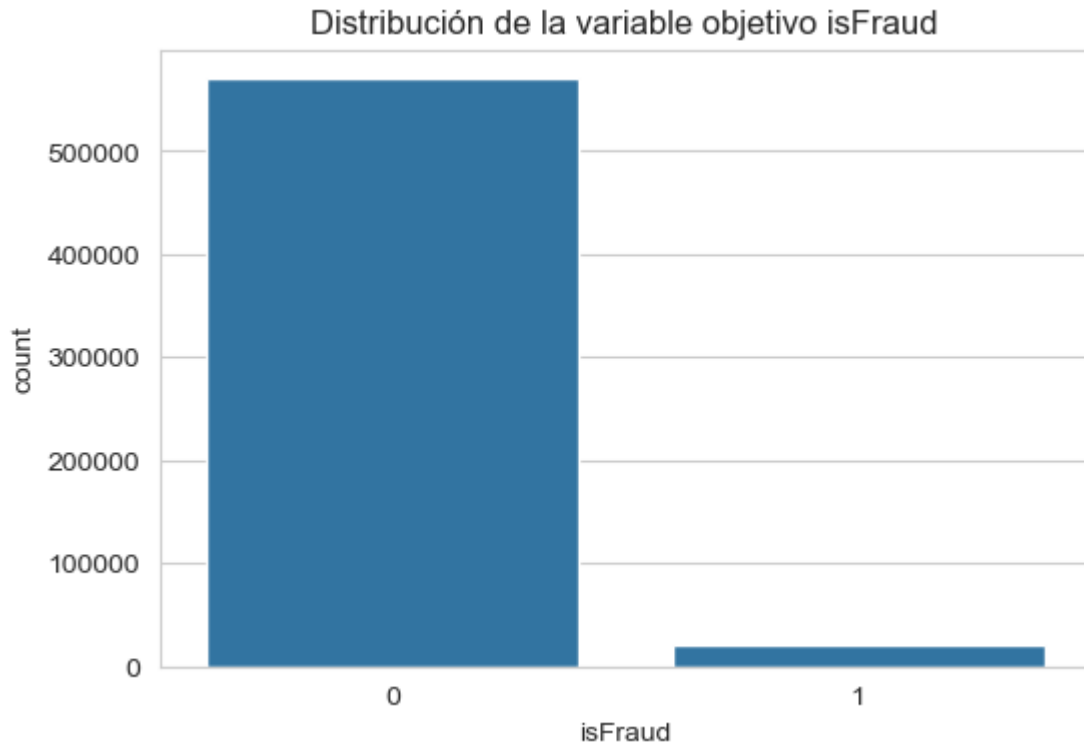
variable	nulos	porcentaje nulos	base
DeviceInfo	471874.0	79.91%	df_eda
id_13	463220.0	78.44%	df_eda
id_16	461200.0	78.10%	df_eda
V276	460110.0	77.91%	df_eda
V262	460110.0	77.91%	df_eda
V249	460110.0	77.91%	df_eda
V252	460110.0	77.91%	df_eda
V253	460110.0	77.91%	df_eda
V254	460110.0	77.91%	df_eda
V257	460110.0	77.91%	df_eda
V258	460110.0	77.91%	df_eda
V260	460110.0	77.91%	df_eda
V261	460110.0	77.91%	df_eda

Elaboración propia.

Posteriormente se pasó a revisar la distribución de la variable objetivo en la base para conocer la proporción de fraudes reconocibles dentro de las transacciones normales. El siguiente gráfico presenta la cantidad de fraudes encontrados frente al total, según el valor dentro de la variable.

Figura 15

Distribución de la variable objetivo isFraud



Elaboración propia.

Como se podía intuir, la cantidad de fraudes encontrados fue del 3,5%, mientras que 96,5% resultan de transacciones sin actividad sospechosa. Esto es muy relevante, especialmente en los entrenamientos de los modelos de clasificación binarias, pues al evaluar miles de movimientos registrados, resulta cada vez más difícil encontrar fraudes que se puedan predecir. Entonces, la regularización del *overfitting* es crucial para evitar el sobreajuste con una enorme varianza o multicolinealidad que pueda influir en el procesamiento de los modelos y entregar resultados erróneos, por ejemplo, saltarse una posible transacción sospechosa que se identificó como normal lo que podría provocar graves consecuencias si en verdad era un ataque.

7.1.1. Reducción de dimensionalidad por medio de PCA

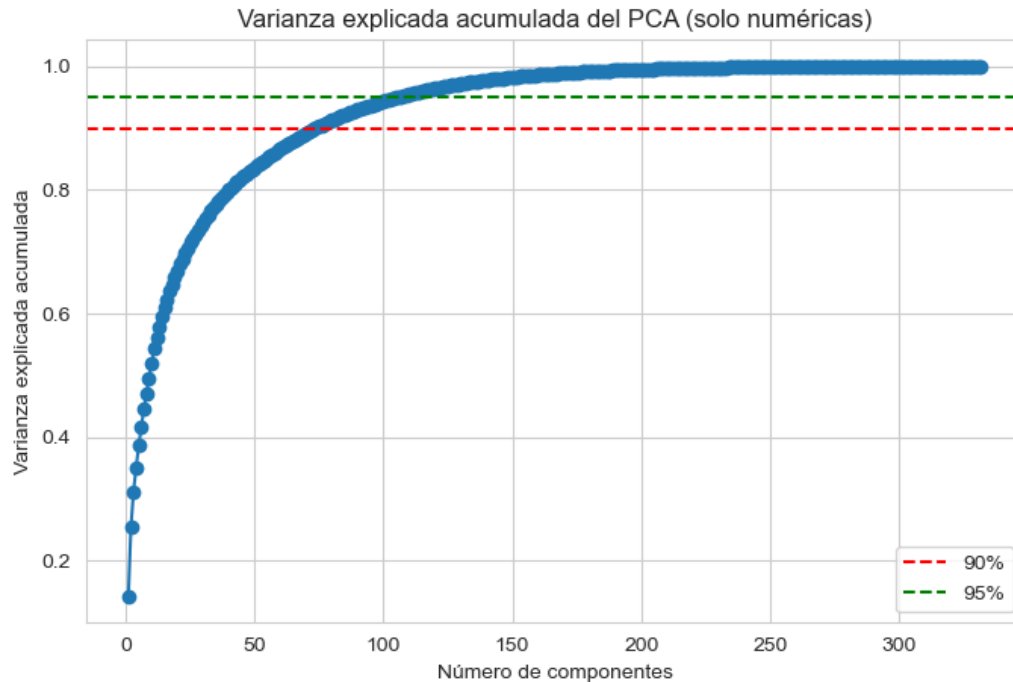
A pesar de haber eliminado columnas por los vacíos, la base sigue siendo bastante grande y compleja para poder trabajar con ella al tener 360 columnas. No obstante, no todas estas variables son las más cercanas al promedio o explican la varianza que es indispensable en las pruebas que hacen los modelos. Así es como aparece el análisis de componentes principales o PCA para intentar reducir la complejidad del problema y sólo utilizar aquellas variables transformadas en componentes que sean indispensables para obtener resultados igual de eficientes que usando todo el *dataset*.

Se utilizaron solo variables numéricas pues son las que permanecen en la base, las cuales se estandarizaron usando el comando de *StandardScaler* para poder trabajar de mejor manera sin las diferencias de magnitud entre columnas. Además, se realizó la primera división de variables, donde X serían todas las explicativas del objetivo Y como la columna del fraude. Los hiper - parámetros escogidos fueron una muestra de 100.000 observaciones de prueba, una semilla de 42 para garantizar reproducibilidad del modelo.

El siguiente gráfico expone la salida del código realizado con las componentes principales que explican la varianza. El número de componentes retenidos se definió a partir de la varianza explicada acumulada, considerando como referencia los umbrales de 90% y 95%.

Figura 16

Aumento de la varianza a medida que aumentan los componentes principales



Nota: Se utilizaron sólo variables numéricas para realizarlo. Elaboración propia.

Se observa que la curva crece rápidamente en los primeros componentes, lo que indica que una parte importante de la información contenida en las variables originales puede concentrarse en un número considerablemente menor de dimensiones. En particular, 74 componentes principales permiten explicar aproximadamente el 90% de la varianza acumulada, mientras que 106 componentes explican alrededor del 96%.

A partir de este resultado, se decidió utilizar 74 componentes principales en la etapa de modelamiento, ya que este valor representa un equilibrio adecuado entre conservación de información y reducción de dimensionalidad. La mejora marginal es menor en comparación con el aumento en complejidad computacional al utilizar las 106 componentes. Por lo tanto, mantener el umbral del 90% permite disminuir el costo de procesamiento y entrenamiento de los modelos.

Posteriormente, se decidió revisar las variables que eran parte de cada componente, especialmente las del 90%. A través de un código que separa los *loadings* en una matriz que señala su relevancia, se ordenaron las principales 30 variables que contribuyen a los 74 componentes principales retenidos. La siguiente tabla muestra las primeras 10 ordenadas de mayor a menor que podrían tener más incidencia en encontrar un fraude o no.

Tabla 7

Ranking de variables numéricas con mayor contribución a las componentes principales retenidas

variable	importancia_global
V168	0.0195
V128	0.0195
V308	0.0194
V97	0.0194
V280	0.0194
V103	0.0193
V179	0.0193
V134	0.0193
V213	0.0193
V295	0.0193

Elaboración propia.

Estos valores no tienen un tópico directo al que se refieren, debido a que pertenecen a datos más confidenciales de *Vesta* y tienen que ver con relaciones contables y flujos propios de la empresa, por lo que no se puede saber su significado con exactitud. Sin embargo, se puede observar que las variables V168, V128, V308, V97 y V280 son las que mayor contribución tienen en la varianza al tener un peso del 1,95% dentro de los componentes principales. Hay que tener en cuenta que PCA usa otro tipo de relaciones matemáticas y

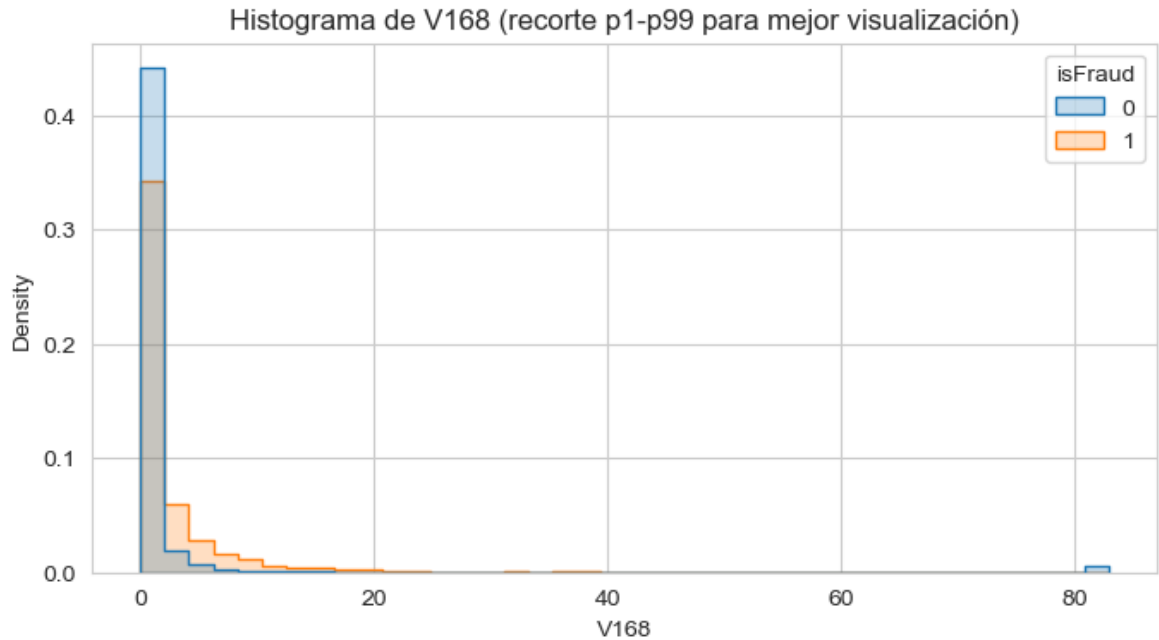
funciones para crear la reducción de dimensionalidad y bajar el costo de procesamiento, aunque los resultados pueden ser similares. En consecuencia, las variables no tienen un peso tan grande en la creación de cada componente y no todos los modelos se pueden entrenar así por esta razón. Por ejemplo, árboles de decisión prefiere utilizar todo el *DataFrame* con el propósito de mantener interpretabilidad, ya que necesita hacer cortes lo más preciso posible. Al reducir la complejidad puede fallar en su factibilidad de predecir correctamente, incurriendo en *underfitting* o problemas semejantes.

7.1.2. Relaciones mediante gráficos y matriz de correlaciones de las principales variables numéricas

De esta forma y gracias al análisis PCA, se pudieron revisar las principales relaciones y grados de importancia de cada variable numérica extraída. Se revisarán histogramas y *boxplots* para cada variable en un inicio y luego la matriz de correlaciones. Para hacerlo más resumido, se señalarán sólo las 3 principales de cada tipo para no extender demasiado esta parte, pues el entrenamiento y prueba de los modelos es lo realmente interesante. Las demás relaciones se expondrán en los **Anexos**. Primero se verán las variables numéricas y después categóricas.

Figura 17

Histograma de V168 (recorte percentil p1 – p99 para mejor visualización)



Nota: Los colores representan la clase con naranja para *isFraud* y azul cuando no lo es. El color gris señala la subida de densidad en los primeros valores. Elaboración propia.

El histograma de la variable V168, segmentado según la variable objetivo *isFraud*, muestra una distribución fuertemente concentrada en valores bajos, especialmente cercanos a cero, tanto para transacciones fraudulentas como no fraudulentas. Sin embargo, se observa que las transacciones clasificadas como fraude presentan una cola más extendida hacia valores mayores, lo que sugiere que valores elevados de V168 podrían estar asociados con patrones transaccionales menos frecuentes y potencialmente más riesgosos.

Figura 18

Histograma de V128

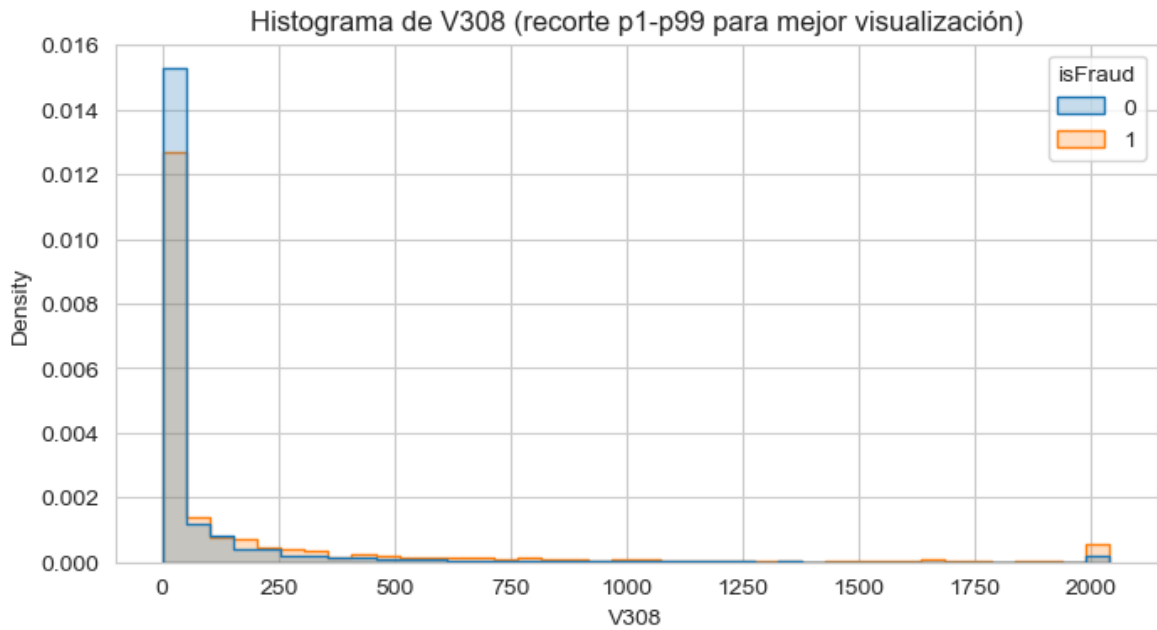


Elaboración propia.

El histograma de V128 evidencia una distribución altamente asimétrica hacia la derecha, con una fuerte concentración de observaciones en valores cercanos a cero y una cola extensa que alcanza valores elevados. Al comparar ambas clases, se observa que las transacciones fraudulentas presentan una mayor presencia relativa en valores intermedios y altos de la variable, mientras que las no fraudulentas se concentran con mayor intensidad en el extremo inferior. Esto sugiere que V128 podría aportar información relevante para diferenciar ciertos patrones de fraude, especialmente en casos menos frecuentes asociados a valores más altos.

Figura 19

Histograma de V308

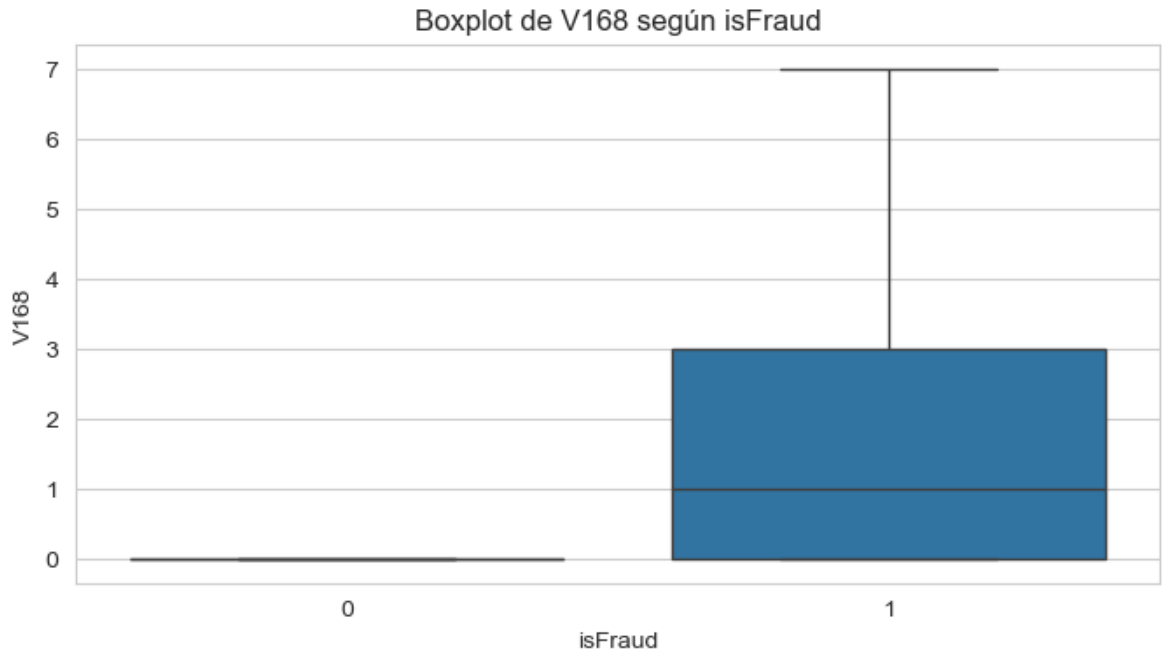


Elaboración propia.

El histograma muestra una distribución fuertemente asimétrica hacia la derecha, con una alta concentración de transacciones en valores bajos y una cola larga que se extiende hacia valores elevados. Al comparar las clases, se observa que las transacciones fraudulentas presentan una presencia relativa más marcada en los valores intermedios y altos, mientras que las no fraudulentas se concentran principalmente cerca de cero. Este comportamiento sugiere que V308 podría capturar patrones poco frecuentes pero relevantes para la detección de fraude.

Figura 20

Boxplot de V168 según IsFraud (variable objetivo)

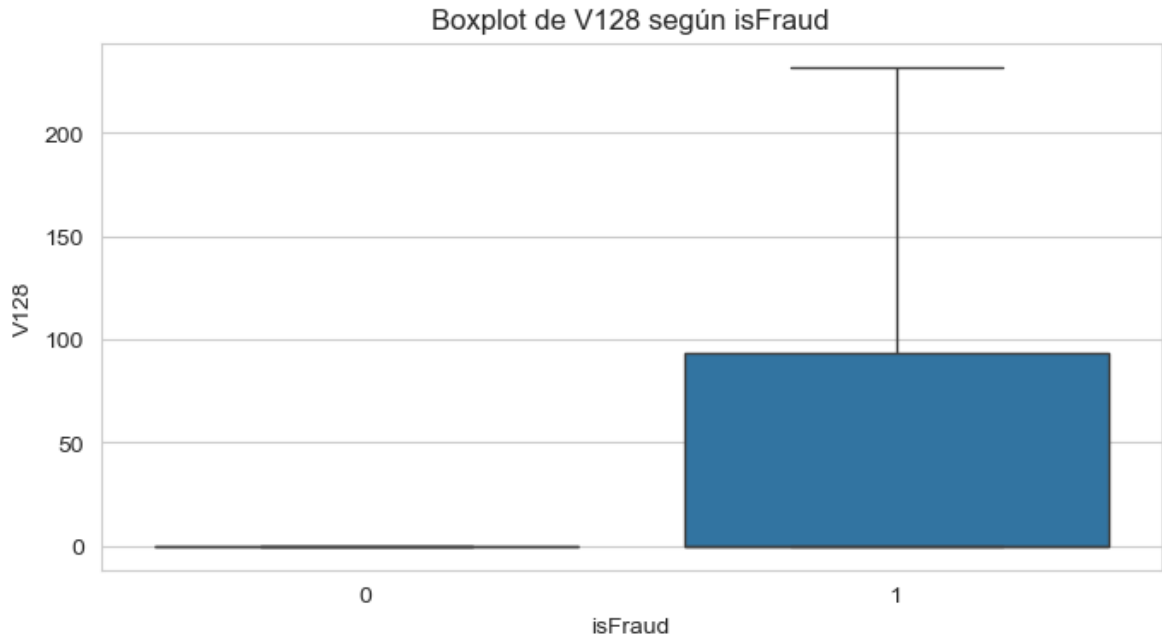


Nota: El valor 0 en *IsFraud* significa que no existe fraude, mientras que el 1 representa un fraude en la transacción. Elaboración propia.

El *boxplot* de V168 muestra una diferencia clara entre las transacciones no fraudulentas y fraudulentas. Para la clase no fraude, los valores se concentran prácticamente en cero, con muy baja dispersión. En cambio, las transacciones fraudulentas presentan una mediana superior y una variabilidad considerablemente mayor, alcanzando valores cercanos a 7 dentro del rango visualizado. Esto refuerza la idea de que V168 aporta información discriminante, ya que los casos de fraude tienden a presentar valores más altos y dispersos.

Figura 21

Boxplot de V128 según IsFraud

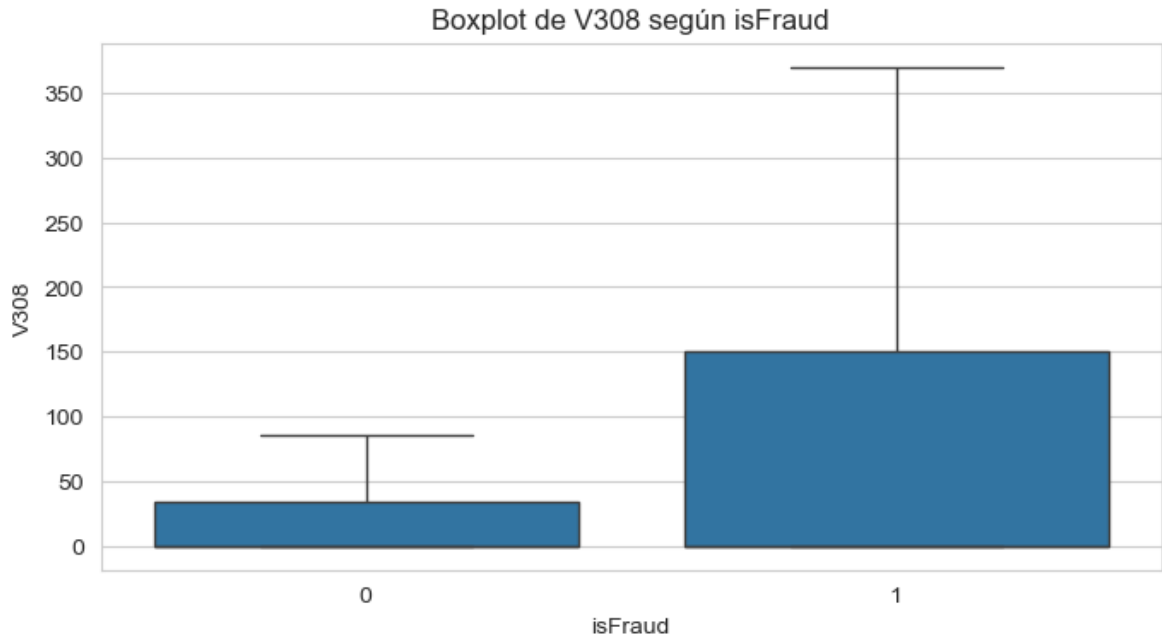


Elaboración propia.

El *boxplot* expone una separación marcada entre las clases. Las transacciones no fraudulentas se concentran casi completamente en valores cercanos a cero, mientras que las transacciones fraudulentas presentan una dispersión mucho mayor, con una mediana superior y valores que alcanzan niveles considerablemente más altos. Esta diferencia sugiere que V128 posee capacidad discriminante para el problema, ya que los casos fraudulentos tienden a presentar valores más elevados y variables.

Figura 22

Diagrama de caja y bigote de V308 según IsFraud



Elaboración propia.

El diagrama de V308 confirma una diferencia relevante entre ambas clases. Las transacciones no fraudulentas presentan una menor dispersión y se concentran en rangos bajos de la variable, mientras que las transacciones fraudulentas muestran una variabilidad considerablemente mayor, con valores que se extienden hasta niveles mucho más altos. Esta separación en la distribución sugiere que V308 puede aportar información útil para distinguir patrones asociados al fraude, especialmente porque los casos fraudulentos tienden a presentar valores más elevados y dispersos que las transacciones legítimas.

La siguiente tabla presenta un resumen de estas variables revisadas, presentando la media, mediana y desviación estándar de cada respecto a si es o no fraude. La V308 presenta mayores valores, por lo que puede ser significativa para revisar tendencias y patrones respecto a la variable objetivo o la varianza del problema.

Tabla 8

Resumen estadístico de las variables más importantes que explican la varianza según PCA

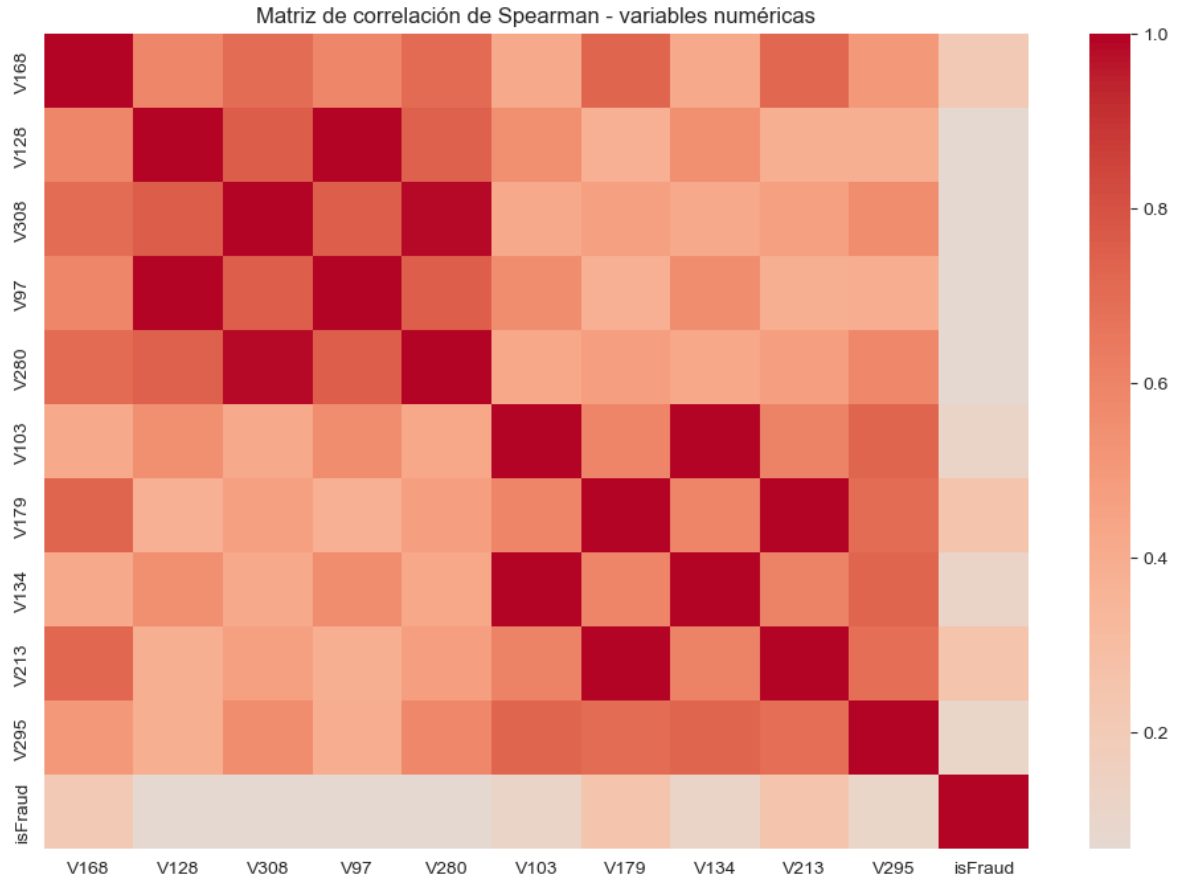
Var	isFraud	0	1
V168	mean	5.627879	2.696889
	median	0.000000	1.000000
	std	53.347255	5.807461
V128	mean	193.880724	163.300594
	median	0.000000	0.000000
	std	2905.685632	525.120312
V308	mean	217.470805	277.318349
	median	0.000000	0.000000
	std	2917.652573	1049.071263

Nota: La tabla presenta los principales índices estadísticos de cada variable según el valor con *IsFraud*. Elaboración propia.

Finalmente, a continuación se presenta el análisis de la matriz de correlación de *Spearman* de las 10 principales variables enteras y decimales obtenidas de los reductores de dimensionalidad sobre la categoría a predecir.

Figura 23

Matriz de correlación de Spearman de las variables numéricas más relevantes



Elaboración propia.

La matriz de correlación de *Spearman* muestra que las variables numéricas seleccionadas presentan asociaciones positivas importantes entre sí, especialmente entre pares como V128–V97, V308–V280, V103–V134 y V179–V213. Esto indica la existencia de relaciones monotónicas entre varias del bloque numérico, lo que sugiere cierto grado de redundancia o información compartida entre atributos. Este comportamiento es coherente con la aplicación de reducción de dimensionalidad para condensar la información contenida en variables altamente relacionadas y disminuir el costo computacional del modelamiento.

Respecto a la variable objetivo *isFraud*, se observa que las correlaciones son positivas, pero considerablemente más bajas que las existentes entre las variables explicativas. Las que

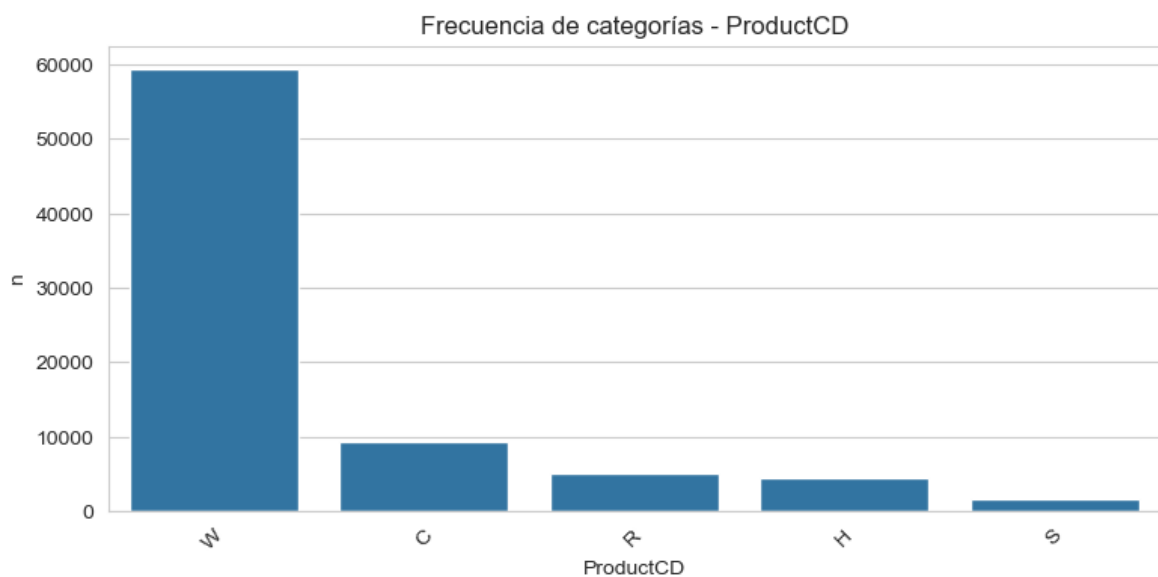
visualmente muestran una mayor cercanía con *isFraud* son principalmente V168, V179 y V213, seguidas por V295, V103 y V134. Sin embargo, la baja intensidad relativa de estas correlaciones sugiere que el fraude no puede explicarse únicamente mediante relaciones monotónicas simples con variables individuales. Por ello, resulta necesario utilizar modelos de clasificación capaces de capturar relaciones más complejas entre múltiples variables.

7.1.3. Relaciones mediante gráficos y matriz de correlaciones de las principales variables categóricas

Ahora se generará un análisis de las variables categóricas, teniendo en cuenta la frecuencia de cada una en la base y la tasa del fraude según las columnas en las que efectivamente se encontró uno frente a todas las demás. Se seleccionarán aquellas con las distribuciones más relevantes en gráficos de barras, aunque todas las demás se pueden revisar más a detalle en los **Anexos**.

Figura 24

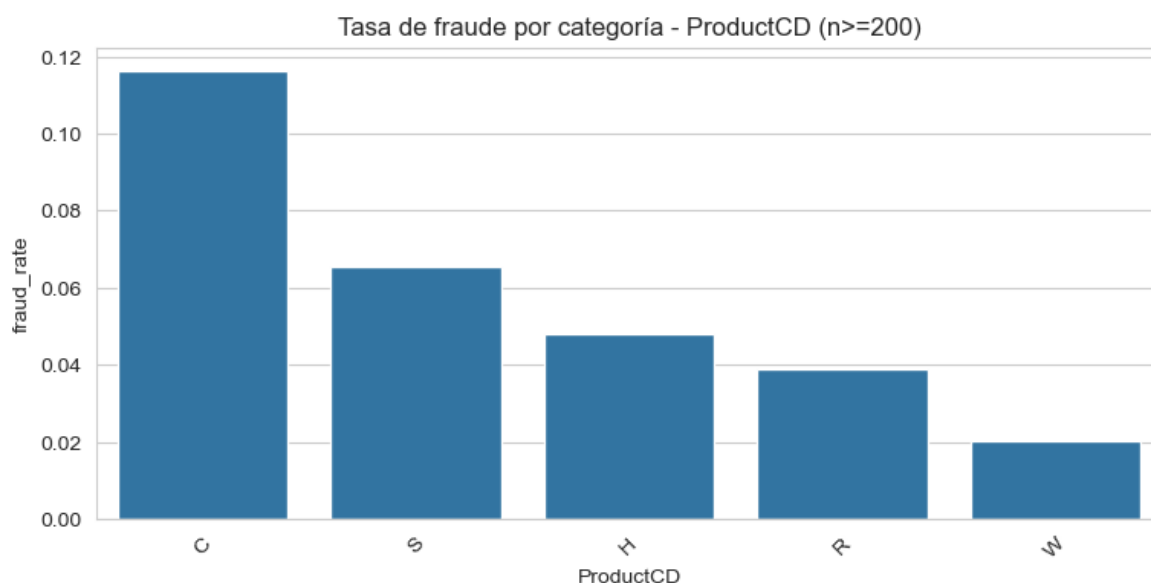
Frecuencia de ProductCD



Elaboración propia.

Figura 25

Tasa de fraude vs ProductCD (n>=200)

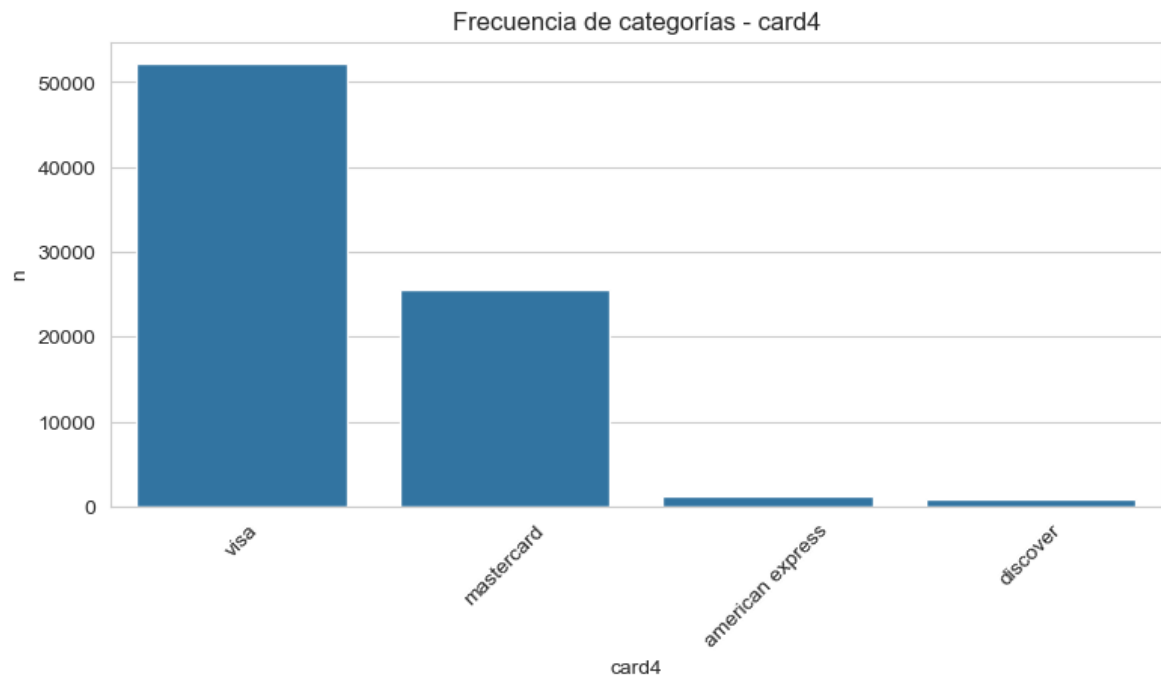


Nota: para obtener los gráficos, se utilizó una frecuencia de sobre 200 de muestra. Elaboración Propia.

La variable *ProductCD* presenta una distribución altamente concentrada en la categoría W, que representa la mayor parte de las transacciones observadas, seguida a bastante distancia por C, R, H y S. Sin embargo, al analizar la tasa de fraude por categoría, se observa que la mayor frecuencia no implica mayor riesgo: aunque W es la categoría dominante, presenta la menor tasa de fraude relativa. En contraste, la categoría C muestra la tasa de fraude más alta, seguida por S, H y R. Esto indica que *ProductCD* es una variable categórica relevante, ya que permite distinguir perfiles de producto con distintos niveles de exposición al fraude, especialmente en categorías menos frecuentes pero proporcionalmente más riesgosas.

Figura 26

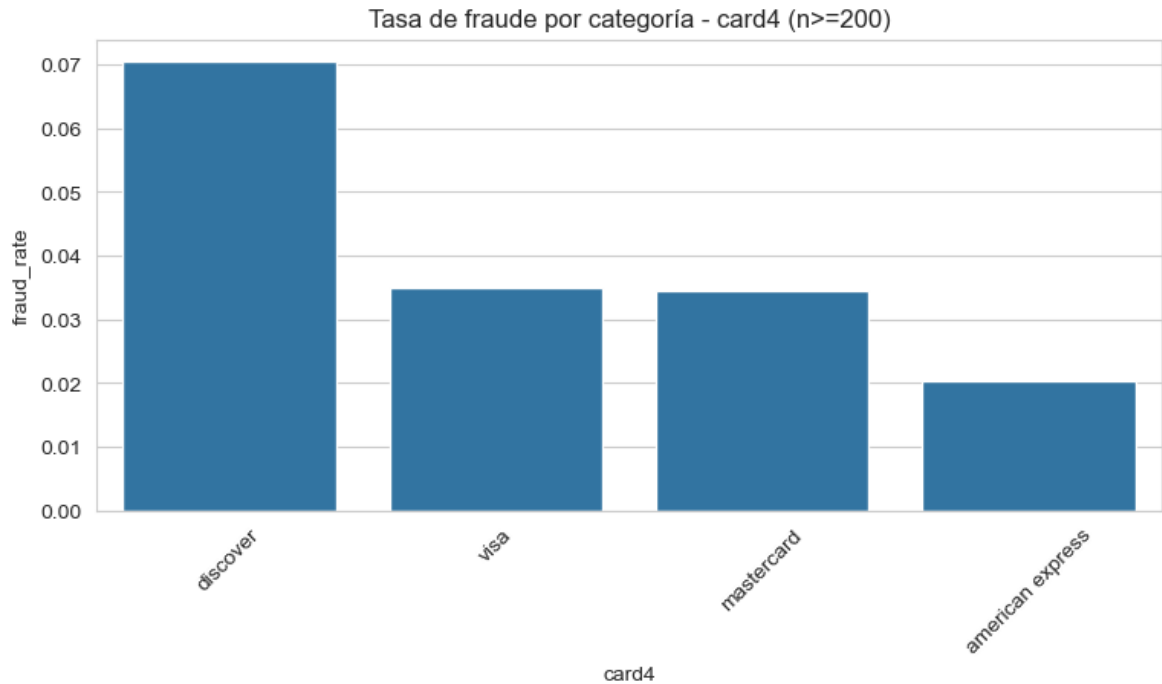
Frecuencia de card4



Elaboración propia.

Figura 27

Tasa de fraude vs card4 ($n \geq 200$)

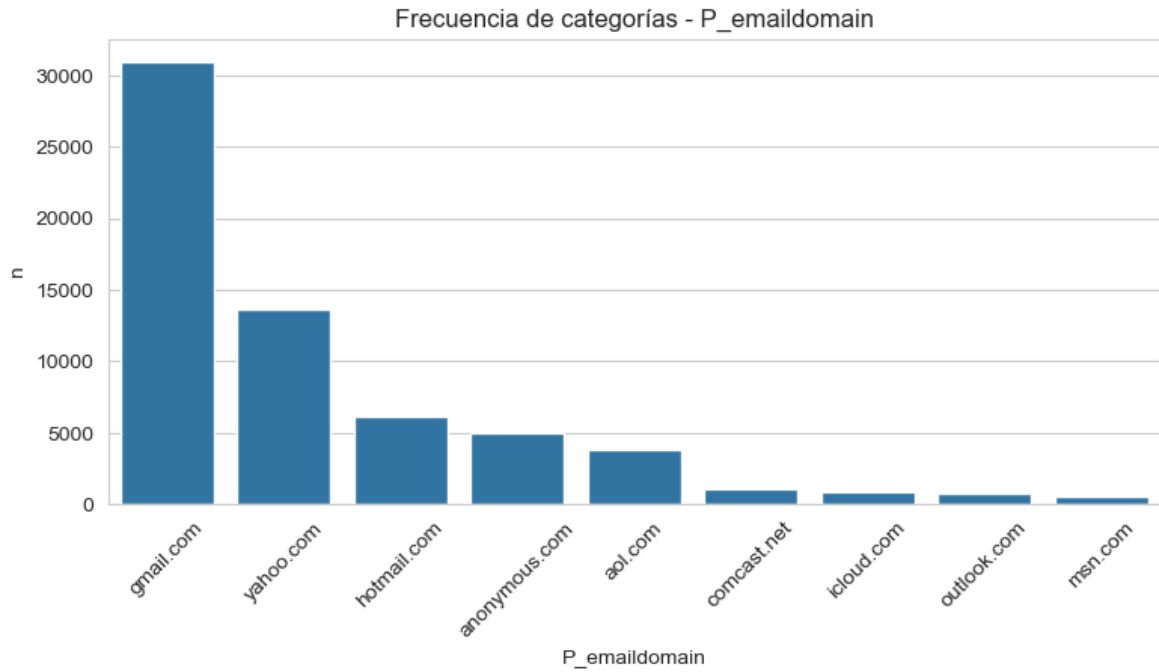


Elaboración propia.

La variable *card4* presenta una concentración importante en las categorías *visa* y *mastercard*, siendo *visa* la más frecuente dentro del conjunto de datos, mientras que *american express* y *discover* tienen una presencia considerablemente menor. Sin embargo, al observar la tasa de fraude por categoría, se aprecia que la frecuencia no necesariamente se relaciona con mayor riesgo: *discover*, pese a ser una de las categorías menos frecuentes, presenta la tasa de fraude más alta. En cambio, *visa* y *mastercard* muestran tasas intermedias y relativamente similares, mientras que *american express* presenta la menor tasa de fraude. Esto sugiere que *card4* puede aportar información relevante, ya que ciertas redes o tipos de tarjeta podrían estar asociadas a distintos niveles de exposición.

Figura 28

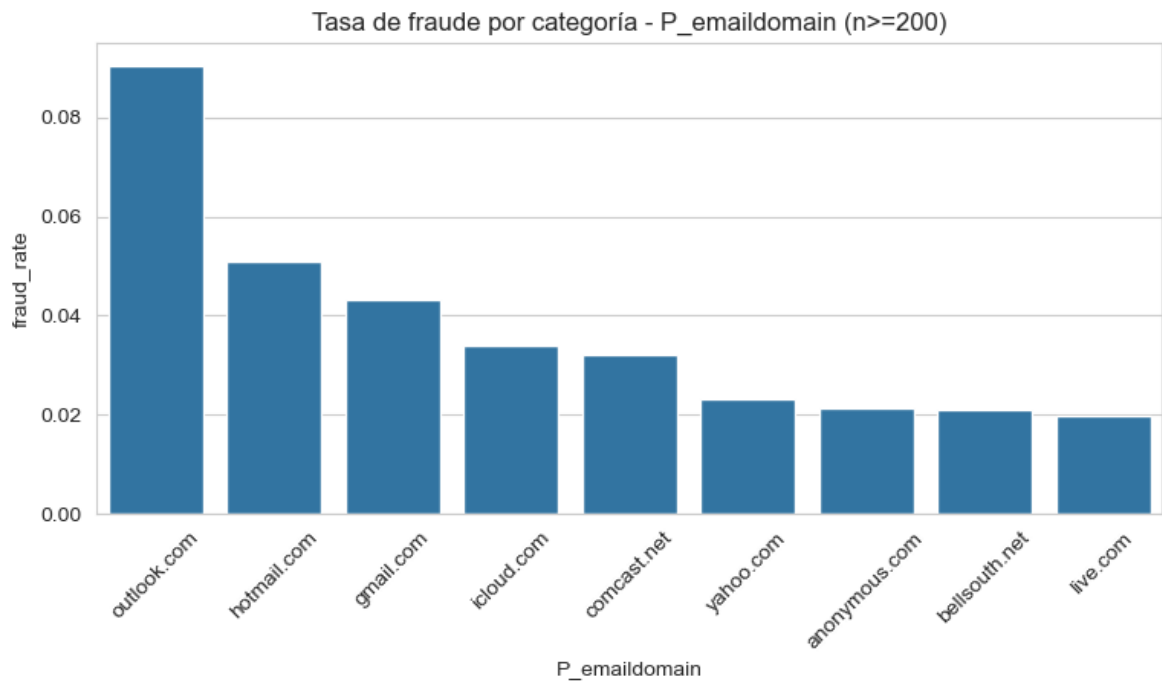
Frecuencia de *P_emaildomain*



Elaboración propia.

Figura 29

Tasa de fraude vs P_emaildomain (n>=200)

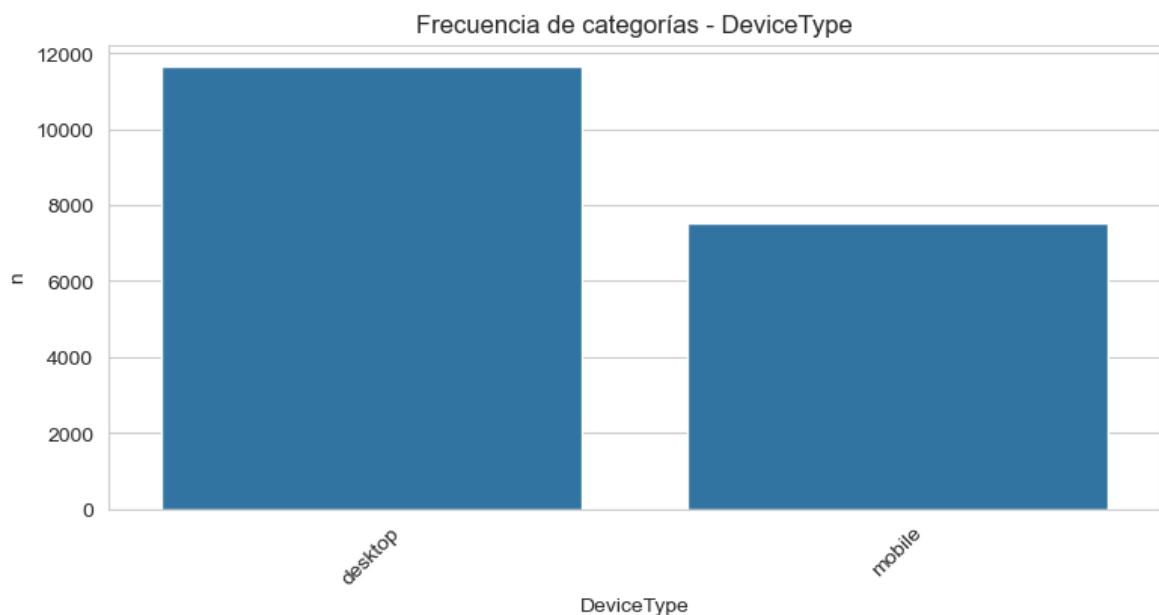


Elaboración propia.

$P_{emaildomain}$ muestra una alta concentración en dominios de correo masivos, especialmente *gmail.com*, seguido por *yahoo.com*, *hotmail.com* y *anonymous.com*. Sin embargo, al analizar la tasa de fraude por categoría, se observa que los dominios más frecuentes no son necesariamente los de mayor riesgo. En particular, *outlook.com* presenta la tasa de fraude más alta entre las categorías con suficiente soporte, seguido por *hotmail.com* y *gmail.com*, mientras que dominios como *yahoo.com*, *anonymous.com* y *live.com* muestran tasas menores. Esto sugiere que el dominio de correo del comprador en *microsoft* (*outlook* y *hotmail*) presentaron mayores tasas observadas de fraude, ya que estos dominios contienen una proporción de fraude superior a la frecuencia esperada.

Figura 30

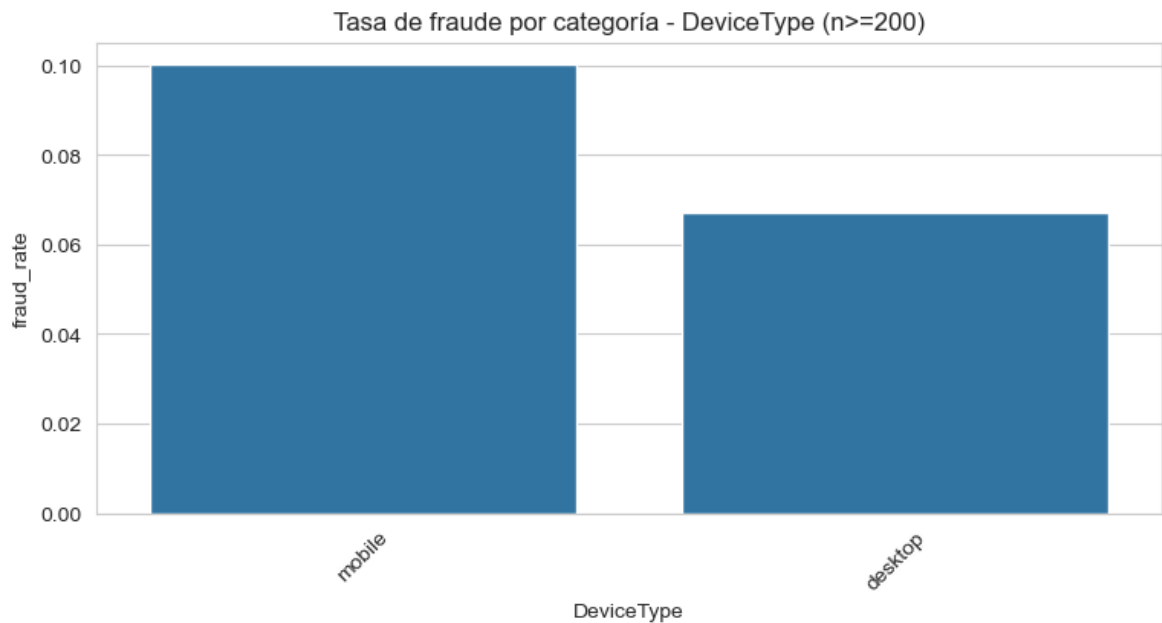
Frecuencia de DeviceType



Elaboración propia.

Figura 31

Tasa de fraude vs DeviceType

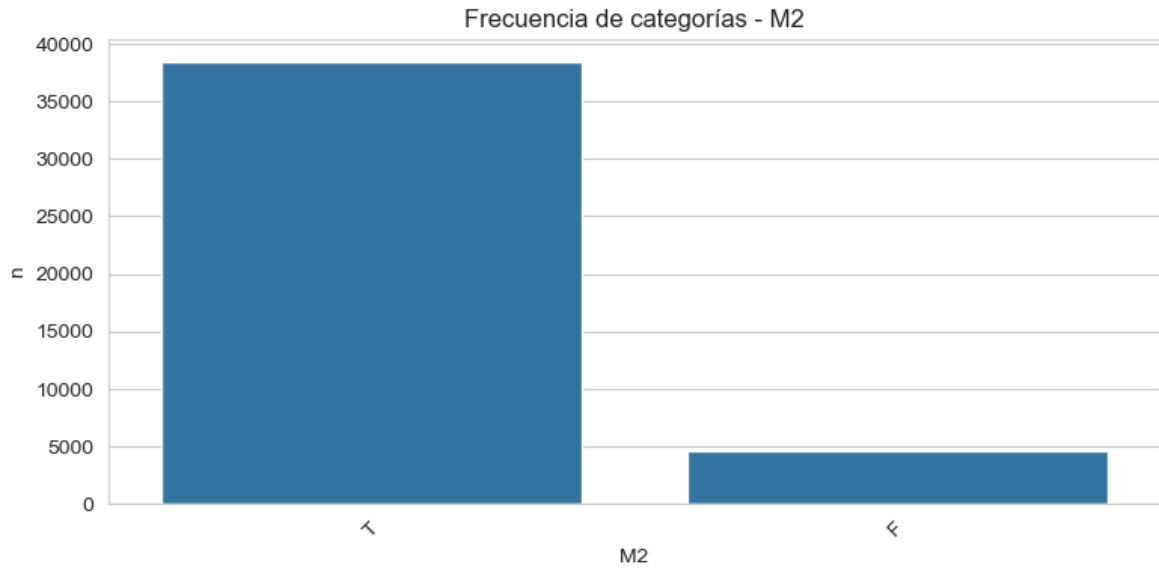


Elaboración propia.

En el caso de *DeviceType*, se observa una diferencia clara en la tasa de fraude entre transacciones realizadas desde dispositivos móviles y de escritorio. Las operaciones asociadas a *mobile* presentan una tasa cercana al 10%, mientras que las realizadas desde *desktop* se ubican alrededor del 6,7%. Esta diferencia sugiere que el tipo de dispositivo está asociado a distintos perfiles de riesgo dentro del conjunto de datos. En particular, el mayor fraude relativo en dispositivos móviles podría estar vinculado hipotéticamente a patrones de compra más dinámicos, menor trazabilidad del entorno o diferencias en el comportamiento de uso respecto a transacciones realizadas desde escritorio.

Figura 32

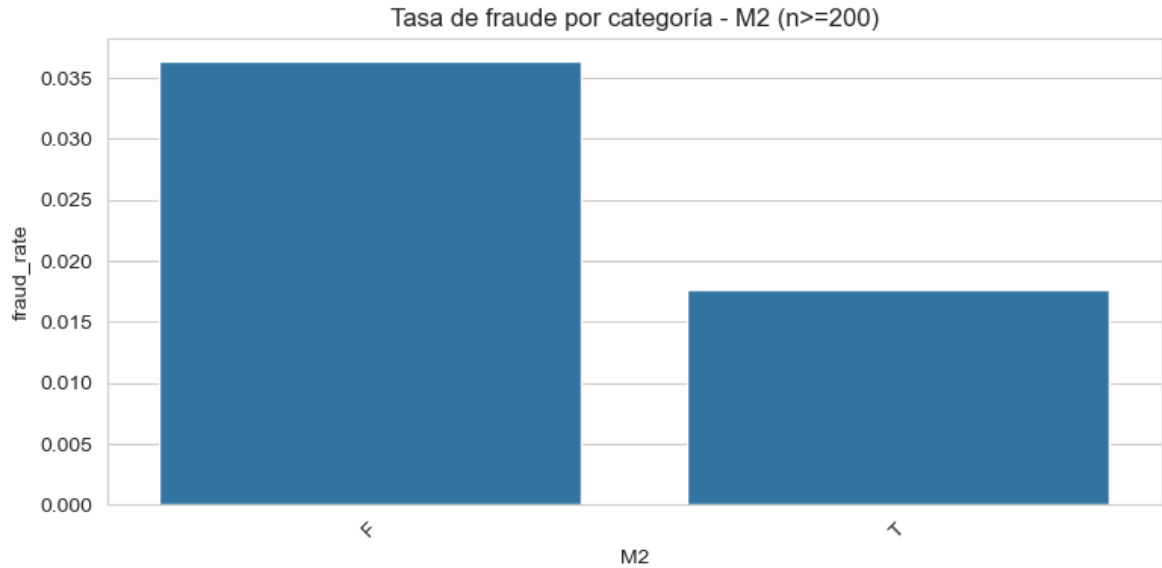
Frecuencia de M2



Elaboración propia.

Figura 33

Tasa de fraude vs M2



Elaboración propia.

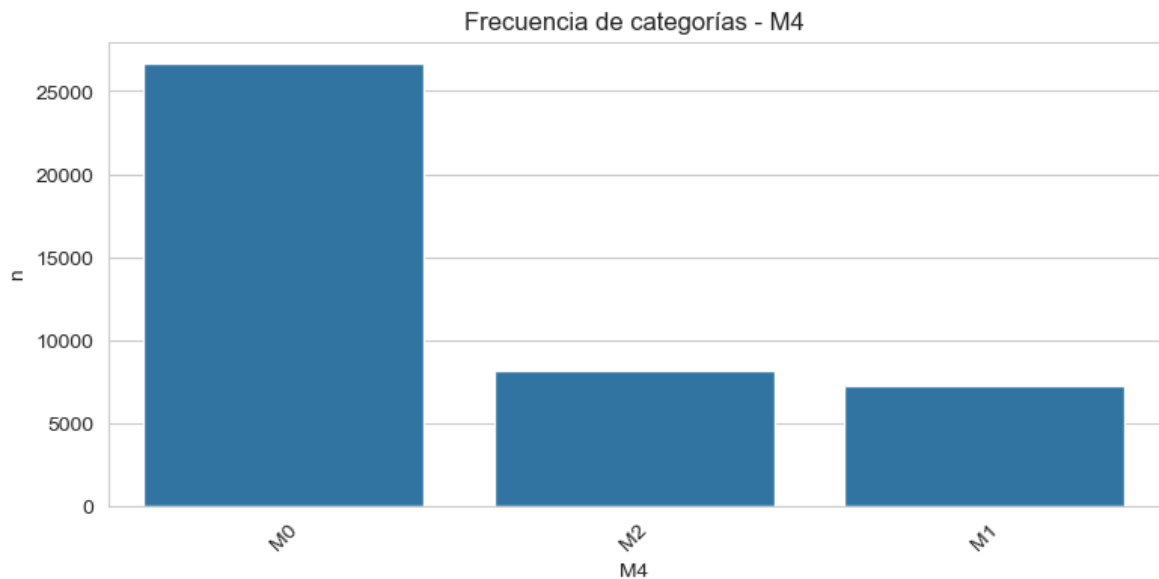
En las variables M presentan características coincidentes como dirección, nombre, etc.

Para M2 la categoría T concentra la mayor cantidad de observaciones, superando

ampliamente a la categoría F. Sin embargo, al comparar la tasa de fraude, se observa el comportamiento contrario: las transacciones con categoría F presentan una proporción de fraude cercana al 3,6%, mientras que la categoría T se ubica alrededor del 1,7%. Esta diferencia indica que, aunque F aparece con menor frecuencia en la base, se asocia proporcionalmente a un mayor nivel de fraude.

Figura 34

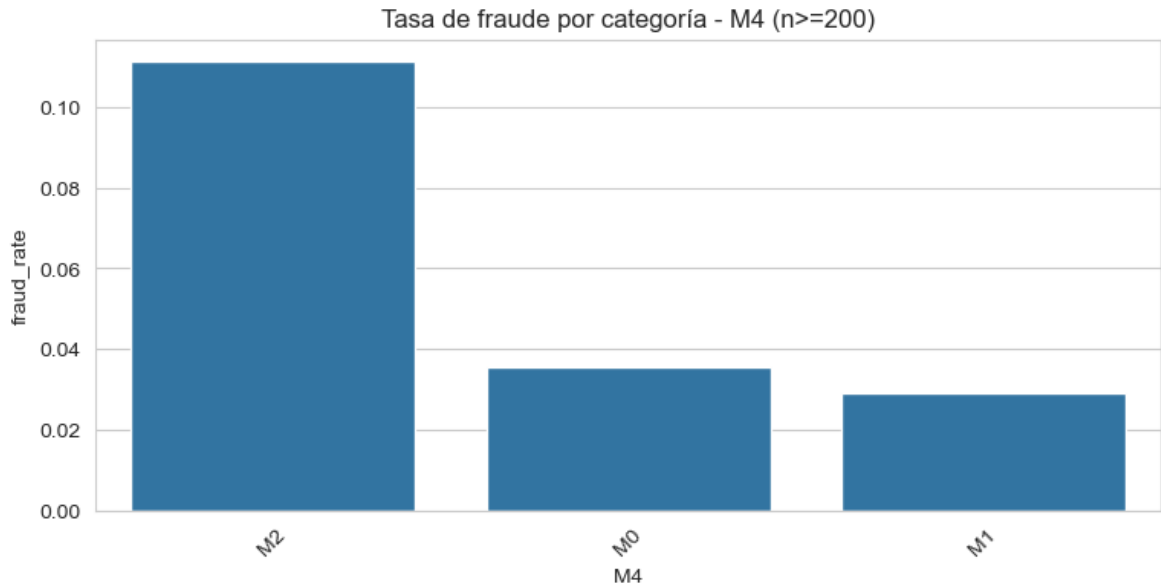
Frecuencia de M4



Elaboración propia.

Figura 35

Tasa de fraude vs M4



Elaboración propia.

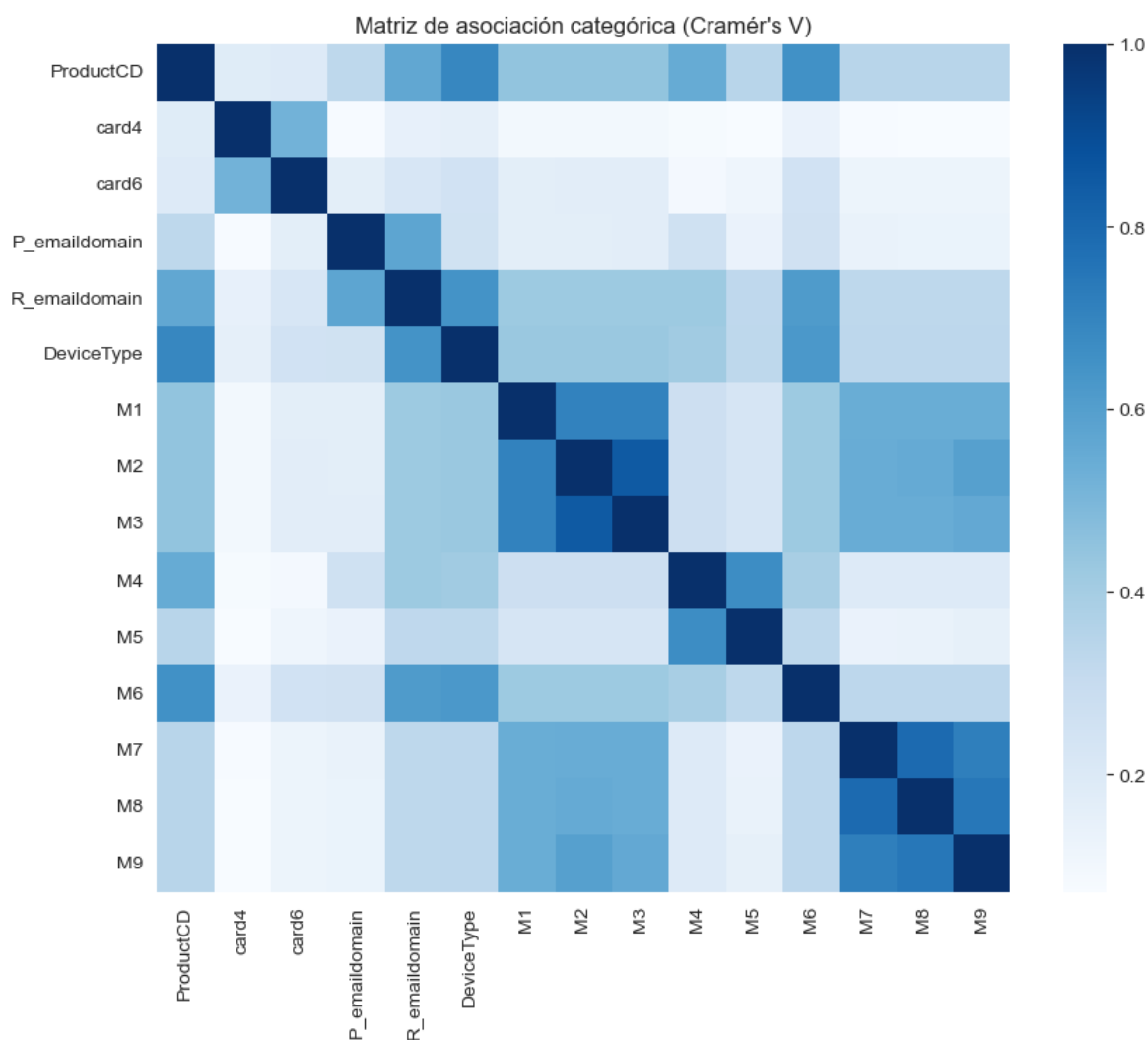
En la variable M4, la categoría M0 concentra la mayor cantidad de observaciones, seguida por M2 y M1, que aparecen con frecuencias menores y relativamente similares. Sin embargo, la tasa de fraude muestra un patrón distinto: M2 presenta una proporción de fraude cercana al 11%, muy superior a M0 y M1. Esto indica que, aunque M2 no es la categoría más frecuente, concentra un perfil proporcionalmente más riesgoso. Dado que M4 pertenece al grupo de variables M, su interpretación debe realizarse en conjunto con el resto de estos indicadores, ya que probablemente capturan condiciones relacionadas entre sí dentro del proceso transaccional.

Luego de estos gráficos, se generó la matriz de correlación de las variables categóricas mediante el coeficiente de *Cramér's V*, el cual permite medir la intensidad de relación entre variables categóricas en una escala de 0 y 1. En este análisis no se incorporó la variable objetivo *isFraud*, ya que el propósito de la matriz fue revisar la asociación interna entre variables predictoras y no evaluar directamente su relación con el fraude. De esta forma, la

matriz permite identificar posibles patrones de dependencia o redundancia entre atributos antes de incorporarlos al modelamiento

Figura 36

Matriz de correlaciones de variables categóricas mediante Cramér's V



Elaboración propia.

Se observa una asociación alta entre algunas variables del grupo M, especialmente entre M2 y M3, así como entre M7, M8 y M9, lo que sugiere que estas variables podrían estar capturando información similar o condiciones relacionadas dentro del proceso transaccional.

También destacan asociaciones relevantes entre *ProductCD*, *DeviceType*, *R_emaildomain* y *M6*, lo que podría indicar que ciertos tipos de producto, dispositivos y dominios de correo receptor tienden a aparecer conjuntamente. Por otro lado, variables como *card4* y *card6* presentan una asociación moderada, esperable al estar ambas relacionadas con características de la tarjeta utilizada. Estos resultados muestran que varias variables categóricas no son completamente independientes entre sí, aspecto relevante al interpretar los modelos y al aplicar transformaciones como la codificación *dummy*.

La tabla final resume el comportamiento de las principales categóricas en función de la tasa de fraude observada. Para cada una se calculó la cantidad de categorías válidas analizadas, la menor tasa de fraude, la mayor tasa de fraude y el *gap_fraud_rate*, entendido como la diferencia entre la tasa máxima y mínima de fraude dentro de una misma variable. Este indicador se incorporó para comparar de manera simple qué variables de texto presentan mayor heterogeneidad interna respecto al fraude, es decir, cuáles muestran diferencias más marcadas entre categorías de bajo y alto riesgo.

Tabla 9

Impacto de variables categóricas en la tasa de fraude

index	variable	n_categorias_validas	fraud_rate_min	fraud_rate_max	gap_fraud_rate
4.0	R_emaildomain	9.0	0.78%	17.33%	16.55%
0.0	ProductCD	5.0	2.01%	11.64%	9.63%
9.0	M4	4.0	1.94%	11.12%	9.18%
3.0	P_emaildomain	18.0	0.27%	9.04%	8.77%
5.0	DeviceType	3.0	2.08%	10.01%	7.93%
11.0	M6	3.0	1.67%	7.13%	5.46%
1.0	card4	5.0	2.03%	7.04%	5.01%
2.0	card6	3.0	2.50%	6.45%	3.95%

Nota: La tabla señala diferencias de tasa de fraude entre categorías con soporte suficiente.

Elaboración Propia.

Las mayores diferencias se observan en *R_emaildomain*, con un *gap* de 16,55 puntos porcentuales, seguida por *ProductCD*, *M4*, *P_emaildomain*, *DeviceType*, *M6*, *card4* y *card6*. Esto muestra que estas variables presentan categorías con tasas de fraude considerablemente distintas entre sí. Por ejemplo, tanto los dominios de correo como el tipo de producto, dispositivo y características de la tarjeta permiten distinguir grupos con comportamientos de fraude más contrastantes.

7.2. Evaluación con modelos de Regresión Lineal

En esta etapa ya se empieza con el primer entrenamiento y testeo de modelos de *machine learning*. Se evalúa el aprendizaje supervisado con modelos de regresión lineal y de regularización. Cabe destacar que para estas instancias se pudieron aplicar dos tipos de testeo para separar los grupos de variables explicativas y generar los entrenamientos correspondientes: *K-Cross Fold Validation* y *Train/Test Split*.

Se decidió utilizar una partición fija de entrenamiento y prueba mediante *train/test split*, en lugar de validación cruzada *K-Fold*. Esto se fundamenta en que el conjunto de datos presenta un tamaño elevado, por lo que una partición del 80% para entrenamiento y 20% para prueba entrega una muestra suficientemente amplia para entrenar los modelos y, al mismo tiempo, reservar un subconjunto independiente para evaluar su desempeño final. Abarca una gran cantidad de datos y es capaz de evaluar generalización sobre observaciones no usadas para entrenar.

Si bien la validación cruzada permite obtener estimaciones más estables en bases de datos pequeñas o medianas, en este caso habría implicado repetir el entrenamiento de cada modelo varias veces sobre un volumen de datos considerable, aumentando la complejidad computacional y dificultando la comparación práctica entre múltiples algoritmos. Además,

algunos modelos evaluados que se presentarán más adelante, como *KNN*, *Random Forest* y redes neuronales, ya presentan un costo de procesamiento relevante, por lo que una estrategia de partición fija permitió mantener un flujo de modelamiento más eficiente sin perder representatividad en la evaluación.

La partición utilizada de 80% para entrenamiento y 20% para prueba resultó consistente en todos los modelos evaluados. Se utilizó una semilla aleatoria igual a 45 en los modelos lineales, definida previamente, con el objetivo de asegurar reproducibilidad en la división de los datos y mantener una muestra estable para comparar resultados. De esta manera, las métricas obtenidas por cada modelo se calcularon sobre el mismo conjunto de prueba, evitando que las diferencias observadas se deban a cambios en la partición de datos.

El preprocesamiento fue integrado dentro de *pipelines*, con el fin de ordenar y automatizar las transformaciones aplicadas antes del entrenamiento. En particular, las variables numéricas fueron escaladas para evitar que diferencias de magnitud afectaran a modelos sensibles a la escala, mientras que las variables categóricas fueron transformadas mediante codificación *dummy*, utilizando *drop_first=True*, para evitar la trampa asociada y reducir redundancia entre variables. El uso de *pipelines* permitió empaquetar estos pasos junto con cada modelo, mejorando la eficiencia del flujo de trabajo, reduciendo errores manuales y asegurando que las mismas transformaciones se aplicaran de manera consistente durante el entrenamiento y la evaluación.

Pasando a los modelos, si bien la regresión lineal no corresponde al clasificador más adecuado para variables dicotómicas, se incorporó como modelo base para evaluar el comportamiento de métodos lineales frente al conjunto de datos. Adicionalmente, se consideraron variantes regularizadas mediante penalizaciones L1 (*Lasso*), L2 (*Ridge*) y

Elastic Net, con el propósito de controlar el sobreajuste y analizar la estabilidad del modelo ante una base de alta dimensionalidad.

El primero fue el de regresión lineal el cual no necesita regularización al ser el más simple de todos, pues es sólo una multiplicación y suma de variables independientes X para predecir la variable dependiente Y . Se implementó mediante el estimador *SGDRegressor* que usa un modelo lineal sin regularización entrenado mediante descenso de gradiente estocástico. Aunque el problema principal corresponde a una tarea de clasificación binaria, este modelo se incorporó como referencia base para analizar el comportamiento de un enfoque lineal simple frente a los modelos posteriores. En este caso, la variable objetivo *isFraud* fue tratada numéricamente, permitiendo obtener una predicción continua que posteriormente puede interpretarse como una aproximación al puntaje de riesgo de fraude.

Dentro de los hiper parámetros más relevantes, se utilizó la pérdida cuadrática (*loss="squared_error"*) y sin penalización (*penalty=None*), con el objetivo de disponer de un modelo lineal base para comparar contra técnicas regularizadas y modelos no lineales. Se definió un máximo de 1000 iteraciones y una tolerancia de $1e-3$ para controlar la convergencia del entrenamiento. Además, se utilizó *random_state=45* para mantener la reproducibilidad de los resultados.

El segundo modelo lineal evaluado correspondió a una regresión *Ridge* con penalización L_2 . Al igual que en la regresión lineal simple, se utilizó *loss="squared_error"* por lo que el modelo busca minimizar el error cuadrático entre los valores observados y las predicciones. La diferencia principal está en *penalty="l2"*, que incorpora una penalización sobre el tamaño de los coeficientes del modelo, aplicando la regularización del modelo. Esta reduce la magnitud de los parámetros, ayudando a controlar el sobreajuste y haciendo que el

modelo sea más estable frente a variables correlacionadas o de alta dimensionalidad. Se utilizó α igual a 0.0001 como penalización baja, permitiendo comparar el efecto de la regularización sin restringir excesivamente el modelo. La configuración se completó con 1000 max_iter , $1e-3$ de tolerancia y una semilla de 45, manteniendo los mismos criterios de convergencia y reproducibilidad utilizados en los demás modelos lineales.

El tercer modelo lineal evaluado correspondió a una regresión *Lasso* con penalización L1. Al igual que en los modelos anteriores, se utilizó una función de pérdida basada en error cuadrático, por lo que el modelo busca minimizar la diferencia entre los valores observados y las predicciones generadas. La principal diferencia se encuentra en la regularización L1, la cual penaliza la suma de los valores absolutos de los coeficientes. Esta característica permite no solo controlar el sobreajuste, sino también reducir la complejidad del modelo, ya que algunos coeficientes pueden disminuir hasta cero, dejando fuera variables con menor contribución relativa. Además, se definió un máximo de 1000 iteraciones, una tolerancia de convergencia de $1e-3$ y una semilla aleatoria igual a 45 para asegurar resultados reproducibles.

El cuarto modelo lineal evaluado correspondió a *Elastic Net*, una técnica de regularización que combina las penalizaciones L1 y L2 utilizadas en *Lasso* y *Ridge*, respectivamente. Al igual que en los modelos anteriores utilizando una función de pérdida cuadrática, por lo que el objetivo del modelo fue minimizar la diferencia entre los valores reales y las predicciones generadas. La principal ventaja de *Elastic Net* es que permite equilibrar dos efectos: por un lado, la regularización L1 puede reducir algunos coeficientes hasta cero, favoreciendo una estructura más simple; por otro lado, la regularización L2 ayuda a controlar la magnitud general de los coeficientes, entregando mayor estabilidad cuando

existen variables correlacionadas. La intensidad de la penalización se fijó en un valor bajo, mientras que la proporción entre L1 y L2 se definió en 0,5, generando un equilibrio entre ambos tipos de regularización. Además, se utilizó un máximo de 1000 iteraciones, una tolerancia de convergencia de 0,001 y una semilla aleatoria igual a 45 para asegurar consistencia y reproducibilidad en el entrenamiento.

Así fue como se corrió un código completo con un ciclo que recorría cada uno de los modelos lineales y fue imprimiendo los resultados uno a uno. Las métricas más importantes fueron el tiempo de procesamiento para obtener los resultados, el error cuadrático medio o MSE para ver el efecto de la varianza, el R^2 o coeficiente de determinación que presenta la dependencia entre variables, y el valor del AUC del ROC a partir de predicciones y clases reales. Aunque no sean modelos de clasificación, el AUC igual puede aportar información importante para revisar la factibilidad de los resultados y que tanto se alejan estos modelos de uno categórico que se dedican a esta clase de casos.

La siguiente tabla resume los principales hallazgos de cada regresión y compara los indicadores para sacar una primera conclusión de estos primeros entrenamientos.

Tabla 10

Comparación de resultados de regresiones lineales y de regularización

modelo	ROC_AUC	MSE	R2	tiempo_entrenamiento_seg
ElasticNet_L1_L2	0.582807	4,69E+28	-1,39E+30	1412.58
Lineal_sin_regularizacion	0.570876	2,96E+28	-8,77E+29	59.095
Ridge_L2	0.507327	2,01E+28	-5,94E+29	53.024
Lasso_L1	0.501084	3,58E+28	-1,06E+30	60.194

Elaboración propia.

Los resultados obtenidos para los modelos lineales muestran un desempeño limitado en la detección de fraude, especialmente al compararlos con modelos de clasificación más adecuados para una variable objetivo-binaria. En términos de **ROC_AUC**, el mejor resultado lo obtuvo *ElasticNet_L1_L2*, con un valor de 0,5828, seguido por el modelo lineal sin regularización, con 0,5709. Si bien ambos superan levemente el comportamiento aleatorio, cuyo valor de referencia es 0,5, su capacidad discriminante sigue siendo baja. Esto indica que, bajo esta formulación, los modelos lineales solo logran separar débilmente las transacciones fraudulentas de las no fraudulentas.

Uno de los aspectos más llamativos es que la incorporación de regularización no produjo una mejora clara en todos los casos. *Ridge* y *Lasso* obtuvieron valores de **ROC_AUC** cercanos a 0,50, específicamente 0,5073 y 0,5011, respectivamente, lo que equivale prácticamente a una clasificación aleatoria. Esto sugiere que, para esta configuración, las penalizaciones L2 y L1 por separado restringieron demasiado la capacidad del modelo para capturar diferencias entre clases, o bien que la relación entre las variables explicativas y el fraude no puede ser representada adecuadamente mediante una estructura lineal simple, confirmando que un modelo de esta índole no es el adecuado para predecir variables categóricas, a pesar de que los valores de *isFraud* eran números (ceros y unos).

Respecto al MSE y al R^2 , los resultados refuerzan las limitaciones de utilizar modelos de regresión lineal para un problema de clasificación binaria. Los valores de error cuadrático medio son extremadamente altos y los valores de R^2 son fuertemente negativos, lo que indica que las predicciones continuas generadas por estos modelos no se ajustan adecuadamente a la variable objetivo. En este contexto, un R^2 negativo significa que el modelo tiene un desempeño peor que una predicción promedio simple, por lo que esta métrica no resulta

especialmente favorable para evaluar el problema de fraude. Esto era esperable, ya que estos modelos pueden generar predicciones continuas no acotadas.

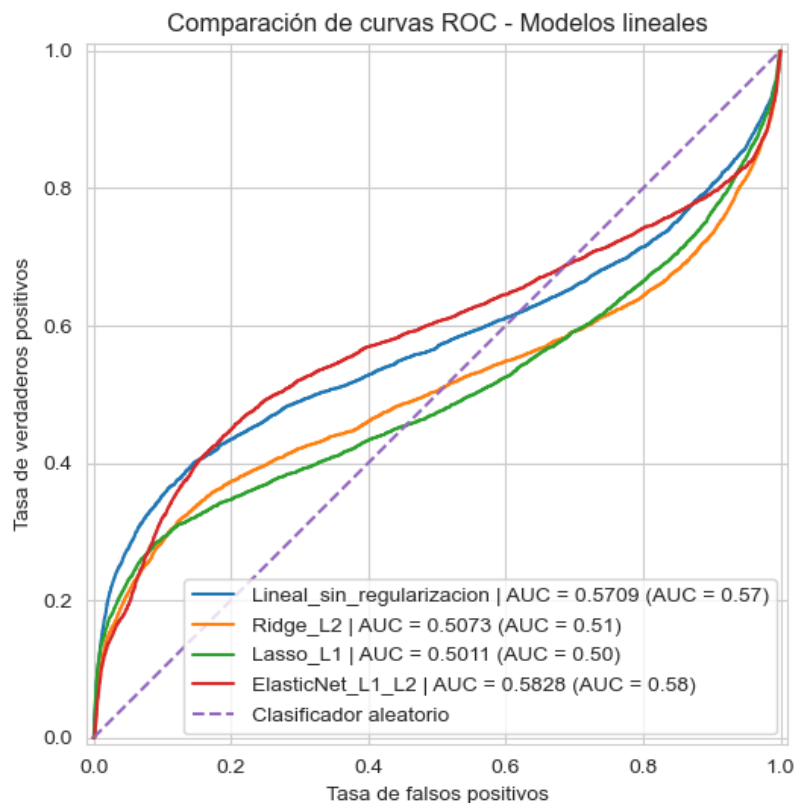
En contraste, *Elastic Net* fue el modelo con mejor desempeño relativo, probablemente porque combina los efectos de *Lasso* y *Ridge*. La penalización L1 permite reducir el peso de variables menos relevantes, mientras que la penalización L2 entrega mayor estabilidad frente a variables correlacionadas. Por lo tanto, *Elastic Net* puede considerarse el mejor modelo dentro de este grupo, pero no necesariamente un modelo competitivo frente a enfoques más flexibles.

En cuanto al tiempo de entrenamiento, todos los modelos presentaron tiempos relativamente similares, entre aproximadamente 41 y 49 segundos. *Ridge* fue el más rápido, seguido por *Elastic Net* y *Lasso*, mientras que el modelo sin regularización fue el más lento. Sin embargo, las diferencias de tiempo no son el factor principal de decisión, ya que el rendimiento predictivo fue bajo en todos los casos. Estos modelos permiten establecer una línea base simple y comparar el efecto de distintas formas de regularización, pero los resultados muestran que el fraude en transacciones digitales requiere modelos con mayor capacidad para capturar relaciones no lineales, interacciones entre variables y patrones complejos presentes en los datos.

Finalmente, el siguiente gráfico expone las tres curvas **ROC_AUC** generadas a partir de las pruebas hechas con los modelos lineales. A simple vista, es evidente que las curvas al inicio están por encima de la línea límite del 50% de clasificación arbitraria, pero comienza a variar el AUC en cada modelo hasta interferir su capacidad de predicción. En consecuencia, se podría clasificar a cualquier valor como fraude probable, cuando podrían ser flujos comunes.

Figura 37

Curvas ROC de los cuatro modelos lineales analizados



Nota: Se observa que cada curva traspasa el límite aleatorio y disminuye su AUC. Elaboración propia.

La línea diagonal punteada representa un clasificador aleatorio, con un **AUC** de referencia cercano a 0,5. Por lo tanto, mientras más se aleje una curva hacia la esquina superior izquierda y mayor sea su área bajo la curva, mejor será la capacidad discriminante del modelo.

En términos generales, las curvas muestran que los modelos lineales presentan un desempeño limitado, ya que todas se mantienen relativamente cercanas a la línea aleatoria. Sin embargo, existen diferencias entre ellos. El modelo *ElasticNet - L1 - L2* obtiene el mejor resultado global, con un AUC de 0,5828, ubicándose por encima del resto durante una parte

importante del recorrido de la curva. Esto confirma que la combinación de regularización L1 y L2 logra una separación algo más efectiva entre clases que las regularizaciones aplicadas de forma individual.

El modelo lineal sin regularización presenta el segundo mejor desempeño, con un AUC de 0,5709. En la parte inicial de la curva, especialmente cuando la tasa de falsos positivos es baja, muestra una subida relativamente rápida, lo que indica cierta capacidad para identificar verdaderos positivos sin aumentar excesivamente los falsos positivos.

Por otro lado, los modelos *Ridge* y *Lasso* muestran curvas muy cercanas a la diagonal aleatoria, con AUC de 0,5073 y 0,5010, respectivamente. Esto implica que, bajo esta configuración, ambas regularizaciones por separado prácticamente no logran diferenciar entre fraude y no fraude. En particular, *Lasso* presenta el desempeño más débil, lo que puede deberse a que la penalización L1 redujo demasiado la contribución de algunas variables, afectando la capacidad del modelo para capturar patrones relevantes.

En síntesis, el orden de desempeño observado es: *Elastic Net* como mejor modelo lineal, seguido por la regresión lineal sin regularización, luego *Ridge* y finalmente *Lasso*. Aun así, los valores de AUC obtenidos son bajos, lo que evidencia que estos enfoques lineales no capturan adecuadamente la complejidad del fraude transaccional. Por ello, estos modelos cumplen principalmente un rol de línea base, permitiendo contrastar posteriormente el desempeño de modelos más flexibles y no lineales.

7.3. Análisis de Clusters con K-means

Antes de pasar a los modelos de clasificación, se realizó una agrupación o *clusterización* (así se le denomina en el área) del *dataset* para revisar los conjuntos de datos

que tengan mayores características en común, puedan ser relevantes para el sesgo de los modelos y poder revisar relaciones relevantes entre las variables. Para esto se utilizó la librería de *K Means* para definir el K óptimo de *clusters* en los que se puedan agrupar las observaciones de cada variable dentro del espacio en el que se inmersa. Esta técnica no supervisada permitirá complementar a los anteriores análisis revisados de la matriz de correlación y PCA, aunque no quiere decir que se agregó como parámetro nuevo a los modelos. Sirve para obtener información nueva que pueda ser necesaria para explicar los resultados de algunos de los modelos y definir mejor los hiper parámetros de cada uno.

La partición de los datos se definió con 80.000 observaciones y una muestra estratificada de la variable Y para un mejor proceso, al igual que la semilla de 45 que asegura reproducibilidad. Para la aplicación de *K-Means* se evaluaron distintos valores de (K), desde 2 hasta 10 *clusters*, utilizando los datos previamente procesados. En cada entrenamiento se configuró el modelo con inicialización *k-means++*, lo que permite seleccionar centroides iniciales de manera más eficiente y reducir la probabilidad de obtener agrupaciones poco representativas. Además, se definieron 20 inicializaciones distintas para cada valor de (K), de modo que el algoritmo pudiera elegir la mejor solución entre varias ejecuciones y disminuir la dependencia de una única inicialización. Se estableció un máximo de 300 iteraciones por entrenamiento, el algoritmo de optimización *lloyd* y una semilla aleatoria igual a 42, con el objetivo de asegurar reproducibilidad en los resultados obtenidos.

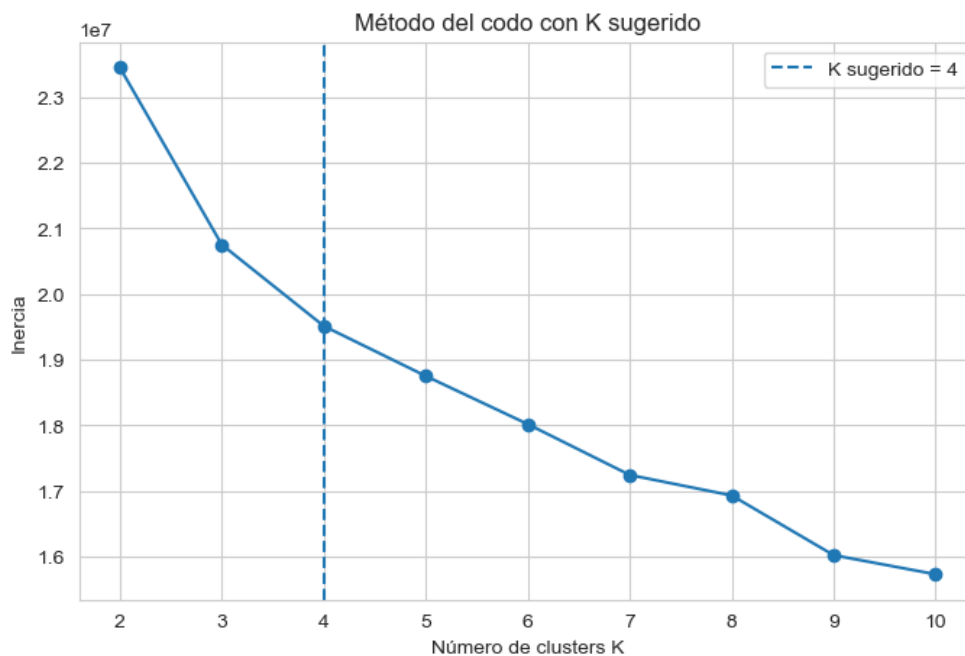
Al ejecutar el modelo para los distintos valores de (K), se observó una disminución progresiva de la inercia a medida que aumentaba el número de *clusters*. Esto era esperable, ya que, al incrementar la cantidad de grupos, las observaciones pueden quedar más cerca de sus centroides correspondientes, reduciendo la suma de distancias internas. La inercia pasó

de aproximadamente 23,46 millones con $K=2$ a 15,73 millones con $K=10$, confirmando un comportamiento decreciente correcto. No obstante, la reducción no fue uniforme: las mayores caídas se produjeron en los primeros valores de (K), especialmente entre $K=2$ y $K=3$, lo que indica que agregar más *clusters* sigue reduciendo la inercia, pero con beneficios marginales más bajos.

Para poder resolver lo de encontrar este número K , se optó por utilizar el método del codo o *Elbow Method* por su traducción en inglés. El siguiente gráfico presenta lo desarrollado por el método, donde se observa una disminución progresiva de la inercia con el aumento de *clusters*, excepto unas leves bajadas abruptas y una pequeña variación donde se encontró el óptimo.

Figura 38

Método del codo con el K óptimo encontrado



Elaboración propia.

La lógica del método consiste en identificar el punto donde la curva deja de disminuir de forma pronunciada y comienza a aplanarse, ya que ese valor representa un equilibrio entre compactación de los grupos y simplicidad del modelo. En este caso, la curva muestra una caída fuerte hasta aproximadamente $K=4$, luego de lo cual la reducción de la inercia se vuelve más gradual. Por esta razón, se seleccionó este valor como número sugerido de *clusters*, ya que permite capturar una parte importante de la estructura de agrupamiento sin aumentar innecesariamente la complejidad del análisis. Es importante señalar que este valor no corresponde al mínimo absoluto de inercia, ya que esta siempre disminuye al aumentar (K), sino al punto donde la mejora adicional comienza a ser menos significativa.

Ya con este número, se procedió a obtener la información relevante de cada *cluster* con el paquete de *MiniBatchKMeans* que utiliza toda la base de datos, en este caso los óptimos, para actualizar centroides de forma incremental y actualiza los conjuntos con cada *batch*. Esto redujo el tiempo de procesamiento y permitió obtener las etiquetas de cada uno para resumir lo relevante en una tabla en la que aparece el tamaño del *cluster*, su tasa de fraude y el porcentaje de muestra de las variables correspondiente a cada uno.

Tabla 11

Resumen de resultados de clusters K-means

cluster_kmeans	n	fraudes	tasa_fraude	porcentaje_muestra
0.0	26502.0	190.0	0.72%	33.13%
1.0	32431.0	1347.0	4.15%	40.54%
2.0	12008.0	234.0	1.95%	15.01%
3.0	9059.0	1028.0	11.35%	11.32%

Elaboración propia.

La tabla presenta una segmentación de la muestra en cuatro *clusters* con comportamientos diferenciados respecto al fraude. Es importante considerar que el algoritmo no utilizó la variable objetivo *isFraud* para formar los grupos, ya que *K-Means* es un método no supervisado. El *cluster* 1 concentra la mayor cantidad de observaciones, con 32.431 registros, equivalente al 40,54% de la muestra. Además, presenta el mayor número absoluto de fraudes, con 1.347 casos, aunque su tasa de fraude es de 4,15%. Esto indica que se trata de un grupo relevante desde el punto de vista operativo, ya que por su tamaño acumula una parte importante de los fraudes totales. En contraste, el *cluster* 3 representa solo el 11,32% de la muestra, pero presenta una tasa de fraude de 11,35%, muy superior al resto de los grupos. Este resultado lo posiciona como el segmento de mayor riesgo relativo, ya que concentra proporcionalmente más fraudes a pesar de tener menos observaciones.

Luego, se generó el gráfico de barras la tasa de fraude de cada grupo que es bien similar a lo encontrado en la tabla. El *cluster* 3 lidera a los demás por bastante, pues su tasa supera el 10%, muy por encima del 1 que lo sigue con un poco más de un 4,2%. El promedio bien sesgado hacia abajo debido a esta diferencia se fijó en 3,8% aproximado con los conjuntos 0 y 2 debajo, por lo que su incidencia en encontrar fraudes es menor, especialmente el cero con menos del 1% de fraudes probables.

Figura 39

Tasa de fraude por cluster



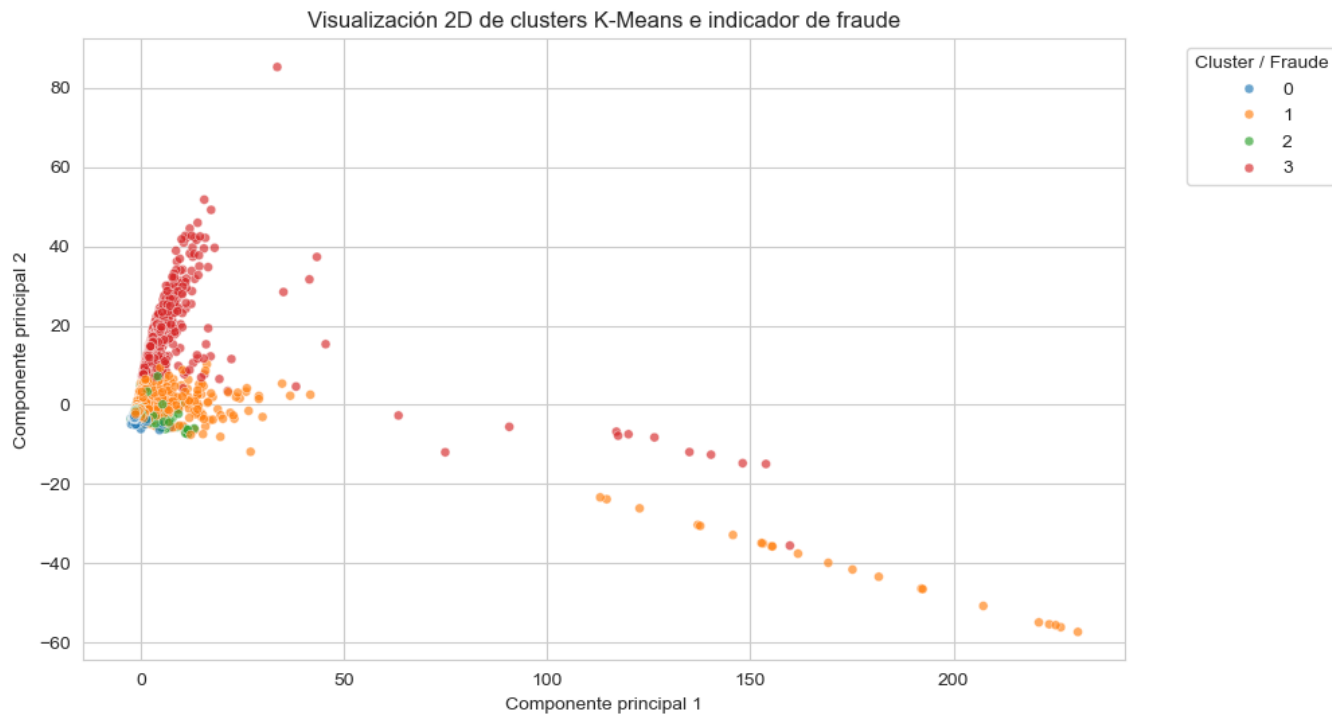
Nota: La tasa promedio es la línea punteada visible. Elaboración propia.

Se realizaron dos gráficos relevantes para mostrar a simple vista la distribución de los *clusters* en 2D. Uno para ver su disposición en el espacio y como se fueron agrupando según los fraudes probables o cercanos, y otro respecto a los fraudes efectivamente encontrados en cada uno frente a los movimientos normales mayormente atípicos.

Hay que destacar que los colores son los separadores principales de cada *cluster* con su significado en la esquina superior derecha. Además, se utilizó PCA para transformar la base a dos componentes para una visualización rápida y clara de las distribuciones señaladas.

Figura 40

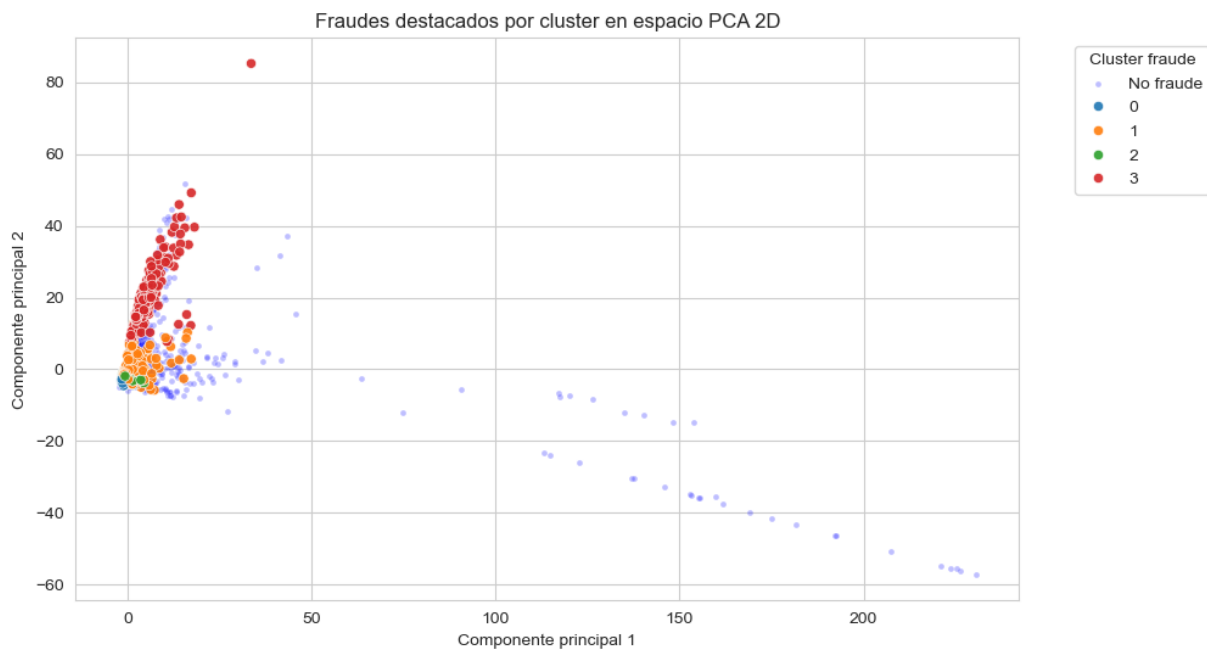
Visualización 2D de clusters K-Means e indicador de Fraude



Elaboración propia.

Figura 41

Fraudes destacados por cluster en espacio PCA 2D



Elaboración propia.

En la primera visualización se observa que los grupos no se encuentran completamente separados, ya que existe una zona de alta concentración cerca del origen donde varios *clusters* se superponen. Esto es esperable, considerando que la proyección PCA en dos componentes resume solo una parte de la información original. Aun así, se distinguen ciertos patrones: algunos grupos se ubican principalmente en la zona inferior y compacta del gráfico, mientras que otros se extienden hacia valores más altos del segundo componente principal o hacia valores más extremos del primer componente. Esto indica que *K-Means* logró identificar segmentos con diferencias estructurales en el espacio de variables transformadas, aunque no necesariamente con fronteras visualmente perfectas en dos dimensiones.

En la segunda visualización, donde se destacan las observaciones fraudulentas, se aprecia que los fraudes no se distribuyen de manera uniforme en todo el espacio, sino que aparecen con mayor concentración en ciertos sectores y *clusters* específicos. En particular, se observa una presencia relevante de fraudes en el grupo que se proyecta hacia valores más altos del segundo componente principal, coherente con lo observado previamente en la tabla de tasas de fraude por *cluster*. También existen fraudes dispersos en zonas más alejadas del primer componente, aunque en menor concentración. Dado que la variable *isFraud* no fue utilizada para construir los *clusters*, este resultado es relevante descriptivamente, ya que muestra que la segmentación no supervisada permitió formar grupos que posteriormente presentan distintos niveles de exposición al fraude. Sin embargo, la superposición entre puntos fraudulentos y no fraudulentos confirma que el fraude no puede separarse únicamente

con dos componentes principales, por lo que estos gráficos deben interpretarse como apoyo visual y no como una regla directa de clasificación.

Para terminar esta etapa, se revisó la importancia de las variables categóricas en cada conjunto y el impacto que podrían tener en la función objetivo dada su naturaleza y frecuencia previamente revisada. La siguiente tabla señala los principales con los valores que más prevalecen y el porcentaje de cada categoría dominante inmersa.

Tabla 12

Importancia de las variables categóricas para cada cluster

cluster_kmeans	ProductCD	card4	card6	P_emaildomain	R_emaildomain	DeviceType	M1	M2	M3
0	W (100.00%)	visa (66.03%)	debit (85.07%)	gmail.com (39.37%)	Missing (100.00%)	Missing (100.00%)	T (72.48%)	T (65.79%)	T (57.48%)
1	W (66.37%)	visa (64.29%)	debit (67.41%)	gmail.com (40.43%)	Missing (71.72%)	Missing (67.64%)	Missing (51.33%)	Missing (51.33%)	Missing (51.33%)
2	W (96.75%)	visa (66.87%)	debit (82.97%)	gmail.com (32.28%)	Missing (97.00%)	Missing (96.94%)	T (68.40%)	T (60.44%)	T (53.25%)
3	C (95.56%)	visa (58.24%)	debit (58.47%)	gmail.com (40.72%)	gmail.com (41.53%)	desktop (48.18%)	Missing (100.00%)	Missing (100.00%)	Missing (100.00%)

Elaboración propia.

La tabla permite caracterizar los *clusters* a partir de las categorías predominantes en algunas variables categóricas seleccionadas. En lugar de incluir todas las variables, se consideraron aquellas que ayudaban a interpretar de forma más clara el perfil de cada grupo. Se observa que varios *clusters* comparten características frecuentes de la base, como W de *ProductCD*, tarjetas visa, tipo débito y presencia de valores faltantes en variables asociadas a correo receptor o dispositivo. Sin embargo, el *cluster 3* presenta un perfil más diferenciado, destacando una alta concentración de C (*ProductCD*), mayor presencia de *R_emaildomain = gmail.com*, uso de *desktop* y valores faltantes en las variables M1, M2 y M3. Este comportamiento es consistente con los resultados previos, donde dicho *cluster* mostraba una tasa de fraude más elevada. En este sentido, la tabla no busca explicar causalmente el fraude,

sino describir qué combinaciones categóricas predominan en cada segmento y cómo estas ayudan a interpretar las diferencias observadas entre *clusters*.

7.4. Modelo de clasificación supervisada: Entrenamiento, testeo y resultados

En esta sección se pondrán a prueba los 6 modelos de clasificación supervisada tradicionales y la red neuronal presentados en el marco teórico. De manera práctica, cada uno tendrá su propio apartado en el que se explicará brevemente su funcionamiento para recordarlo, se argumentarán los parámetros elegidos de cada uno y se expondrán los principales resultados obtenidos que serían la matriz de confusión, las métricas de clasificación, el valor del AUC, entre otros que se detallarán dependiendo del modelo.

Existen 360 variables, donde se definió la variable a predecir *Y* como la columna *isFraud* del *dataset* y las variables explicativas *X* como todas las demás. La partición en general fue de 80% entrenamiento y 20% testeo para evitar el *overfitting*, entregando resultados relativamente rápidos y coherentes con lo que debería conseguir un aprendizaje de este estilo. Las 334 numéricas fueron escaladas para mantener estables los órdenes de magnitud y no provocar alteraciones, mientras que en las 26 categóricas se evitó la trampa *dummy* gracias al comando de *drop_first = True* de *Python* para la disminución al máximo de la multicolinealidad o posibles sesgos que se enfrenten los modelos. Cabe recalcar que algunos utilizaron los componentes PCA por las similitudes con la base completa en su comportamiento y otorgar logros más eficientes en tiempo y métricas.

7.4.1. Regresión Logística

La regresión logística fue incorporada como uno de los primeros modelos de clasificación supervisada, debido a que permite estimar la probabilidad de que una

transacción pertenezca a la clase fraudulenta o no fraudulenta. A diferencia de una regresión lineal tradicional, este modelo está diseñado para problemas de clasificación binaria, por lo que resulta adecuado para trabajar con la variable objetivo *isFraud*, compuesta por las categorías 0 y 1.

Su funcionamiento se basa en combinar las variables explicativas de manera lineal y transformar ese resultado en una probabilidad entre 0 y 1. Posteriormente, a partir de un umbral de decisión, el modelo clasifica cada transacción como fraude o no fraude. En este contexto, la regresión permite contar con un modelo base interpretable, eficiente y útil para comparar el desempeño de métodos más complejos.

Dentro de los parámetros, la regresión logística utilizó regularización L2, con una intensidad estándar, con el objetivo de controlar la magnitud de los coeficientes y reducir el riesgo de sobreajuste en una base con alta dimensionalidad. Se seleccionó el optimizador *saga*, adecuado para conjuntos de datos grandes y matrices dispersas generadas por la codificación de variables categóricas. Además, se incorporó ponderación balanceada de clases, debido al fuerte desbalance entre transacciones fraudulentas y no fraudulentas, permitiendo que el modelo otorgara mayor importancia relativa a la clase minoritaria.

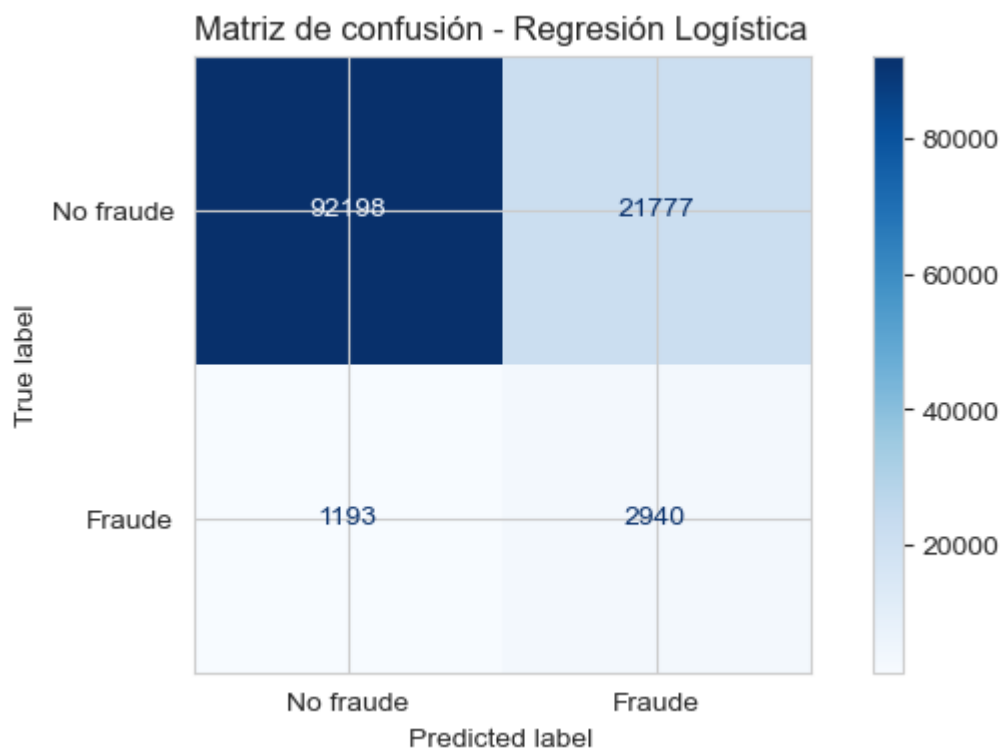
En el entrenamiento se definió un máximo de 800 iteraciones y una tolerancia de convergencia de 0,001, buscando equilibrar estabilidad del ajuste y eficiencia computacional. También se fijó una semilla aleatoria de 42 para asegurar reproducibilidad y se utilizaron todos los núcleos disponibles del procesador para acelerar el procesamiento. Finalmente, se registró el tiempo de ejecución del modelo

como una variable adicional de comparación, con el fin de evaluar no solo su desempeño predictivo, sino también su capacidad práctica para procesar una base de gran tamaño.

Después de 718,85 segundos, algo así como 12 minutos, de entrenamiento y prueba, se obtuvieron los primeros resultados del modelo. La figura presenta la matriz de confusión obtenida, seguido de las métricas calculadas con los valores correspondientes.

Figura 42

Matriz de confusión de la regresión logística



Elaboración propia.

Tabla 13

Métricas obtenidas de la regresión logística

modelo	accuracy	recall	precision	specificity	f1_score	roc_auc	tiempo_entrenamiento_seg
Regresión Logística	80.55%	71.13%	11.89%	80.89%	20.38%	84.26	718.85

Elaboración propia.

La matriz de confusión de la regresión logística muestra que el modelo clasificó correctamente 92.198 transacciones no fraudulentas y detectó 2.940 fraudes reales. Además, dejó sin detectar 1.193 fraudes, mientras que 21.777 transacciones legítimas fueron clasificadas erróneamente como fraude. Este comportamiento refleja un modelo sensible a la clase fraudulenta, ya que logra capturar una proporción importante de los fraudes reales, aunque a costa de generar una cantidad elevada de falsos positivos.

En términos métricos, el modelo obtuvo un *accuracy* de 80,55% y un *specificity* de 80,89%, lo que indica un desempeño razonable al clasificar correctamente transacciones no fraudulentas. Sin embargo, la métrica más relevante en este contexto es el *recall*, que alcanzó 71,13%, evidenciando que el modelo identificó aproximadamente siete de cada diez fraudes reales. Esto resulta positivo para un problema de detección de fraude, donde no detectar una transacción fraudulenta puede ser más costoso que revisar manualmente una transacción legítima.

Por otro lado, la precisión fue de solo 11,89%, lo que significa que, entre todas las transacciones clasificadas como fraude, una proporción baja correspondía

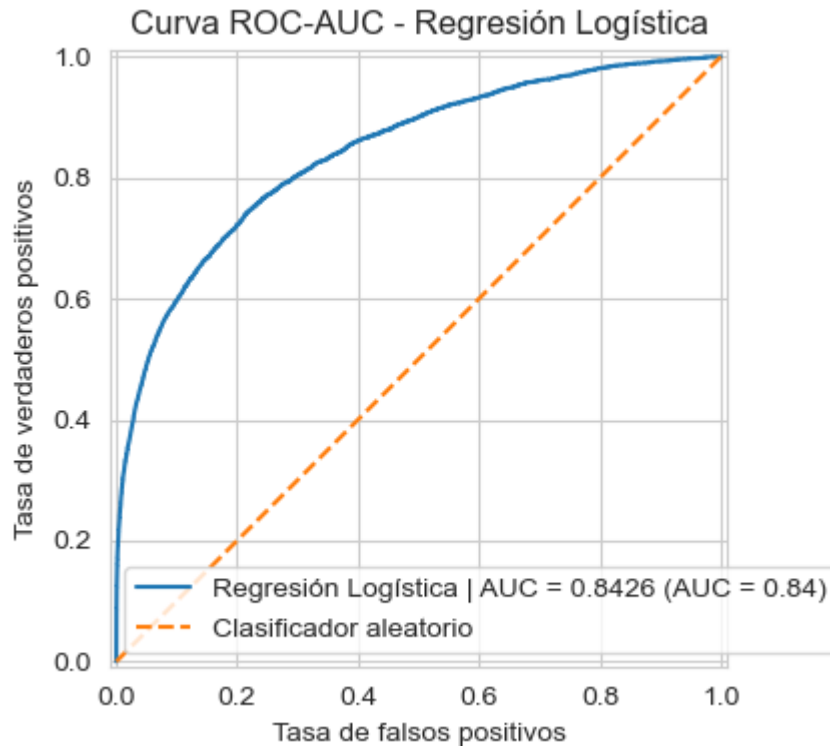
efectivamente a casos fraudulentos. Esto significa que el modelo comete algunos errores al clasificar transacciones normales como fraude, lo cual es válido por el tamaño y el desbalance de clases que generan mayor probabilidad de generar falsas alarmas. Esto se refleja también en un *F1-score* de 20,38%, afectado por la estrategia de ponderación balanceada, que aumenta la sensibilidad hacia la clase minoritaria. A pesar de ello, el **ROC-AUC** de 84,26% muestra que el modelo sí posee una buena capacidad general para ordenar las transacciones según su nivel de riesgo, aunque el umbral de clasificación utilizado genera muchos falsos positivos.

Respecto a la eficiencia computacional, el tiempo de entrenamiento fue de 718,85 segundos, lo que evidencia un costo de procesamiento considerable. Esto puede explicarse por el tamaño de la base, la alta dimensionalidad posterior al preprocesamiento y el uso del optimizador seleccionado.

Para cada modelo se graficó la curva ROC para tener a simple vista una mejor percepción del comportamiento del modelo y verificar que el AUC esté sobre el 50%. Sólo se presentará en este caso el gráfico, ya que sería redundante hacerlo para todos los modelos y tampoco la investigación quiere saturar de información irrelevante. De igual manera estos serán presentados en los **Anexos** para mayor detalle.

Figura 43

Curva ROC-AUC de la regresión logística



Elaboración propia.

La curva ROC-AUC de la regresión logística se ubica claramente por encima del clasificador aleatorio, lo que indica que el modelo posee una buena capacidad para diferenciar transacciones fraudulentas y no fraudulentas. El AUC obtenido fue de 0,8426, valor que refleja un desempeño adecuado en el ordenamiento de las transacciones según su riesgo de fraude. En términos prácticos, esto significa que el modelo logra capturar patrones relevantes para priorizar casos sospechosos, aunque su efectividad final depende del umbral utilizado para clasificar una transacción como fraude.

En síntesis, la regresión logística presentó un desempeño adecuado como modelo base de clasificación, destacando por su capacidad para detectar una

proporción importante de fraudes reales y por un **ROC-AUC** favorable. Sin embargo, su baja precisión evidencia una tendencia a generar muchos falsos positivos, por lo que, aunque resulta útil para priorizar transacciones sospechosas, no sería suficiente por sí sola como modelo final sin ajustes adicionales de umbral o comparación con algoritmos más robustos.

7.4.2. K-Nearest Neighbors (KNN)

El modelo K vecinos más cercanos, conocido como KNN, fue incorporado como un método basado en similitud entre observaciones. Su funcionamiento consiste en comparar una nueva transacción con los registros más cercanos del conjunto de entrenamiento y asignarle una clase según el comportamiento de sus vecinos más próximos. En este caso, si las transacciones cercanas presentan patrones asociados a fraude, el modelo tenderá a clasificar la nueva observación como fraudulenta.

A diferencia de la regresión logística, KNN no estima una relación lineal entre las variables y la probabilidad de fraude, sino que clasifica a partir de la cercanía entre observaciones dentro del espacio de variables procesadas. Esto permite capturar ciertos patrones locales que podrían no ser evidentes en modelos lineales. Sin embargo, también es un modelo sensible a la escala de las variables y al tamaño de la base, por lo que el preprocesamiento numérico y la reducción de dimensionalidad resultan especialmente importantes para mejorar su funcionamiento y eficiencia.

Para el modelo se trabajó con una submuestra del conjunto de entrenamiento y prueba, debido a que este algoritmo requiere calcular distancias entre observaciones, lo que puede volverse costoso cuando se trabaja con una base de gran tamaño. Por esta

razón, desde el conjunto de entrenamiento original se seleccionaron 80.000 observaciones, mientras que desde el conjunto de prueba se seleccionaron 30.000 observaciones. Esta decisión permitió reducir el tiempo de procesamiento del modelo sin perder una cantidad significativa de información para evaluar su comportamiento.

La selección de las muestras se realizó de forma estratificada, es decir, manteniendo la proporción original entre transacciones fraudulentas y no fraudulentas. Esto era especialmente importante debido al fuerte desbalance de la variable objetivo. Como resultado, tanto en entrenamiento como en prueba se mantuvo aproximadamente un 96,5% de transacciones no fraudulentas y un 3,5% de transacciones fraudulentas. Además, ambas bases quedaron con 358 variables explicativas, lo que confirma que se conservó la misma estructura de atributos utilizada en el proceso de modelamiento.

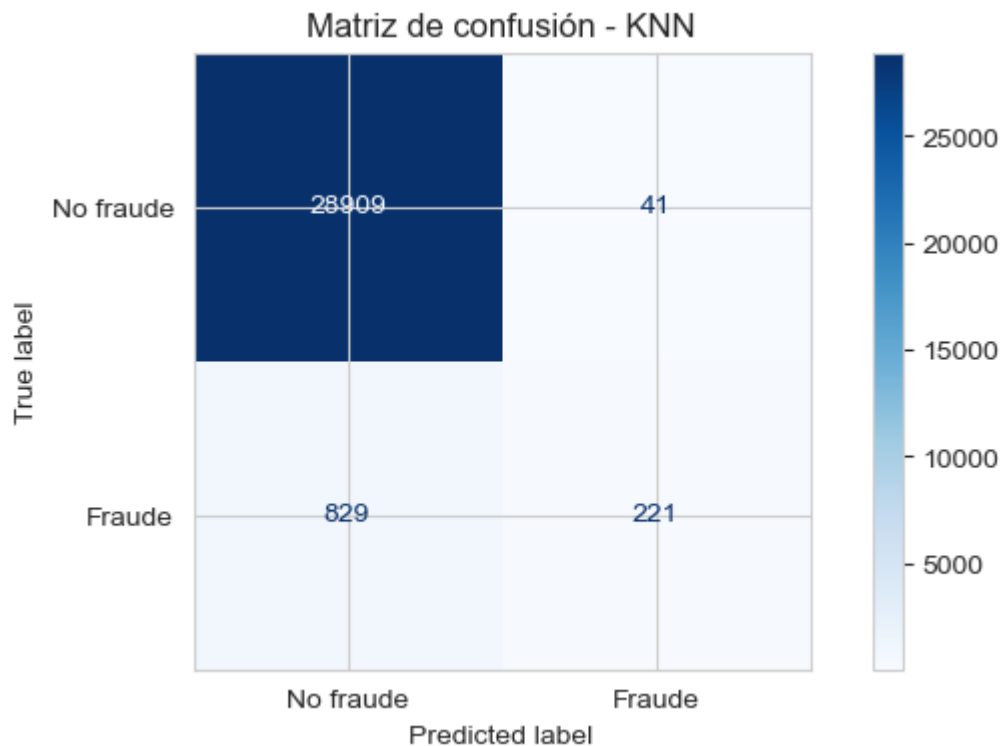
Por otro lado, se definieron 21 vecinos cercanos como criterio de clasificación. Este valor permite suavizar la decisión del modelo, evitando que una transacción sea clasificada únicamente por pocos casos cercanos que podrían ser atípicos. Además, al utilizar un número impar de vecinos se reduce la posibilidad de empates entre clases. La ponderación se realizó según la distancia, por lo que las observaciones más cercanas a la transacción evaluada tienen mayor influencia que aquellas ubicadas más lejos. Esta decisión resulta adecuada para KNN, ya que no todos los vecinos deberían aportar con el mismo peso cuando existen diferencias relevantes en cercanía y desbalance de clases.

Como medida de distancia se utilizó la métrica de *Minkowski* con parámetro 2, equivalente a la distancia euclidiana, una de las formas más comunes para medir similitud entre observaciones numéricas. También se habilitó el uso de todos los núcleos disponibles del procesador, con el objetivo de acelerar el cálculo de distancias y mejorar la eficiencia computacional. Finalmente, el modelo fue integrado dentro de un *pipeline* junto con el preprocesamiento.

Así se obtuvieron los siguientes resultados iniciales del modelo al ejecutar sobre la submuestra definida para entrenamiento y prueba.

Figura 44

Matriz de confusión inicial de KNN



Elaboración propia.

Tabla 14

Resultados iniciales de KNN.

modelo	accuracy	recall	precision	specificity	f1_score	roc_auc	tiempo_entrenamiento_seg	tiempo_prediccion_seg
KNN	97.1%	21.05%	84.35%	99.86%	33.69%	82.12%	7.92	7.52

Elaboración propia.

La matriz muestra que el modelo clasificó correctamente 28.909 transacciones no fraudulentas y cometió solo 41 falsos positivos, lo que se refleja en una especificidad muy alta de 99,86%. Sin embargo, en la clase fraudulenta el desempeño fue más limitado: de 1.050 fraudes reales presentes en la muestra de prueba, el modelo detectó 221 y dejó sin identificar 829. Esto explica que el *recall* haya sido de 21,05%, indicando que el modelo fue conservador al momento de clasificar transacciones como fraude. Ser más estrictos genera algoritmos más escépticos al momento de evaluar transacciones sospechosas para prevenir falsas alarmas, aunque eso signifique sacrificar algunas que sí lo eran, evidenciando el *trade-off* con la *precision*.

En términos generales, KNN obtuvo un *accuracy* de 97,10%, pero esta métrica debe interpretarse con cautela debido al fuerte desbalance de clases. Lo más destacable es su alta precisión de 84,35%, lo que significa que cuando el modelo predijo fraude, en la mayoría de los casos efectivamente correspondía a fraude, evitando el desbalance de clases y diferenciando la carga operativa en cada entrenamiento. El *F1-score* de 33,69% refleja este equilibrio parcial entre una precisión alta y una sensibilidad baja.

Por otro lado, el **ROC-AUC** de 82,12% muestra que el modelo sí posee una capacidad razonable para ordenar las transacciones según riesgo, aunque el umbral inicial de clasificación resultó demasiado estricto para detectar fraudes. Respecto a la eficiencia computacional, el modelo presentó un tiempo de entrenamiento de 7,92 segundos y un tiempo de predicción de 7,52 segundos, valores adecuados considerando que KNN requiere calcular distancias entre observaciones.

Con el objetivo de mejorar la clasificación inicial del modelo KNN, se evaluaron distintos umbrales de decisión entre 0,10 y 0,50. Esta etapa fue necesaria porque el umbral estándar de 0,50 generaba un modelo muy conservador: clasificaba pocos casos como fraude, obteniendo alta precisión, pero dejando sin detectar una proporción importante de fraudes reales. Por ello, se compararon distintos puntos de corte y se seleccionó aquel que maximizaba el *F1-score*, ya que esta métrica permite equilibrar *precision* y *recall*, aspecto especialmente relevante en un problema con fuerte desbalance de clases.

A partir de esto, los siguientes resultados muestran las métricas obtenidas para cada umbral y su evolución a medida que avanza este.

Tabla 15

Métricas obtenidas al evaluar distintos umbrales de decisión

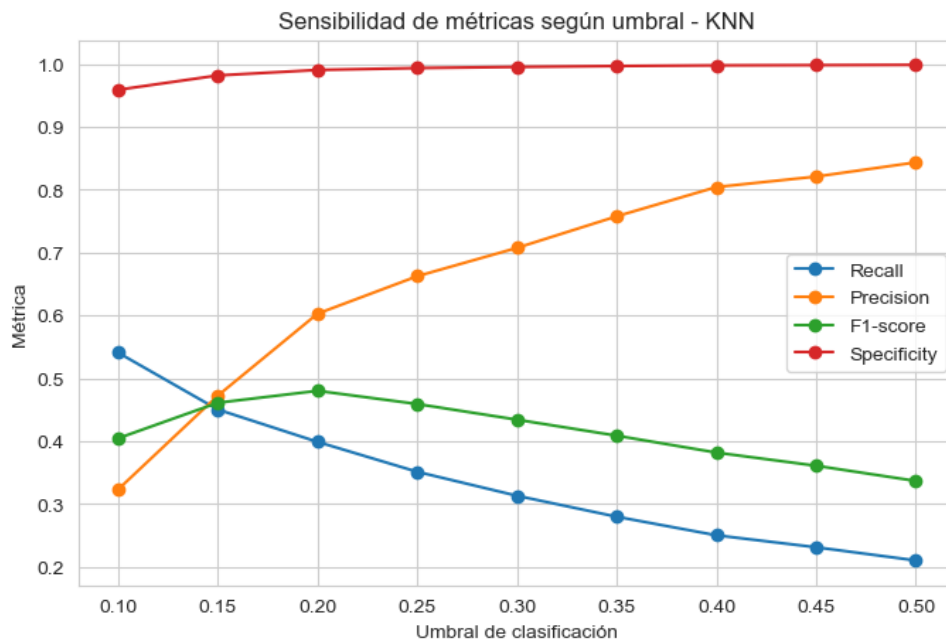
umbral	accuracy	recall	precision	specificity	f1_score	roc_auc
0.10	94.43%	54.1%	32.33%	95.89%	40.47%	82.12%
0.15	96.32%	45.05%	47.25%	98.18%	46.12%	82.12%
0.20	96.98%	39.9%	60.29%	99.05%	48.02%	82.12%

0.25	97.1%	35.14%	66.25%	99.35%	45.92%	82.12%
0.30	97.14%	31.33%	70.75%	99.53%	43.43%	82.12%
0.35	97.17%	28.0%	75.77%	99.68%	40.89%	82.12%
0.40	97.16%	25.05%	80.43%	99.78%	38.2%	82.12%
0.45	97.13%	23.14%	82.09%	99.82%	36.11%	82.12%
0.50	97.1%	21.05%	84.35%	99.86%	33.69%	82.12%

Elaboración propia.

Figura 45

Sensibilidad de métricas según distintos umbrales para KNN



Nota: Cada métrica está representada por un color. Elaboración propia.

Los resultados muestran que, al disminuir el umbral de validación, el modelo aumenta su capacidad para detectar fraudes, pero también incrementa la cantidad de

falsos positivos. Por ejemplo, con un umbral de 0,10 el *recall* alcanza 54,10%, pero la *precisión* disminuye a 32,33%, lo que implica una mayor cantidad de transacciones legítimas clasificadas erróneamente como fraude. En cambio, al utilizar umbrales más altos, como 0,50, la precisión aumenta hasta 84,35%, pero el *recall* cae a 21,05%, dejando pasar muchos fraudes reales. Este comportamiento refleja el intercambio habitual entre detectar más fraudes y mantener baja la cantidad de falsas alarmas.

El mejor equilibrio se obtuvo con un umbral de 0,20, donde el *F1-score* alcanzó su valor máximo de 48,02%. En este punto, el modelo logró una precisión de 60,29% y un *recall* de 39,90%, detectando 419 fraudes reales y reduciendo los falsos positivos a 276. Además, mantuvo una especificidad alta de 99,05% y una exactitud de 96,98%. El gráfico confirma esta evolución: a medida que aumenta el umbral, la precisión y la especificidad suben, mientras que el *recall* disminuye. Por ende, el ajuste de umbral permitió mejorar considerablemente el equilibrio del modelo KNN, pasando de un clasificador demasiado restrictivo a una alternativa más útil para detectar fraudes sin generar una cantidad excesiva de alertas falsas.

Se decidió mantener el umbral de 0,2 para KNN al maximizar el *F1-score* y ser una de las mejores alternativas para detectar fraudes. La matriz de confusión confirma estos valores, pues 28.674 transacciones fueron normales, mientras que 419 fueron fraudes reales. Su implementación podría ser primordial para la ciberseguridad de un banco, dada la cautela con el que es capaz de detectar este peligro, sin dejar de lado los 631 falsos negativos que predijo, por lo que igual se tiene que seguir mejorando su avance.

7.4.3. Support Vector Machine (SVM)

Support Vector Machine o SVM, fue incorporado como un método de clasificación supervisada capaz de separar las observaciones en distintas clases mediante la construcción de una frontera de decisión. Su funcionamiento se basa en encontrar una separación entre clases que maximice el margen entre los casos más cercanos a la frontera, lo que permite generar una clasificación más robusta frente a observaciones difíciles de separar.

En este proyecto se trabajó con una versión lineal de SVM, adecuada para bases de datos grandes y de alta dimensionalidad. Este enfoque resulta útil cuando existen muchas variables explicativas, como ocurre después del preprocesamiento y la codificación de variables categóricas. A diferencia de modelos basados en vecinos, SVM no clasifica comparando directamente cada nueva observación con todas las anteriores, sino que aprende una frontera de decisión durante el entrenamiento.

Para implementar SVM se utilizó *SGDClassifier* con pérdida tipo *hinge*, lo que permite aproximar un clasificador SVM lineal de forma eficiente en bases de datos grandes. Además, se incorporó regularización L2, con el objetivo de controlar la magnitud de los coeficientes y mejorar la estabilidad del modelo frente a variables correlacionadas o con alta dispersión.

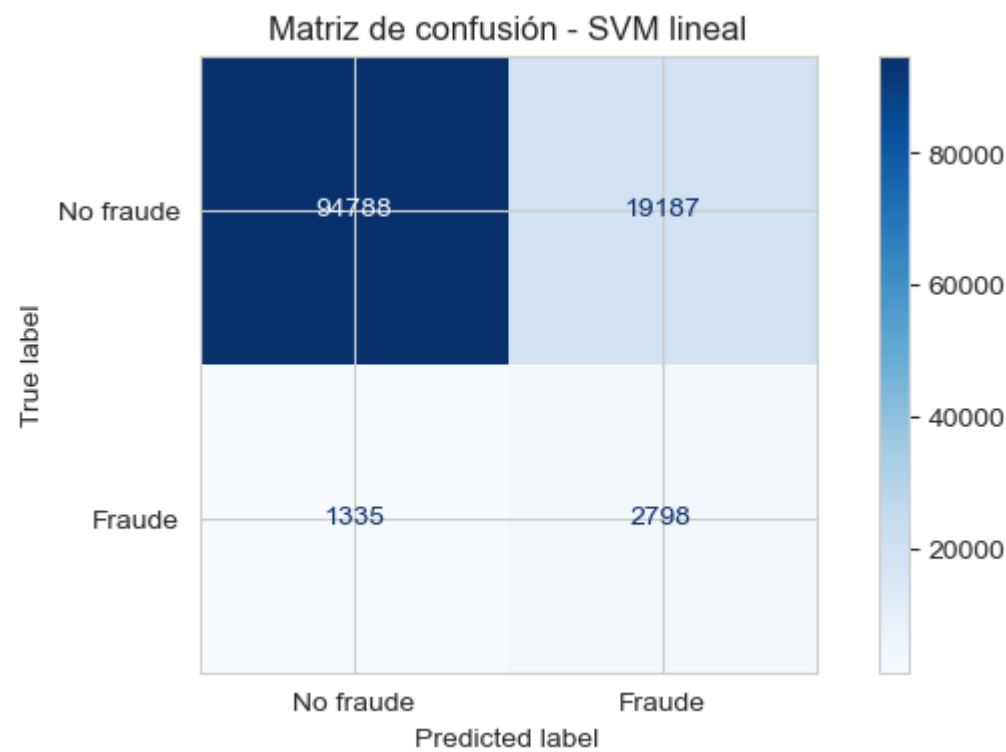
La intensidad de regularización se definió en un nivel bajo, permitiendo mantener capacidad predictiva sin restringir excesivamente la frontera de decisión. También se utilizó ponderación balanceada de clases, debido al fuerte desbalance entre transacciones fraudulentas y no fraudulentas, otorgando mayor peso relativo a la clase minoritaria

durante el entrenamiento. Finalmente, se estableció un máximo de 1000 iteraciones, una tolerancia de convergencia de 0,001, una semilla aleatoria igual a 42 para asegurar reproducibilidad y el uso de todos los núcleos disponibles del procesador para mejorar la eficiencia computacional.

Se consiguieron los siguientes resultados una vez ejecutados los códigos de entrenamiento del modelo, ya con las variables separadas y escaladas.

Figura 46

Matriz de confusión de SVM



Elaboración propia.

Tabla 16

Resultados aplicados a SVM

modelo	accuracy	recall	precision	specificity	f1_score	roc_auc	tiempo_entrenamiento_seg
SVM lineal	82.62%	67.7%	12.73%	83.17%	21.43%	83.45%	196.18

Elaboración propia.

La matriz de confusión del SVM lineal muestra que el modelo clasificó correctamente 94.788 transacciones no fraudulentas y 2.798 transacciones fraudulentas. Sin embargo, también generó 19.187 falsos positivos, es decir, transacciones legítimas clasificadas como fraude, y dejó sin detectar 1.335 fraudes reales. Esto refleja un comportamiento similar al observado en otros modelos sensibles a la clase minoritaria: logra identificar una parte importante de los fraudes, pero a costa de aumentar las alertas falsas.

En términos de métricas, el modelo obtuvo un *accuracy* de 82,62% y un *specificity* de 83,17%, lo que indica un desempeño razonable al reconocer transacciones no fraudulentas. Su *recall* fue de 67,70%, por lo que logró detectar cerca de dos tercios de los fraudes reales. No obstante, el *precision* alcanzó solo 12,73%, evidenciando que muchas de las transacciones clasificadas como fraude no correspondían efectivamente a casos fraudulentos. Lo complementa un *F1_score* de 21,43%, que muestra un equilibrio limitado entre detección de fraude y precisión de las alertas.

El **ROC_AUC** de 83,45% indica que el SVM lineal posee una buena capacidad general para ordenar las transacciones según su nivel de riesgo, aunque su desempeño práctico se ve afectado por la baja precisión. En cuanto al costo computacional, el tiempo de entrenamiento fue de 196,18 segundos, 3,27 minutos, menor que el de la regresión logística, pero superior al de modelos más simples. En resumen, el SVM lineal funcionó

como una alternativa competitiva dentro de los modelos base, con buena capacidad de separación global, aunque todavía con una cantidad importante de falsos positivos al momento de clasificar fraudes.

7.4.4. Árboles de decisión

El árbol de decisión fue incorporado como un modelo supervisado capaz de segmentar las transacciones mediante reglas sucesivas basadas en las variables explicativas. Su funcionamiento consiste en dividir la base en grupos cada vez más homogéneos, utilizando condiciones sobre las variables que permiten separar progresivamente las transacciones fraudulentas de las no fraudulentas. De esta manera, el modelo genera una estructura similar a un flujo de decisiones, donde cada nodo representa una regla y cada rama conduce a una clasificación final.

A diferencia de modelos lineales, el árbol de decisión puede capturar relaciones no lineales e interacciones entre variables sin requerir una especificación previa de dichas relaciones. Además, ofrece una ventaja importante en términos de interpretabilidad, ya que permite observar qué variables fueron utilizadas para realizar las principales divisiones del modelo. En el contexto de detección de fraude, esto resulta útil no solo para clasificar transacciones, sino también para comprender qué patrones o condiciones están asociados a un mayor riesgo dentro de la base analizada.

Para el árbol de decisión se utilizó el criterio de *Gini*, el cual permite seleccionar las divisiones que generan grupos más homogéneos respecto a la variable objetivo. En otras palabras, el modelo evalúa las variables disponibles y escoge aquellas que permiten separar mejor las transacciones fraudulentas de las no fraudulentas en cada nodo. Por esta

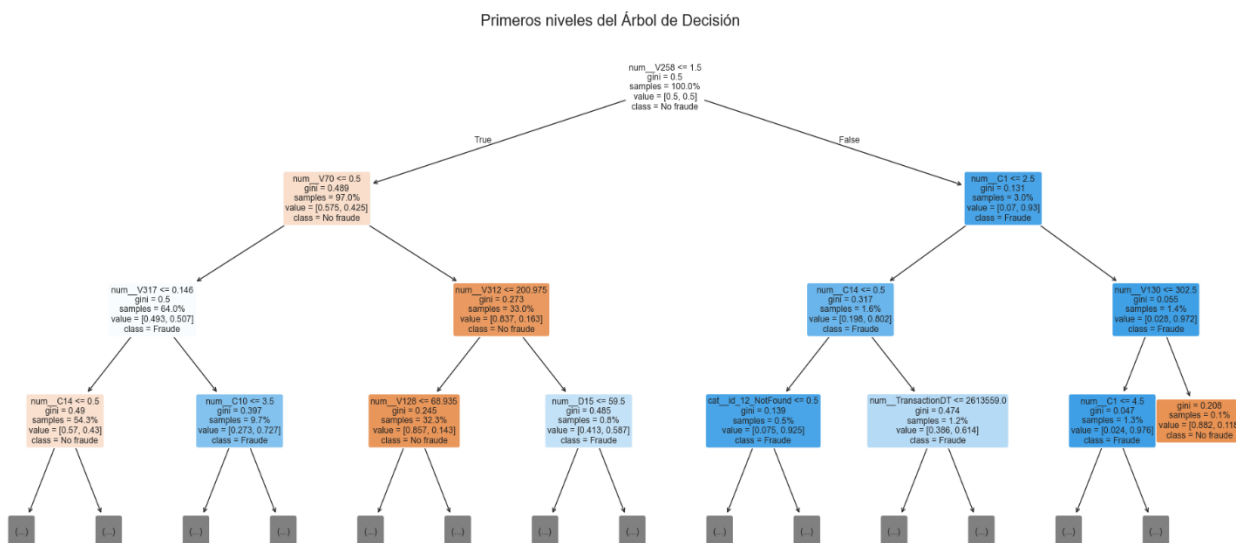
razón, el árbol no utiliza todas las variables con la misma importancia, sino que va priorizando aquellas que generan mayores mejoras en la clasificación.

Se limitó la profundidad máxima del árbol a 5 niveles, con el objetivo de evitar que el modelo creciera demasiado y memorizara patrones específicos de la base de entrenamiento. Además, se exigió un mínimo de 1000 observaciones para realizar una nueva división y un mínimo de 300 observaciones por hoja final. Estos parámetros ayudan a que las reglas generadas sean más generales y no dependan de grupos muy pequeños o atípicos. También se incorporó ponderación balanceada de clases, debido al desbalance entre fraudes y normales, permitiendo que el modelo considere con mayor peso los casos fraudulentos. Finalmente, se fijó una semilla aleatoria igual a 42 para asegurar reproducibilidad en los resultados. Esta configuración permitió obtener un árbol más controlado, interpretable y enfocado en identificar variables con mayor capacidad de segmentación.

El árbol de decisión generado permite observar de manera directa algunas de las variables que el modelo consideró más relevantes para segmentar las transacciones. En los primeros niveles del árbol se aprecia que la primera división se realiza con la variable V258, lo que indica que esta fue la variable inicial con mayor capacidad para separar los grupos según la clase objetivo. Posteriormente, aparecen variables como V70, C1, V317, V312, C14, V130 y otras variables numéricas y categóricas transformadas, mostrando que el modelo combina distintos tipos de información para construir reglas de clasificación. Además, debido al uso de ponderación balanceada, los nodos no solo reflejan la cantidad total de observaciones, sino también la importancia relativa entregada a la clase fraudulenta durante el entrenamiento.

Figura 47

Esquema de los primeros niveles del árbol de decisión

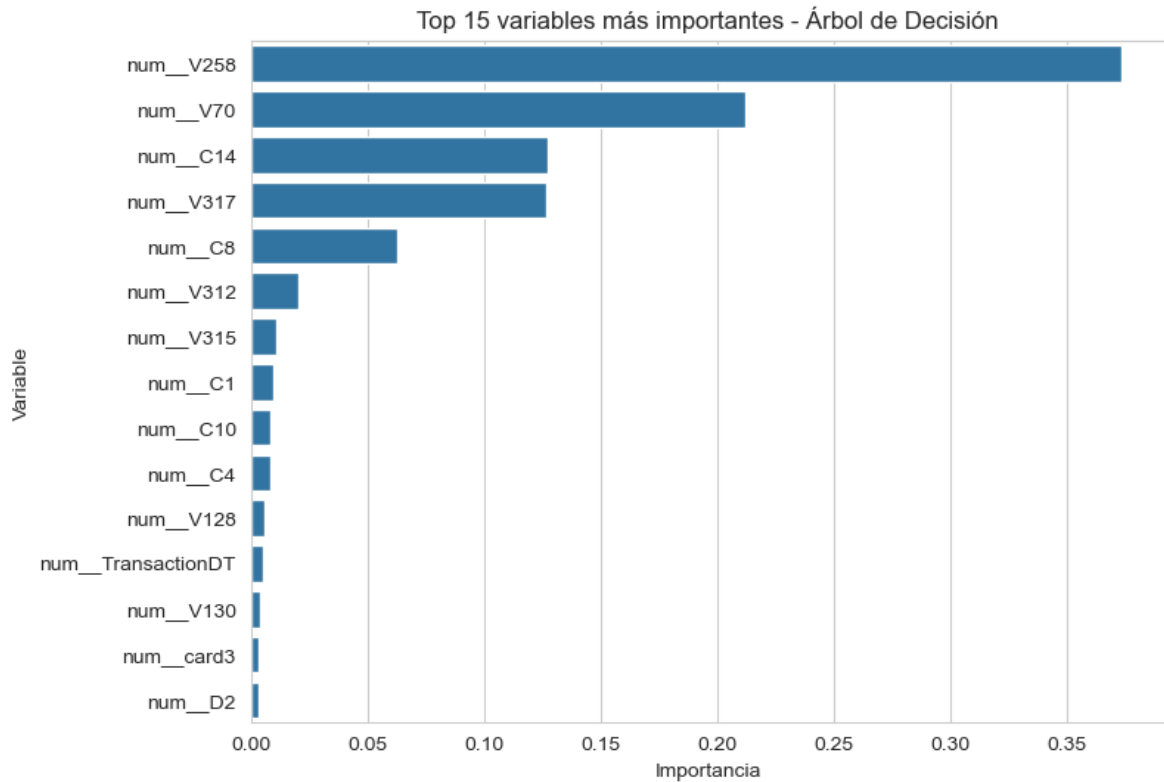


Nota: Sólo se incluyeron los cuatro primeros niveles para una mejor visualización. Elaboración propia.

Por otro lado, se generó el gráfico de importancia de variables que muestra que el árbol concentró gran parte de su capacidad predictiva en un número reducido de atributos.

Figura 48

Variables más importantes según árboles de decisión



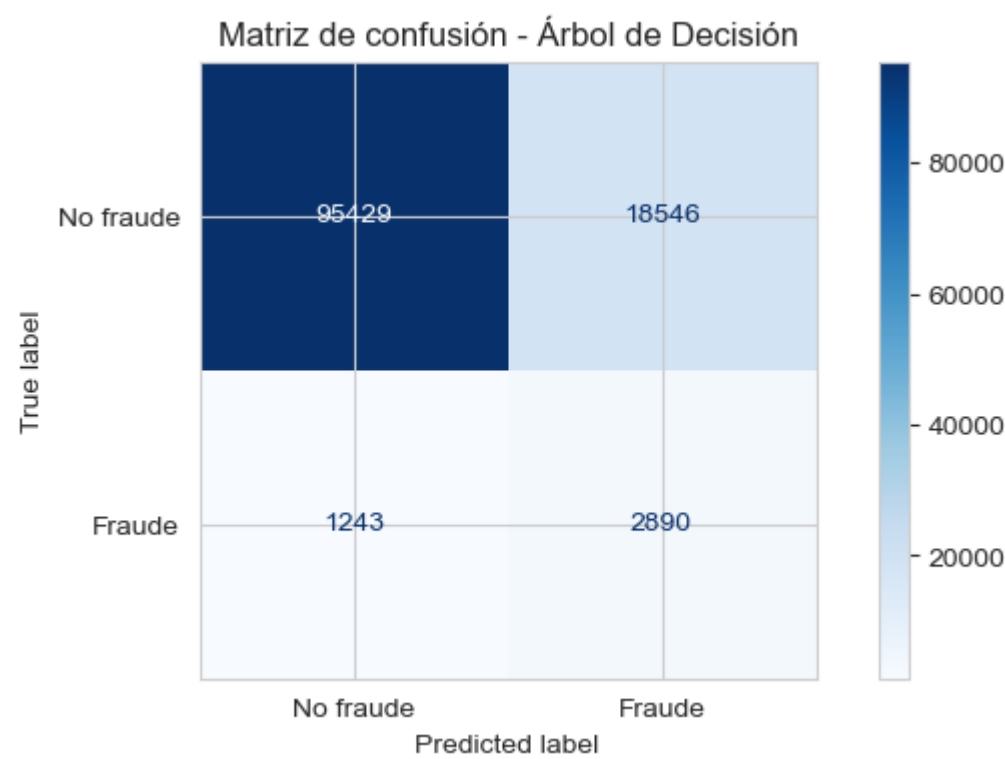
Elaboración propia.

La variable V258 aparece como la más importante por un margen considerable, seguida por V70, lo que indica que ambas fueron claves para realizar las principales divisiones del modelo. Luego destacan variables como C14 y V317, con una contribución intermedia, mientras que el resto de las variables presenta una importancia bastante menor. Este comportamiento sugiere que el árbol no distribuyó el peso de decisión de forma homogénea entre todas las variables, sino que identificó ciertos atributos específicos con mayor capacidad para segmentar transacciones fraudulentas y no fraudulentas. No obstante, esta importancia debe interpretarse como una medida interna del modelo y no como una relación causal directa con el fraude.

Después se pasó a obtener la matriz de confusión e indicadores significativos del modelo para la evaluación.

Figura 49

Matriz de confusión de árboles de decisión



Elaboración propia.

Tabla 17

Indicadores del árbol de decisión obtenido

modelo	accuracy	recall	precision	specificity	f1_score	roc_auc	tiempo_entrenamiento_seg
Árbol de Decisión	83.24%	69.92%	13.48%	83.73%	22.61%	82.59%	446.15

Elaboración propia.

En la matriz de confusión se observa que el árbol clasificó correctamente 95.429 transacciones no fraudulentas y detectó 2.890 fraudes reales. Sin embargo, también generó 18.546 falsos positivos y dejó sin detectar 1.243 fraudes. Esto muestra que el modelo logra capturar una proporción importante de los fraudes, aunque mantiene una cantidad considerable de alertas incorrectas. Su *recall* fue de 69,92%, lo que indica que identificó cerca de siete de cada diez fraudes reales, mientras que su *precision* fue de 13,48%, reflejando que muchas de las transacciones predichas como fraude correspondían realmente a casos no fraudulentos.

Se observa que, en los modelos vistos hasta ahora, la *precision* baja se repite lo que indica la importancia del desbalance de clases y cómo afecta la mayor cantidad de movimientos legítimos en el entrenamiento de los modelos, a pesar de haber utilizado una ponderación balanceada dentro de los parámetros. Aunque esto no los invalida, se debe tener en cuenta que ser más estrictos en la detección puede aumentar la cantidad de falsas alarmas que en una instancia real puede ser perjudicial si empieza a ser muy reiterativo.

Respecto a las métricas generales, el árbol alcanzó un *accuracy* de 83,24% y un *specificity* de 83,73%, mostrando un desempeño razonable en la clasificación de transacciones legítimas. El *F1_score* fue de 22,61%, ligeramente superior al de modelos lineales como regresión logística y SVM lineal, lo que sugiere un mejor equilibrio entre detección de fraude y precisión de las alertas. Por otro lado, el **ROC_AUC** de 82,59% indica una capacidad adecuada para ordenar las transacciones según su riesgo, aunque menor que la obtenida por otros modelos más robustos. En conjunto, el árbol de decisión

entrega un resultado útil e interpretable, destacando por mostrar reglas y variables relevantes.

7.4.5. Random Forest

Random Forest es un modelo de clasificación de ensamble basado en el uso conjunto de múltiples árboles de decisión. A diferencia de un árbol individual, que puede ser sensible a pequeñas variaciones en los datos, *Random Forest* construye varios árboles y combina sus resultados para obtener una predicción más robusta. En este caso, cada árbol analiza distintas combinaciones de observaciones y variables, y la clasificación final se obtiene a partir del voto agregado del conjunto de árboles.

Este enfoque permite capturar relaciones no lineales e interacciones entre variables, manteniendo al mismo tiempo una mayor estabilidad que un árbol de decisión simple. En el contexto de detección de fraude, *Random Forest* resulta especialmente útil porque puede identificar patrones complejos distribuidos en distintas variables, sin depender de una única secuencia de reglas. Además, permite analizar la importancia relativa de las variables utilizadas, lo que facilita interpretar qué atributos aportan más a la clasificación de transacciones fraudulentas y no fraudulentas.

Se utilizaron 150 árboles de decisión, lo que permite combinar múltiples clasificadores individuales y obtener una predicción más estable que la de un árbol aislado. Cada árbol fue construido utilizando el criterio de *Gini*, orientado a seleccionar divisiones que generen grupos más homogéneos respecto a la variable objetivo. Además, se limitó la profundidad máxima a 12 niveles, con el fin de controlar el crecimiento de los árboles y reducir el riesgo de sobreajuste.

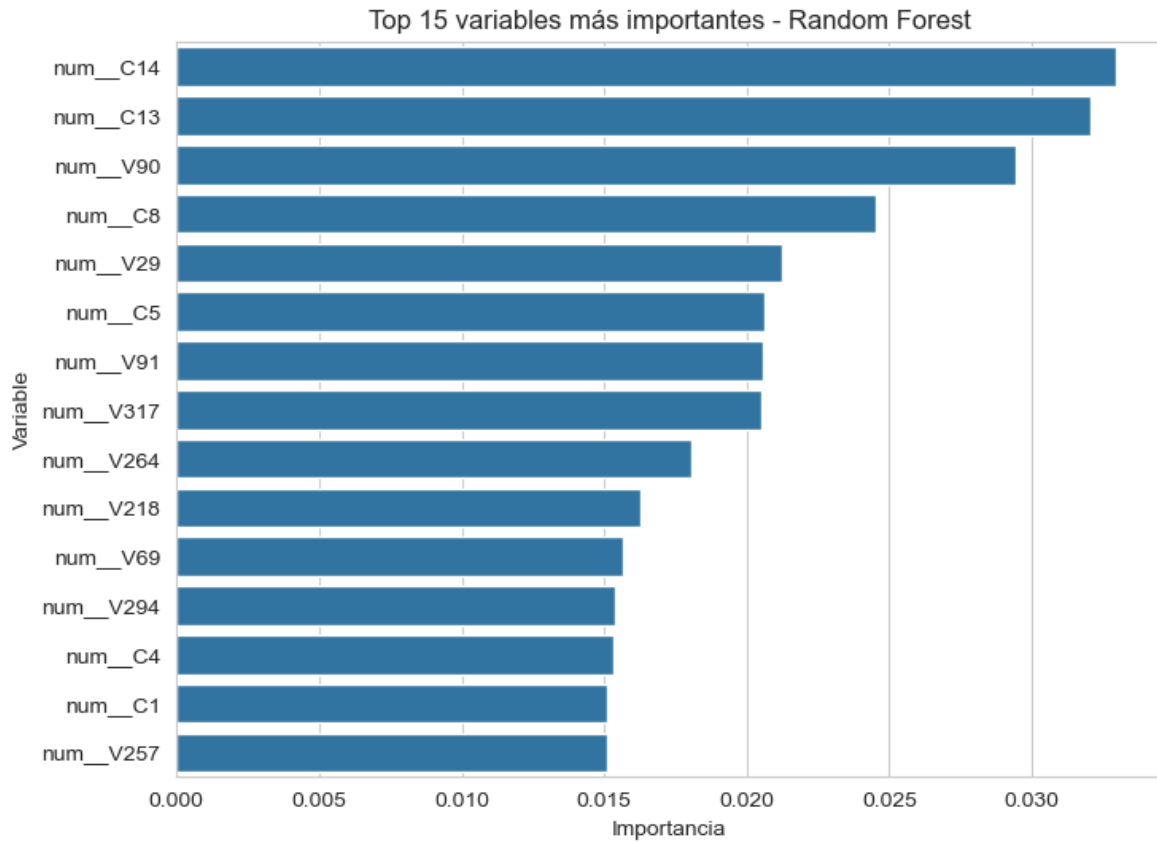
También se definió un mínimo de 1000 observaciones para permitir una nueva división y un mínimo de 300 observaciones por hoja final. Estos valores ayudan a evitar reglas demasiado específicas basadas en pocos casos. Para aumentar la diversidad entre árboles, se utilizó una selección aleatoria de variables en cada división mediante la raíz cuadrada del total de atributos disponibles, junto con muestreo *bootstrap* y un 60% de la muestra en cada árbol. Esto permite que cada árbol aprenda desde una porción distinta de los datos, favoreciendo la robustez del modelo.

Dado el fuerte desbalance entre transacciones fraudulentas y no fraudulentas, se incorporó ponderación balanceada por submuestra, ajustando los pesos de las clases dentro de cada árbol para dar mayor relevancia relativa a los fraudes. Nuevamente, se fijó una semilla aleatoria igual a 42 y se utilizaron todos los núcleos disponibles del procesador.

El gráfico de importancia de variables muestra que *Random Forest* distribuye la capacidad predictiva entre un conjunto más amplio de atributos, a diferencia del árbol de decisión individual, que concentraba gran parte de su importancia en V258 y V70. En este caso, las variables más relevantes fueron C14, C13, V90, C8, V29, C5, V91 y V317, lo que indica que el modelo aprovecha múltiples señales del conjunto de datos para clasificar las transacciones. Esto es coherente con la lógica de *Random Forest*, ya que al combinar varios árboles puede detectar patrones desde distintas combinaciones de variables, reduciendo la dependencia de una única regla principal.

Figura 50

Variables más importantes según Random Forest

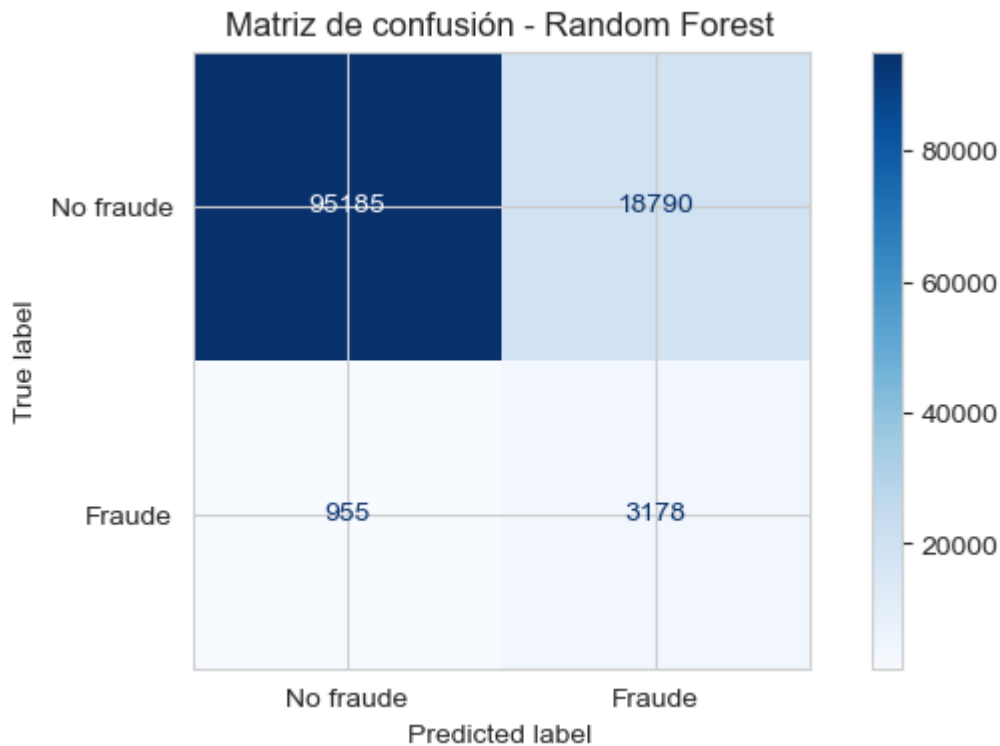


Elaboración propia.

Luego, se pasó a realizar la matriz de confusión y medidas correspondientes, de forma preliminar, este es uno de los modelos que mayores fraudes ha detectado.

Figura 51

Matriz de confusión de Random Forest



Elaboración propia.

Tabla 18

Indicadores de Random Forest

modelo	accuracy	recall	precision	specificity	f1_score	roc_auc	tiempo_entrenamiento_seg
Random Forest	83.28%	76.89%	14.47%	83.51%	24.35%	87.79%	335.81

Elaboración propia.

La matriz de confusión muestra que *Random Forest* clasificó correctamente 95.185 transacciones no fraudulentas y detectó 3.178 fraudes reales. En comparación con el árbol de decisión, logró identificar una mayor cantidad de fraudes, reduciendo los falsos negativos de 1.243 a 955. Esto se refleja directamente en un mayor *recall*, que alcanzó 76,89%, frente al 69,92% obtenido por el árbol individual. No obstante, también generó 18.790 falsos

positivos, una cifra levemente superior a la del árbol de decisión, lo que implica una pequeña disminución en *specificity*.

En términos de métricas, *Random Forest* obtuvo un *accuracy* de 83,28% con mejoras más relevantes en las métricas asociadas a fraude. El *precision* alcanzó 14,47%, el *F1_score* subió a 24,35% y el **ROC_AUC** llegó a 87,79%, superando claramente al árbol individual en capacidad general de discriminación. Esto indica que el modelo no solo detecta más fraudes reales, sino que también ordena mejor las transacciones según su nivel de riesgo. Además, el tiempo de entrenamiento fue de 335,81 segundos, unos 5,6 minutos, menor que el del árbol de decisión en esta ejecución, probablemente favorecido por la paralelización y por la configuración restringida de crecimiento de los árboles. Bajo esta evidencia, el modelo representa una mejora frente al árbol simple, especialmente por su mayor sensibilidad ante la clase fraudulenta y su mejor desempeño global.

7.4.6. Naive Bayes

Naive Bayes se introdujo como un modelo probabilístico base, debido a su simplicidad y bajo costo computacional. Su funcionamiento se basa en estimar la probabilidad de que una transacción pertenezca a la clase fraude o no fraude a partir de las variables observadas, asumiendo independencia entre los atributos. En este caso se utilizó *ComplementNB*, una variante adecuada para escenarios con clases desbalanceadas, junto con un preprocesamiento que aseguró valores no negativos en las variables.

Para *Naive Bayes* se utilizó un suavizado igual a 1,0, conocido como suavizado de *Laplace*. Este parámetro evita que categorías o combinaciones de variables con baja frecuencia generen probabilidades nulas, lo que resulta útil cuando existen muchas variables

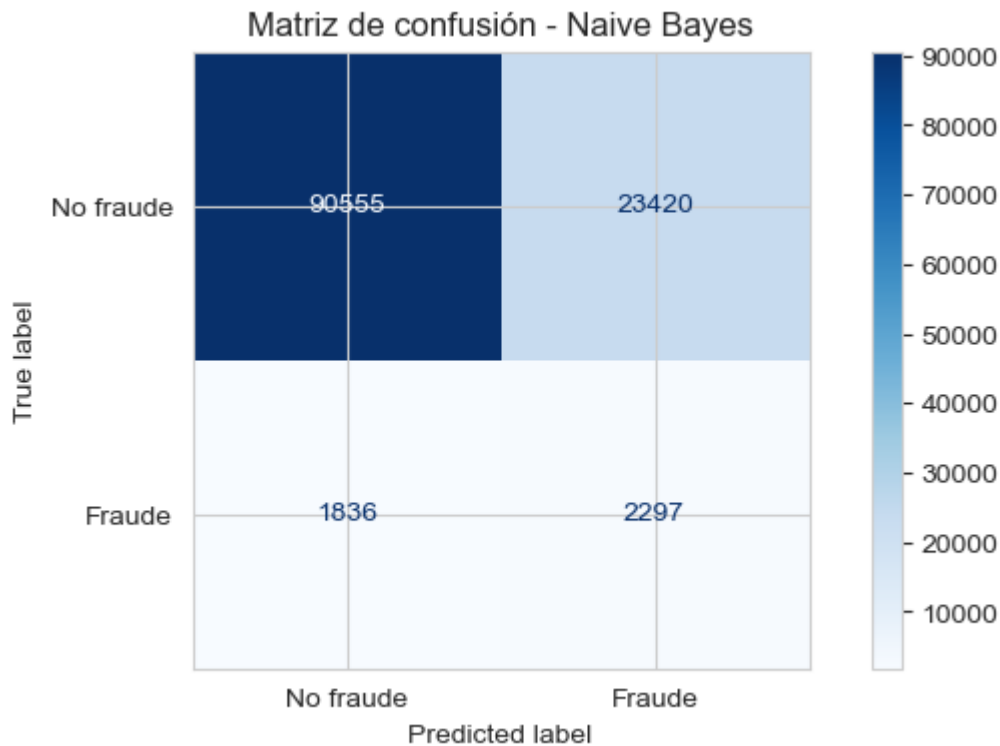
transformadas y algunas categorías aparecen pocas veces. De esta manera, el modelo mantiene mayor estabilidad al estimar probabilidades para la clase fraude y no fraude.

Además, se decidió no ajustar probabilidades previas de clase a partir de la frecuencia observada en los datos. Esta decisión busca evitar que el fuerte desbalance de la base incline excesivamente el modelo hacia la clase mayoritaria, considerando que las transacciones fraudulentas representan una proporción mucho menor del total. Finalmente, no se aplicó normalización adicional sobre los pesos internos del modelo, manteniendo la formulación estándar de *ComplementNB* y priorizando una implementación simple.

De esta forma, se obtuvieron los resultados del método, aunque no son los que más resaltan y podría decirse que son los más simples en comparación a los demás modelos, incluso a regresión logística que también usa probabilidades para generar las predicciones.

Figura 52

Matriz de confusión de Naive Bayes



Elaboración propia.

Tabla 19

Indicadores de Naive Bayes

modelo	accuracy	recall	precision	specificity	f1_score	roc_auc	tiempo_entrenamiento_seg
Naive Bayes	78.62%	55.58%	8.93%	79.45%	15.39%	76.31%	146.2

Elaboración propia.

La matriz de confusión de *Naive Bayes* muestra que el modelo clasificó correctamente 90.555 transacciones no fraudulentas y detectó 2.297 fraudes reales. Sin embargo, también generó 23.420 falsos positivos y dejó sin identificar 1.836 fraudes. Esto evidencia que el modelo logra reconocer una parte de las transacciones fraudulentas, pero con una capacidad menor que otros modelos evaluados, especialmente al compararlo con *Random Forest*, árbol de decisión o regresión logística.

En términos de métricas, obtuvo una exactitud de 78,62% y una especificidad de 79,45%, valores inferiores a los observados en la mayoría de los modelos anteriores. Su *recall* fue de 55,58%, lo que indica que logró detectar un poco más de la mitad de los fraudes reales, pero dejó pasar una proporción considerable de casos fraudulentos. Además, el *precision* fue de 8,93%, reflejando que muchas de las transacciones clasificadas como fraude correspondían realmente a operaciones legítimas. Esto se traduce en un *F1_score* de 15,39%, el más bajo dentro de los modelos revisados hasta este punto, mostrando un equilibrio débil entre detección y precisión. El AUC de 76,31% indica que el modelo posee cierta capacidad para ordenar las transacciones, pero claramente menor que la de modelos más robustos. Bajo estos resultados, *Naive Bayes* funciona principalmente como una referencia probabilística simple, pero sus supuestos de independencia entre variables limitan su rendimiento en una base compleja como la de fraude transaccional, donde probablemente existen interacciones importantes entre atributos.

7.4.7. Redes Neuronales (RRNN)

La red neuronal fue incorporada como el modelo de mayor flexibilidad dentro del proceso de clasificación, debido a su capacidad para aprender relaciones complejas y no lineales entre las variables explicativas y la probabilidad de fraude. A diferencia de modelos más simples, como regresión logística o *Naive Bayes*, una red neuronal puede combinar múltiples capas de procesamiento, permitiendo detectar patrones que no necesariamente son evidentes. Esto resulta especialmente relevante en un problema de fraude transaccional, donde las señales de riesgo pueden depender de interacciones entre muchas variables numéricas y categóricas.

Para su implementación se utilizaron los paquetes *TensorFlow* y *Keras*, los cuales permiten construir, entrenar y evaluar modelos de aprendizaje profundo de manera estructurada. *TensorFlow* entrega la base computacional para trabajar con redes neuronales y operaciones de alto rendimiento, mientras que *Keras* facilita la definición de la arquitectura del modelo mediante capas, funciones de activación, optimizadores y métricas de evaluación.

La red neuronal fue definida mediante una arquitectura secuencial en *Keras*, donde la dimensión de entrada se estableció automáticamente a partir del número de variables disponibles después del preprocesamiento. El modelo comienza con una capa de entrada y continúa con tres capas densas ocultas de 128, 64 y 32 neuronas, respectivamente. En estas capas se utilizó la función de activación *ReLU*, adecuada para capturar relaciones no lineales entre las variables y mejorar la capacidad de aprendizaje del modelo. La capa final contiene una sola neurona con activación sigmoide, ya que el objetivo corresponde a una clasificación binaria, donde la salida representa la probabilidad estimada de que una transacción sea fraudulenta.

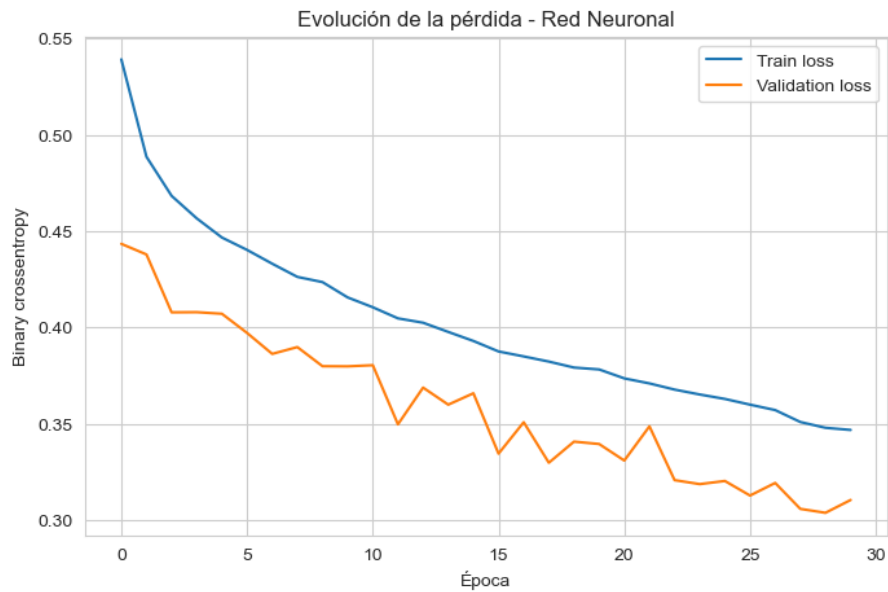
Para mejorar la estabilidad del entrenamiento y reducir el riesgo de sobreajuste, se incorporó una capa de normalización por lotes después de la primera capa densa, junto con capas de *dropout* de 30% en las dos primeras secciones del modelo. El *dropout* permite desactivar aleatoriamente una proporción de neuronas durante el entrenamiento, evitando que la red dependa excesivamente de combinaciones específicas de variables. Posteriormente, el modelo fue compilado con el optimizador *Adam*, utilizando una tasa de aprendizaje de 0,001, y con pérdida de entropía cruzada binaria, adecuada para problemas de clasificación entre dos clases.

Durante el entrenamiento se utilizaron hasta 30 épocas, lotes de 2048 observaciones y ponderación de clases para dar mayor importancia relativa a las transacciones fraudulentas. También se aplicaron dos *callbacks*: detención temprana (*early stopping*), que interrumpe el entrenamiento si el **PR_AUC** (área bajo la curva de un gráfico *Precision-Recall*) de validación deja de mejorar, y reducción automática (*reduce lr*) de la tasa de aprendizaje, que disminuye la velocidad de ajuste cuando el modelo se estanca. Finalmente, las predicciones se generaron sobre el conjunto de prueba en lotes de 4096 observaciones, obteniendo probabilidades de fraude que luego fueron transformadas en clases mediante un umbral base de 0,5.

Se generaron tres curvas para evaluar el desempeño de la red neuronal a medida que se iba entrenando y aprendiendo en cada una de sus épocas y capas: la pérdida de la entropía o *binary cross - entropy* que iba a disminuyendo a medida que avanzaban las épocas, la evolución de la curva **ROC_AUC** que es fundamental para monitorear el aprendizaje del modelo y si es capaz de predecir correctamente, y el gráfico de **PR_AUC** que mide el aumento de la precisión y sensibilidad a medida que se detectan más fraudes con una mejor selección de patrones que logren predicciones más certeras.

Figura 53

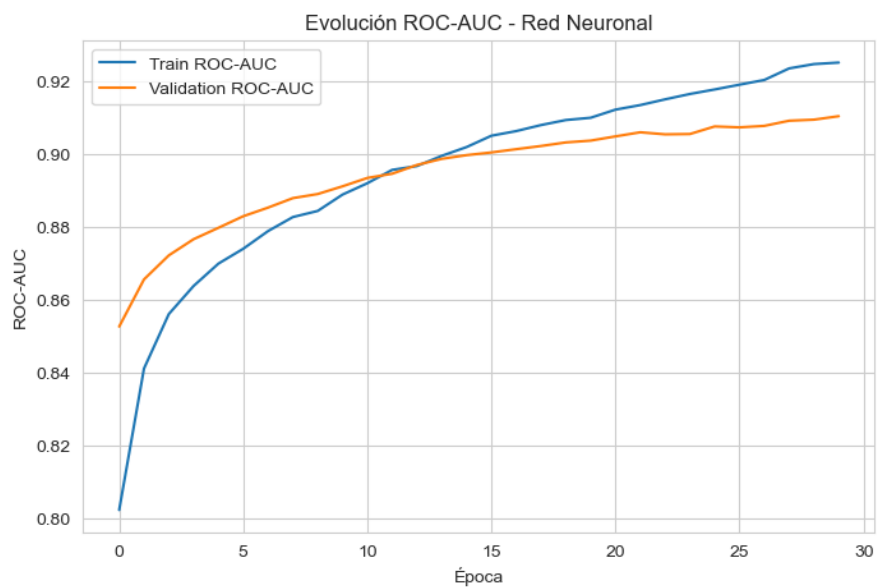
Evolución de la binary cross-entropy loss con el avance del entrenamiento en épocas



Elaboración propia.

Figura 54

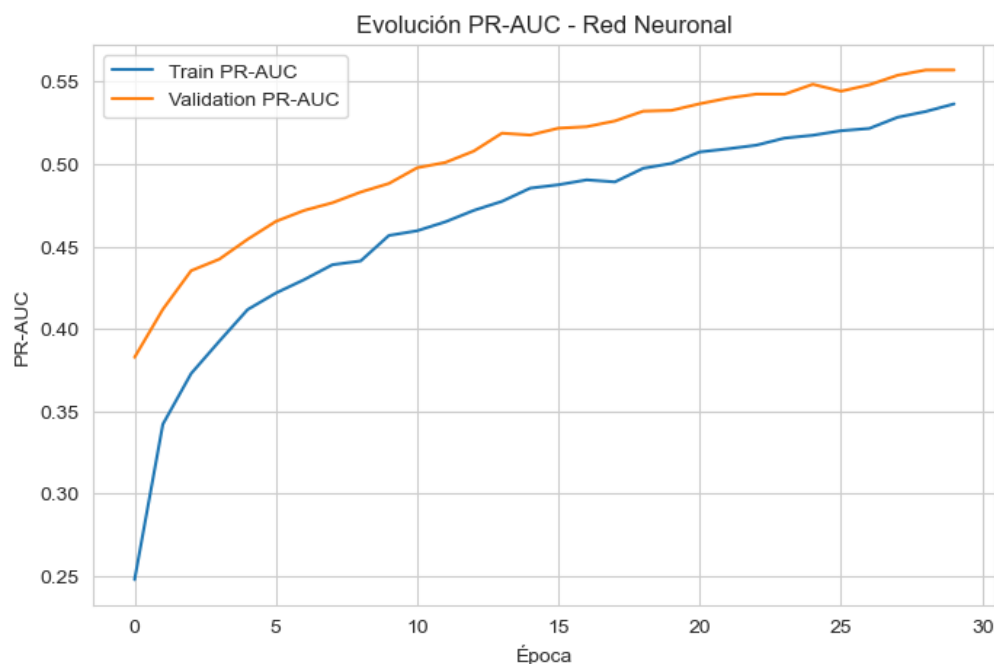
Evolución del ROC_AUC en la red neuronal



Elaboración propia.

Figura 55

Evolución del PR_AUC de la red neuronal



Elaboración propia.

La evolución del entrenamiento de la red neuronal muestra un comportamiento estable y consistente. En primer lugar, la pérdida disminuye de forma progresiva tanto en entrenamiento como en validación, lo que indica que el modelo fue aprendiendo patrones relevantes a medida que avanzaban las épocas. Además, la pérdida de validación se mantiene incluso por debajo de la de entrenamiento, lo que puede explicarse por el uso de mecanismos de regularización como *dropout* activo durante el entrenamiento, y sugiere que no se observa un sobreajuste evidente.

Respecto al **ROC-AUC**, ambas curvas presentan una mejora sostenida hasta estabilizarse en valores cercanos a 0,9 en validación y ligeramente superiores en

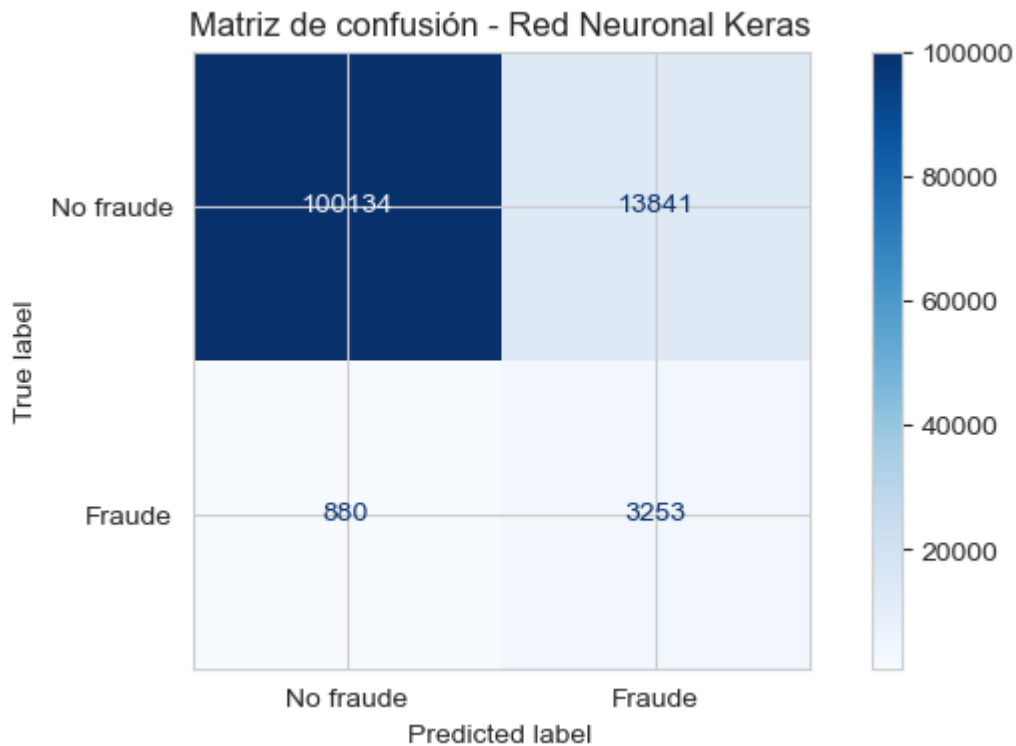
entrenamiento. Esta cercanía entre ambas curvas indica que la red logró una buena capacidad de discriminación entre transacciones fraudulentas y no fraudulentas, manteniendo un desempeño similar en datos no vistos durante el ajuste del modelo.

El **PR-AUC** también aumenta de manera gradual, alcanzando valores superiores a 0,55 en validación. Este resultado es especialmente importante dado el desbalance de clases, ya que refleja una mejora en la capacidad del modelo para detectar fraudes manteniendo cierto nivel de precisión en las alertas. De forma general, los tres gráficos muestran un entrenamiento adecuado, con mejoras consistentes, ausencia de divergencias importantes entre entrenamiento y validación, y un desempeño prometedor para el problema de detección de fraude.

Por último, se obtuvieron las métricas conocidas y matriz de confusión de la red. De forma preliminar, hay una mejora significativa respecto a los demás modelos con un grado de detección superior.

Figura 56

Matriz de confusión de la red neuronal



Elaboración propia.

Tabla 20

Métricas y resultados de la red neuronal

modelo	accuracy	recall	precision	specificity	f1_score	roc_auc	tiempo_entrenamiento_seg
Red Neuronal Keras	87.54%	78.71%	19.03%	87.86%	30.65%	91.18%	205.85

Elaboración propia.

La matriz de confusión de la red neuronal muestra el mejor comportamiento general entre los modelos evaluados hasta este punto. El modelo clasificó correctamente 100.134 transacciones no fraudulentas y detectó 3.253 fraudes reales, dejando sin identificar 880 casos fraudulentos. Además, generó 13.841 falsos positivos, una cifra menor que la observada en modelos como regresión logística, SVM, árbol de decisión y *Random Forest*.

Esto refleja una mejora importante, ya que la red no solo identifica más fraudes, sino que también reduce parcialmente la cantidad de alertas incorrectas.

En términos de métricas, la red neuronal alcanzó un *accuracy* de 87,54% y un *specificity* de 87,86%, mostrando una buena capacidad para clasificar correctamente transacciones legítimas. Su *recall* fue de 78,71%, el valor más alto entre los modelos revisados, lo que indica que logró detectar cerca de ocho de cada diez fraudes reales. Además, obtuvo un *precision* de 19,03% y un *F1_score* de 30,65%, superando a los modelos anteriores, pero siguen siendo niveles bajos para lo que se quiere lograr. Por tanto, la sutileza de detectar fraudes por sobre las transacciones normales sigue prevaleciendo, aunque eso signifique tolerar alarmas de falsos positivos. El **ROC_AUC** de 91,18% confirma una alta capacidad para ordenar las transacciones según su nivel de riesgo, siendo uno de los resultados más sólidos del proceso de modelamiento.

A nivel computacional, el tiempo de entrenamiento fue de 205,85 segundos, lo que representa un costo razonable considerando la complejidad del modelo y el tamaño de la base utilizada. Así es como la red neuronal se posiciona como el modelo con mejor desempeño global, especialmente por su mayor capacidad para detectar fraudes reales, su mejor balance entre métricas y su alta capacidad discriminante.

Así como con KNN, se intentó comparar diversos umbrales de probabilidad de clasificación para ver si existían resultados aún mejores que los vistos con el de 0,5. Sin embargo, en un rango de 0,1 a 0,5 como límite se consiguieron resultados mucho peores a los analizados, sin ninguna mejora considerable. Por tanto, se decidió conservar este umbral y dejar el modelo dentro de los mejores clasificadores de aprendizaje supervisado en este caso particular.

7.5. Comparación y selección final de los modelos supervisados

Como parte final de los resultados de este proyecto, se realizó una comparación conjunta de todos los modelos analizados para detectar fraudes dentro del **IEEE-CIS Fraud Detection**, incluidos los modelos lineales, pero sólo con el tiempo de entrenamiento y AUC como indicadores a comparar.

Primero se revisará una tabla resumen que enseña todas las métricas revisadas hasta el momento ordenadas de mayor a menor AUC como principal medida de buen desempeño de *machine learning*.

Tabla 21

Comparación general de desempeño de todos los modelos evaluados

modelo	accuracy	recall	precision	specificity	f1_score	roc_auc	time_ent	cond_eva
Red Neuronal Keras	87.54%	78.71%	19.03%	87.86%	30.65%	91.18%	205,85	Dataset completo
Random Forest	83.28%	76.89%	14.47%	83.51%	24.35%	87.79%	335,81	Dataset completo
Regresión Logística	80.55%	71.13%	11.89%	80.89%	20.38%	84.26%	718,85	Dataset completo
SVM lineal	82.62%	67.70%	12.73%	83.17%	21.43%	83.45%	196,18	Dataset completo
Árbol de Decisión	83.24%	69.92%	13.48%	83.73%	22.61%	82.59%	446,15	Dataset completo
KNN, umbral 0.20	96.98%	39.90%	60.29%	99.05%	48.02%	82.12%	7,92	Submuestra estratificada
Naive Bayes	78.62%	55.58%	8.93%	79.45%	15.39%	76.31%	146,2	Dataset completo
ElasticNet_L1_L2						58.28%	1412,58	Dataset completo / modelos lineales
Lineal_sin_reg						57.09%	59,09	Dataset completo / modelos lineales
Ridge_L2						50.73%	53,02	Dataset completo / modelos lineales
Lasso_L1						50.11%	60,19	Dataset completo / modelos lineales

Elaboración propia.

Entre los modelos entrenados con el *dataset* completo, la red neuronal *Keras* obtuvo el mejor rendimiento general, alcanzando el mayor *accuracy*, *recall*, *F1_score* y **ROC_AUC**. En particular, su *recall* de 78,71% y su AUC de 91,18% indican que fue el modelo con mayor capacidad para detectar fraudes reales y ordenar correctamente las transacciones según su nivel de riesgo.

Random Forest también mostró un desempeño sólido, especialmente al compararlo con modelos más simples como regresión logística, SVM lineal, árbol de decisión y *Naive Bayes*. Aunque su *accuracy* fue menor que la red neuronal, obtuvo un *recall* de 76,89% y un AUC de 87,79%, posicionándose como el mejor modelo tradicional del análisis. Además, mantuvo una estructura más interpretable que la red neuronal, especialmente mediante la importancia de variables, lo que lo convierte en una alternativa atractiva cuando se busca equilibrar desempeño predictivo e interpretabilidad.

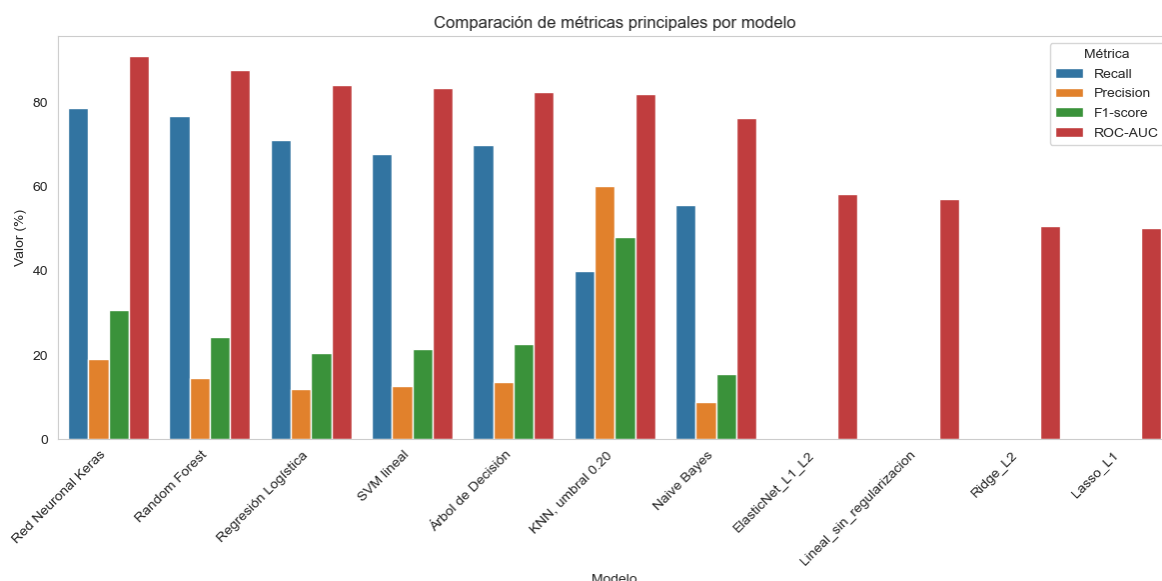
El caso de KNN debe interpretarse con cuidado, ya que fue evaluado sobre una submuestra estratificada y no sobre el *dataset* completo. Aun así, con el umbral ajustado a 0,20 obtuvo el mayor *precision* y *F1_score* de la tabla, lo que indica un buen equilibrio dentro de su muestra de evaluación. Sin embargo, su *recall* de 39,90% fue considerablemente menor que el de la red neuronal y *Random Forest*, por lo que detectó una menor proporción de fraudes reales. Por esta razón, sus resultados son útiles como referencia, pero no son completamente comparables con los modelos entrenados y evaluados sobre toda la base.

Los modelos lineales de regresión y regularización presentaron los resultados más débiles, con valores de **ROC_AUC** cercanos al clasificador aleatorio, especialmente *Ridge* y *Lasso*. Estos modelos fueron incluidos principalmente como línea base para contrastar enfoques simples frente a modelos de clasificación más robustos. En términos generales, la comparación evidencia que los modelos con mayor capacidad para capturar relaciones no lineales fueron los más adecuados para abordar el problema de detección de fraude en transacciones digitales.

Con la misma información de antes, se desarrolló un gráfico de barras que visualiza de mejor manera las relaciones entre las métricas y cuáles fueron los modelos más eficientes en la predicción de fraudes y que menos fallan tuvieron al predecir.

Figura 57

Comparación de métricas principales de los modelos



Elaboración propia.

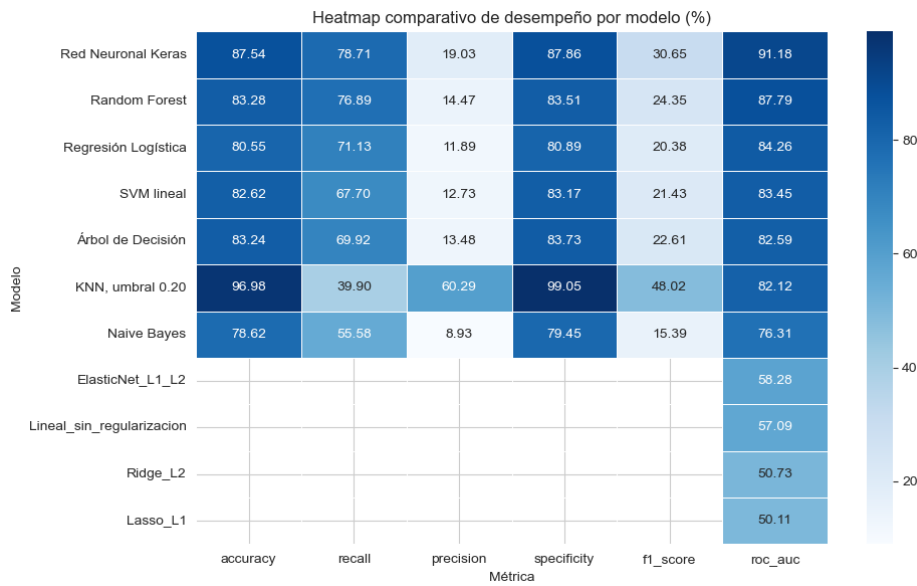
El gráfico resume visualmente las principales diferencias entre modelos en términos de *recall*, *precision*, *F1_score* y AUC. Se observa que la red neuronal lidera en la mayoría de las métricas relevantes, especialmente en *recall*, *precision* y AUC, lo que confirma su mayor capacidad para detectar fraudes y ordenar las transacciones según riesgo. *Random Forest* aparece como el segundo modelo más sólido, con resultados cercanos en detección de fraude y mejor desempeño que los modelos lineales, SVM, árbol de decisión y *Naive Bayes*.

El gráfico refuerza que los modelos más flexibles y elaborados fueron los más adecuados para capturar los patrones complejos asociados al fraude transaccional. No obstante, se debe tener los conocimientos apropiados para manejarlos ya que se codifican con funciones y parámetros más sensibles que los otros que son más simples en su generación.

Posteriormente, se desarrolló un *Heatmap* o matriz comparativa con los desempeños de cada uno de los modelos con las métricas en distintos colores según su cercanía al 100%. La red neuronal es la que tiene los mejores resultados con los valores en azul más oscuro, mientras que modelos como *Naive Bayes* tienen valores más claros con menor capacidad de predicción.

Figura 58

Heatmap comparativo de desempeño del modelo



Elaboración propia.

El *heatmap* permite visualizar de forma rápida qué modelos concentran los mejores desempeños según cada métrica. La red neuronal *Keras* destaca con los valores más altos dentro de los modelos comparables con *dataset* completo y un mejor equilibrio general reflejado en el *F1_score*. *Random Forest* también presenta un desempeño robusto, especialmente en *recall*, confirmándose como el mejor modelo tradicional evaluado. En contraste, regresión logística, SVM lineal y árbol de decisión muestran resultados intermedios, con buena capacidad de detección, pero menor precisión y menor balance global.

El caso de KNN con umbral 0,20 resalta por su alta precisión, *specificity* y *F1_score*, aunque debe interpretarse considerando que fue evaluado sobre una submuestra estratificada. Por otro lado, *Naive Bayes* presenta los valores más bajos entre los modelos de clasificación completos, especialmente en *precision* y *f1_score*. Finalmente, los modelos lineales y regularizados solo muestran valores de AUC, los cuales se mantienen cercanos al 50%, evidenciando una capacidad discriminante limitada.

Para ordenar los modelos de manera más integral, se construyó un *score_compuesto* (puntaje ponderado) como un criterio heurístico que generó pesos para cada métrica dependiendo de los valores que tenían en los modelos. Su cálculo consistió en una combinación ponderada de las métricas más relevantes para el problema de detección de fraude. Este indicador se definió otorgando un 40% de peso al **ROC_AUC**, un 30% al *F1_score*, un 20% al *recall* y un 10% al *precision*. La decisión de no incluir directamente el *accuracy* se debe a que la base se encuentra fuertemente desbalanceada, por lo que un modelo podría alcanzar un alto *accuracy* clasificando correctamente la mayoría de las transacciones no fraudulentas, pero sin necesariamente detectar bien los fraudes.

El mayor peso fue asignado al AUC, ya que esta métrica permite evaluar la capacidad general del modelo para distinguir entre transacciones fraudulentas y no fraudulentas a distintos umbrales de clasificación. Luego, se otorgó un peso importante al *F1_score*, debido a que combina dos métricas, entregando una medida más equilibrada del desempeño frente a clases desbalanceadas. El *recall* recibió también una ponderación relevante, ya que en un problema de fraude es especialmente importante detectar la mayor cantidad posible de casos fraudulentos reales. Finalmente, se incluyó el *precision* con menor peso, ya que en este contexto resulta más crítico no dejar pasar fraudes, aunque eso indique tener que aceptar una proporción de falsas alarmas dentro del análisis.

Con este criterio, el ranking final permite comparar los modelos no solo por una métrica aislada, sino considerando simultáneamente su capacidad discriminante, su balance entre detección y precisión, y su eficiencia para identificar la clase fraudulenta.

Tabla 22

Ranking de modelos según score compuesto

modelo	accuracy	recall	precision	specificity	f1_score	roc_auc	time_ent_seg	score_compuesto
Red Neuronal Keras	87.54%	78.71%	19.03%	87.86%	30.65%	91.18%	205,85	63.31%
KNN, umbral 0.20	96.98%	39.9%	60.29%	99.05%	48.02%	82.12%	7,92	61.26%
Random Forest	83.28%	76.89%	14.47%	83.51%	24.35%	87.79%	335,81	59.25%
Regresión Logística	80.55%	71.13%	11.89%	80.89%	20.38%	84.26%	718,85	55.23%
Árbol de Decisión	83.24%	69.92%	13.48%	83.73%	22.61%	82.59%	446,15	55.15%
SVM lineal	82.62%	67.7%	12.73%	83.17%	21.43%	83.45%	196,18	54.62%
Naive Bayes	78.62%	55.58%	8.93%	79.45%	15.39%	76.31%	146,2	47.15%

Elaboración propia.

La tabla ordena los modelos de clasificación considerando simultáneamente su capacidad discriminante, equilibrio entre métricas y desempeño frente a la clase fraudulenta. Bajo este criterio, la red neuronal obtuvo el mayor puntaje, con 63,31%, posicionándose

como el modelo de mejor rendimiento global. Esto se explica por su alto *recall*, *AUC* y un *F1_score* superior al resto de los modelos entrenados sobre la base de datos. Por lo tanto, representa la alternativa más sólida para detectar fraudes reales sin deteriorar excesivamente la clasificación de transacciones legítimas.

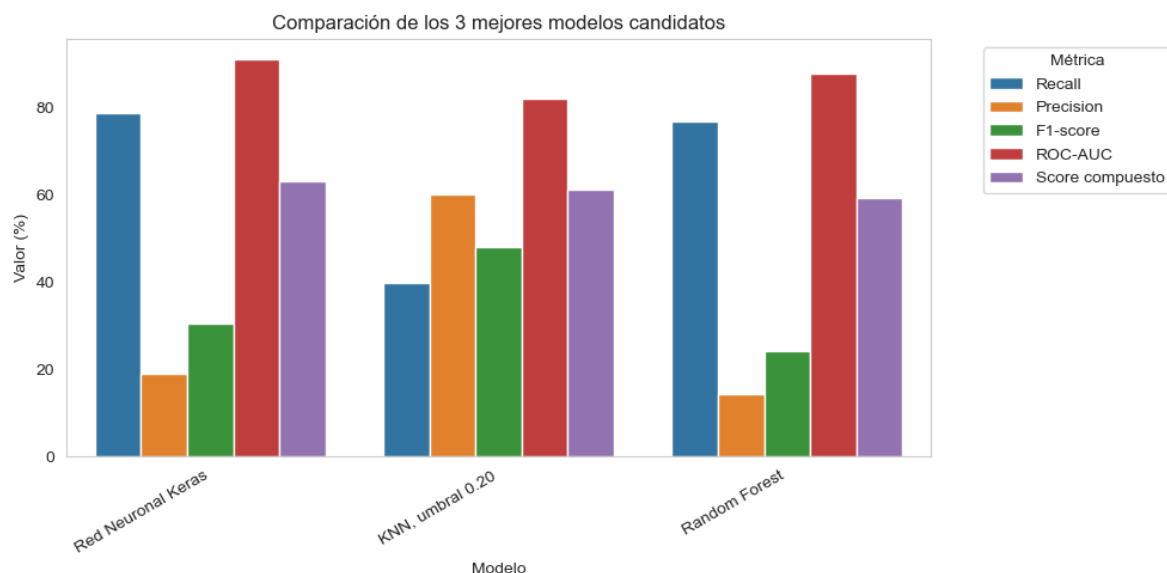
El segundo lugar lo ocupa KNN con umbral 0,20, con un *score_compuesto* de 61,26%. Su posición se explica principalmente por su alta *precision* y el mayor *F1_score* de la tabla, lo que indica un buen equilibrio entre calidad de alertas y clasificación dentro de la submuestra utilizada. Sin embargo, KNN fue evaluado sobre una submuestra estratificada, entonces los resultados no reflejan su desempeño al 100% todavía. Además, su *recall* fue menor que el de la red neuronal y *Random Forest*, por lo que detecta una menor proporción de fraudes reales.

El tercer modelo seleccionado corresponde a *Random Forest*, con un *score* de 59,25%. Aunque queda levemente por debajo de KNN en el ranking, su resultado es especialmente relevante, porque fue entrenado y evaluado sobre el *dataset* completo. Además, combina un alto *recall*, buen **ROC_AUC** y mayor interpretabilidad mediante importancia de variables. De esta forma, el top 3 final queda compuesto por red neuronal *Keras* como mejor modelo global, KNN ajustado por umbral como alternativa con mejor balance en submuestra, y *Random Forest* como el mejor modelo tradicional aplicado sobre la base completa.

Para finalizar, se realizó un gráfico final que representa el top 3 mejores modelos para clasificar y son los que mejor desempeño tuvieron en la base de datos seleccionados. Esto no quiere decir que tengan los mismos resultados con otra base de datos ni mucho menos.

Figura 59

Comparación de los 3 mejores modelos candidatos



Elaboración propia.

Como era de suponerse, la red neuronal aparece como el modelo más equilibrado para este caso. Esto indica que fue el modelo con mejor capacidad para detectar fraudes reales y para ordenar las transacciones según su nivel de riesgo. Aunque su precisión no es la más alta, logra un balance sólido entre detección de fraude, capacidad discriminante y desempeño general.

KNN con umbral 0,20 destaca principalmente por su mejor *F1_score*, lo que muestra que, dentro de la submuestra evaluada, fue efectivo generando alertas de mejor calidad. Sin embargo, su *recall* es bastante menor y se aplicó sobre una submuestra estratificada, por lo que detecta una menor proporción de fraudes reales.

En tercer lugar, *Random Forest* se posiciona como el mejor modelo tradicional aplicado sobre la base completa. Aunque su *score_compuesto* queda levemente por debajo de KNN, presenta mayor interpretabilidad mediante importancia de variables. Esto lo convierte en una alternativa relevante cuando se busca combinar desempeño predictivo con mayor capacidad de explicación del modelo.

Para esta jerarquía se debe tener en cuenta que solo un 3,5% de la base corresponde a fraude, evidenciando el fuerte desbalance de clases que los modelos deben afrontar al momento de trabajar. Por ello, una pequeña tasa de error sobre la clase mayoritaria puede generar una gran cantidad de falsos positivos y reducir la *precision*. Dado este contexto aparece el *trade off* con el *recall*, porque para la detección de fraude resulta crítico identificar la mayor cantidad posible de fraudes reales, aunque esto implique aceptar un cierto número de falsas alertas. Sin embargo, en una situación de prueba real hay que evaluar en detalle si es más importante el costo relativo de solucionar un porcentaje de falsas alarmas, frente al costo de dejar pasar un fraude que puede tener mayores consecuencias dependiendo de su gravedad.

Por otro lado, estos resultados deben entenderse como válidos para esta base de datos, esta partición de entrenamiento y prueba, el preprocesamiento aplicado y los hiperparámetros definidos. En otra base, con otra distribución de fraude, nuevas variables o distintos costos asociados a falsos positivos y falsos negativos, el *ranking* podría cambiar. El *dataset* seleccionado pertenece a una competencia basado en transacciones reales y no corresponde a una institución bancaria específica, entonces, las conclusiones que se puedan deducir son netamente académicas y un apoyo para quien lea esta investigación.

8. Conclusiones

A partir del desarrollo realizado, el objetivo general de la investigación fue desarrollado en gran parte, ya que se evaluó y comparó la aplicación de distintos modelos de inteligencia artificial para la detección de fraude en transacciones digitales, utilizando el *dataset* **IEEE-CIS Fraud Detection**. El trabajo permitió abordar el problema desde una metodología ordenada, comenzando con la comprensión del contexto financiero y tecnológico, siguiendo con la limpieza y preparación de los datos, y finalizando con la implementación, evaluación y comparación de modelos supervisados, no supervisados y redes neuronales. De esta manera, la investigación no se limitó únicamente a entrenar modelos, sino que construyó un proceso completo de análisis orientado a identificar qué técnicas resultan más adecuadas para enfrentar un problema de clasificación altamente desbalanceado.

Respecto a los objetivos específicos, primero se identificaron y describieron distintos enfoques de *machine learning* aplicables al contexto financiero, incluyendo modelos lineales, modelos de clasificación, modelos no supervisados y redes neuronales. Luego, se exploró y preparó el *dataset*, separando atributos numéricos y categóricos, aplicando escalamiento, codificación de variables categóricas y técnicas de reducción de dimensionalidad mediante PCA. Esta etapa fue clave, ya que permitió disminuir la complejidad de una base extensa y preparar los datos para que los modelos pudieran entrenarse de manera más estable y comparable.

El análisis exploratorio confirmó uno de los principales desafíos del problema: la fuerte desproporción entre transacciones fraudulentas y no fraudulentas. La clase fraudulenta representó aproximadamente un 3,5% del total, por lo que el uso exclusivo de *accuracy* como

indicador de desempeño, obtenido de la matriz de confusión, habría sido insuficiente para evaluar el desempeño real de los modelos. Por esta razón, se incorporaron métricas como *recall*, *precision*, *specificity*, *F1-score* y el valor del **AUC** de la curva **ROC**. Esta decisión metodológica fue relevante, ya que permitió comparar los modelos considerando no solo cuántas transacciones clasificaban correctamente, sino qué tan capaces eran de detectar fraudes reales en el caso.

Los resultados iniciales obtenidos de los modelos lineales de regresión y regularización muestran que no fueron adecuados para abordar directamente el problema de fraude, debido a que la variable objetivo es binaria y presenta una estructura más compleja que una relación lineal simple. Aunque el modelo de *Elastic Net* destacó dentro de este grupo, sus valores de **ROC-AUC** fueron bajos en comparación con los modelos de clasificación. Esto permitió establecer una línea base útil, pero también confirmó la necesidad de utilizar algoritmos más flexibles, capaces de capturar relaciones no lineales e interacciones complejas entre variables.

El análisis no supervisado mediante *K-Means* aportó una visión complementaria al modelamiento, ya que permitió segmentar las transacciones en grupos con distintos niveles de exposición al fraude. Aunque el algoritmo no utilizó la variable objetivo para formar los *clusters*, posteriormente se observó que algunos grupos presentaban tasas de fraude considerablemente más altas que otros. Esto sugiere que existen perfiles transaccionales diferenciados dentro de la base, lo que puede ser útil para entender patrones de riesgo y apoyar futuras estrategias de monitoreo o priorización de alertas.

En cuanto a los modelos supervisados de clasificación, la red neuronal con *Keras* fue el modelo con mejor desempeño global sobre el *dataset* completo. Obtuvo el mayor *recall*,

el mayor **ROC-AUC** y el mejor equilibrio general entre las métricas principales, lo que indica una mayor capacidad para detectar fraudes reales y ordenar las transacciones según su nivel de riesgo. Este resultado es coherente con la naturaleza del problema, ya que las redes neuronales pueden aprender relaciones complejas entre variables numéricas y categóricas más allá de la matemática estándar, especialmente en *datasets* grandes y con múltiples patrones ocultos.

El modelo de ensamble *Random Forest* también destacó como una alternativa sólida, especialmente por ser el mejor modelo tradicional evaluado sobre la base completa, a pesar de utilizar múltiples árboles de decisión en su trabajo. Con un desempeño inferior al de la red neuronal, logró presentar buenos resultados en *recall* y AUC, además de entregar mayor interpretabilidad mediante el análisis de importancia de variables. Esta característica lo convierte en una opción relevante cuando se busca comprender qué atributos están influyendo con mayor fuerza en la clasificación.

El modelo de KNN ajustado con umbral de 0,20 obtuvo un *F1-score* y una *precision* destacables, pero su interpretación debe ser más cautelosa, ya que fue entrenado y evaluado sobre una submuestra estratificada. Su resultado muestra que el ajuste de umbrales puede mejorar significativamente el equilibrio entre *recall* y *precision*, especialmente en contextos desbalanceados. Sin embargo, al presentar un *recall* menor que la red neuronal y *Random Forest*, su aplicación práctica requeriría mayor validación sobre bases completas o entornos de producción antes de considerarlo como una alternativa principal. No obstante, la validación de la submuestra de KNN obtuvo un *accuracy* del 97,1%, por lo que su capacidad de discernir entre fraudes es bastante avanzada en comparación a los demás, pero un poco

más sesgado su actuar, especialmente con el cambio de umbral utilizado y las razones ya señaladas.

En términos generales, los resultados muestran que no existe un único modelo ideal para todos los escenarios, sino que la elección depende del objetivo operativo. Si la prioridad es detectar la mayor cantidad posible de fraudes, la red neuronal aparece como la mejor alternativa dentro de esta investigación, teniendo en cuenta la poca precisión de sus resultados. Si se requiere mayor interpretabilidad y trazabilidad de variables, *Random Forest* ofrece un equilibrio más adecuado. En cambio, si se busca reducir falsas alertas en un subconjunto específico de datos, KNN ajustado puede ser útil como referencia, aunque con menor capacidad de detección global.

Desde una perspectiva aplicada, los hallazgos permiten afirmar que los modelos de inteligencia artificial pueden aportar valor en la detección preliminar de fraude transaccional, así como en otros casos de clasificación que sea tan compleja como la base utilizada. Sin embargo, estos modelos no deberían entenderse como reemplazos absolutos de la supervisión humana, sino como herramientas de apoyo para mejorar la eficiencia del análisis, reducir tiempos de revisión y focalizar recursos en transacciones con mayor probabilidad de riesgo.

En un entorno real, los resultados obtenidos y el análisis posterior no son todavía suficientes para recomendar implementación operacional sin validación adicional, calibración de umbrales, revisión de otras técnicas de entrenamiento, disminuir el impacto del desbalance y monitoreo de *drift*. Una sugerencia relevante es la evaluación de costos reales de cada métrica, ya que pueden tener un peso distinto en una institución financiera o empresa relacionada dependiendo de su alcance respecto al tema. Si su intención es prevenir fraudes como prioridad máxima, aunque eso signifique clasificar transacciones normales

como tal, le darán mayor peso al *recall* y *F1-score*, mientras que, si prefieren reducir el impacto de las falsas alarmas y evitar analizar fraudes erróneos, se enfocarán en el *accuracy* y *precision*, aun cuando eso pueda terminar en dejar pasar una transacción sospechosa.

El **ROC_AUC** es relevante, dado que analiza el rendimiento de la capacidad general del modelo para distinguir entre una y otra clasificación a distintos umbrales, pero al tener todos valores muy parecidos, no termina siendo una métrica decisiva en la selección final del modelo. Además, el *score compuesto* utilizado para ordenar los modelos corresponde a un criterio diseñado exclusivamente para esta investigación y no constituye una fórmula objetiva o estandarizada. Su cálculo asigna distintas ponderaciones a los indicadores de desempeño según la importancia considerada para el problema de detección de fraude. Por lo tanto, el orden obtenido depende de los pesos seleccionados y podría variar si se utilizaran ponderaciones diferentes.

La siguiente recomendación de modelos de alerta preliminar para priorizar revisión de transacciones se hizo a partir de todo el análisis visto en los resultados de la aplicación metodología CRISP-DM sobre el *dataset*. La red neuronal *Keras* como modelo candidato principal para escenarios donde se priorice el rendimiento predictivo, mientras que *Random Forest* debería considerarse como alternativa de *benchmark* interpretable por la categorización de variables más riesgosas que inciden en una transacción. En una aplicación institucional, ambos modelos podrían incluso complementarse: la red neuronal para generar puntajes de riesgo más precisos, y *Random Forest* para apoyar la explicación de patrones relevantes y disminución de la incertidumbre ante posibles ataques. Esta combinación permitiría mejorar la detección sin perder completamente la capacidad de interpretación, aunque se debe tener en cuenta el *trade off* entre sensibilidad y precisión, y el desbalance de

clases que pueda existir, a pesar de tener una base de gran tamaño. *KNN* sirve con un umbral ajustado, pero procurando que sus resultados sean consecuentes con toda la base de datos y comparar el desempeño a uno sin el ajuste que podría ser igual de eficiente

En suma, los resultados de esta investigación pueden ser útiles para bancos, instituciones financieras, áreas de riesgo, equipos de ciberseguridad, empresas *Fintech*, plataformas de comercio electrónico y analistas de datos interesados en detección de fraude. El trabajo ofrece una referencia metodológica para comparar modelos bajo un contexto de datos desbalanceados, mostrando la importancia de evaluar múltiples métricas comparables y no depender únicamente de uno general. Además, entrega una base práctica para diseñar sistemas de alerta que apoyen la prevención de fraudes digitales y la toma de decisiones en ambientes transaccionales de alta complejidad.

9. Discusión

Para la discusión se compararon los resultados con los trabajos similares revisados en el marco teórico para encontrar algún aporte diferente a la visión de esos autores que pueda ser aplicable fuera del caso. La investigación puede ser crucial para alguna empresa que necesite una defensa ante posibles ataques en sus sistemas y los resultados encontrados pueden hacer la distinción entre ser víctima de este delito o prevenirlo antes de que suceda.

En primer lugar, en la investigación de Sheyla Nieves aún era incipiente el uso de la inteligencia artificial en España, por lo que los resultados fueron menos concluyentes al sólo analizar herramientas propias de los bancos, como *chatbots* o *autenticación biométrica* (Nieves, 2019). Sin embargo, logró hacer una red neuronal con el 90% de precisión, mucho mayor que 20% de la encontrada, lo que quiere decir que obtuvo una menor cantidad de falsos positivos, por lo que efectivamente clasificó transacciones probadas como sospechosas.

Otro punto importante fue que las variables con las que trabajó también su significado eran incógnitas como las de la base de IEEE-CIS. Por su lado, esto fue una restricción importante, ya que no tenía como defender cuáles variables eran importantes para el *machine learning* aplicado y el desbalance de clases existente (Nieves, 2019). En este caso, las variables más relevantes que explicaban la varianza según PCA eran las del conjunto V o del C, que no se sabía su significado por su confidencialidad. Por tanto, solo se pueden hacer deducciones parciales sobre el impacto en la predicción, pero de forma arbitraria sin saber de qué se trataban, lo que podría ser relevante en una institución real.

El trabajo de Cristián Rayo en el banco de Perú tuvo un enfoque distinto, utilizó el método *design thinking* para desarrollar su proyecto y si logró modelar un sistema para el

banco en el que trabajaba, normalizando variables numéricas como en este trabajo. En su análisis, *Random Forest* fue el que obtuvo mejores resultados, teniendo en cuenta que no modeló redes neuronales en su tesis (Rayo, 2020). En consecuencia, coincide con el tercer lugar de los resultados que obtuve buenos resultados en comparación a los demás utilizando toda la base de datos a diferencia de KNN. Su principal diferencia fue bajar en una nube virtual el modelo para la interfaz con el banco, lo que podría introducir riesgos importantes si no contiene los sistemas de alerta o peligro adecuados.

Por último, el proyecto de Joaquín Beristain con el *churn* en telecomunicaciones fue el que tuvo resultados más similares a los conseguidos. Su red neuronal fue la de mejor desempeño con el 34% de sus configuraciones (Beristain, 2025). Su *recall* fue el de mayor importancia y fue el más robusto respecto a detectar clientes que abandonan compañías, así como encontró fraudes en este trabajo con tema completamente distinto.

Las principales diferencias fueron que *Random Forest* no tuvo el mejor desempeño frente a otros métodos de *boosting*. Además, no utilizó una base de datos de una institución nacional, coincidiendo con esta investigación, aunque sus resultados estuvieron sesgados frente al desbalance de clases y variables reales que podrían tener mayor repercusión en su base (Beristain, 2025). Esto es fundamental teniendo en cuenta que las variables aportadas por Vesta son en su mayoría reales y tuvieron repercusiones significativas en los entrenamientos, dada la explicación de la varianza en cada uno con PCA y los *clusters* que se conformaron en *K-means*.

De igual forma, no se puede concluir respecto a cuál de todos los modelos revisados es el mejor de todos, ya que depende netamente de la información de la empresa, los parámetros de evaluación de cada modelo y el objetivo al que se quiere llegar, porque es muy

distinto clasificar en base a fraude respecto a otros temas, como el abandono del *churn* o el precio de una acción. Entonces, dependerá del usuario o empresa la elección del modelo adecuado en base a su juicio y a las capacidades de los que tengan que usarlos. Cabe recalcar que las redes neuronales son las más eficientes en este aspecto, ya que tienen un nivel de eficiencia muy alto, son aptos de encontrar relaciones complejas que les son útiles en el entrenamiento y son la base de la inteligencia artificial de este siglo. Por tanto, se recomienda su uso precavido como una herramienta de apoyo en problemas de negocio que necesiten un razonamiento más avanzado y rápido que el de los sistemas tradicionales.

10. Limitaciones

Las principales limitaciones de esta investigación se relacionan con la naturaleza del *dataset* utilizado y con el nivel de interpretación disponible sobre sus variables. Si bien la base corresponde a transacciones reales de comercio electrónico aportadas por *Vesta Corporation*, no proviene directamente de un banco chileno ni de una institución financiera popular. Por esta razón, los resultados obtenidos deben entenderse como una evaluación académica y experimental, útil para analizar el comportamiento de distintos modelos de inteligencia artificial, pero no como una solución inmediatamente aplicable a una entidad bancaria sin una validación previa en datos reales de dicha organización y evaluación de costos entre *recall* y *precision* como indicador más relevante en el que se basaría el trabajo.

Una segunda limitación importante corresponde al significado de varias variables del *dataset*. Gran parte de los atributos más relevantes para los modelos pertenecen a conjuntos enmascarados, especialmente variables del tipo V, C y M, cuyo significado exacto no se encuentra disponible por razones de confidencialidad. Esto restringe la interpretación de los resultados, ya que, aunque fue posible identificar variables relevantes mediante PCA, árboles de decisión, *Random Forest* y análisis exploratorio, no siempre fue posible explicar con precisión qué característica transaccional o de comportamiento representaba cada una. En una aplicación real, esta falta de trazabilidad podría dificultar la justificación operativa de una alerta de fraude.

También debe considerarse que los resultados obtenidos dependen directamente de la base de datos utilizada, de su distribución de clases, de las variables disponibles, del

preprocesamiento aplicado y de los hiper parámetros definidos para cada modelo. En otra base de datos con distintas variables, el desempeño de los modelos podría variar de manera significativa. Por lo tanto, no se puede afirmar que el *ranking* obtenido sea universal, sino que representa el mejor resultado encontrado para este caso particular bajo las condiciones metodológicas definidas.

Otra restricción relevante corresponde al fuerte desbalance de clases presente en el *dataset*, donde las transacciones fraudulentas representan una proporción muy baja respecto del total. Esto obligó a utilizar métricas más sensibles al fraude, como *F1-score* y **ROC-AUC**, pero también generó una dificultad persistente: aumentar la detección de fraudes suele implicar un mayor número de falsos positivos. En un contexto real, este aspecto es crítico, ya que una cantidad excesiva de alertas puede sobrecargar a los equipos de revisión o afectar la experiencia de clientes legítimos.

En relación con las limitaciones revisadas en el marco teórico sobre inteligencia artificial y ciberseguridad, los resultados del trabajo realizado coinciden con varios puntos relevantes. Tal como se señaló, la IA requiere grandes volúmenes de información, datos de calidad, personal capacitado y una correcta selección del modelo para entregar resultados confiables. En este trabajo se contó con una base amplia, lo que permitió entrenar distintos algoritmos, pero la presencia de variables enmascaradas, valores faltantes y clases desbalanceadas confirma que el rendimiento de la IA depende fuertemente de la calidad y comprensión de los datos. Asimismo, el uso de modelos avanzados no elimina la necesidad de analistas humanos capaces de interpretar alertas, ajustar umbrales y tomar decisiones frente a posibles fraudes detectados por IA.

A pesar de esto, la investigación logró su propósito del objetivo general, ya que permitió evaluar y comparar distintos modelos de inteligencia artificial aplicados a la detección de fraude en transacciones digitales. El principal aporte del trabajo radica en mostrar que los modelos de inteligencia artificial pueden apoyar la detección preliminar de fraude mediante alertas tempranas, especialmente cuando se utilizan métricas adecuadas para bases desbalanceadas y se comparan distintos enfoques bajo una misma metodología. En particular, la red neuronal *Keras* y *Random Forest* mostraron los mejores resultados sobre el *dataset* completo, lo que sugiere que los modelos capaces de capturar relaciones no lineales e interacciones entre variables pueden ser especialmente útiles en contextos financieros.

Finalmente, si bien los resultados no deben aplicarse directamente sin validación previa, entregan una base metodológica útil para seleccionar modelos, definir métricas relevantes y comprender la importancia de contar con datos completos, interpretables y representativos. En un entorno real, el valor de este tipo de herramientas no estaría en reemplazar la supervisión humana, sino en apoyar la priorización de transacciones sospechosas, reducir tiempos de revisión y fortalecer la prevención de delitos que puedan afectar la calidad de vida de los usuarios de estas plataformas.

11. Anexos

Anexo 1: Tabla de variables con más del 80% de nulos en la base inicial.

index	variable	nulos	porcentaje_nulos	base
0.0	id_24	585793.0	99.20%	df_train
1.0	id_25	585408.0	99.13%	df_train
2.0	id_07	585385.0	99.13%	df_train
3.0	id_08	585385.0	99.13%	df_train
4.0	id_21	585381.0	99.13%	df_train
5.0	id_26	585377.0	99.13%	df_train
6.0	id_27	585371.0	99.12%	df_train
7.0	id_23	585371.0	99.12%	df_train
8.0	id_22	585371.0	99.12%	df_train
9.0	dist2	552913.0	93.63%	df_train
10.0	D7	551623.0	93.41%	df_train
11.0	id_18	545427.0	92.36%	df_train
12.0	D13	528588.0	89.51%	df_train
13.0	D14	528353.0	89.47%	df_train
14.0	D12	525823.0	89.04%	df_train
15.0	id_03	524216.0	88.77%	df_train
16.0	id_04	524216.0	88.77%	df_train
17.0	D6	517353.0	87.61%	df_train
18.0	id_33	517251.0	87.59%	df_train
19.0	id_10	515614.0	87.31%	df_train
20.0	id_09	515614.0	87.31%	df_train
21.0	D9	515614.0	87.31%	df_train
22.0	D8	515614.0	87.31%	df_train
23.0	id_30	512975.0	86.87%	df_train
24.0	id_32	512954.0	86.86%	df_train
25.0	id_34	512735.0	86.82%	df_train
26.0	id_14	510496.0	86.45%	df_train
27.0	V142	508595.0	86.12%	df_train
28.0	V158	508595.0	86.12%	df_train
29.0	V140	508595.0	86.12%	df_train
30.0	V162	508595.0	86.12%	df_train
31.0	V141	508595.0	86.12%	df_train
32.0	V161	508595.0	86.12%	df_train

index	variable	nulos	porcentaje_nulos	base
33.0	V157	508595.0	86.12%	df_train
34.0	V146	508595.0	86.12%	df_train
35.0	V156	508595.0	86.12%	df_train
36.0	V155	508595.0	86.12%	df_train
37.0	V154	508595.0	86.12%	df_train
38.0	V153	508595.0	86.12%	df_train
39.0	V149	508595.0	86.12%	df_train
40.0	V147	508595.0	86.12%	df_train
41.0	V148	508595.0	86.12%	df_train
42.0	V163	508595.0	86.12%	df_train
43.0	V139	508595.0	86.12%	df_train
44.0	V138	508595.0	86.12%	df_train
45.0	V160	508589.0	86.12%	df_train
46.0	V151	508589.0	86.12%	df_train
47.0	V152	508589.0	86.12%	df_train
48.0	V145	508589.0	86.12%	df_train
49.0	V144	508589.0	86.12%	df_train

Anexo 2: Resumen estadístico de variables numéricas por clase

level_0	level_1	0	1
V168	mean	5.6279	2.6969
V168	median	0.0	1.0
V168	std	53.3473	5.8075
V128	mean	193.8807	163.3006
V128	median	0.0	0.0
V128	std	2905.6856	525.1203
V308	mean	217.4708	277.3183
V308	median	0.0	0.0
V308	std	2917.6526	1049.0713
V97	mean	1.6408	1.04
V97	median	0.0	0.0
V97	std	26.9231	2.2528
V280	mean	1.8982	1.9139
V280	median	0.0	0.0
V280	std	27.1362	9.3671
V103	mean	1.2082	0.5586

level_0	level_1	0	1
V103	median	0.0	0.0
V103	std	24.94	1.5091
V179	mean	4.6271	1.9788
V179	median	0.0	0.0
V179	std	49.7959	4.7771
V134	mean	136.9199	86.6948
V134	median	0.0	0.0
V134	std	2686.3316	378.4996
V213	mean	514.7151	139.0091
V213	median	0.0	0.0
V213	std	5384.9572	563.8426
V295	mean	1.3684	1.254
V295	median	0.0	0.0
V295	std	25.283	8.9565

Anexo 3: Varianza explicada de los primeros 43 componentes principales

componente	varianza_explicada	varianza_acumulada
PC1	14.31%	14.31%
PC2	11.10%	25.42%
PC3	5.73%	31.15%
PC4	3.71%	34.86%
PC5	3.45%	38.31%
PC6	2.95%	41.26%
PC7	2.84%	44.11%
PC8	2.65%	46.76%
PC9	2.58%	49.34%
PC10	2.41%	51.76%
PC11	2.07%	53.83%
PC12	1.94%	55.77%
PC13	1.78%	57.56%
PC14	1.60%	59.16%
PC15	1.51%	60.67%
PC16	1.41%	62.07%
PC17	1.31%	63.39%
PC18	1.23%	64.62%

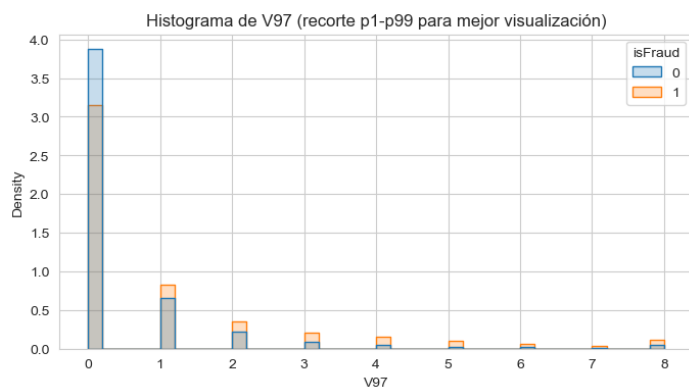
componente	varianza_explicada	varianza_acumulada
PC19	1.16%	65.78%
PC20	1.10%	66.88%
PC21	1.03%	67.91%
PC22	0.96%	68.87%
PC23	0.91%	69.78%
PC24	0.87%	70.65%
PC25	0.80%	71.45%
PC26	0.78%	72.23%
PC27	0.71%	72.94%
PC28	0.67%	73.60%
PC29	0.64%	74.25%
PC30	0.63%	74.87%
PC31	0.61%	75.49%
PC32	0.60%	76.08%
PC33	0.56%	76.64%
PC34	0.54%	77.18%
PC35	0.51%	77.69%
PC36	0.51%	78.20%
PC37	0.48%	78.68%
PC38	0.46%	79.15%
PC39	0.46%	79.61%
PC40	0.43%	80.04%
PC41	0.42%	80.45%
PC42	0.39%	80.84%
PC43	0.38%	81.22%

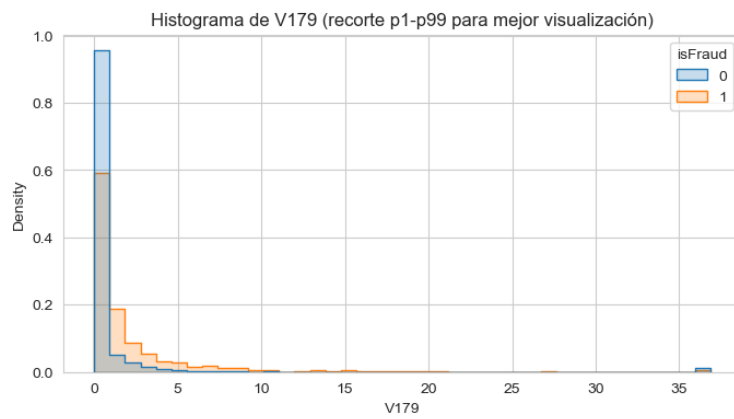
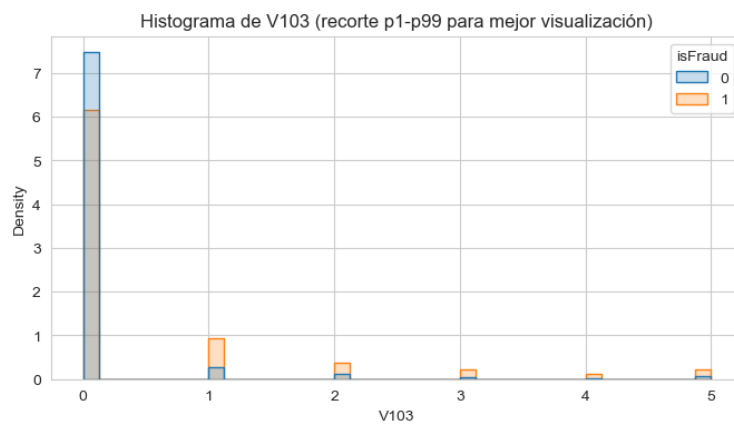
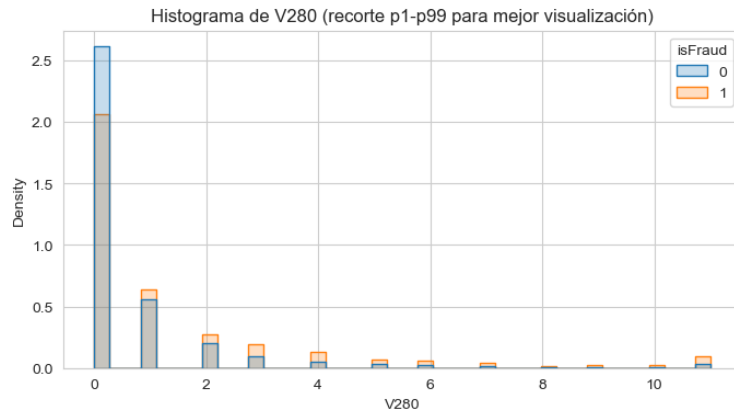
Anexo 4: Variables numéricas más influyentes según PCA

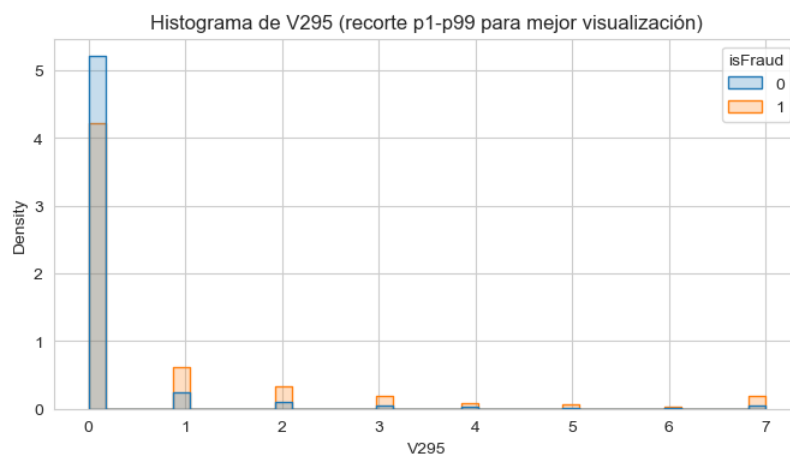
variable	importancia_global
V168	0.0195
V128	0.0195
V308	0.0194
V97	0.0194
V280	0.0194
V103	0.0193

variable	importancia_global
V179	0.0193
V134	0.0193
V213	0.0193
V295	0.0193
V318	0.0192
V96	0.0192
V127	0.0192
V204	0.0192
V178	0.0191
V102	0.0191
V133	0.0191
V212	0.0189
V105	0.0189
V202	0.0187
V306	0.0187
V279	0.0187
V307	0.0187
V126	0.0187
V95	0.0187
V167	0.0186
V293	0.0186
V298	0.0186
V101	0.0185
V132	0.0185

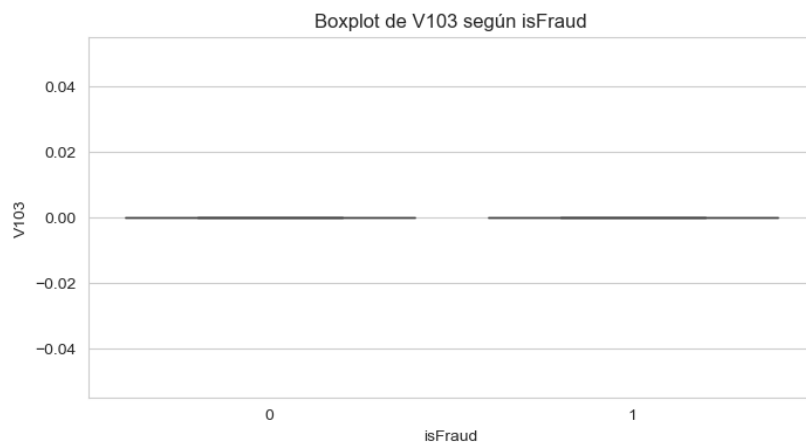
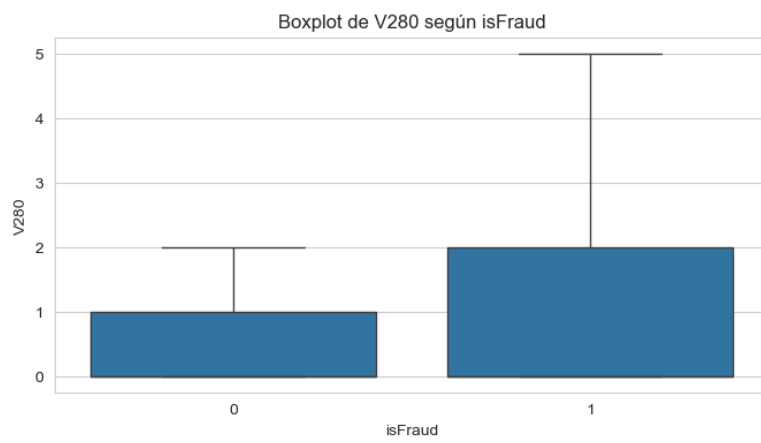
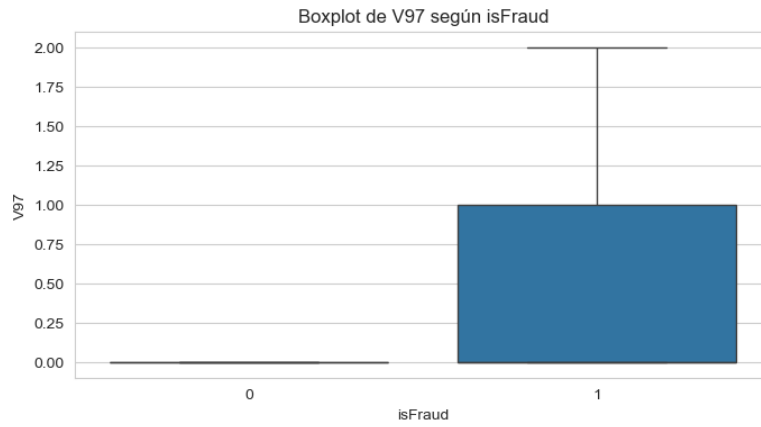
Anexo 5: Histogramas variables numéricas más relevantes según PCA

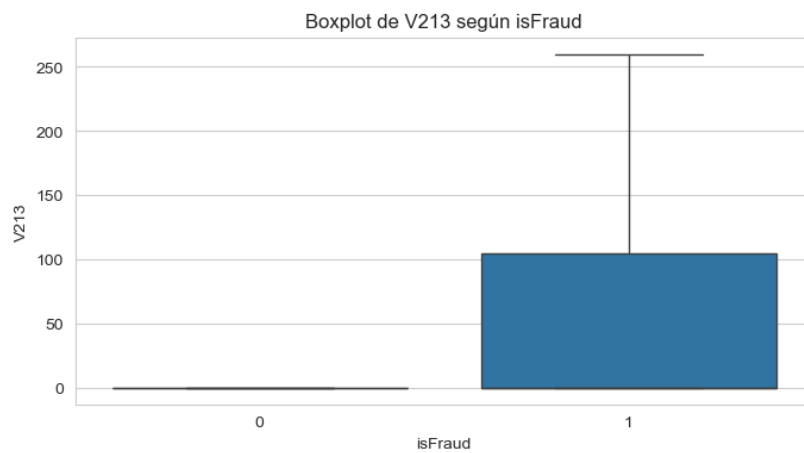
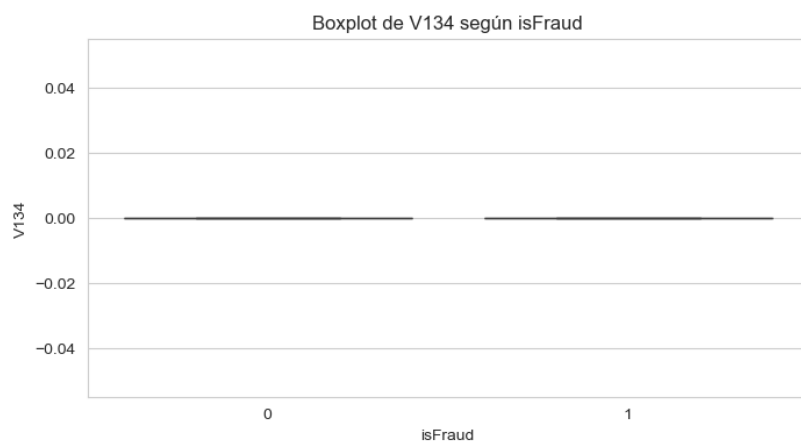
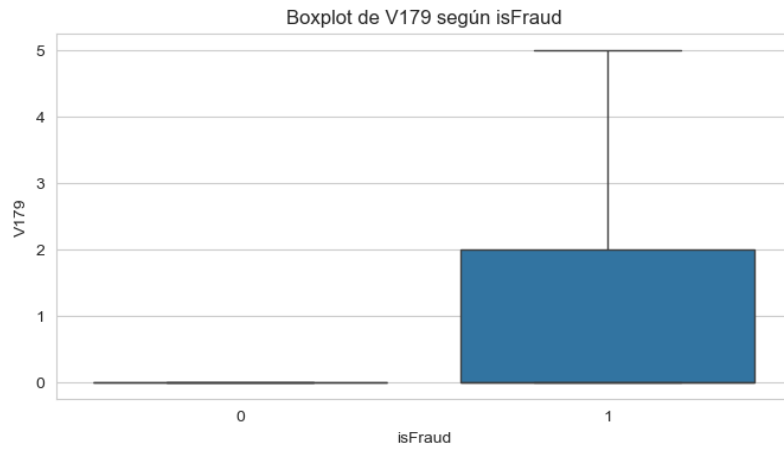


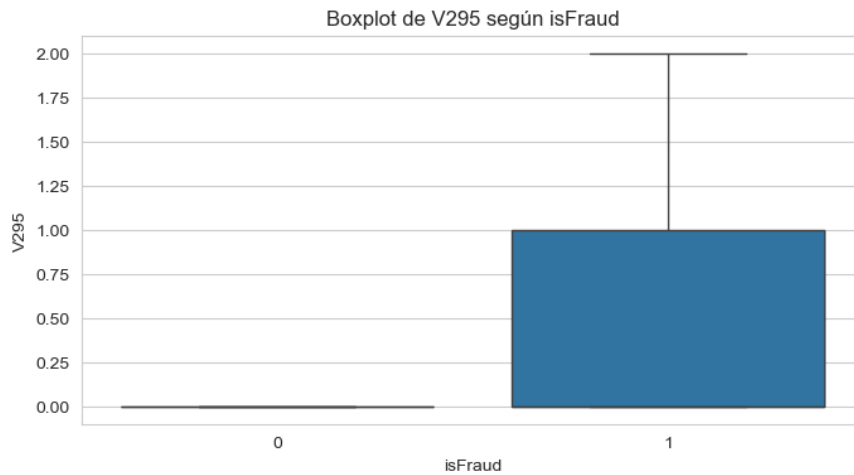




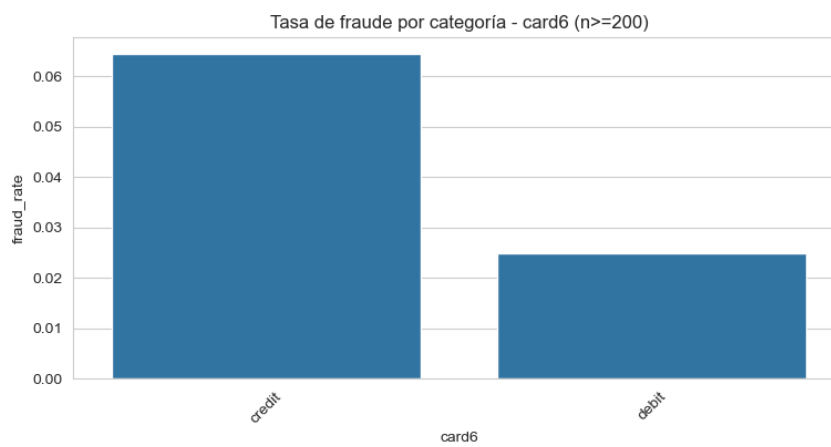
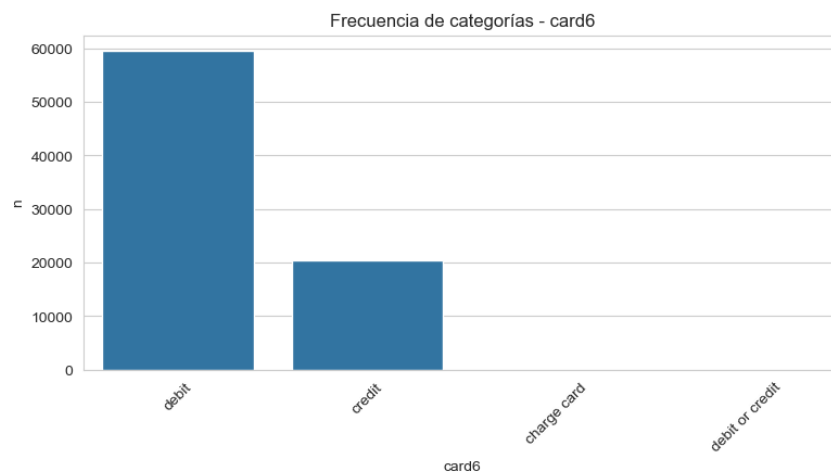
Anexo 6: *Boxplots* más relevantes de variables numéricas

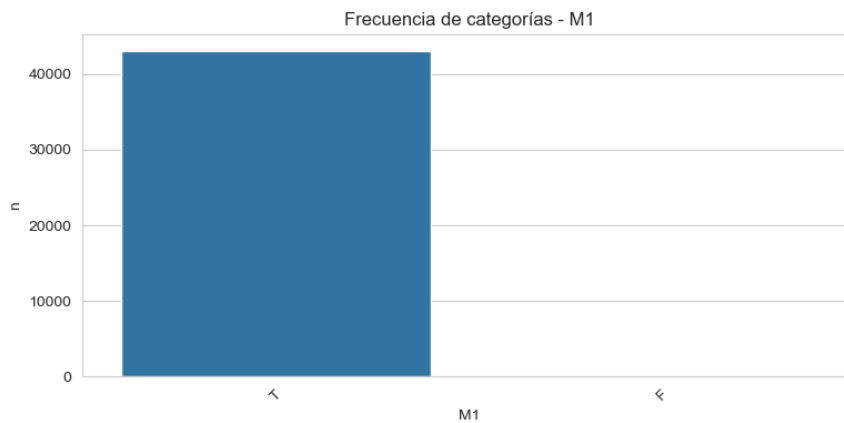
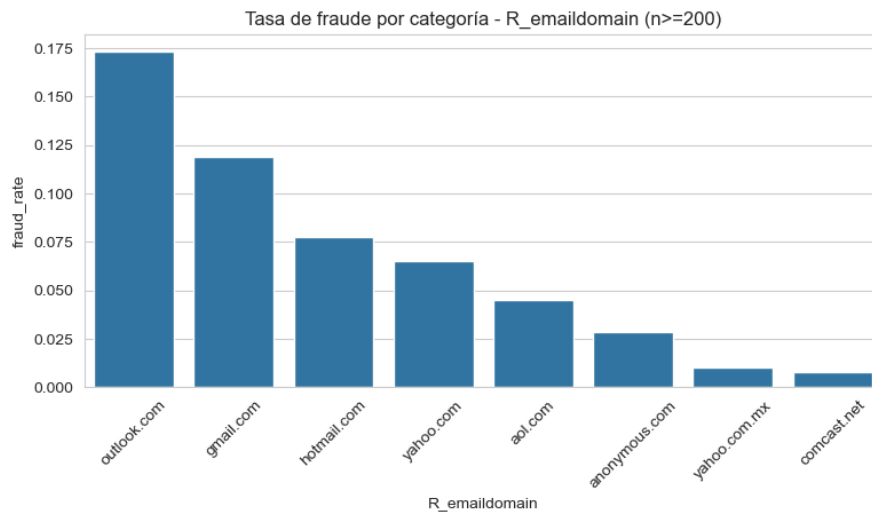
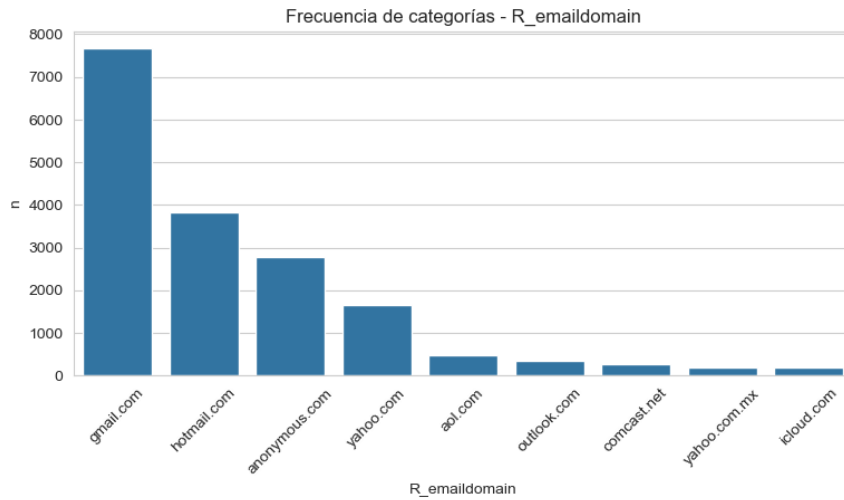


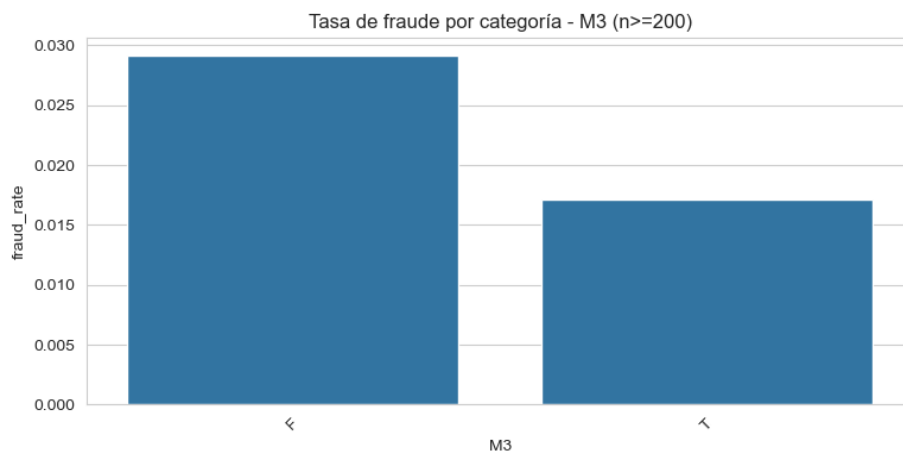
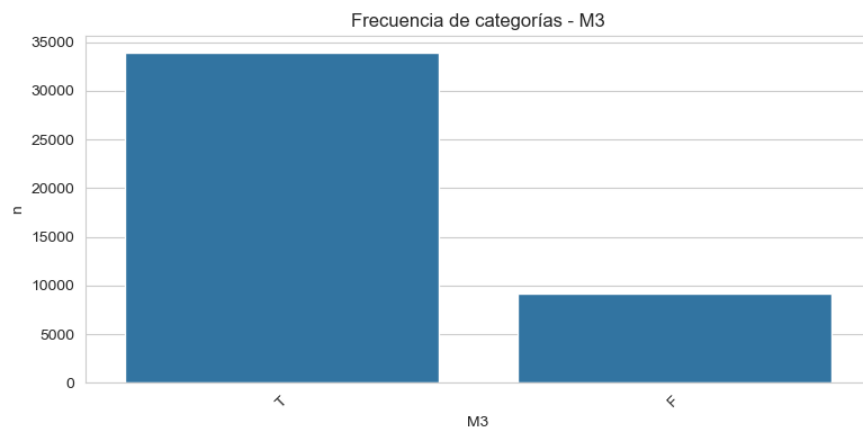
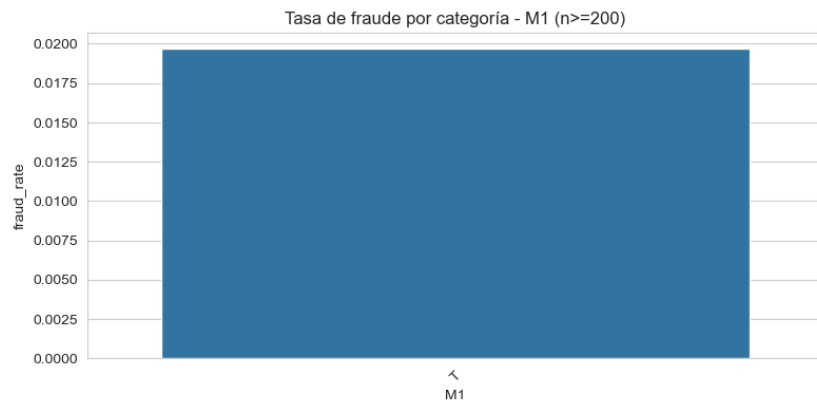


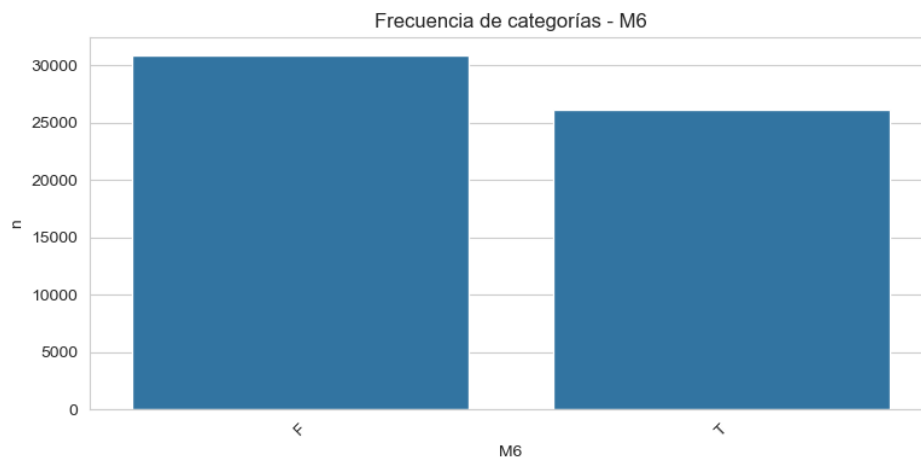
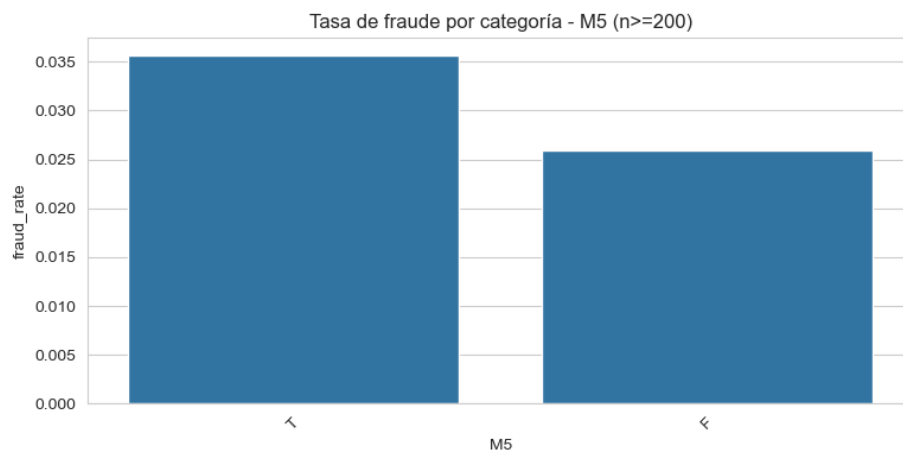
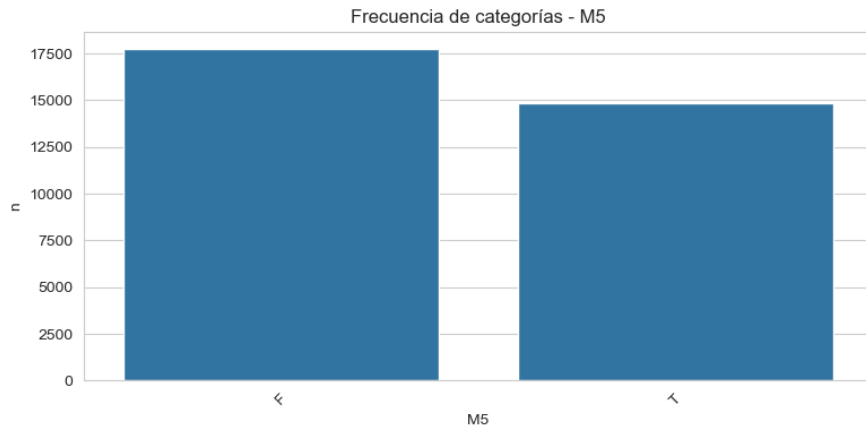


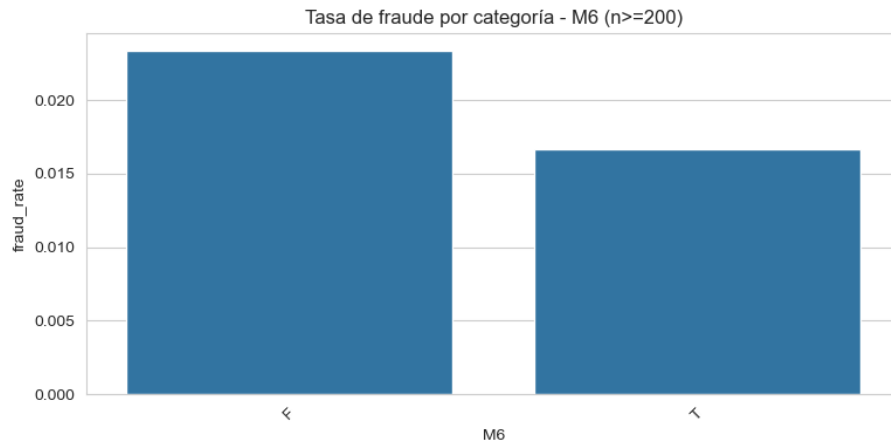
Anexo 7: Frecuencia y tasa de fraude de variables categóricas



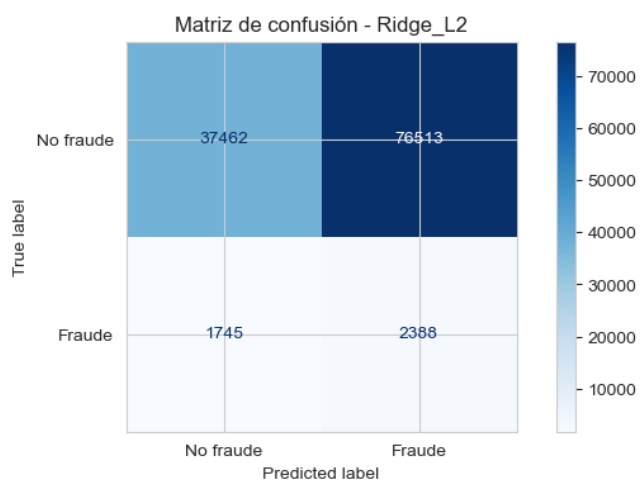
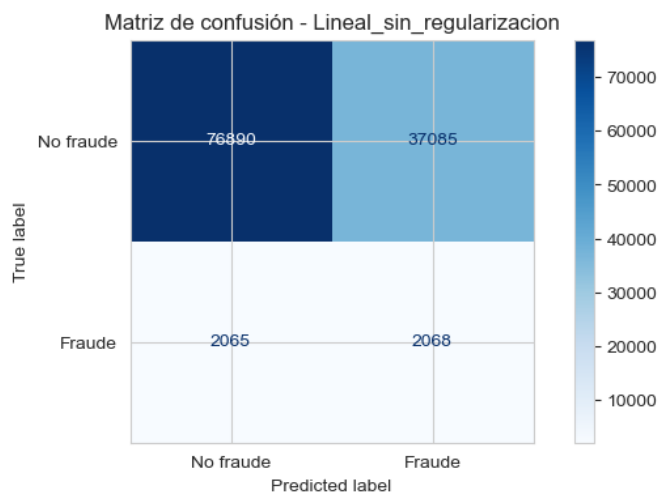


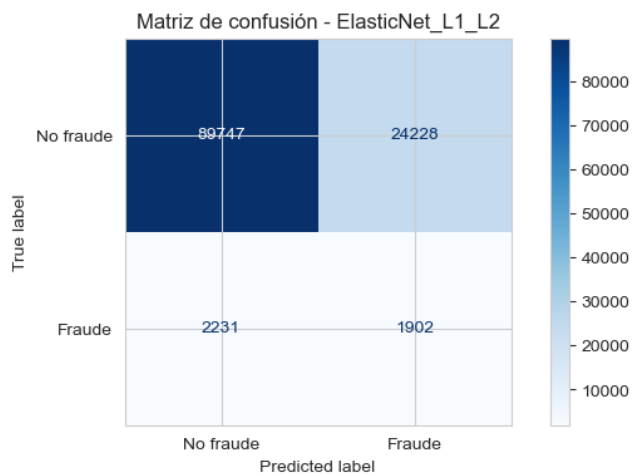
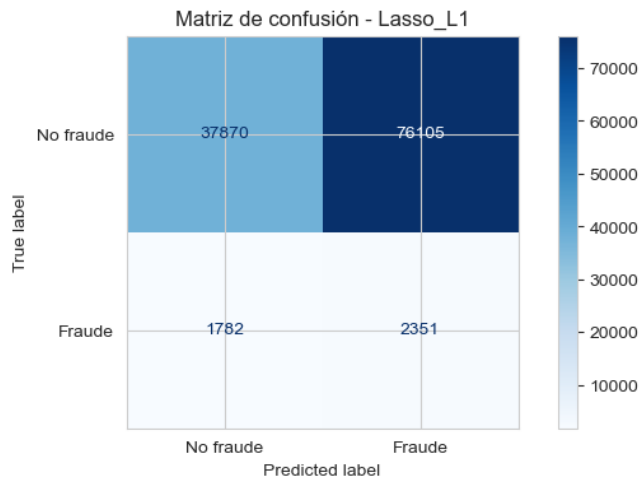




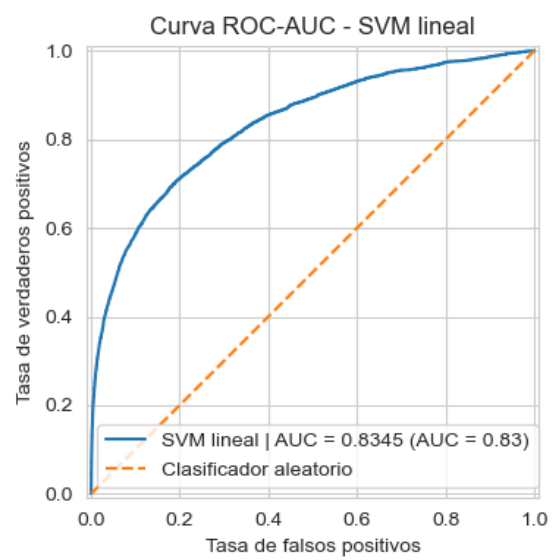
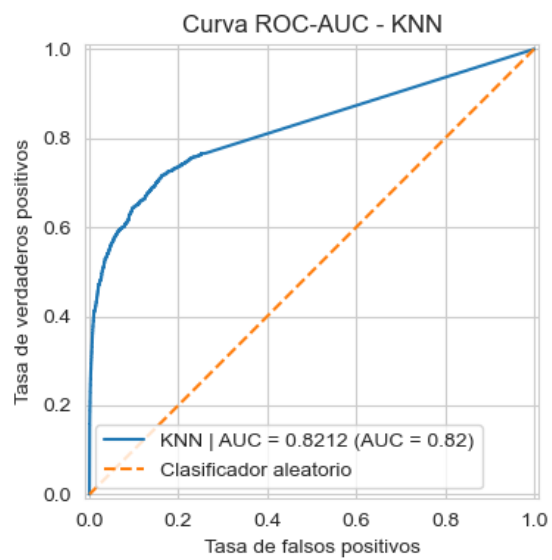


Anexo 8: Matrices de confusión de modelos de regresión lineal

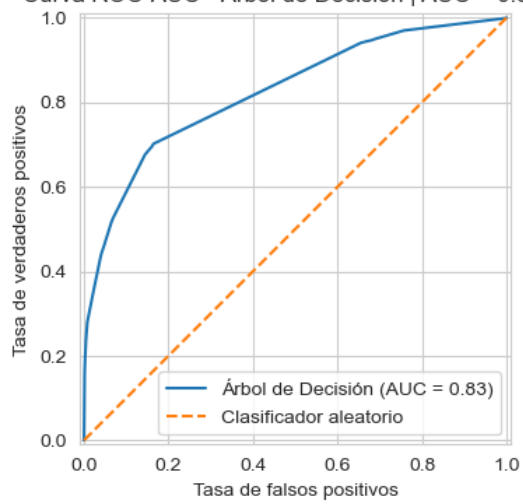




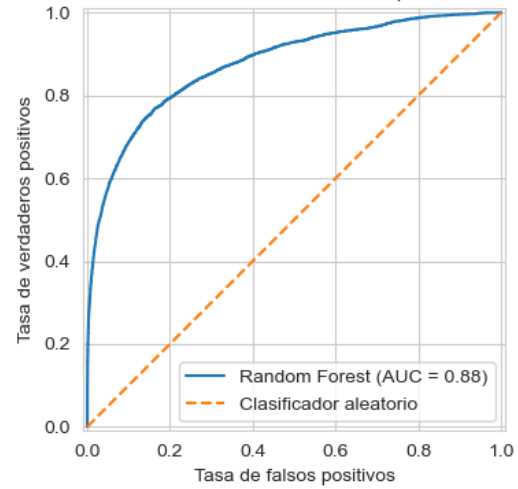
Anexo 9: Curvas ROC-AUC de los modelos de clasificación evaluados



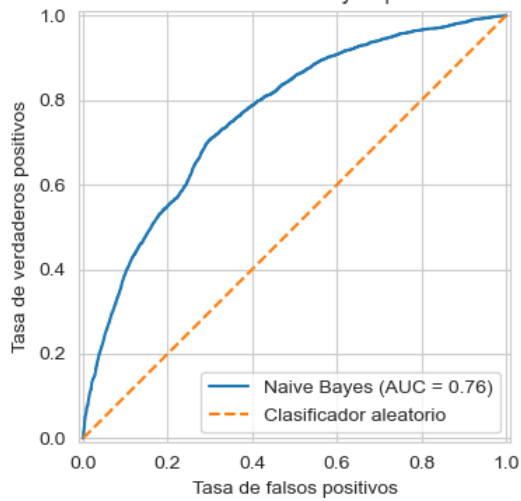
Curva ROC-AUC - Árbol de Decisión | AUC = 0.8259



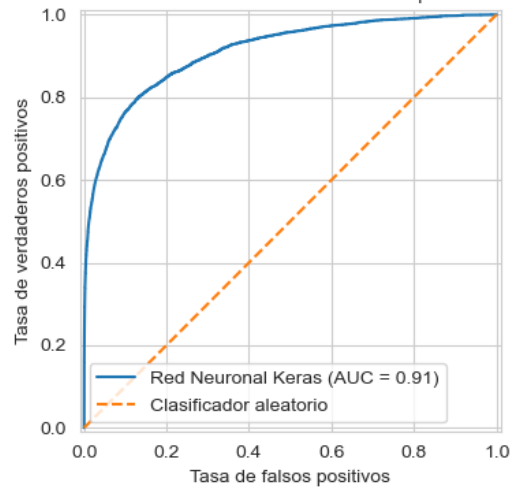
Curva ROC-AUC - Random Forest | AUC = 0.8779



Curva ROC-AUC - Naive Bayes | AUC = 0.7631



Curva ROC-AUC - Red Neuronal Keras | AUC = 0.9118



12. Referencias

- Alvarez, F. (2020). *Machine Learning en la detección de fraudes de comercio electrónico aplicado a los servicios bancarios*. Software Socialconsultores.
- Ávila, T. (2020). La inteligencia artificial y sus aplicaciones en medicina. *EL SEVIER*, 779. doi:<https://doi.org/10.1016/j.aprim.2020.04.013>
- Baroni, M. J. (2019). Las narrativas de la inteligencia artificial. *Revista de Bioética y Derecho.*, 7. doi:<https://doi.org/10.1344/RBD2019.0.27280>.
- Beristain, J. (2025). Modelos predictivos de churn en telecomunicaciones: evaluación comparativa y análisis de aplicabilidad con técnicas de machine learning. *Tesis de grado*. Universidad Técnica Federico Santa María, Santiago de Chile.
- Breiman, L. (2001). RANDOM FORESTS. *Memoria de título*. Universidad de California, California.
- Chugh, V. (1 de Octubre de 2024). *AUC y ROC en aprendizaje automático*. Obtenido de datacamp: <https://www.datacamp.com/es/tutorial/auc>
- Dávila, A. L. (2018). *Neural Networks: The backpropagation Algorithmic*. Estados Unidos: College of William and Mary.
- EY Chile. (24 de Septiembre de 2025). *¿Cómo utilizar la IA Generativa de manera efectiva, estratégica y responsable en la industria financiera?* Obtenido de EY: https://www.ey.com/es_cl/insights/banking-capital-markets/utilizar-ia-generativa-efectiva-estrategica-responsable
- Goodfellow, I. (2016). *Deep learning*. MIT. Obtenido de www.deeplearningbook.org

- Hussain Mahdi, M. D. (2010). Credit Fraud Detection in the Banking Sector in UK: A Focus on E-Business. *Fourth International Conference on Digital Society*, (pág. 234). Reino Unido.
- IEEE Computational Intelligence Society (IEEE-CIS). (3 de Octubre de 2019). *Detección de fraude IEEE-CIS*. Obtenido de kaggle: <https://www.kaggle.com/competitions/ieee-fraud-detection/overview>
- LeCun, Y. (2015). Deep Learning. *Review Insight*, 436-442. doi:10.1038/nature14539
- Lucidchart. (s.f.). *Qué es un árbol de decisiones*. Obtenido de Lucidchart: <https://www.lucidchart.com/pages/es/tutorial/arbol-de-decisiones>
- Manrique, E. (2020). Machine Learning: análisis de lenguajes de programación y herramientas para desarrollo. *Revista Ibérica de Sistemas y Tecnologías de Información*, 586-599.
- Mcgrath, A. (24 de Julio de 2025). *¿Qué es la inteligencia artificial (IA) en las finanzas?* Obtenido de IBM: <https://www.ibm.com/mx-es/think/topics/artificial-intelligence-finance>
- Mucci, T. (5 de Noviembre de 2025). *Sobreajuste vs Subajuste*. Obtenido de IBM: <https://www.ibm.com/think/topics/overfitting-vs-underfitting>
- Nieves, S. (2019). *Influencia de la inteligencia artificial en el sector financiero. Desarrollo de un modelo de predicción de transacciones futuras*. Valencia, España: Universidad Politécnica de Valencia.

- Nilson Report. (2022). Global network cards. *Nilson Report*, 1. Obtenido de <https://nilsonreport.com/articles/card-fraud-losses-worldwide>
- OpenAI. (15 de Septiembre de 2025). *Cómo la gente usa ChatGPT*. Obtenido de OpenIA: https://openai.com/index/how-people-are-using-chatgpt/?utm_source=chatgpt.com
- Python. (s.f.). *Naive Bayes*. Obtenido de Scikit Learn: https://scikit-learn.org/stable/modules/naive_bayes.html
- Python. (s.f.). *Random Forest Classifier*. Obtenido de Scikit Learn: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>
- Rajesh, K. (14 de Febrero de 2025). *Guía de algoritmo de agrupación DBSCAN*. Obtenido de datacamp: <https://www.datacamp.com/es/tutorial/dbscan-clustering-algorithm>
- Rayo, C. (2020). *Prototipo de detección de fraudes con tarjetas de crédito basado en inteligencia artificial aplicado a un banco peruano*. Lima, Perú: Universidad de Lima.
- Reguera, P. F. (2021). *Cuadernos de Econometría*. Santiago: Departamento de Industrias, Universidad Técnica Federico Santa María.
- Robins, N. (10 de Mayo de 2024). *El director ejecutivo de la empresa de publicidad más grande del mundo fue blanco de una estafa deepfake*. Obtenido de The Guardian: <https://www.theguardian.com/technology/article/2024/may/10/ceo-wpp-deepfake-scam>
- Rodríguez, V. (2020). Fraudes financieros, salud y calidad de vida: un estudio cualitativo. *Elsevier*, 268 - 275. doi:<https://doi.org/10.1016/j.gaceta.2019.11.006>

Sanhueza, J. M. (4 de Septiembre de 2024). *BancoEstado descubre fraude por \$6100 millones y se querella por asociación ilícita y lavado de dinero* . Obtenido de Interferencia: <https://interferencia.cl/articulos/bancoestado-descubre-fraude-por-6100-millones-y-se-querella-por-asociacion-ilicita-y>

Sarker, I. H. (8 de Marzo de 2021). MachineLearning: Algorithms, Real-World Applications and Research Directions. *Preprints*, 1-23.
doi:<https://doi.org/10.20944/preprints202103.0216.v1>

Schmidhuber, J. (2015). Neural Network. *El Sevier*, 85-117.
doi:<http://dx.doi.org/10.1016/j.neunet.2014.09.003>

Selvaraj, N. (25 de Abril de 2024). *8 modelos de machine learning explicados en 20 minutos* . Obtenido de datacamp: <https://www.datacamp.com/es/blog/machine-learning-models-explained>

Shahroudnejad, A. (2021). *A Survey on Understanding, Visualizations, and Explanation of Deep Neural Networks*. Montreal: Concordia Institute for Information Systems Engineering.

Teletrece. (19 de Septiembre de 2025). *Presidente Boric víctima de la IA: usan su voz e imagen en estafa digital*. Obtenido de T13: <https://www.t13.cl/noticia/nacional/presidente-boric-victima-ia-usan-su-voz-e-imagen-estafa-digital-19-9-2025>

Universidad de Cambridge. (2009). 13 Text classification and Naive Bayes. *Cambridge University Press*, 256 - 287. Obtenido de <https://nlp.stanford.edu/IR-book/pdf/13bayes.pdf>

Vallalta, J. F. (s.f.). *CRISP-DM: una metodología para minería de datos en salud*. Obtenido de IA Health Data Miner : <https://healthdataminer.com/data-mining/crisp-dm-una-metodologia-para-mineria-de-datos-en-salud/>

Valor Añadido Fundaciones y Servicios. (20 de Octubre de 2023). *Una guía de machine learning y detección de fraude*. Obtenido de VASS: <https://vasscompany.com/latam/es/insights/blogs-articles/deteccion-fraude-machine-learning/#2-3>

Vention. (2025). *Estadísticas de adopción de IA por industrias y países: resumen de 2025*. Obtenido de Vention S.A.: <https://ventionteams.com/solutions/ai/adoption-statistics>

Vodworks. (27 de Noviembre de 2024). *How much does AI cost: A financial analysis for C-level executives*. Obtenido de Vodworks: <https://vodworks.com/blogs/how-much-does-ai-cost/>

Zhou, S. (2024). *Comparison of machine learning models* . Foshan: Dalian Polytechnic University.