UIDE
Powered by
Arizona State University

eig Business School

Maestría en

Ciencia de Datos y Máquinas de Aprendizaje mención en Inteligencia Artificial

Trabajo de investigación previo a la obtención del título de Magíster en Ciencia de Datos y Máquinas de Aprendizaje con mención en Inteligencia Artificial

AUTORES:

Freddy Gonzalo Lituma Perero

Iván Patricio Ortega Salas

Juan Carlos Suárez León

Eleanor Alexandra Varela Tapia

TUTORES:

Alejandro Cortés López MSc.

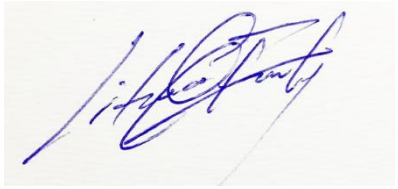
Jefferson Rodríguez Chivatá MSc.

Técnicas de minería de datos aplicadas a la predicción de fraude en tarjetas de crédito

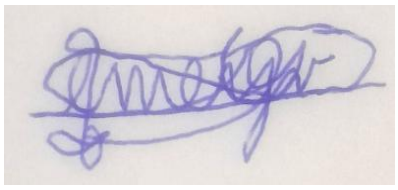
Certificación de autoría

Nosotros, **Freddy Gonzalo Lituma Perero, Iván Patricio Ortega Salas, Juan Carlos Suárez León y Eleanor Alexandra Varela Tapia,** declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido presentado anteriormente para ningún grado o calificación profesional y que se ha consultado la bibliografía detallada.

Cedemos nuestros derechos de propiedad intelectual a la Universidad Internacional del Ecuador (UIDE), para que sea publicado y divulgado en internet, según lo establecido en la Ley de Propiedad Intelectual, su reglamento y demás disposiciones legales.



Firma del graduando
Freddy Gonzalo Lituma Perero



Firma del graduando
Iván Patricio Ortega Salas

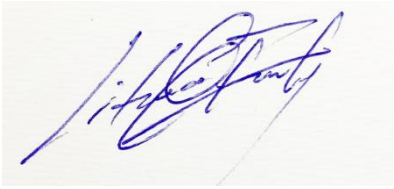
Firma del graduando
Juan Carlos Suárez León

Firma del graduando
Eleanor Alexandra Varela Tapia

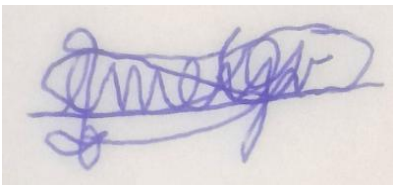
Autorización de Derechos de Propiedad Intelectual

Nosotros, **Freddy Gonzalo Lituma Perero, Iván Patricio Ortega Salas, Juan Carlos Suárez León y Eleanor Alexandra Varela Tapia**, en calidad de autores del trabajo de investigación titulado ***Técnicas de minería de datos aplicadas a la predicción de fraude en tarjetas de crédito***, autorizamos a la Universidad Internacional del Ecuador (UIDE) para hacer uso de todos los contenidos que nos pertenecen o de parte de los que contiene esta obra, con fines estrictamente académicos o de investigación. Los derechos que como autores nos corresponden, lo establecido en los artículos 5, 6, 8, 19 y demás pertinentes de la Ley de Propiedad Intelectual y su Reglamento en Ecuador.

D. M. Quito, enero 2025



Firma del graduando
Freddy Gonzalo Lituma Perero



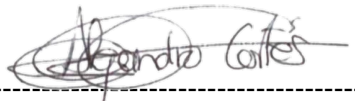
Firma del graduando
Iván Patricio Ortega Salas

Firma del graduando
Juan Carlos Suárez León

Firma del graduando
Eleanor Alexandra Varela Tapia

Aprobación de dirección y coordinación del programa

Nosotros, **Alejandro Cortés Msc. e Iván Reyes Ch. Mgtr.,** declaramos que: Freddy Gonzalo Lituma Perero, Iván Patricio Ortega Salas, Juan Carlos Suárez León y Eleanor Alexandra Varela Tapia, son los autores exclusivos de la presente investigación y que ésta es original, auténtica y personal de ellos.



Alejandro Cortés Msc.
Director de la
Maestría en Ciencia de datos y Maquinas de Aprendizaje mención Inteligencia Artificial EIG

Iván Reyes Ch. Mgtr.
Coordinador de la
Coordinador de Posgrados Escuela de Ciencia de la Computación UIDE

DEDICATORIA

A mi madre, Mirna Perero, quien ha sido mi mayor apoyo, mi guía y mi refugio. Su amor incondicional, sacrificios y ejemplo de dedicación me han fortalecido cada día. A mi padre, Freddy Lituma, por sus enseñanzas de esfuerzo y perseverancia. A mi hermana, Nohelia, y a toda mi familia, por su compañía y motivación constante.
Freddy Gonzalo Lituma Perero

Dedico este trabajo a mis hijos Nikolái y Gary, a mi gran esposa Lily que han estado muchos años conmigo dándome fuerza y alegría. También dedico a mi querida madre María y a mi buen hermano Boris, a mi abuelita Edita y a mi tía Elenita por haberme criado; a mi sobrina Emily, a mis suegros Marcia y Marcelo y a mi padre Patricio. También a mis tías Martha, Mariana, Matilde, Amparito, Beatriz y Teresa. Finalmente, también a mis abuelitos Cornelio, Víctor y Lucrecia.
Iván Patricio Ortega Salas

Dedico este logro a mi familia y compañeros, cuyo apoyo incansable, constancia e inspiración han sido fundamentales para que pudiera culminarlo con éxito. Agradezco profundamente cada momento de aliento y guía que han compartido conmigo.
Juan Carlos Suarez León

Dedico el esfuerzo realizado a Dios, a mi familia y a todas las personas que me conocen que de una forma u otra han sido el soporte en todo momento de mi vida para crecer tanto en lo personal como en lo profesional.
Eleanor Alexandra Varela Tapia

AGRADECIMIENTOS

A Dios, por darme la fuerza y las oportunidades para llegar hasta aquí. A mis excompañeros del Parque Nacional Galápagos, cuyas enseñanzas y guía marcaron una diferencia en mi vida. A mis mentores, Julio Cortez, Jack Román y Juan Mite, por su orientación; a mis compañeros de tesis, por su colaboración durante este proceso, y a mis amigos, por su apoyo incondicional. Todos ustedes han sido una gran inspiración en mi crecimiento personal y profesional.

Freddy Gonzalo Lituma Perero

Agradezco mucho a mi esposa Lily y a mis hijos Nikolai y Gary por su amor y apoyo constante. Agradezco también a mi madre María y mi hermano Boris por estar conmigo en los buenos y malos momentos dándome fortaleza. También agradezco por su ayuda a mis compañeros de grupo Eleanor, Freddy y Juan quienes mostraron mucha dedicación y profesionalismo en esta maestría. Finalmente agradezco a los profesores de esta maestría por los conocimientos proporcionados.

Iván Patricio Ortega Salas

Dedico este logro a mi familia y compañeros, cuyo apoyo incansable, constancia e inspiración han sido fundamentales para que pudiera culminarlo con éxito. Agradezco profundamente cada momento de aliento y guía que han compartido conmigo.

Juan Carlos Suarez León

Agradezco a Dios, a mi madre Edith, a mi padre Jorge (+), a mi esposo Iván, a mi hijo Christopher y a mis hermanos por brindarme el apoyo en cada momento de mi vida para ser una mejor persona. También a los profesores del máster y compañeros de grupo Freddy, Iván y Juan por compartir el conocimiento aprendido en cada sesión.

Eleanor Alexandra Varela Tapia

RESUMEN

El fraude en tarjetas de crédito pone en peligro a los consumidores y a las entidades financieras, empleando datos no permitidos para llevar a cabo operaciones fraudulentas, provocando pérdidas financieras y perjudicando la confianza del usuario. Con el crecimiento de los pagos con tarjetas de crédito, las estrategias de los delincuentes se han agudizado, aumentando la demanda de sistemas sofisticados de identificación y prevención de fraudes por lo que se necesita la aplicación de modelos predictivos de aprendizaje automático para detectar y minimizar estos fraudes, salvaguardando a las instituciones financieras y a sus consumidores. El objetivo del proyecto es el desarrollo de modelos predictivos de detección de fraudes mediante técnicas de minería de datos para la identificación de transacciones fraudulentas en tarjetas de crédito con alta precisión. Durante el entrenamiento de los modelos se aplicó la metodología del proceso KDD, haciendo uso de técnicas de preprocesamiento de datos, algoritmos de aprendizaje supervisado, evaluación de los modelos con métricas de clasificación. Adicionalmente, se usaron técnicas de optimización con RandomSearchCV y validación cruzada para selección de los hiperparámetros que evite el sobreajuste de los modelos y poder generalizar los hallazgos de la predicción frente a datos nuevos. Como resultado se obtuvo la predicción de fraude en una aplicación web local mediante los 3 mejores modelos entrenados con los algoritmos de Árbol de decisión con accuracy 0.977, auc 0.961, falsos negativos 13; Random Forest con accuracy 0.987, auc 0.968, falsos negativos 20 y Gradient Boosting con accuracy 0.989, auc 0.973, falsos negativos 17.

Palabras Clave: predicción, fraude, tarjetas de crédito, minería de datos, aprendizaje automático, aprendizaje supervisado

ABSTRACT

Credit card fraud endangers consumers and financial institutions by using unauthorized data to carry out fraudulent transactions, causing financial losses and damaging user confidence. With the growth of credit card payments, criminals' strategies have become more sophisticated, increasing the demand for sophisticated fraud identification and prevention systems, requiring the application of predictive machine learning models to detect and minimize these frauds, safeguarding financial institutions and their consumers. The objective of the project is to develop predictive fraud detection models using data mining techniques to identify fraudulent credit card transactions with high accuracy. During model training, the KDD process methodology was applied, making use of data preprocessing techniques, supervised learning algorithms, model evaluation with classification metrics. Additionally, optimization techniques with RandomSearchCV and cross validation were used to select the hyperparameters to avoid overfitting the models and to generalize the prediction findings to new data. As a result, fraud prediction in a local web application was obtained using the 3 best models trained with the algorithms Decision Tree with accuracy 0.977, auc 0.961, false negatives 13; Random Forest with accuracy 0.987, auc 0.968, false negatives 20 and Gradient Boosting with accuracy 0.989, auc 0.973, false negatives 17.

Keywords: prediction, fraud, credit card, data mining, machine learning, machine learning, supervised learning

TABLA DE CONTENIDOS

Contenido

LISTA DE TABLAS

LISTA DE FIGURAS

CAPITULO 1

INTRODUCCION

1. PLANTEAMIENTO DEL PROBLEMA E IMPORTANCIA DEL ESTUDIO

1.1. Introducción

El fraude con tarjetas de crédito es uno de los delitos más comunes y desafiantes en el mundo del comercio digital. En el año 2022, se registró una pérdida de 33.500 millones de dólares en fraudes con tarjetas a nivel mundial, en relación con los 28.400 millones de 2020 y los 27.900 millones de 2018 (Nilson Report, 2023) . En la Unión Europea, se presentaron como fraudulentas 1.530 millones de euros en transacciones con tarjeta en el 2021, con una baja del 7% en la cantidad total de transacciones fraudulentas con tarjeta respecto a 2020 (BCE, 2023). En el Ecuador, el 22% de las entidades expresan preocupación por el fraude a través de medios de pago digital (Superintendencia de Economía Popular y Solidaria, 2021).

En Ecuador, la Ley del Código Orgánico Integral Penal (COIP), establece en el que:

> Artículo 186. Estafa. - La persona que, para obtener un beneficio patrimonial para sí misma o para una tercera persona, mediante la simulación de hechos falsos o la deformación u ocultamiento de hechos verdaderos, induzca a error a otra, con el fin de que realice un acto que perjudique su patrimonio o el de una tercera, será sancionada con pena privativa de libertad de cinco a siete años. La pena máxima se aplicará a la persona que:
>
> 1. Defraude mediante el uso de tarjeta de crédito, débito, pago o similares, cuando ella sea alterada, clonada, duplicada, hurtada, robada u obtenida sin legítimo consentimiento de su propietario (Registro Oficial Suplemento, 2021).

Debido a la naturaleza cambiante y adaptativa del fraude, las técnicas tradicionales de detección de fraudes, como las reglas heurísticas, han demostrado ser insuficientes para

enfrentar los desafíos actuales (Sosa, 2007). En este contexto, los avances en la ciencia de los datos, particularmente el Machine Learning (aprendizaje automático) y el Deep Learning (aprendizaje profundo), ofrecen una solución más sólida para abordar este problema. Estos modelos pueden encontrar patrones complejos y no lineales, lo que los convierte en una herramienta útil para la detección temprana de fraudes (Ameijeiras et al., 2021; Espinosa-Zúñiga, 2020).

Uno de los estudios relacionados al tema, donde se analiza varios tipos de fraude con tarjetas de crédito, dividiéndolos en dos categorías: fraude conductual y fraude por solicitud. El fraude conductual se refiere a actividades como el uso de tarjetas robadas o perdidas, la falsificación de tarjetas y el robo de información a través de correo, en cambio el fraude por solicitud incluye técnicas más sofisticadas como el robo de identidad para abrir nuevas cuentas de tarjeta de crédito a nombre de otras personas, la clonación de transacciones y el uso de dispositivos para copiar la información de las tarjetas, conocidos como skimmers. También, se mencionan prácticas como el fraude CNP (Card Not Present), donde los criminales utilizan datos de tarjetas en transacciones en línea sin necesidad de la presencia física de la tarjeta. Se aplican algoritmos como Random Forest, SVM, Árboles de Decisión, y Redes Neuronales, destacando que Random Forest se destaca por su precisión al detectar transacciones fraudulentas (Elhusseny et al., 2022).

Según Tiwari et al. (2021), en un estudio mencionan que la detección de fraude se ha convertido en una prioridad para las instituciones financieras, por lo que proponen varias metodologías basadas en el aprendizaje automático para identificar transacciones fraudulentas y prevenir pérdidas financieras. Entre los métodos que utilizan, son los siguientes:

- Hidden Markov Model (HMM): Utilizado para modelar el comportamiento del titular de la tarjeta, ofrece una precisión cercana al 80% y su rendimiento se degrada

cuando hay poca información disponible.

- Bayesian Belief Networks (BBN): Calcula la probabilidad de que una transacción sea fraudulenta mediante teoremas de probabilidad. Útiles cuando se cuenta con grandes cantidades de datos, pueden ser precisas y costosas de entrenar.
- Genetic Algorithm (GA): Basado en la selección natural, este algoritmo mejora la eficiencia del sistema al seleccionar las mejores características para predecir fraudes, sin embargo, depende de parámetros complejos.
- Redes Neuronales: Estas técnicas son capaces de detectar fraudes con alta precisión. Se analizan tres tipos de redes neuronales: Artificiales (ANN), Convolucionales (CNN) y Recurrentes (RNN), cada una con ventajas específicas.

En la investigación realizada por Domor & Jere (2024), revisan diferentes enfoques propuestos de investigaciones previas, destacando la aplicación de técnicas de aprendizaje automático como máquinas de vectores de soporte, regresión logística y algoritmos genéticos. También se menciona el uso de modelos híbridos que combinan varios algoritmos para mejorar la precisión en la detección de fraudes.

Uno de los modelos propuestos sigue una arquitectura que incluye las siguientes etapas:

- Preprocesador: Limpia los datos de entrenamiento y prueba, eliminando valores duplicados o irrelevantes.
- Extractor de características: Clasifica los datos para facilitar su análisis mediante medidas estadísticas como la media y la desviación estándar.
- Modelo de Machine Learning: Entrena el sistema utilizando diferentes algoritmos, seleccionando el más preciso.
- Clasificador: Clasifica transacciones fraudulentas y no fraudulentas con alta precisión.
- Análisis de rendimiento y visualización de resultados: Utiliza gráficos para analizar el rendimiento del modelo y su precisión.

El modelo final se destaca por su precisión en la predicción de fraudes, utilizando técnicas como SMOTE para evitar el sobreajuste de datos, utiliza un conjunto de datos públicos para entrenar y probar el modelo, logrando una precisión 99.5% con algoritmos de aprendizaje supervisado y un 99.8% con algoritmos de redes neuronales.

En el estudio realizado por Intan et al. (2021), tiene como objetivo aplicar algoritmos de aprendizaje automático para detectar fraudes basándose en un conjunto de datos históricos de una empresa de financiamiento al consumidor. Se aplican cinco métodos de aprendizaje automático: Regresión Logística, K-Nearest Neighbor (KNN), Árbol de Decisión, Random Forest, Máquina de Soporte Vectorial (SVM), utilizando datos de transacciones financiación al consumidor, que se dividen en conjuntos de datos de 80% entrenamiento y 20% prueba. Se analizaron 46.536 transacciones con 26 variables categóricas que se agruparon en dos categorías principales: el perfil del cliente y el perfil de financiamiento. De los resultados obtenidos, el algoritmo de Random Forest mostró la mayor precisión, con un accuracy de entrenamiento de 99.4999% y un accuracy de prueba de 74.5437%.

De acuerdo con Prateeksha et al. (2023), realizan una investigación que tiene como objetivo abordar el creciente problema del fraude financiero, especialmente el fraude con tarjetas de crédito, en el contexto del aumento de los sistemas de comercio y pagos electrónicos. El estudio propone un modelo de detección de fraude basado en técnicas de aprendizaje automático, utilizando clasificadores como el Árbol de Decisión (DT), Regresión Logística (LR) y Redes Neuronales Artificiales (ANN). Los autores destacan la importancia de seleccionar adecuadamente las características de los fraudes con tarjetas de crédito para mejorar la eficacia del modelo, entre las variables utilizadas se tiene: monto de la transacción, hora-fecha de la transacción, ubicación geográfica, tipo de transacción, historial del titular de la tarjeta y características del comerciante. Para validar el rendimiento del modelo de detección propuesto, se utiliza un conjunto de datos generado a partir de

transacciones de titulares de tarjetas de crédito europeos. Los resultados mostraron que el enfoque propuesto supera a los sistemas existentes en términos de precisión y eficacia en la detección de fraudes. Su mejor modelo fue con el algoritmo de Random Forest con un accuracy del 98.60%.

Según Ileberi et al. (2022), abordan el estudio del creciente problema del fraude con tarjetas de crédito en el contexto del comercio electrónico. Exponen que a medida que el uso de servicios en línea se ha expandido, también lo han hecho las actividades fraudulentas relacionadas con las transacciones con tarjetas de crédito. El estudio propone un motor de detección de fraude basado en aprendizaje automático (ML) que utiliza un algoritmo genético (GA) para la selección de características, indicando que este enfoque es crucial, ya que la elección adecuada de las características es fundamental para mejorar la efectividad de los modelos de detección de fraude. Los algoritmos de ML utilizados en el estudio incluyen Árboles de Decisión (DT), Bosques Aleatorios (RF), Regresión Logística (LR), Redes Neuronales Artificiales (ANN) y Naive Bayes (NB). El mejor modelo presentado en el estudio fue el RF, que logró una notable precisión de accuracy de 99.98% en la detección de fraude con tarjetas de crédito.

En la investigación realizada por (Gaikwad et al., 2023), presentan un sistema integral para la detección de fraudes que combina el aprendizaje automático y la tecnología blockchain. El estudio destaca el aumento significativo del fraude financiero, con pérdidas de aproximadamente $8.8 mil millones en 2022. Esto ha llevado a la necesidad de desarrollar sistemas más efectivos para prevenir y detectar fraudes. El sistema propuesto utiliza algoritmos de aprendizaje automático para identificar patrones y anomalías en las transacciones que podrían indicar actividades fraudulentas. Utilizan específicamente tres algoritmos: Regresión Logística, Árboles de Decisión y Bosques Aleatorios. Además, utilizan la tecnología blockchain para crear una aplicación bancaria descentralizada que asegura la

transparencia y la seguridad de las transacciones, registrando todas las transacciones en la blockchain, garantizando que sean legítimas y a prueba de manipulaciones. El mejor modelo fue el de Random Forest con un accuracy 99.99%.

En el estudio de Byrapu et al. (2024), exponen la creciente problemática del fraude en el comercio electrónico y propone soluciones innovadoras mediante el uso de técnicas de aprendizaje automático y análisis de big data. Destacan que el fraude en línea es un problema global que afecta tanto a consumidores como a empresas, y que la sofisticación de las técnicas de fraude está en constante evolución. Ellos critican la efectividad de los métodos tradicionales de detección de fraude, que a menudo dependen de reglas estáticas y no pueden adaptarse rápidamente a las nuevas tácticas de los defraudadores. La investigación propone un método automatizado de detección de fraude que utiliza aprendizaje supervisado. Este sistema se basa en la combinación de descubrimiento de características informadas por expertos, procesamiento de datos personalizado y selección de modelos, lo que permite una mayor precisión en las predicciones. Utilizan diversos algoritmos de machine learning, obteniendo como mejor modelo con Regresión Logística con un accuracy de 80%.

1.2. Problema de investigación

Pese a los progresos tecnológicos y las medidas de protección aplicadas por las entidades financieras, el fraude en las operaciones con tarjetas de crédito continúa representando un peligro considerable. Los métodos convencionales de identificación de fraudes frecuentemente resultan insuficientes para reconocer patrones de comportamiento cada vez más avanzados de los que cometen fraudes, lo que resulta en un elevado porcentaje de falsos positivos y en la imposibilidad de identificar nuevos tipos de fraude (Smith & Valverde, 2021).

El problema consiste en elaborar modelos predictivos más eficaces y exactos que sean capaces de ajustarse con rapidez a las modificaciones en las tácticas de fraude, reduciendo así las pérdidas económicas para las entidades y el efecto adverso en los clientes. En estas circunstancias, se presenta la necesidad de implementar y utilizar diversas técnicas de minería de datos con el fin de incrementar la exactitud en la predicción de fraudes en tarjetas de crédito, mejorando el equilibrio entre la identificación temprana de fraudes y la disminución de falsificaciones positivas.

Por lo cual, se plantea la siguiente pregunta de investigación:
¿Qué técnicas de minería de datos ofrecen un mejor desempeño en cuanto a exactitud, sensibilidad y disminución de falsos positivos en la predicción de fraudes en tarjetas de crédito para encontrar el mejor modelo que permite la optimización en la adaptación de patrones de fraude emergentes y cambiantes en el tiempo?

1.3. Objetivos

1.3.1. Objetivo general

- Desarrollar modelos predictivos de detección de fraudes mediante técnicas de minería de datos para la identificación de transacciones fraudulentas en tarjetas de crédito con alta precisión.

1.3.2. Objetivos específicos

- Analizar un conjunto de datos históricos de transacciones con tarjetas de crédito para identificación de características que permitan distinguir transacciones legítimas de transacciones fraudulentas.
- Diseñar modelos de clasificación con técnicas de minería de datos como Machine Learning y Deep Learning para la detección de fraudes con tarjetas de crédito.
- Comparar el desempeño de los modelos obtenidos en términos de las métricas de

accuracy, precisión, recall, F1-Score, matriz de confusión, falsos positivos y falsos negativos utilizando métricas como la AUC-ROC para selección del mejor modelo.

- Desarrollar una API como medio de comunicación entre la interfaz visual y el mejor modelo predictivo para realizar predicciones con nuevos datos.

1.4. Justificación e importancia del trabajo de investigación

El fraude con tarjetas de crédito no solo afecta la rentabilidad de las instituciones financieras, sino que también hace que los clientes pierdan la confianza en los sistemas de pago digitales. Por lo tanto, la implementación de modelos inteligentes de detección de fraudes es esencial para reducir estas pérdidas y, al mismo tiempo, mejorar la experiencia del cliente al reducir los falsos positivos, que son transacciones legítimas que se identifican incorrectamente como fraudulentas.

Los métodos de aprendizaje automático y profundo ofrecen una ventaja significativa sobre los métodos tradicionales porque estos modelos aprenden de grandes volúmenes de datos, lo que permite identificar transacciones fraudulentas que no siguen patrones predeterminados. Además, se pueden encontrar relaciones complejas y patrones anómalos que pasarían desapercibidos con los métodos tradicionales utilizando modelos como Random Forest, Gradient Boosting, Deep Neural Networks, entre otros.

El proyecto no solo ayudará en la detección de fraudes, sino que también ampliará el campo de la investigación en ciencia de datos aplicada a la seguridad financiera, ofreciendo un marco sólido que se puede aplicar a otros sectores financieros y comerciales como bancos y cooperativas. Además, permitirá desarrollar conocimientos sobre el uso de modelos matemáticos en el campo del aprendizaje automático y profundo.

Algunas de las ventajas del proyecto se pueden mencionar:

- Mayor precisión: Incremento de la capacidad para detectar fraudes con menos falsos

positivos.

- Escalabilidad: Los modelos de aprendizaje profundo en relación a los modelos de aprendizaje supervisado pueden manejar grandes cantidades de datos y continuar aprendiendo a medida que el sistema recibe nuevas transacciones.
- Adaptabilidad: Mejor adaptación ante la búsqueda de nuevos patrones de fraude.

Por lo tanto, la importancia del proyecto es el desarrollo de modelos de detección de fraudes basado en técnicas de minería de datos, tanto de aprendizaje automático como de aprendizaje profundo, para encontrar el mejor modelo con el propósito de aumentar la precisión y capacidad de adaptación como apoyo a futuro de los sistemas de fraudes de tarjetas de créditos de instituciones financieras.

1.5. Alcance y limitaciones del proyecto

El proyecto de titulación contempla lo siguiente como entregables:

- Desarrollo de modelos matemáticos entrenados con datos históricos mediante algoritmos de Machine Learning como Regresión Logística, Random Forest, Gradient Boosting, entre otros y Deep Learning con Redes Neuronales para la detección de fraudes de tarjeta de crédito.
- Análisis comparativo del desempeño de los modelos entrenados para medir la efectividad en la detección de fraudes.
- Código en lenguaje Python sobre el proceso de diseño y evaluación de los modelos de Machine Learning y Deep Learning realizados en notebooks de Google Colab subidos en un repositorio de GitHub.
- Interfaz web básica visual y una API para comunicación con el mejor modelo entrenado para realizar predicciones en tiempo real.

Con relación a las limitaciones de lo que no se considera se menciona:

- El proyecto no aborda la implementación de los modelos en sistemas productivos reales de entidades financieras.
- No se proporciona una interfaz web ni el API tanto a empresas o instituciones financieras.
- Tampoco se consideran temas relacionados con la ejecución de políticas de seguridad interna de instituciones bancarias, por lo tanto, no se contempla la recuperación financiera o la compensación de fondos en caso de transacciones fraudulentas detectadas.

1.6. Marco Teórico

1.6.1. Minería de datos

1.6.1.1. Definición

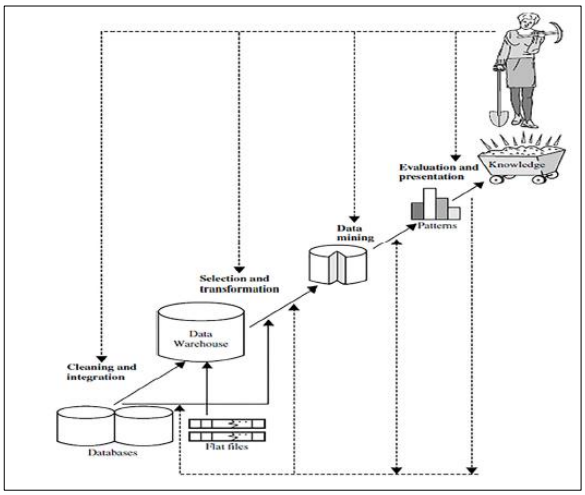
La minería de datos es un proceso fundamental dentro del descubrimiento de conocimiento en bases de datos (KDD) que se enfoca en encontrar patrones interesantes y estructuras en grandes volúmenes de datos. Es una disciplina interdisciplinaria que surgió de la necesidad de analizar grandes cantidades de datos más allá de los enfoques estadísticos tradicionales (Shu y Ye, 2023).

1.6.1.2. Proceso KDD

El proceso de minería de datos generalmente implica varias etapas, incluyendo la preparación de los datos, la aplicación de técnicas y algoritmos de minería y la evaluación e interpretación de los resultados. Además, utiliza métodos tanto deductivos como inductivos y puede emplear enfoques supervisados y no supervisados para descubrir conocimientos ocultos en los datos (Shu y Ye, 2023).

Figura 1.

Proceso KDD



Nota: La figura representa las fases del proceso KDD. Tomado de (Murillo et al., 2018)

En la Figura 1, se observa las etapas del proceso KDD, las mismas que se describen a continuación de acuerdo con (Murillo et al., 2018):

1. Recopilación de datos: Se identifican y seleccionan los datos que serán pertinentes al análisis. Esto puede incluir recopilar datos de varias fuentes, asegurándose de que la información seleccionada sea relevante para el problema que se busca resolver.
2. Preprocesamiento de datos: Los datos seleccionados se limpian y se transforman para preparar los datos para el análisis, esto incluye eliminar errores y duplicados, manejo de valores nulos y valores faltantes. Además, se realizan transformaciones necesarias que pueden modificar o reducir los datos, esto puede incluir agregar, normalizar o crear variables nuevas que sean más útiles para la minería de datos.
3. Minería de datos: Se utilizan algoritmos particulares para extraer patrones y obtener modelos de datos. Dependiendo del tipo de patrón que se busca, se utilizan diferentes técnicas de minería de datos, como la clasificación, la regresión, el agrupamiento y la asociación.
4. Interpretación y evaluación del modelo: Una vez obtenido los patrones deben

evaluarse para determinar su validez y utilidad. Esto implica verificar si los patrones son significativos y satisfacen los objetivos del análisis. Para la evaluación de la calidad de los patrones descubiertos, se pueden utilizar métricas específicas.

1.6.1.3. Técnicas de minería de datos

Las técnicas de minería de datos abarcan una amplia gama de métodos para extraer conocimiento valioso de grandes conjuntos de datos. Entre las técnicas se pueden mencionar de acuerdo con (Shu y Ye, 2023):

- Regresión: Predecir valores numéricos.
- Clasificación: Asignar elementos a categorías predefinidas.
- Agrupamiento: Identificar grupos de elementos similares mediante clusters.
- Análisis de asociación: Descubrir relaciones entre elementos.
- Redes neuronales: Simular el funcionamiento del cerebro humano para aprender patrones complejos.

1.6.1.4. Aplicaciones

Las técnicas de minería de datos tienen diversas aplicaciones en varios campos y se emplean varios métodos para el descubrimiento de conocimientos y el reconocimiento de patrones: Marketing, Finanzas, Salud, Ciencias, Educación, entre otras (Shu & Ye, 2023).

1.6.2. Machine Learning

1.6.2.1. Definición

El aprendizaje automático (o machine learning en inglés) es una rama de la inteligencia artificial que se centra en el desarrollo de algoritmos y modelos que permiten a las computadoras aprender de los datos y hacer predicciones o tomar decisiones sin ser programadas explícitamente para realizar una tarea específica (Majumdar, 2023).

1.6.2.2. Tipos de aprendizaje

El aprendizaje automático abarca varios tipos de enfoques de aprendizaje, siendo las categorías principales el aprendizaje supervisado, el aprendizaje no supervisado y el aprendizaje de refuerzo.

A continuación, se describe cada uno de acuerdo (Joshi, 2023):

- Aprendizaje Supervisado: En este enfoque, el modelo se entrena con un conjunto de datos etiquetado, donde cada entrada tiene una salida correspondiente. El objetivo es aprender a predecir la salida para nuevas entradas.
- Aprendizaje No Supervisado: El modelo trabaja con datos no etiquetados y busca patrones o agrupaciones en los datos sin una guía explícita sobre lo que se debe aprender.
- Aprendizaje por Refuerzo: Este tipo de aprendizaje implica que un agente aprende a tomar decisiones mediante la interacción con un entorno, recibiendo recompensas o penalizaciones en función de sus acciones.

1.6.2.3. Algoritmos de Aprendizaje Supervisado

Entre los algoritmos básicos machine learning de aprendizaje supervisado se pueden mencionar a Regresión Logística, SVM, KNN, Naive Bayes, Árbol de decisión. También se tiene algoritmos ensemble como Random Forest y Gradient Boosting.

Regresión Logística

Es un algoritmo utilizado para problemas de clasificación binaria. La regresión logística predice la probabilidad de que una instancia pertenezca a una clase específica. Utiliza la función logística (o sigmoide) para transformar la salida de una combinación lineal de las características en un valor entre 0 y 1, lo que permite interpretar este valor como una probabilidad (Majumdar, 2023).

SVM (Soporte de Máquina Vectorial)

Es un algoritmo utilizado para clasificación y regresión. Su objetivo principal es encontrar el hiperplano que mejor separa las clases en un espacio de características. SVM busca maximizar el margen entre las instancias de diferentes clases, lo que significa que intenta encontrar el hiperplano que tiene la mayor distancia a los puntos de datos más cercanos de cada clase, conocidos como vectores de soporte (Majumdar, 2023).

KNN (K-Nearest Neighbors)

Es un algoritmo utilizado para clasificación y regresión. Funciona identificando los 'k' vecinos más cercanos a un punto de datos nuevo y tomando decisiones basadas en la mayoría de las clases (en clasificación) o el promedio de los valores (en regresión) de esos vecinos. KNN es un método basado en instancias, lo que significa que no construye un modelo explícito, sino que almacena todos los datos de entrenamiento y realiza cálculos en tiempo de consulta (Majumdar, 2023).

Naive Bayes

Es un conjunto de algoritmos de clasificación basados en el teorema de Bayes, que asume que las características son independientes entre sí, dado el valor de la clase. Este enfoque es "naive" (ingenuo) porque simplifica la realidad al suponer que la presencia de una característica en una clase no está relacionada con la presencia de cualquier otra característica. Naive Bayes es especialmente útil para problemas de clasificación de texto, como el filtrado de spam y la clasificación de documentos (Majumdar, 2023).

Árbol de Decisión

Es un algoritmo utilizado para clasificación y regresión. Funciona dividiendo el conjunto de datos en subconjuntos más pequeños basados en características específicas, creando una estructura en forma de árbol donde cada nodo interno representa una prueba

en una característica, cada rama representa el resultado de la prueba y cada nodo hoja representa una clase o un valor de predicción.

Los árboles de decisión son fáciles de interpretar y visualizar, lo que los hace populares en aplicaciones donde la interpretabilidad es importante (Majumdar, 2023).

Ensemble

Se refiere a un enfoque en el aprendizaje automático que combina múltiples modelos para mejorar la precisión y la robustez de las predicciones. La idea detrás de los modelos de ensemble es que, al combinar las predicciones de varios modelos, se puede reducir el riesgo de sobreajuste y mejorar la generalización del modelo final. Los modelos de ensemble pueden ser particularmente efectivos porque diferentes modelos pueden capturar diferentes patrones en los datos (Shah et al., 2023).

Tipos de modelos ensemble

Entre los principales modelos que son ensemble se tiene a los Bagging, Boosting y Stacking, se describen cada uno:

- Bagging: Se crean múltiples subconjuntos de datos a partir del conjunto de datos original mediante muestreo con reemplazo. Se aplica el mismo algoritmo a cada subconjunto y luego se agregan los resultados. Un ejemplo popular de bagging es el algoritmo Random Forest.
- Boosting: Se construye modelos de manera secuencial, donde cada nuevo modelo se entrena para corregir los errores del modelo anterior. Los modelos se combinan de manera que los errores de los modelos anteriores se ponderan más en las predicciones finales. Ejemplos de algoritmos de boosting incluyen AdaBoost y Gradient Boosting.
- Stacking: Se entrenan varios modelos base y luego se utiliza un modelo de nivel superior (meta-modelo) para combinar las predicciones de los modelos base. Este

enfoque puede utilizar diferentes tipos de modelos en la capa base y en la capa de meta-modelo.

Ventajas de los modelos ensemble:

Algunas de las ventajas se pueden mencionar (Li et al., 2018; Mabdeh et al., 2021):

- Mejora de la precisión: Al combinar múltiples modelos, se puede lograr una mayor precisión en las predicciones.
- Robustez: Los modelos de ensemble son menos sensibles a las variaciones en los datos y pueden manejar mejor el ruido.
- Reducción del sobreajuste: Al promediar las predicciones de varios modelos, se puede reducir el riesgo de sobreajuste que puede ocurrir con un solo modelo complejo.

Random Forest

Es un algoritmo que pertenece a la categoría de modelos de ensemble, específicamente al enfoque de bagging. Fue desarrollado por Leo Breiman y se utiliza principalmente para problemas de clasificación y regresión. El algoritmo crea un "bosque" de árboles de decisión, donde cada árbol se entrena utilizando un subconjunto aleatorio de los datos y un subconjunto aleatorio de las características. La predicción final se obtiene mediante la agregación de las predicciones de todos los árboles en el bosque, lo que mejora la precisión y reduce el riesgo de sobreajuste (Majumdar, 2023).

Gradient Boosting

Es un algoritmo que se utiliza para problemas de clasificación y regresión. Se basa en la idea de construir modelos de manera secuencial, donde cada nuevo modelo se entrena para corregir los errores del modelo anterior. A diferencia de los modelos de ensemble como Random Forest, que combinan múltiples modelos de manera paralela, el

Gradient Boosting construye modelos de forma aditiva, lo que significa que cada nuevo modelo se ajusta a los residuos (errores) de los modelos anteriores (Majumdar, 2023).

En la Tabla 1, se describen los principales hiperparámetros que se pueden considerar al momento de realizar el entrenamiento del modelo con los diferentes algoritmos ensemble de aprendizaje supervisado. En la Tabla 2, se describen los principales hiperparámetros que se consideran al momento de realizar el entrenamiento del modelo con los algoritmos de aprendizaje supervisado.

Tabla 1.

Hiperparámetros de algoritmos ensemble de aprendizaje supervisado

Algoritmo	**Hiperparámetros**
Random forest	*n_estimators:* define el número de árboles en el bosque *criterion*: determina la función de calidad utilizada para medir la división en los nodos *max_depth*: controla la profundidad máxima del árbol *min_samples_split*: especifica el número mínimo de muestras requeridas para dividir un nodo *min_samples_leaf*: establece el número mínimo de muestras que debe haber en un nodo hoja *max_features*: controla el número de características a considerar al buscar la mejor división *bootstrap*: indica si se debe usar el muestreo con reemplazo al construir árboles
Gradient boosting	*n_estimators:* define el número de árboles *learning_rate*: controla la contribución de cada árbol al modelo final *max_depth*: controla la profundidad máxima del árbol *min_samples_split*: especifica el número mínimo de muestras requeridas para dividir un nodo *min_samples_leaf*: establece el número mínimo de muestras que debe haber en un nodo hoja *subsample*: controla la fracción de muestras que se utilizan para entrenar cada árbol *loss*: define la función de pérdida que se utilizará para optimizar el modelo *alpha*: se utiliza en la regresión cuantílica y controla el nivel de cuantiles que se desea predecir.

Nota: La tabla describe una explicación de los hiperparámetros de los algoritmos ensemble de aprendizaje supervisado. Obtenido de (Majumdar, 2023)

Tabla 2.

Hiperparámetros de algoritmos de aprendizaje supervisado

Algoritmo	**Hiperparámetros**
Regresión logística	*C*: controla la regularización *penalty*: especifica el tipo de regularización *solver*: determina el algoritmo para optimizar la función de costo *max_iter*: define el número máximo de iteraciones para encontrar los coeficientes óptimos *tol*: establece el criterio de tolerancia para la convergencia
SVM	*C*: controla el margen de tolerancia para los errores de clasificación. *kernel*: especkfica la función del npucleo que se usa para transformar los datos en un espacio de mayor dimensión *degree*: se usa solo si el núcleo es polinómico *gamma*: define el alcance de la influencia de un solo ejemplo de entrenamiento *max_iter*: define el número máximo de iteraciones para encontrar los coeficientes óptimos *tol*: establece el criterio de tolerancia para la convergencia
KNN	n_neighbors: número de vecinos más cercanos weights: determina como se pondera los vecinos en la predicción algorithm: especifica el algoritmo para calcular los vecinos más cercanos metric: define la métrica de distancia para calcular la proximidad de los puntos leaf_size: define el tamaño de las hojas en el árbol
Naive Bayes	*alpha*: se usa en el suavizado de Laplace para manejar el problema de las probabilidades cero *fit_prior*: indica si se debe aprender la probabilidad a priori de las clases a partir de los datos *class_prior*: permite especificar las probabilidades a priori de las clases *var_smoothing*: se utiliza en el caso de Naive Bayes Gaussiano y controla la suavización de la varianza
Árbol de decisión	*criterion*: determina la función de calidad utilizada para medir la división en los nodos *max_depth*: controla la profundidad máxima del árbol *min_samples_split*: especifica el número mínimo de muestras requeridas para dividir un nodo *min_samples_leaf*: establece el número mínimo de muestras que debe haber en un nodo hoja *max_features*: controla el número de características a considerar al buscar la mejor división *bootstrap*: indica si se debe usar el muestreo con reemplazo al construir árboles

Nota: La tabla describe una explicación de los hiperparámetros de los algoritmos de aprendizaje supervisado. Obtenido de (Majumdar, 2023)

1.6.3. Deep Learning

1.6.3.1. Perceptrón

El perceptrón es considerado la forma más elemental de una red neuronal. Fue desarrollado en 1958 por Frank Rosenblatt para clasificar patrones que podían separarse de forma lineal. Se define como un modelo de clasificación binaria que determina la clase a la que pertenece una muestra, basándose en una combinación lineal de sus entradas (Haykin, 2009).

Funcionamiento

El perceptrón consta de una sola neurona y realiza la suma ponderada de las variables de entrada, ajustadas por pesos sinápticos y adicionando un sesgo. La función de activación transforma esta suma en una salida binaria, representando la clase a la que pertenece la muestra (Haykin, 2009).

Es posible obtener la salida y mediante las siguientes ecuaciones (1) y (2):

$$a = \sum_{i=1}^{n} \omega_i x_i + b \qquad (1)$$

$$y = f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases} \qquad (2)$$

donde:

X=[x_1,x_2,…..,x_n] corresponde a las variables de entrada.

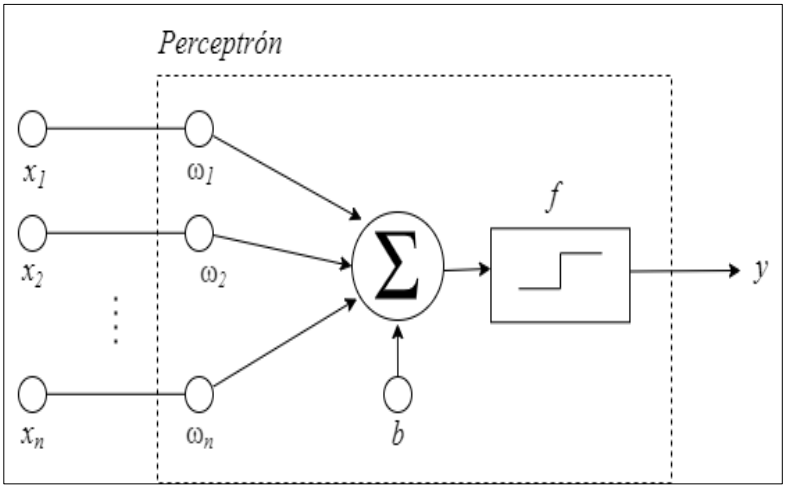
ω=[ω_1,ω_2,…..,ω_n] es el vector de pesos que se asigna a cada entrada.

b es el bias (o sesgo) que permite ajustar la salida (Haykin, 2009).

f es la función de activación que está dada por una función escalonada (Bishop, 2006).

Figura 2.

Modelo de red perceptrón



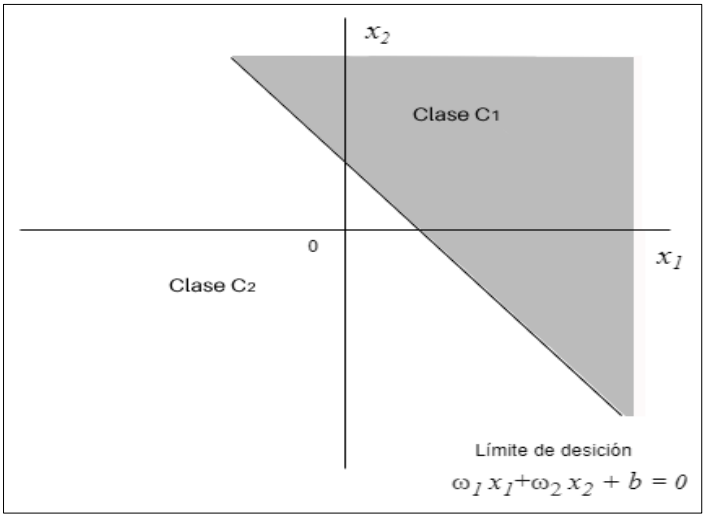
Nota: La figura muestra los componentes de la red neuronal artificial del Perceptrón. Basado en conceptos de (Haykin, 2009). Obtenido: Elaboración propia

El perceptrón tiene como objetivo clasificar correctamente una muestra o conjunto de entradas [x_1,x_2,…..,x_n] en una de dos clases c1 o c2, ver Figura 2. La regla de decisión que se considera al momento de clasificar una muestra o conjunto de entradas [x_1,x_2,…..,x_n] es asignar la clase c1 si la salida del perceptrón y es +1 y la clase c2 si es -1 (Haykin, 2009).

Para entender mejor el funcionamiento de este clasificador, podemos crear un gráfico como se observa la Figura 3 que muestra las zonas donde clasifica los datos. Si consideramos muestras con dos variables de entrada x1 y x2, donde una línea recta determina el límite de decisión, los puntos (x1,x2) situados por encima de la línea se asignan a la clase c1, mientras que los que se encuentran por debajo se corresponderán a la clase c2. Al variar el valor de b se modifica la intersección de la recta con el eje vertical de las ordenadas desplazando así el límite de decisión con respecto al origen del plano. (Haykin, 2009).

Figura 3.

Hiperplano del clasificador del Perceptrón



Nota: La figura describe el hiperplano (en este ejemplo, una línea recta) como frontera de decisión para un problema de clasificación de patrones de dos dimensiones y dos clases. (Traducido al español y modificado). Adaptado de (Haykin, 2009).

Algoritmo de entrenamiento

Según Goodfellow et al. (2016), el entrenamiento de un perceptrón se realiza utilizando un algoritmo supervisado, se trata de un proceso iterativo donde se busca ajustar los pesos a fin de reducir el error entre la salida esperada y la salida real.

Este proceso consiste en lo siguientes pasos:

1. Inicialización de pesos: Se realiza una inicialización aleatoria de los pesos antes del entrenamiento, la cual puede ser aleatoria o a cero. La velocidad de convergencia del algoritmo está influida por la selección de la inicialización.
2. Propagación hacia adelante: Durante la propagación hacia adelante, cada entrada se multiplica por su peso correspondiente, y se obtiene la suma ponderada, luego se calcula la salida del perceptrón a través de la función de activación.
3. Cálculo del error: El error se determina comparando la salida real y con la salida deseada d. Podemos expresar el error mediante la siguiente ecuación (3).

$$Error = d - y \qquad (3)$$

4. Actualización de pesos: Se realiza una actualización de los pesos mediante la regla de aprendizaje del perceptrón, buscando reducir el error a la salida. La ecuación (4) para la actualización para el peso es ω_i está dada por:

$$\omega_i \leftarrow \omega_i + \eta(d - y)x_i \qquad (4)$$

donde:

- η es la tasa de aprendizaje, un parámetro que controla la magnitud de la actualización.
- d es la salida deseada.
- y es la salida obtenida.
- x_i es el valor de la entrada i-ésima que se está utilizando para calcular la salida.

Se realizan múltiples iteraciones o épocas hasta que se alcance un nivel de error aceptable o se agoten los ciclos de entrenamiento.

Convergencia del Perceptrón

Un punto importante para considerar con respecto al perceptrón es que su método de garantiza la convergencia cuando los datos cumplen con la condición de ser separables linealmente. Esto implica que en el hiperplano se formará una recta que permitirá separar de forma muy clara dos clases, y con ello clasificar las muestras. Por otra parte, si los datos no son separables linealmente, el perceptrón no podrá encontrar una solución adecuada y continuará ajustando los pesos sin alcanzar un resultado óptimo (Haykin, 2009).

Limitaciones

A pesar ser un modelo relativamente simple, el Perceptrón tiene importantes limitaciones. Su incapacidad para resolver problemas no lineales se evidencia en el famoso problema XOR, donde no existe una línea recta que pueda separar las clases (Haykin, 2009).

Esta limitación motivó el desarrollo de modelos más sofisticados que permitan solucionar problemas que presenten no linealidad, un ejemplo de aquello es el Perceptrón multicapa.

1.6.3.2. Perceptrón Multicapa (MLP)

Es una arquitectura de red neuronal estructurada en varias capas de neuronas o nodos interconectados, donde cada neurona en una capa está conectada a todas las neuronas de la capa siguiente. Esto permite modelar relaciones complejas y no lineales en los datos (Haykin, 2009; Taye, 2023).

Los perceptrones multicapa (MLP) han ganado enorme popularidad en el contexto del aprendizaje profundo y son esenciales para una gama de aplicaciones contemporáneas, como el reconocimiento de patrones y clasificación.

Arquitectura del MLP

En la arquitectura de un MLP se identifican de tres tipos de capas como se observa en la Figura 4, estas capas son:

Capa de Entrada

La capa de entrada está formada por un conjunto de neuronas que representan las características del set de datos. Cada neurona de esta capa se asocia a una característica específica, que se transmite a la siguiente capa (Haykin, 2009).

Capas Ocultas

El rol de las capas ocultas es realizar el procesamiento de datos y por lo general contienen múltiples neuronas. Cada neurona en estas capas aplica una función de activación a una combinación lineal de sus entradas, lo que permite al MLP aprender

representaciones jerárquicas y características complejas de los datos (Goodfellow et al., 2016; Taye, 2023).

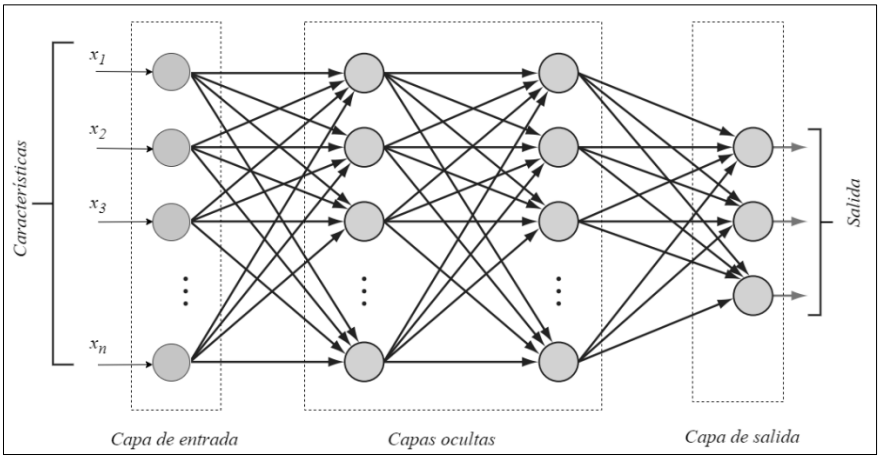
Las neuronas de las capas ocultas trabajan en conjunto para capturar patrones en los datos que son difíciles de detectar mediante un Perceptrón.

Capa de Salida

La capa de salida es la encarga de generar el resultado final en la red. El número de neuronas en esta capa depende de la tarea específica que se esté realizando, pudiendo tener una o varias neuronas. Por ejemplo, en problemas de clasificación binaria, se emplea una sola neurona que produce una salida que representa la probabilidad de que la entrada pertenezca a la clase positiva. En contraste, en problemas de clasificación multiclase, generalmente se asigna una neurona a cada clase. Además, se utilizan funciones de activación como la sigmoide o softmax, que transforman las salidas en probabilidades que indican la pertenencia a cada clase (Bishop, 2006; Haykin, 2009).

Figura 4.

Arquitectura de la red MLP



Nota: La figura detalla los componentes de la arquitectura de un perceptrón multicapa con dos capas ocultas. (Traducido al español y modificado). Adaptado de (Haykin, 2009).

Proceso de Entrenamiento

El entrenamiento de un MLP tiene un enfoque supervisado, es decir se realiza utilizando muestras etiquetadas. Se busca minimizar la diferencia entre las predicciones que genera el modelo y las etiquetas verdaderas o reales (Goodfellow et al., 2016).

Podemos describir al proceso de entrenamiento con las siguientes etapas:

Inicialización de pesos

Al comenzar, los pesos de las conexiones entre neuronas son inicializados aleatoriamente. Esta inicialización es crucial, ya que puede afectar la convergencia del modelo (Bishop, 2006).

Propagación hacia adelante (Forward Propagation)

Consiste en el procesamiento de las entradas a través de las capas de la red. A la suma ponderada de cada neurona se le aplica una función de activación, de esta forma se obtiene una salida que se transmite a la capa siguiente. Este proceso continúa hasta que alcanzar la capa de salida, generando las predicciones del modelo (Haykin, 2009).

Cálculo del error

El error se determina al contrastar las predicciones producidas por la red MLP con las etiquetas correctas del conjunto de entrenamiento. Esta diferencia se cuantifica mediante una función de costo o pérdida, que varía dependiendo del problema; por ejemplo, es común utilizar el error cuadrático medio para tareas de regresión y la entropía cruzada en problemas de clasificación (Goodfellow et al., 2016).

Retropropagación (Backpropagation)

Rumelhart et al. (1986), explican que la retropropagación es un fundamental en el entrenamiento del perceptrón multicapa. Se calcula el gradiente del error respecto a cada

peso en la red mediante la regla de la cadena, el error se propaga hacia atrás desde la capa de salida a las capas ocultas, de esta forma se ajustan los pesos de minimizando el error.

Actualización de pesos

Basándose en los gradientes calculados durante la retropropagación, los pesos se actualizan utilizando un algoritmo de optimización, como el descenso de gradiente. El aprendizaje se ajusta mediante una tasa de aprendizaje, que determina el tamaño del paso en la dirección del gradiente (Bishop, 2006).

Optimizadores

Los optimizadores son algoritmos que ajustan los pesos del modelo para minimizar la función de pérdida durante el proceso de entrenamiento. La elección del optimizador puede tener un impacto considerable en la velocidad de convergencia y en el rendimiento del modelo (Kingma y Ba, 2014).

Existen varios optimizadores utilizados en MLP, entre ellos:

Descenso de Gradiente Estocástico (SGD).

Este método es simple y consiste en el ajuste de los pesos del modelo tomando una sola muestra del entrenamiento en cada iteración. Aunque es fácil de implementar, el SGD puede ser ineficiente y requerir muchas iteraciones para lograr converger, por otra parte, es sensible a la tasa de aprendizaje elegida (Bottou, 2010).

Adam (Adaptive Moment Estimation)

El optimizador Adam, que significa Estimación de Momentos Adaptativa, es uno de los algoritmos más efectivos y muy utilizados en para entrenar redes neuronales, integra los beneficios del descenso de gradiente estocástico (SGD), el método del momento y el escalado adaptativo. Este optimizador tiene la capacidad de ajustar automáticamente la

tasa de aprendizaje para cada parámetro del modelo, lo que puede facilitar la convergencia y mejorar el rendimiento global de las redes neuronales (Kingma y Ba, 2014).

Parámetros e Hiperparámetros

Parámetros

Los parámetros del MLP incluyen:

1. Pesos

Son valores que determina la importancia de las variables de entrada en la salida de una neurona. Las conexiones entre neuronas tienen asignado un peso que se ajusta durante el entrenamiento (Bishop, 2006).

2. Biases

Es un valor asociado a cada neurona que permite ajustar la salida de manera independiente de las entradas, lo que mejora la capacidad de modelado del MLP. Esto permite que las neuronas se activen incluso cuando todas las entradas son cero (Goodfellow et al., 2016).

Hiperparámetros

Son ajustes o configuraciones en la arquitectura del modelo que son fundamentales para su rendimiento, y repercuten en la capacidad del modelo para aprender y generalizar a partir de los datos (Bengio et al., 2013). Entre los hiperparámetros más importantes se encuentran:

1. Epochs.

Indican el número de veces que se realiza un recorrido completo sobre el set de datos durante el entrenamiento. Un número inapropiado de épocas puede resultar en sobreajuste (overfitting) o subajuste (underfitting), lo que impacta negativamente en la capacidad de generalización del modelo (Goodfellow et al., 2016).

2. Tamaño del lote (Batch Size)

Indica el número de muestras que se utilizan en cada actualización de los pesos. Un tamaño de lote reducido puede generar estimaciones del gradiente más imprecisas, lo que facilita escapar de mínimos locales, aunque esto puede prolongar el tiempo de entrenamiento. En contraste, un tamaño de lote mayor ofrece estimaciones más consistentes, pero puede necesitar más memoria y resultar en una convergencia más lenta (Kingma y Ba, 2014). Durante el entrenamiento el tamaño del lote es un factor importante para la eficiencia y efectividad de este.

3. Tasa de aprendizaje

Regula la rapidez con la que se modifican los pesos. Un valor demasiado alto puede causar inestabilidad, mientras que uno demasiado bajo puede dar lugar a un entrenamiento lento y poco eficiente (Bengio et al., 2013).

4. Función de activación

La selección de la función de activación es un factor determinante en el rendimiento de una red neuronal. Las funciones ReLU, sigmoide y tanh introducen no linealidades en el modelo. Cada una tiene sus propias ventajas y desventajas en términos de convergencia y capacidad de modelado (Goodfellow et al., 2016).

5. Función de pérdida

Son esenciales en el entrenamiento de modelos de machine learnig, incluidos los perceptrones multicapa (MLP). Tienen como objetivo medir la diferencia entre las predicciones del modelo y los valores reales. La función de pérdida proporciona una señal de retroalimentación que el modelo utiliza para ajustar sus parámetros (Goodfellow et al., 2016).

Existen varias funciones de pérdida comúnmente utilizadas en MLP:

- Pérdida Cuadrática Media (MSE): Se define como el promedio de los cuadrados de

las diferencias entre los valores predichos y los reales, como se observa en ecuación (5):

$$MSE = \frac{1}{n}\sum_{i}^{n}(y_i - \hat{y}_i) \qquad (5)$$

donde y_i representa el valor real y yˆ_i la predicción del modelo (Hastie et al., 2009).

- Entropía Cruzada: Es una función de pérdida común en problemas de clasificación. Mide la distancia entre la distribución de probabilidad real y la predicha por el modelo. Para problemas de clasificación binaria, ver ecuación (6) se define como:

$$L = -\frac{1}{n}\sum_{i=1}^{n}[y_i\, log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \qquad (6)$$

En problemas de clasificación multiclase, ver ecuación (7) se extiende a:

$$L = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} y_{ic}\, log(\hat{y}_{ic}) \qquad (7)$$

donde C es el número de clases (Bishop, 2006).

La entropía cruzada es particularmente útil porque penaliza más severamente las predicciones incorrectas, facilitando un aprendizaje más eficaz en clasificación.

1.6.4. Optimización

RandomSearchCV

Es una técnica de optimización comúnmente utilizada en el ajuste de hiperparámetros, independiente de otras opciones los hiperparámetros se escogen al azar. Random Search selecciona aleatoriamente subconjuntos de combinaciones, lo que puede reducir significativamente el tiempo de cálculo. A pesar de la aleatoriedad; este enfoque es especialmente útil cuando el espacio de búsqueda de hiperparámetros es muy grande o no está claro qué valores son más prometedores (Shekhar et al., 2022).

RandomSearchCV es simple de implementar y es muy conveniente para el aprendizaje de funciones gradient-free. Como limitaciones se tiene que el método necesita mucho tiempo porque evaluar la función resulta costoso. No tiene la capacidad de selección de modelos, ya que se usa solo con un solo modelo. Al realizar comparaciones con la optimización bayesiana se encuentra que este método no aprovecha el conocimiento de un espacio de búsqueda de buen rendimiento (Shekhar et al., 2022).

GridSearchCV

Es una técnica utilizada para la optimización de hiperparámetros en modelos de machine learning. Esta metodología busca de manera exhaustiva en un espacio definido de hiperparámetros para identificar la combinación óptima que maximice el desempeño del modelo. A través de la validación cruzada, evalúa el rendimiento del modelo en diferentes subconjuntos de datos, asegurando que la selección de hiperparámetros no esté sesgada por la partición de entrenamiento. En comparación con técnicas más modernas como RandomSearchCV, GridSearchCV puede ser menos eficiente cuando el espacio de búsqueda es muy amplio o cuando los modelos son costosos de entrenar (Shekhar et al., 2022).

Bayesiano

La optimización bayesiana es una técnica que se emplea para ajustar hiperparámetros en modelos de aprendizaje automático cuando se desea optimizar una función que es compleja y costosa de evaluar ya sea en términos de tiempo, dinero u otros recursos. Es particularmente útil en situaciones donde el proceso a optimizar se comporta como una caja negra; en lugar de probar todas las combinaciones posibles de hiperparámetros como en GridSearchCV o usar un enfoque más simple como RandomSearchCV, la optimización bayesiana construye un modelo probabilístico del problema y utiliza ese modelo para seleccionar de manera inteligente las combinaciones de

hiperparámetros que se prueban. El uso de la optimización bayesiana puede agilizar este proceso de búsqueda y ayudarnos a encontrar el óptimo de la función de la manera más rápida posible (Nguyen, 2023).

1.6.5. Métricas de rendimiento

Para la explicación de las diferentes métricas se deben considerar las siguientes variables:

- TP = Verdaderos Positivos,
- TN = Verdaderos Negativos
- FP = Falsos Positivos
- FN = Falsos Negativos

Accuracy

El accuracy se calcula dividiendo la cantidad de ejemplos correctamente clasificados entre el total de ejemplos clasificados como se observa en ecuación (8):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (8)$$

Es una métrica valiosa cuando los errores al predecir todas las clases tienen la misma relevancia. Sin embargo, por ejemplo, en el caso de la clasificación de correo como spam o no spam, esto no siempre se aplica. Es posible que se toleren menos los falsos positivos que los falsos negativos. Un falso positivo en la detección de spam ocurre cuando un correo electrónico de un amigo se clasifica erróneamente como spam y, por lo tanto, no se muestra. En contraste, un falso negativo es menos problemático: si el modelo no detecta un pequeño porcentaje de mensajes spam, no resulta tan grave (Burkov, 2020).

Precisión y Recall

Las métricas más comúnmente utilizadas para evaluar un modelo son la precisión y el recall.

La precisión se define como la proporción de predicciones positivas correctas en relación con el total de predicciones positivas realizadas como se observa en la ecuación (9):

$$Precision = \frac{TP}{TP + FP} \quad (9)$$

El recall se calcula como la proporción de predicciones positivas correctas en comparación con el total de ejemplos positivos en el conjunto de prueba como se observa en la ecuación (10):

$$Recall = \frac{TP}{TP + FN} \quad (10)$$

Para comprender el significado y la relevancia de la precisión y el recall en la evaluación de modelos, es útil considerar el problema de predicción como una búsqueda de documentos en una base de datos mediante una consulta. La precisión se refiere a la proporción de documentos relevantes en la lista de todos los documentos devueltos. Por otro lado, el recall mide la cantidad de documentos relevantes que devuelve el motor de búsqueda en comparación con el total de documentos relevantes que podrían haber sido devueltos.

En la práctica, suele ser necesario decidir entre obtener una alta precisión o un alto recall (sensibilidad), ya que rara vez es posible maximizar ambos a la vez. Esto se puede lograr de varias maneras, tales como:

- Asignar un mayor peso a las clases como parte de su entrada.
- Ajustar los hiperparámetros de modo que se maximicen la precisión o el recall en el conjunto de validación.
- Al ajustar el umbral de decisión en algoritmos que generan probabilidades de clases,

entonces se puede modificar la precisión y el recall. Por ejemplo, al utilizar regresión logística o un árbol de decisión, se puede optar por clasificar una predicción como positiva solo si la probabilidad obtenida por el modelo es superior a 0.9, lo que resulta en una mayor precisión, aunque a expensas de un menor recall.

Aunque la precisión y el recall se definen para el caso de clasificación binaria, también se pueden utilizar para evaluar un modelo de clasificación multicategoría. Para ello, primero se debe seleccionar una clase específica que se desea analizar. Luego, se consideran todos los ejemplos de esa clase como positivos y todos los ejemplos de las otras clases como negativos (Burkov, 2020).

F1 score

F1 score se define como la media armónica de la precisión y el recall, como se observa en la ecuación (11) (Chicco y Jurman, 2020):

$$F1\ score = \frac{2 * TP}{2 * TP + FP + FN} = 2 * \frac{precision * recall}{precision * recall} \quad (11)$$

Matriz de Confusión

La matriz de confusión en una clasificación multiclase tiene tantas filas y columnas como clases diferentes existan. Esta herramienta puede ser útil para identificar patrones de error. Por ejemplo, una matriz de confusión podría mostrar que un modelo entrenado para reconocer diferentes especies de animales tiende a predecir erróneamente gato en lugar de pantera o ratón en vez de rata. En este caso, se podría optar por añadir más ejemplos etiquetados de estas especies para ayudar al algoritmo de aprendizaje a distinguir entre ellas. Las matrices de confusión se utilizan para calcular dos métricas de rendimiento clave que son la precisión y el recall (Burkov, 2020).

1.6.6. Despliegue de modelos

API

Una API es un conjunto de normas y protocolos que permite la interacción entre aplicaciones de software. La definición de una API se entiende mejor al considerar su función principal: facilitar la comunicación entre componentes de software distintos. Las APIs establecen métodos y estructuras de datos que permiten a las aplicaciones intercambiar información, lo que proporciona un medio para compartir funcionalidades y datos de manera segura y eficiente (Johnson, 2024).

La estructura de una API se asemeja a un contrato entre un solicitante y un proveedor, donde la aplicación que solicita llama a operaciones específicas ofrecidas por la API. Luego, la API procesa esa solicitud y devuelve los datos o funcionalidades requeridos. Este contrato se refleja a través de endpoints, que incluyen un URI (Identificador Uniforme de Recursos) que indica la ubicación del recurso, el método de acceso y el formato esperado de los datos que se transferirán (Johnson, 2024).

REST

REST es un estilo arquitectónico caracterizado por un conjunto de restricciones que sirven como guía para el diseño de aplicaciones en red. Fue presentado por Roy Fielding en su tesis doctoral en el año 2000. REST no se considera un protocolo o un estándar, sino más bien una filosofía de diseño que busca crear interfaces simplificadas y escalables en la web. Una de las principales características de REST es su uso de comunicación sin estado a través de métodos HTTP (Johnson, 2024).

En el contexto de REST, los recursos, que pueden ser cualquier entidad como un documento o una imagen, se identifican mediante URIs (Identificadores Uniformes de

Recursos). La interacción con estos recursos se lleva a cabo mediante un conjunto de métodos HTTP, tales como:

- GET: Utilizado para obtener datos de un servidor en el recurso especificado.
- POST: Utilizado para enviar datos a ser procesados en un recurso específico.
- PUT: Utilizado para reemplazar todas las representaciones actuales del recurso objetivo con los datos proporcionados.
- DELETE: Utilizado para eliminar el recurso especificado.
- PATCH: Utilizado para aplicar modificaciones parciales a un recurso (Johnson, 2024).

Aplicaciones Web.

Una aplicación web es un programa interactivo accesible a través de navegadores web. Las aplicaciones web son herramientas dinámicas que permiten a los usuarios realizar diversas tareas. A diferencia de las aplicaciones de escritorio que se instalan en una máquina local, las aplicaciones web se instalan en servidores. Los desarrolladores suelen dividir las aplicaciones web en dos componentes principales: el front end y el back end. El front end incluye todo lo que el usuario ve en su pantalla, mientras que el back end abarca las partes ocultas al usuario, como los scripts que soportan el funcionamiento de la aplicación (Saia et al., 2022).

Framework Web Django.

Django es un framework para aplicaciones web de código abierto y alto nivel, desarrollado en Python, que destaca por su capacidad para crear aplicaciones de manera eficiente, segura y con menos líneas de código en comparación con otros frameworks. Esto facilita el desarrollo ágil sin comprometer la seguridad ni la funcionalidad. Utiliza el principio Don’t Repeat Yourself (No te repitas) para evitar que exista una duplicación innecesaria en el código (Espinosa-Hurtado, 2021).

Servidor Web

Un servidor web se encarga de procesar solicitudes HTTP y devolver respuestas. El término de servidor web puede hacer referencia al software del servidor web o también al dispositivo o equipo computacional dedicado a proporcionar las páginas web. Los softwares de servidores web implementan HTTP y manejan las conexiones relacionadas con TCP; se encargan de administrar las respuestas entregadas por el servidor web y gestionan la administración de configuraciones, recursos, control, seguridad, desempeño y aspectos de mejora del servidor web. El servidor web comparte la responsabilidad de administrar las conexiones TCP junto con el sistema operativo (Gourley et al., 2022).

Linux

Linux es un sistema operativo creado por Linus Torvalds. Linux es seguro, es gratuito y su personalización es total, permite modificar el kernel y los programas del sistema gracias a las opciones de compilación y la licencia GPL que da acceso al código fuente; además Linux es eficiente funciona en hardware de bajo costo y su diseño busca aprovechar al máximo las capacidades del sistema; es portable ya que pueden funcionar aplicaciones desarrolladas en diferentes lenguajes de programación y es adecuado para ejecutar aplicaciones web (Bovet y Cesati, 2020).

CAPITULO 2

METODOLOGÍA

2. METODOLOGÍA Y DESARROLLO

2.1 Metodología

2.1.1. Metodología de la investigación

La metodología que se ha seguido para el desarrollo del proyecto comprende en el ámbito de investigación de las siguientes etapas: investigación bibliográfica e investigación cuasiexperimental.

Investigación bibliográfica

Para la revisión literaria sobre los antecedentes y estado del arte de estudios relacionados al proyecto de titulación que se han considerado en la introducción del capítulo 1, se realizó la búsqueda de artículos de revistas en bases indexadas como Redalyc, Dialnet, IEEE, Science Direct, Springer y Scopus.

Para el desarrollo de los temarios de las bases teóricas que sustentan el desarrollo técnico del proyecto que están incluidos en el marco teórico del capítulo 1, se realizó la consulta de artículos de revistas en bases indexadas como Proquest, Ebsco, Dialnet, Science Direct y Springer. Adicionalmente, también se consideran libros para las definiciones conceptuales relacionados a la temática del proyecto.

Entre los criterios de búsqueda que se aplican son los siguientes:

- De preferencia información relevante de los últimos 5 años.
- Conectores lógicos: and, or
- Para el estado de arte: artículos de revistas indexadas
- Para el marco teórico: artículos de revistas indexadas y libro.
- Palabras de búsqueda en inglés como:

- fraud and credit card and machine learning
- fraud and credit card and deep learning
- fraud and credit card and machine learning or deep learning
- techniques and data mining
- machine learning and supervised learning
- supervised learning and algorithms
- deep learning and perceptron
- deep learning and MLP
- optimization and machine learning or deep learning
- metrics and evaluation and machine learning and classification

- Palabras de búsqueda en español como:
  - fraude and tarjeta crédito and aprendizaje automático
  - fraude and tarjeta crédito and aprendizaje profundo
  - fraude and tarjeta crédito and aprendizaje automático or aprendizaje profundo
  - técnicas y minería de datos
  - aprendizaje automático and aprendizaje supervisado
  - aprendizaje supervisado and algoritmos
  - aprendizaje profundo and perceptrón
  - aprendizaje profundo and MLP
  - optimización and aprendizaje automático or aprendizaje profundo
  - métricas and evaluación and aprendizaje automático and clasificación

Investigación básica

Se inicia con la recolección y estudio de datos históricos de transacciones, lo que facilita el análisis del comportamiento de los usuarios y los patrones que podrían señalar sospechas de fraude. Se investigan varios algoritmos de aprendizaje automático y minería

de datos, como la regresión logística, árboles de decisión, redes neuronales entre otras técnicas para identificar fraudes.

En este estudio también se abarca la investigación de la valoración de exactitud, sensibilidad, especificidad y la tasa de falsos positivos en los modelos, además del estudio de las características más significativas para la identificación de fraudes con tarjetas de crédito, entendiendo los principios esenciales que respaldan los algoritmos y procedimientos empleados para detectar transacciones fraudulentas.

Investigación cuasi-experimental

Se trabaja con conjuntos de datos históricos de transacciones, dividiendo el dataset en 80% datos de entrenamiento y 20% datos de pruebas, con transacciones etiquetadas como fraudulentos o no fraudulentos. Se implementan diversos modelos entre algoritmos de aprendizaje supervisado y aprendizaje profundo.

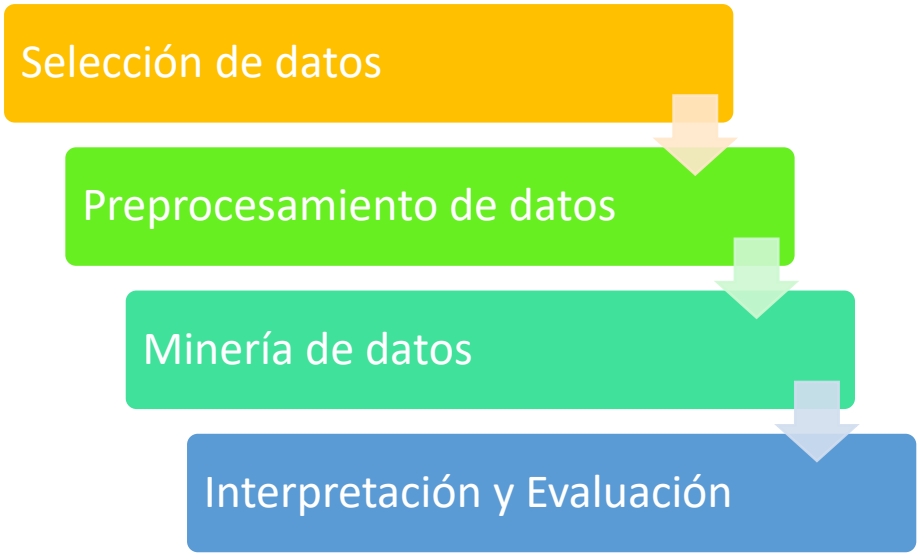
Durante el proceso de entrenamiento, se aplican técnicas de optimización y validación cruzada en la búsqueda de hiperparámetros de los algoritmos de aprendizaje automático. También técnicas de regularización en los algoritmos de aprendizaje profundo para evaluar el rendimiento del modelo bajo diferentes condiciones, sin la necesidad de manipular directamente las variables de estudio (predictoras).

2.1.2. Metodología del proyecto

En la metodología del proyecto de titulación, se considera el proceso KDD para el desarrollo de proyectos relacionados a Ciencia de Datos, como se observa en Figura 5.

Figura 5.

Metodología del proceso KDD



Nota: La figura describe las fases del proceso KDD ejecutados en el proyecto de titulación.

Obtenido: Elaboración propia

2.2 Desarrollo

2.2.1. Desarrollo del modelo predictivo

Se describen las fases realizadas durante el proceso del modelamiento aplicando las técnicas de minería de datos, tanto de aprendizaje supervisado y aprendizaje profundo.

Fase: Selección de datos

Los datos fueron obtenidos del repositorio abierto Kaggle, el dataset Credit Card Fraud. Este conjunto de datos consta de transacciones con tarjetas de crédito en el oeste de Estados Unidos. Incluye información sobre cada transacción, incluidos los datos del cliente, el comerciante y la categoría de la compra, y determina si la transacción fue o no un fraude.

El dataset comprende de 14 variables predictoras entre variables categóricas y variables numéricas y 1 variable target con 14.446 registros, en la Tabla 3 se observa el diccionario de datos y en la Figura 6 se observa los tipos de datos del dataset.

El link del dataset es:

https://www.kaggle.com/datasets/neharoychoudhury/credit-card-fraud-data

Tabla 3.

Diccionario de datos del dataset Credit Card Fraud

Variable	Descripción
trans_date_trans_time	Fecha y hora de la transacción
merchant	Nombre del comerciante
category	Categoría de Comerciante
amt	Monto de la transacción
city	Ciudad del Titular de la Tarjeta de Crédito
state	Estado/lugar del titular de la tarjeta de crédito
lat	Latitud Lugar de compra
long	Longitud Lugar de Compra
city_pop	Población de la ciudad del titular de la tarjeta de crédito
job	Trabajo del titular de la tarjeta de crédito
dob	Fecha de nacimiento del titular de la tarjeta de crédito
trans_num	Número de transacción
merch_lat	Ubicación de latitud del comerciante
merch_long	Ubicación de longitud del comerciante
is_fraud	Si la transacción es fraudulenta (1) o no (0)

Nota: La tabla describe las variables del dataset Credit Card Fraud. Obtenido: Elaboración propia

Figura 6.

Tipo de datos del dataset Credit Card Fraud

```
RangeIndex: 14446 entries, 0 to 14445
Data columns (total 15 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   trans_date_trans_time  14446 non-null  object
 1   merchant               14446 non-null  object
 2   category               14446 non-null  object
 3   amt                    14446 non-null  float64
 4   city                   14446 non-null  object
 5   state                  14446 non-null  object
 6   lat                    14446 non-null  float64
 7   long                   14446 non-null  float64
 8   city_pop               14446 non-null  int64
 9   job                    14446 non-null  object
 10  dob                    14446 non-null  object
 11  trans_num              14446 non-null  object
 12  merch_lat              14446 non-null  float64
 13  merch_long             14446 non-null  float64
 14  is_fraud               14446 non-null  object
dtypes: float64(5), int64(1), object(9)
memory usage: 1.7+ MB
```

Nota: La figura describe los tipos de datos del dataset Credit Card Fraud. Obtenido: Elaboración propia

Fase: Preprocesamiento de datos

Se aplican técnicas de limpieza de datos para lo siguiente:

Verificación de nulos y duplicados

- No se encuentran nulos.
- Se encuentran 68 registros duplicados, se proceden a eliminarlos.

Transformación de los datos

La variable ‘trans_date_trans_time’ de la transacción se desagrega en día de la semana, día del mes, mes, hora y año. Se obtienen nuevas variables: ‘transaction_day_of_week’, ‘transaction_day_of_month’, ‘transaction_month’, ‘transaction_hour' y ‘transaction_year’ respectivamente.

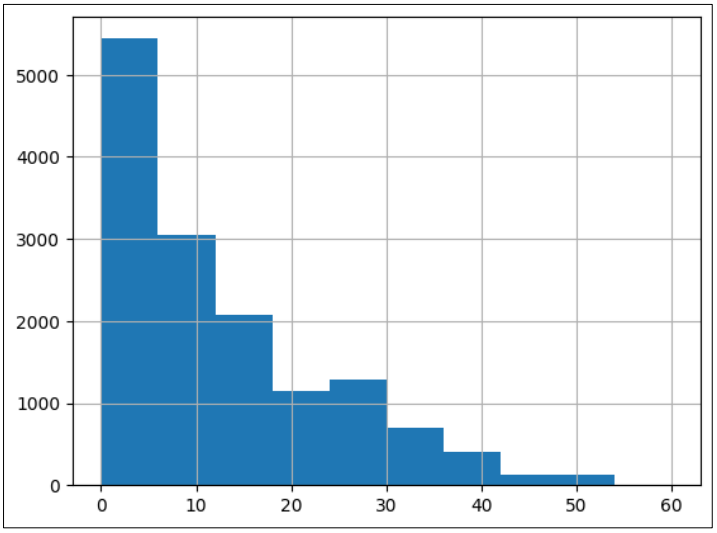
La variable ´merchant’ que es el nombre del establecimiento, se realiza una categorización en base a la cantidad de fraudes que tiene cada negocio, para lo cual se contabiliza la variable ‘is_fraud’, encontrándose que hay locales que no tienen fraudes y otros que si tienen muchos fraudes.

Primero, se procede a realizar el tratamiento de la variable ‘merchant’ para estandarizar el texto que algunos tienen el valor con “ ” y otros no tienen las comillas, posteriormente se realiza la categorización de la variable ‘merchant’ en base a la contabilización del total de transacciones con fraudes por cada establecimiento.

Luego, se calcula el valor porcentual que representa la frecuencia de fraude de todas las transacciones multiplicado por 100. Se obtiene los siguientes resultados porcentuales, donde se observa que un poco menos del 60% de las transacciones son con fraudes, como se observa en la Figura 7.

Figura 7.

Resultados porcentuales del conteo de establecimientos con Fraudes



Nota: La figura describe los porcentuales del conteo de establecimientos con Fraudes para con esa información encontrar . Obtenido: Elaboración propia

Una vez obtenido los valores porcentuales se realiza la categorización tomando en cuenta los percentiles para obtener 5 categorías con la siguiente función:

```
# Crear la nueva columna fraud_category_merchant
def categorize_fraud(row):
   # Entre 0 y el 10vo percentil
   if row['porcent_fraud'] == 0:
      return 0
   # Superior al 10vo y el 20vo percentil
   elif row['porcent_fraud'] <= percentiles[1]:
      return 1
   # Superior al 20vo y el 30vo percentil
   elif row['porcent_fraud'] <= percentiles[2]:
      return 2
   elif row['porcent_fraud'] <= percentiles[3]:
      return 3
   # Superior al 30vo y el 40vo percentil
   elif row['porcent_fraud'] <= percentiles[4]:
      return 4
   else: # Por encima del 40vo percentil
      return 5
```

Las categorías por fraude para cada establecimiento son las siguientes: 0: no fraude, 1: bajo fraude, 2: medianamente bajo fraude, 3: intermedio fraude, 4: medianamente alto fraude, 5: alto fraude. Se obtiene una nueva variable ´fraud_category_merchant'

La variable 'dob' que es la fecha de nacimiento del cliente, se desagrega para obtener la edad del cliente, creando la variable 'age'.

La variable 'job' que es el tipo de trabajo del cliente, se agrupa de acuerdo con tipos de trabajos similares mediante la siguiente función:

```
# Definimos grupos de profesiones
def classify_profession(profession):
    profession_lower = profession.lower()
    if any(keyword in profession_lower for keyword in ['engineer',
   'scientist', 'technologist', 'technologist']):
        return 'Ingeniería y Técnicos'
    elif any(keyword in profession_lower for keyword in ['nurse',
   'therapist', 'psychologist']):
        return 'Salud'
    elif any(keyword in profession_lower for keyword in ['teacher',
   'lecturer', 'educator']):
        return 'Educación'
    elif any(keyword in profession_lower for keyword in ['manager',
   'executive', 'officer', 'consultant']):
        return 'Ejecutivos'
    elif any(keyword in profession_lower for keyword in ['artist',
   'designer', 'writer', 'musician']):
        return 'Arte'
    elif any(keyword in profession_lower for keyword in
   ['administrator', 'accountant', 'officer']):
        return 'Administradores'
    else:
        return 'Other'
```

Obteniéndose la variable 'profession_group' con los grupos obtenidos de las profesiones: Ingeniería y Técnicos, Salud, Educación, Ejecutivos, Arte, Administradores y Otros.

EDA - Exploración de los datos limpios

Se realiza diversas gráficas como pastel, histogramas, dispersión, boxplot, barras agrupadas y mapas para poder entender los datos y determinar los patrones más relevantes entre las variables predictoras y la variable objetivo 'is_fraud'.

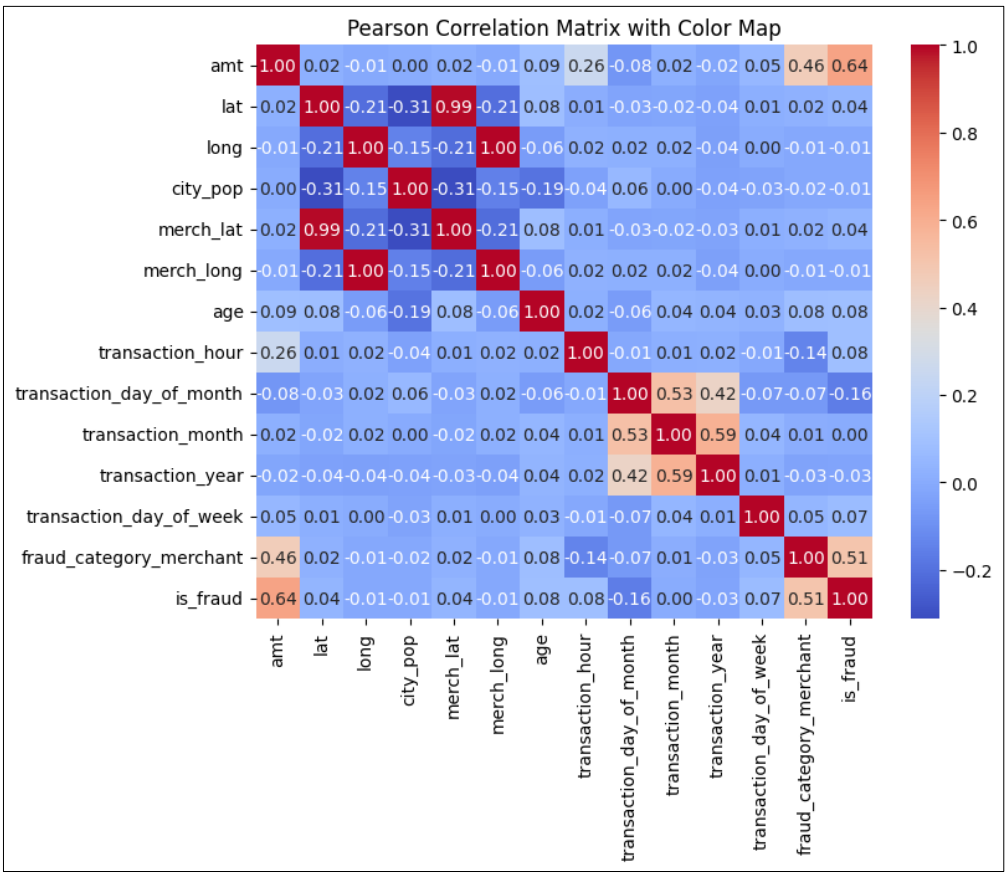
Estas graficas se encuentran en los resultados del capítulo 3.

Selección de variables

En la selección de variables se aplica la técnica estadística de correlación de Pearson para las variables numéricas como se observa en la Figura 8.

Figura 8.

Diagrama con la Matriz de Correlación de Pearson.



Nota: La figura describe la correlación de Pearson en un Mapa de calor de las variables numéricas del dataset para selección de variables con alta relación a la salida. Obtenido: Elaboración propia

Las variables que tienen una correlación alta sobre la variable ‘is_fraud’ son: ‘amt’ y ‘fraud_category_merchant’.

Las variables que tienen muy mínima relación con la variable de salida son: ‘lat’, ‘long’, ‘city_pop’, ‘merchant_lat’, ‘merchant_long’, ‘transaction_month’, ‘transaction_year’.

Se observa que las variables ‘merchant_lat’ y ‘lat’ están altamente correlacionadas, al igual que las variables ‘merchant_long’ y ‘long’.

Se seleccionan 10 variables para el entrenamiento de los modelos, estas son las siguientes: 'category', 'amt', 'city', 'state', 'age', 'transaction_hour', 'transaction_day_of_month', 'transaction_day_of_week', 'profession_group' y 'fraud_category_merchant'.

Partición de los datos

Se ejecuta la partición de los datos utilizando la técnica de train_test_split de la librería sklearn, realizando la distribución con datos estratificados de ‘y’ para que se repartan uniformemente con un 80% datos de entrenamiento y 20 % datos de pruebas.

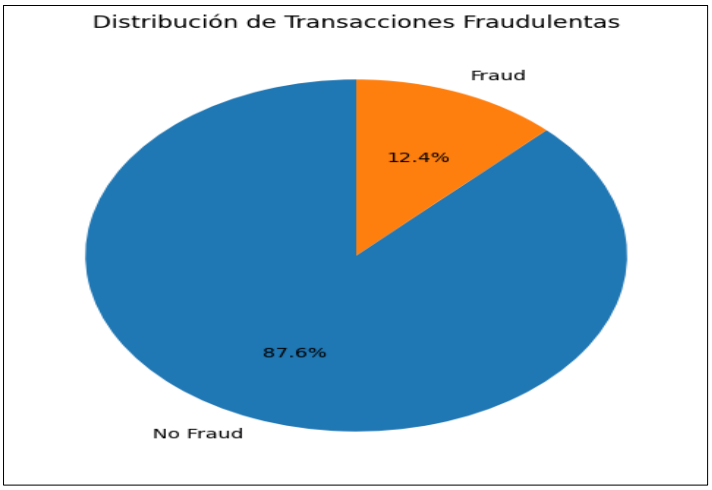
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=45, stratify=y)
```

Balanceo de los datos

Se analiza el balanceo de los datos de entrenamiento X_train, y_train de la variable ‘is_fraud’ mediante un diagrama de pastel como se observa en Figura 9.

Figura 9.

Distribución original de Transacciones Fraudulentas

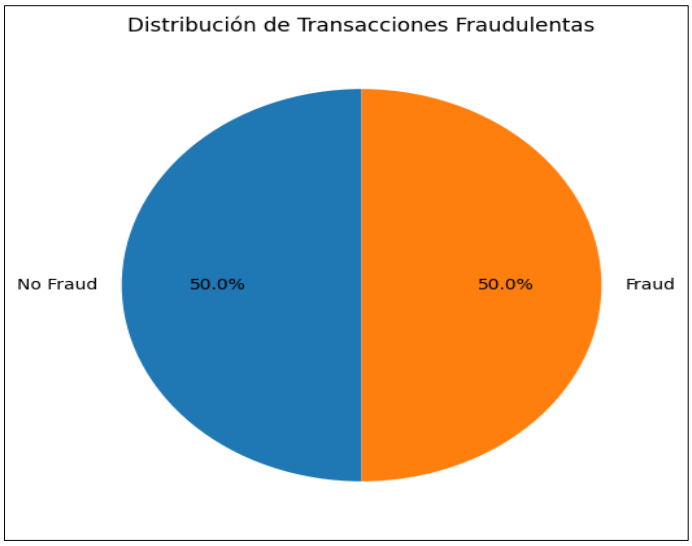


Nota: La figura muestra el porcentaje de datos de transacciones con fraude y sin fraude, observando que hay desbalance en los datos. Obtenido: Elaboración propia

Una vez obtenidos los datos del entrenamiento se procede a unificar los datos de X_train, y_train para aplicar la técnica de balanceo upsampled a la clase minoritaria quedando 10080 tanto para la clase 0 y la clase 1 de la variable ‘is_fraud’ como se observa en la Figura 10.

Figura 10.

Distribución con datos balanceados de Transacciones Fraudulentas



Nota: La figura muestra el porcentaje de datos balanceados de transacciones con fraude y sin fraude luego de aplicar la técnica SMOTE. Obtenido: Elaboración propia

Codificación de las variables categóricas

Se aplica la codificación de las variables de acuerdo con la naturaleza de los datos y requisitos del algoritmo: Label encoder: Para variables que no tienen un orden ordinal en algoritmos de aprendizaje supervisado.

Escalado de los datos

Se aplica el escalado de los datos a las variables numéricas del X_train y X_test para los algoritmos que requiere de datos escalados como: Regresión logística, SVM, KNN, Naive Bayes y la red MLP. No se aplica escalado de datos a los algoritmos que no requieren como Árbol de decisión, Random Forest y Gradient Boosting.

Fase: Minería de datos

Algoritmos de machine learning:

Los algoritmos de aprendizaje supervisados que se utilizan en el entrenamiento de los modelos se encuentran en la Tabla 4 que detalla los algoritmos ensemble utilizados como Random Forest y Gradient Boosting, mientras que en la Tabla 5 se mencionan los algoritmos tradicionales: Regresión Logística, Naive Bayes, SVM, KNN y Árbol de decisión.

En todos los algoritmos, se aplica las técnicas de optimización en la búsqueda de los mejores hiperparámetros: RandomSearchCV y Validación cruzada con un valor de 5, esta técnica permite validar el modelo durante el entrenamiento con los algoritmos de aprendizaje supervisado de caja blanca y RandomSearchCV para encontrar los mejores hiperparámetros de cada modelo. La técnica RandomSearchCV se alimenta de un diccionario con los hiperparámetros y un rango de valores de cada uno, de acuerdo con el algoritmo de aprendizaje supervisado, se describe en la Tabla 4 y Tabla 5 respectivamente.

Tabla 4.

Algoritmos ensemble de aprendizaje supervisado aplicados a los datos del proyecto

Modelo	Algoritmo	Diccionario de datos	Optimización y Validación cruzada

1	Random Forest	paramsRF = {
		'n_estimators': [50, 75, 100, 125, 150, 200],
		'max_features': ['sqrt', 'log2', None],
		'max_depth': randint(1, 20),
		'min_samples_split': randint(2, 20),
		'min_samples_leaf': randint(1, 20),
		'criterion': ['gini', 'entropy']
		}
			RandomSearchCV
			Validación cruzada 5

2	Gradient Boosting	param_distGB = {
		"n_estimators": [50, 100, 200, 300],
		"learning_rate": [0.01, 0.1, 0.2],
		"max_depth": [3, 4, 5],
		"min_samples_split": [2, 5, 10],
		"min_samples_leaf": [1, 2, 4],
		"subsample": [0.8, 0.9, 1.0],
		"loss": ["log_loss", "exponential"]
		}
			RandomSearchCV
			Validación cruzada 5

Nota: La tabla describe los algoritmos ensemble y el diccionario con los hiperparámetros a seleccionarse en el entrenamiento de los modelos. Obtención: Elaboración propia.

Tabla 5.

Algoritmos de aprendizaje supervisado aplicados a los datos del proyecto

Modelo	Algoritmo	Diccionario de datos	Optimización y Validación cruzada

1	Regresión logística	param_distRL = {
		'C': [0.01, 0.1, 1, 10, 100],
		'penalty': ['l1', 'l2'],
		'solver': ['liblinear', 'saga'],
		'max_iter': [100, 200, 500],
		'class_weight': [None, 'balanced']
		}
			RandomSearchCV
			Validación cruzada 5

2	Naive Bayes	param_distNB = {
		'var_smoothing':np.logspace(0,-9, num=100),
		'priors': [None] + [[1 / num_classes] * num_classes]
		
		}
			RandomSearchCV
			Validación cruzada 5

3	SVM	param_distSVC = {
		"C": [0.1, 1, 10, 100],
		"kernel": ["linear", "poly", "rbf", "sigmoid"],
		"degree": [2, 3, 4],
		"gamma": ["scale", "auto"],
		"coef0": [0, 0.1, 0.5, 1],
		"tol": [1e-3, 1e-4, 1e-5],
		}
			RandomSearchCV
			Validación cruzada 5

4	KNN	param_distKNN = {
		"n_neighbors": [3, 5, 7, 9, 11],
		"weights": ["uniform", "distance"],
		"metric":["euclidean","manhattan", "minkowski"],
		"p": [1, 2]
		}
			RandomSearchCV
			Validación cruzada 5

5	Árbol de decisión	paramsDT = {
		'criterion': ['gini', 'entropy'],
		'splitter': ['best', 'random'],
		'max_depth': randint(1, 20),
		'min_samples_split': randint(2, 20),
		'min_samples_leaf': randint(1, 20),
		'max_features': ['sqrt', 'log2', None]
		}
			RandomSearchCV
			Validación cruzada 5

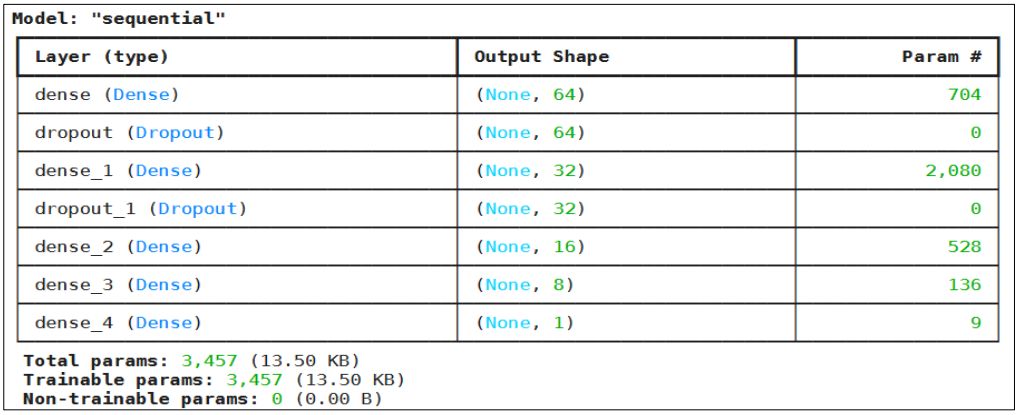
Nota: La tabla describe los algoritmos y el diccionario con los hiperparámetros a seleccionarse en el entrenamiento de los modelos. Obtención: Elaboración propia.

Algoritmo de aprendizaje profundo

Se aplica el algoritmo de la arquitectura de una red neuronal MLP (Multilayer Perceptron) como se describe en la Figura 11 y en la Tabla 6 los componentes de la arquitectura, compilado, técnicas de regularización y entrenamiento del modelo.

Figura 11.

Arquitectura de la red MLP del dataset Fraude Tarjeta de Crédito



Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	704
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 32)	2,080
dropout_1 (Dropout)	(None, 32)	0
dense_2 (Dense)	(None, 16)	528
dense_3 (Dense)	(None, 8)	136
dense_4 (Dense)	(None, 1)	9

Total params: 3,457 (13.50 KB)
Trainable params: 3,457 (13.50 KB)
Non-trainable params: 0 (0.00 B)

Nota: La figura describe los componentes de la arquitectura MLP. Obtención: Elaboración propia.

Tabla 6.

Componentes de la Arquitectura de la red MLP del dataset Fraude Tarjeta de Crédito

Componente red MLP	**Descripción**
Arquitectura del modelo + Técnicas de regularización	64 neuronas con 10 entradas con función de activación 'relu', kernel_regularizer=l2(0.001) (capa entrada, capa intermedia) Dropout con 0.05 (desconexión de las neuronas) 32 neuronas con función de activación 'relu' (capa intermedia) Dropout con 0.05 (desconexión de las neuronas) 16 neuronas con función de activación 'relu' (capa intermedia) 8 neuronas con función de activación 'relu' (capa intermedia) 1 neurona con función de activación 'sigmoid' (capa salida para clasificación binaria)
Compilado del modelo	Función de pérdida 'binary_crossentropy' (clasificación binaria) Optimizador 'adam' (tasa de aprendizaje variable - decaimiento por pasos)
Callback EarlyStopping	monitor='val_loss' (monitorea con la pérdida de validación para hacer la parada temprana) patience=10 (se detiene si en 10 epoch consecutivos ya no generaliza la red)
Entrenamiento del modelo	epochs=100 batch_size=32 validation_split=0.2 (20% de datos de validación) callbacks=[early_stop, lr_scheduler] (detección de parada, tasa de aprendizaje variable)

Nota: La tabla describe los componentes de la arquitectura MLP. Obtención: Elaboración propia.

Fase: Interpretación y Evaluación

En la interpretación de los modelos obtenidos se calcula las métricas de clasificación como accuracy, precision, recall, f1score y espicificidad, matriz de reporte de la matriz de confusión, gráfica de la matriz de confusión, auc y curva roc de cada modelo. Adicionalmente en la red MLP se obtiene las métricas - gráficas de pérdidas y exactitud del entrenamiento para analizar el overfitting (sobreajuste) de la red neuronal.

En los modelos que se entrenan con los árboles de decisión, random forest y gradient boosting, se extrae la importancia de las variables para el análisis de que variables aportan más con mayor conocimiento durante el entrenamiento a través de estos algoritmos.

También se elabora las gráficas de frontera de decisión para el análisis de la eficiencia del modelo al separar los datos entre la Clase 0 y la Clase 1 de la variable objetivo ‘is_fraud’ mediante la técnica tSNE. Por último, se obtiene las gráficas para la explicabilidad e interpretabilidad mediante la librería SHARP de los modelos mejor evaluados con las métricas de clasificación

El análisis de los resultados obtenidos de los modelos tanto de aprendizaje supervisado de caja blanca y de aprendizaje profundo de caja negra, se encuentran el capítulo 3 en la sección de resultados donde se describe el mejor modelo obtenido en el proceso de entrenamiento.

2.2.2. Desarrollo de la página web y API

Herramienta de desarrollo

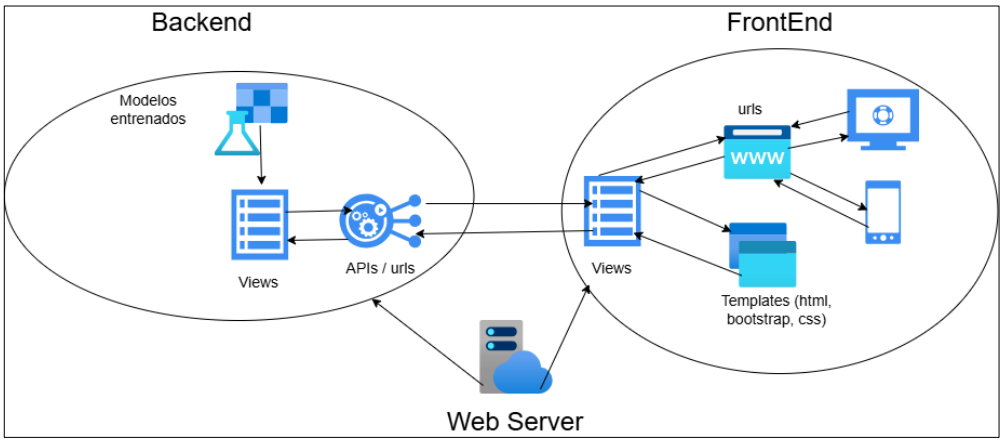
El framework de desarrollo que se usa es Django, el mismo que utiliza el lenguaje de programación Python, ya que es muy útil para el desarrollo web.

Diagrama Arquitectónico de la solución web

En la Figura 12 se detalla los componentes de la arquitectura web tanto en el lado del back end y en el lado del front end que son considerados en la implementación de la página web que permite utilizar los mejores modelos obtenidos de los entrenamientos realizados con los algoritmos de aprendizaje supervisado.

Figura 12.

Diseño de arquitectura web con el modelo predictivo de



Nota: La figura describe los componentes de la arquitectura web implementados en el proyecto.

Obtención: Elaboración propia

Descripción de la Arquitectura y el API

El framework que se utiliza para el desarrollo del backend y el frontend es Django. Los componentes de Views tienen los servicios del backend, en este punto se integran los modelos entrenados a través de la librería pickle.

En el frontend se utilizan los servicios que se encuentran en las Views las mismas que invocan el API del backend para enviar los parámetros y obtener una respuesta acerca de si existe fraude o no en los datos de la tarjeta de crédito ingresados. También se utilizan los templates de Django en los cuales se hace uso de Bootstrap, HTML y CSS para

proporcionar una interfaz gráfica web amigable. Finalmente se tienen los urls que serán llamados desde el browser para acceder a la aplicación web.

2.3 Código de las experimentaciones

En este apartado se incluye el link de enlace de GitHub donde están todo el código de la aplicación web en Django, la API de conexión y los notebooks:

https://github.com/iportegas/titulacion_grupo3

En la carpeta tarjetas, se tiene todo el código de proyecto de la web en Django, la API de conexión. En el Apéndice A se encuentra un manual de instalación sobre la instalación de la aplicación web del proyecto.

El primer notebook, contiene el preprocesamiento de los datos, exploración de los datos, separación de los datos en train y test, balanceo de los datos de train obteniendo dos ficheros train_balanceado.csv y test.csv.

El segundo notebook, limpieza de los outliers del train, codificación de las variables categóricas, escalado de las variables numéricas, entrenamientos de los 8 modelos con los algoritmos de aprendizaje supervisado de machine learning, deep learning y evaluación con las métricas de clasificación.

CAPITULO 3

RESULTADOS

3. ANÁLISIS DE RESULTADOS

3.1 Resultados

EDA - Exploración de los datos

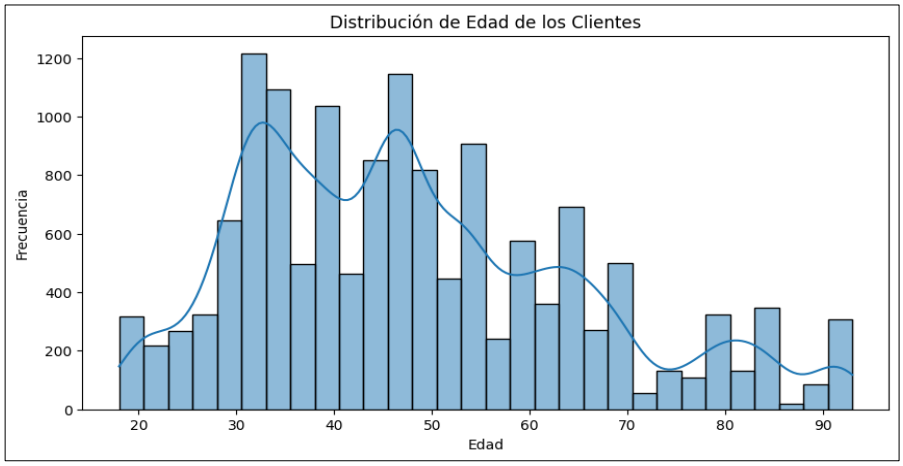
En la exploración de los datos, se obtuvieron gráficos de análisis univariado y gráficos de análisis bivariado para poder entender la estructura de los datos, examinando las características del dataset como objeto de estudio Credit Card Fraud y examinar la distribución de los datos, valores atípicos, descubrir relaciones y patrones entre las variables predictoras y la variable objetivo.

3.1.1. Análisis univariado

En la Figura 13 se observa que la mayoría de los clientes tienen entre 30 y 70 años.

Figura 13.

Distribución edad de los clientes



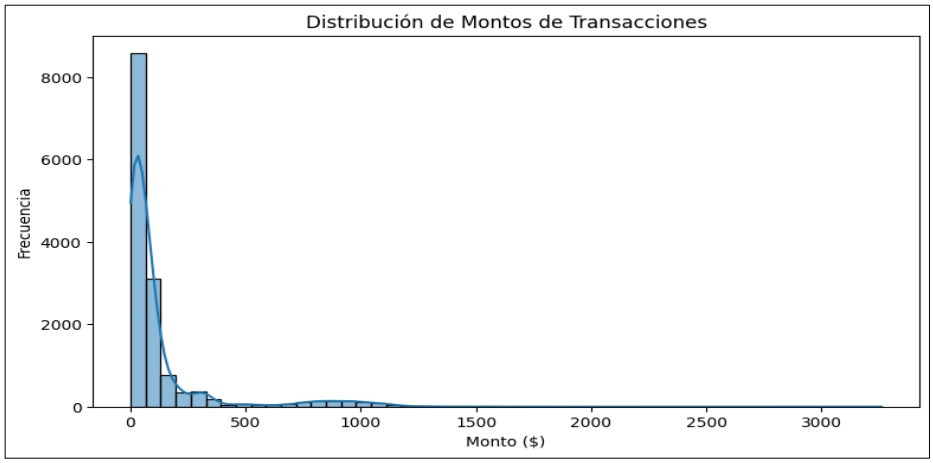
Nota: La figura describe la distribución de edad de los clientes, la mayoría entre 30 a 70 años .

Obtención: Elaboración propia

En la Figura 14 se observa que la distribución más alta del monto de las transacciones es inferior a $500,00, seguido por las transacciones con montos superiores a $500,00.

Figura 14.

Distribución montos de transacciones



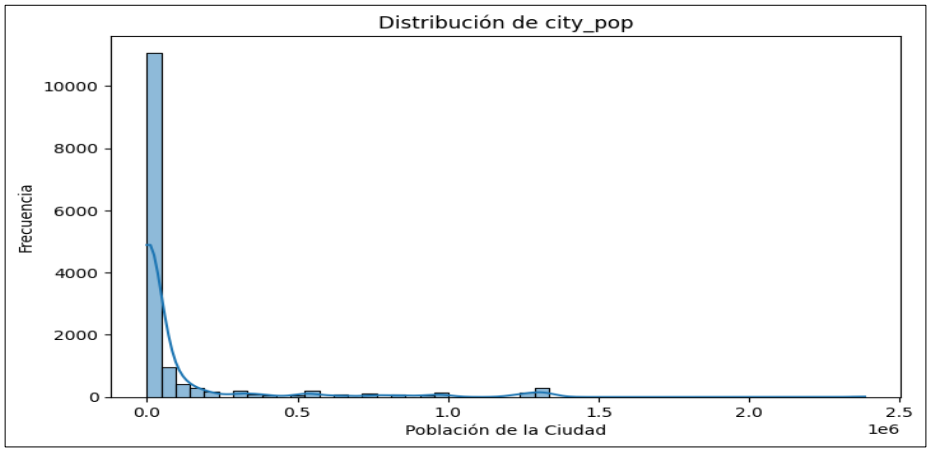
Nota: La figura describe la distribución montos de transacciones, mayoría menos de $500,00.

Obtención: Elaboración propia

En la Figura 15 se observa que la distribución más alta de la población por ciudad es inferior a 500.000 habitantes, seguido por la población con valores superiores a los 500.000 habitantes.

Figura 15.

Distribución de población de ciudad



Nota: La figura describe la distribución montos de población de ciudad, mayoría población inferior a 500.000 habitantes. Obtención: Elaboración propia.

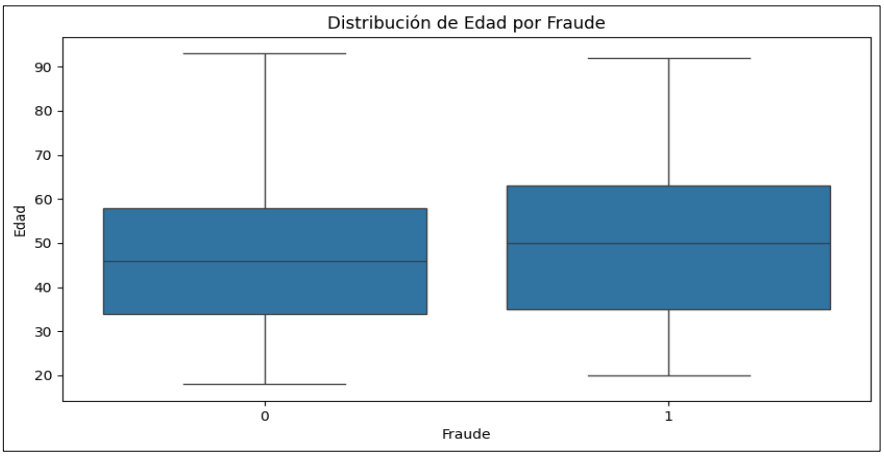
En el Apéndice B se encuentran otras figuras desde la Figura 90 a Figura 93 del resto del análisis univariado.

3.1.2. Análisis bivariado

En la Figura 16 se observa que los clientes que comenten fraude hasta el 25% tienen un poco más de 30 años, el 50% llegan a los 50 años y el 75% superan los 60 años.

Figura 16.

Distribución de edad de clientes por fraude

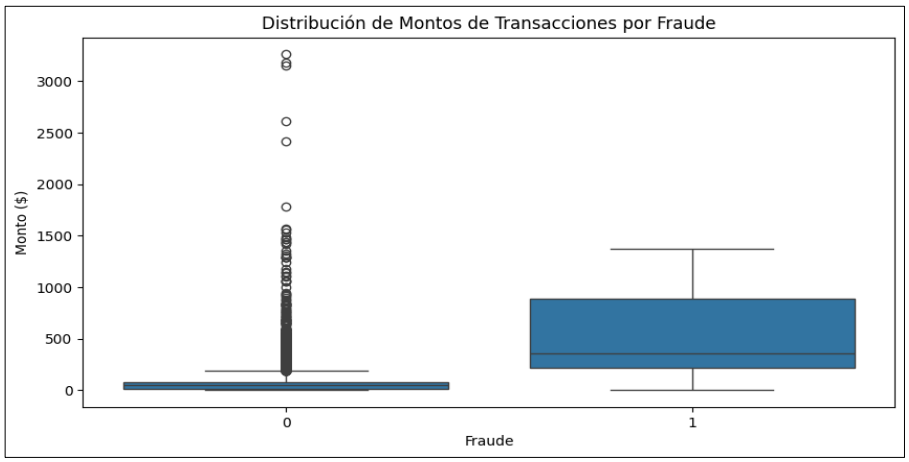


Nota: La figura describe la distribución de edad de clientes por fraude. Obtención: Elaboración propia

En la Figura 17 se observa que el monto de las transacciones fraudulentas hasta el 25% es inferior a $200,00, el 50% es inferior a $500,00 y el 75% no supera los $1000,00. El monto máximo de los fraudes no supera los $1.500,00.

Figura 17.

Distribución de montos de transacciones por fraude

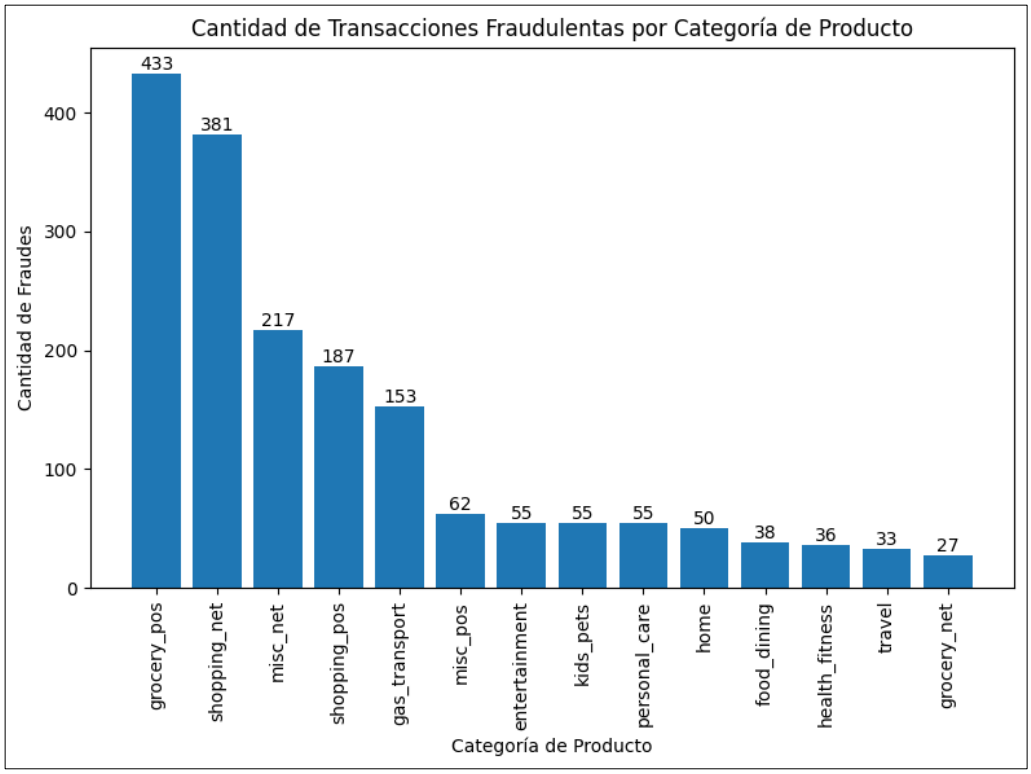


Nota: La figura describe la distribución de montos de transacciones por fraude distribuidos por los cuartiles en el gráfico boxplot. Obtención: Elaboración propia

En la Figura 18 se observa que los productos con mayor cantidad de transacciones fraudulentas son los productos: grocery_pos, shopping_net, misc_net, shopping_pos y gas_transport.

Figura 18.

Distribución de cantidad de transacciones fraudulentas por categoría de producto

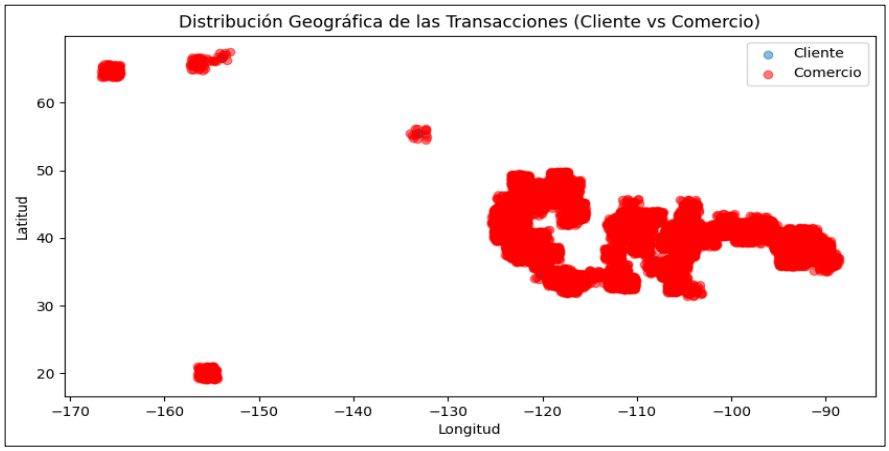


Nota: La figura describe la distribución cantidad de transacciones fraudulentas por categoría de producto, mayoría hay en grocery_pos. Obtención: Elaboración propia

En la Figura 19 se observa que la distribución geográfica de los clientes y del negocio del establecimiento son similares, esto se debe a que las compras de los clientes ocurren en sitio del comercio.

Figura 19.

Distribución geográfica de las transacciones Clientes vs Comercio

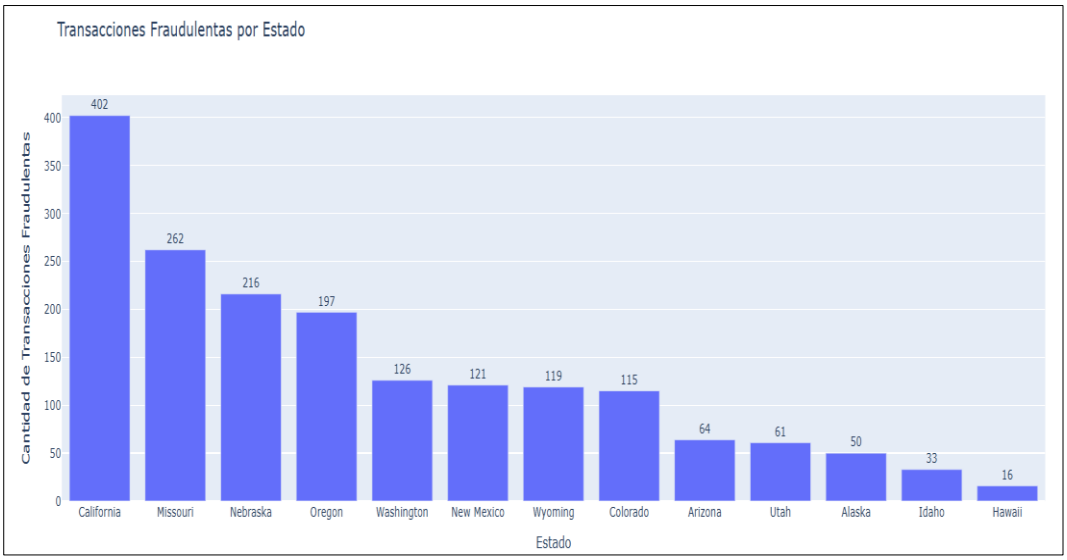


Nota: La figura describe la distribución geográfica de las transacciones Clientes vs Comercio que son compras en sitio. Obtención: Elaboración propia

En la Figura 20 se observa que los estados con mayores transacciones de fraude son: California, Missouri, Nebraska, Oregon y Washington.

Figura 20.

Distribución de transacciones fraudulentas por estado

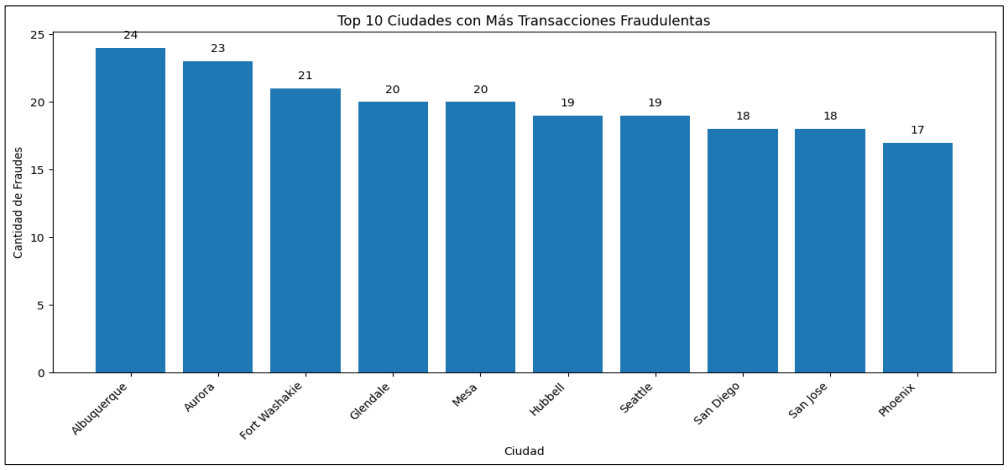


Nota: La figura describe la distribución de transacciones fraudulentas por estado siendo California con mayor fraudes. Obtención: Elaboración propia

En la Figura 21 se observa el Top 10 de las ciudades con más transacciones con fraudes, siendo las primeras 5 cuidades: Albuquerque, Aurora, Fort Washakie, Glendale y Mesa.

Figura 21.

Top 10 de cuidades con más transacciones con fraudes

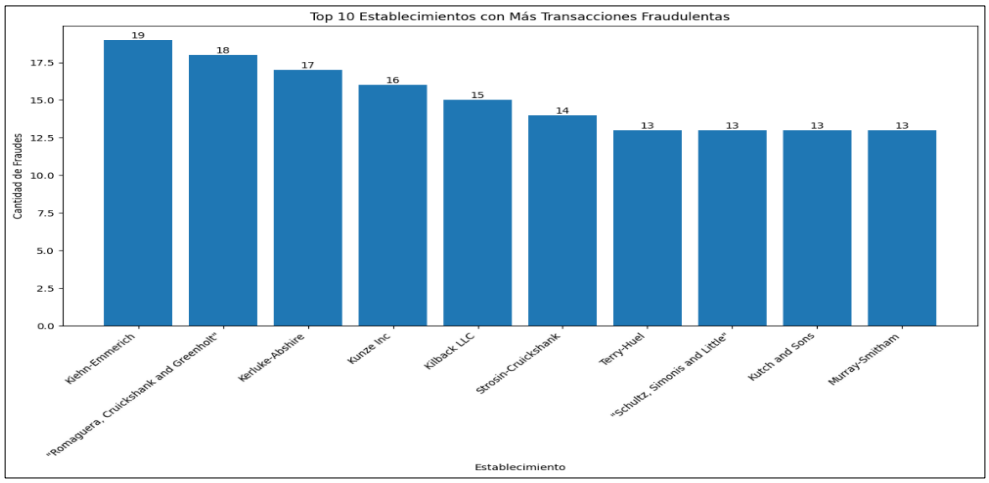


Nota: La figura describe las 10 ciudades con más transacciones con fraudes, siendo Albuquerque con mayor fraude. Obtención: Elaboración propia

En la Figura 22 se observa el Top 10 de los establecimientos con mayores cantidades de fraudes, siendo los principales: Kiehn-Emmerich, Romaguera Cruickshank and Greenholt y Kerluke-Abshire.

Figura 22.

Top 10 de establecimientos con más transacciones de fraude

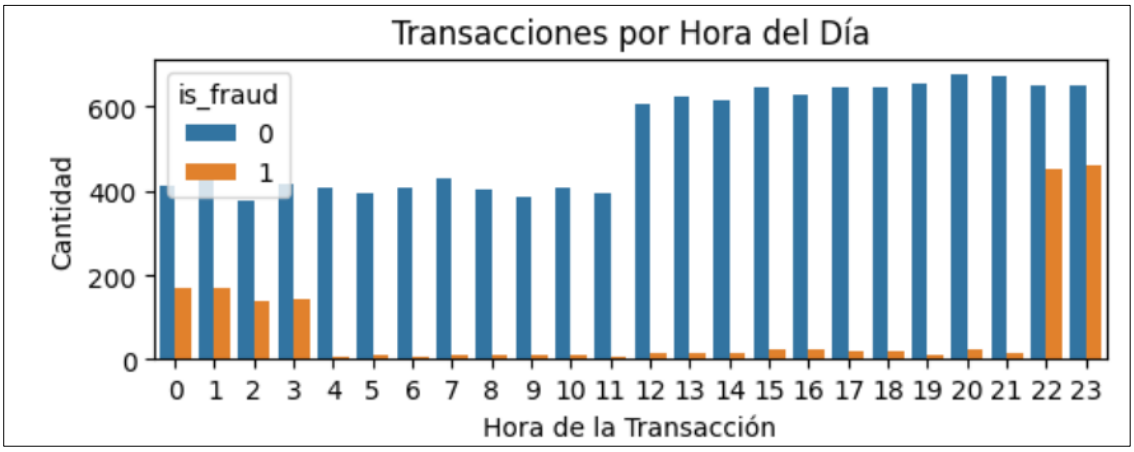


Nota: La figura describe los 10 establecimientos con más transacciones con fraudes, siendo Kiehn-Emmerich con mayor fraude. Obtención: Elaboración propia

En la Figura 23, Figura 24, Figura 25 y Figura 26 se observa que las transacciones con fraudes se ejecutan entre las 22h00 hasta las 03h00 del siguiente día, todos los días de cada mes hay transacciones fraudulentas y el dataset cuenta con datos históricos de fraudes entre el 2019 y 2020.

Figura 23.

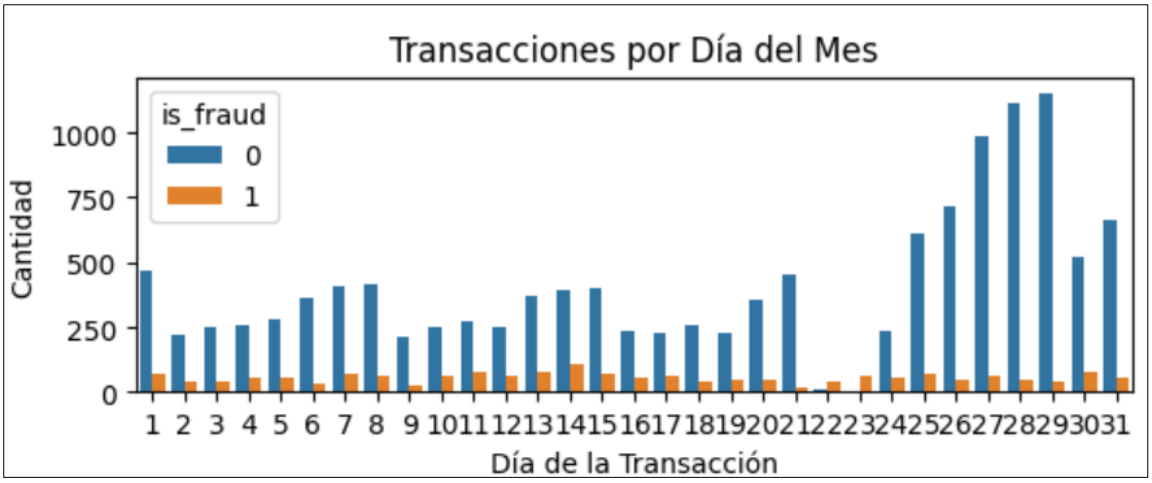
Transacciones con fraudes por hora



Nota: La figura describe Transacciones con fraudes por hora, siendo mayor en las últimas horas del día. Obtención: Elaboración propia

Figura 24.

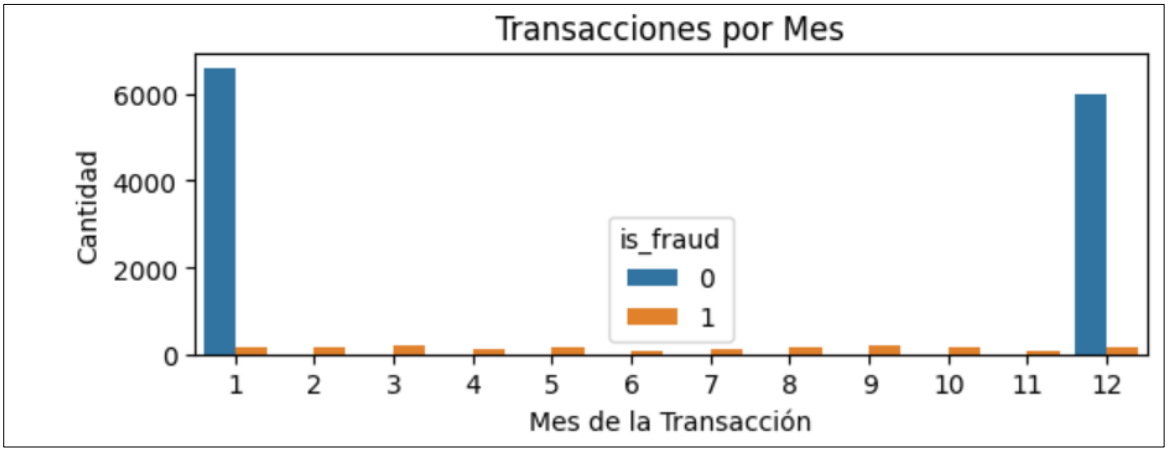
Transacciones con fraudes por día del mes



Nota: La figura describe Transacciones con fraudes por día del mes, con mayor incidencia en los días intermedios del mes. Obtención: Elaboración propia

Figura 25.

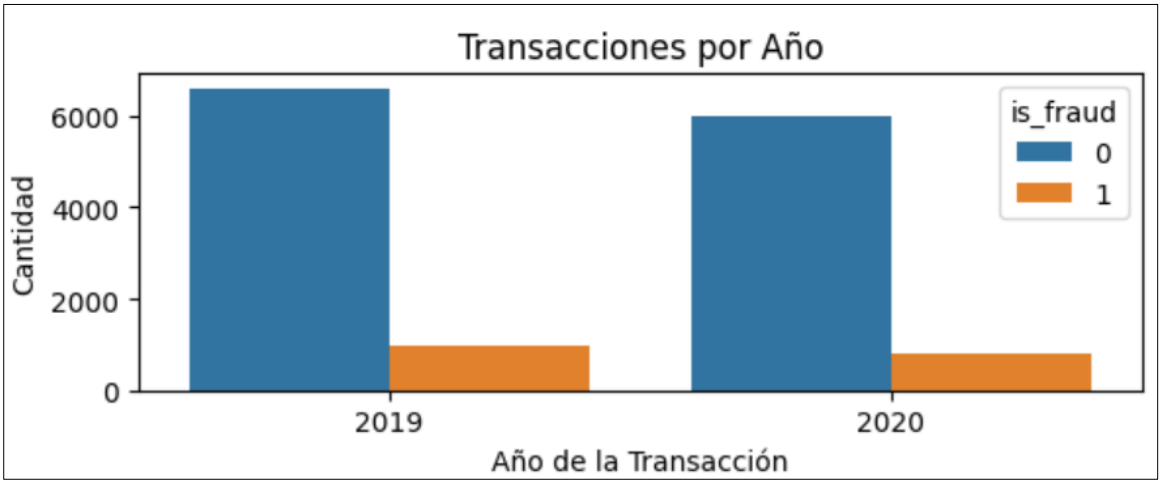
Transacciones con fraudes por mes



Nota: La figura describe Transacciones con fraudes por mes, hay en todos los meses. Obtención: Elaboración propia

Figura 26.

Transacciones con fraudes por año

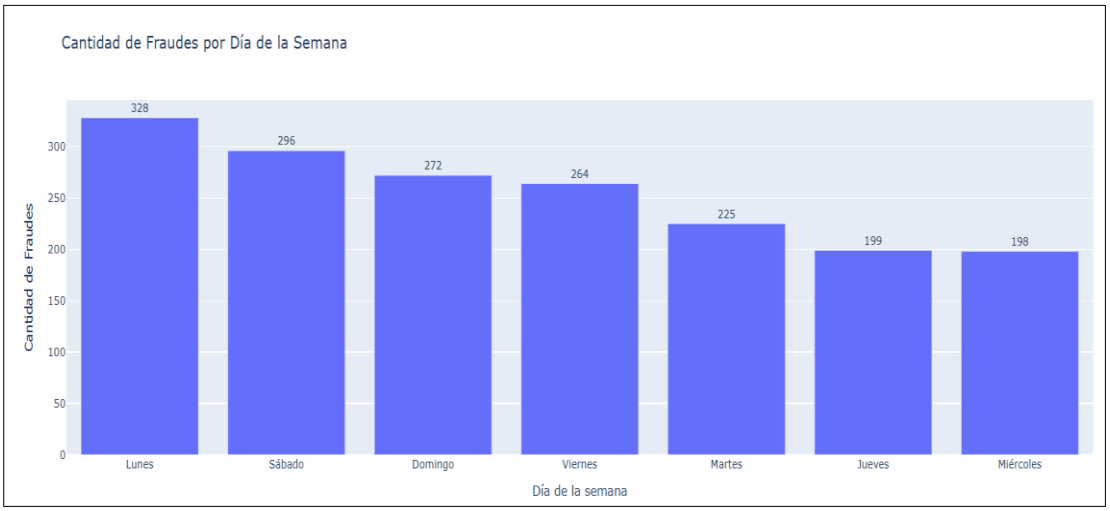


Nota: La figura describe Transacciones con fraudes por año 2019 y 2020. Obtención: Elaboración propia

En la Figura 27 se observa que los fraudes con mayores ocurrencias son los lunes, sábado y domingo.

Figura 27.

Cantidad de fraudes por día de la semana

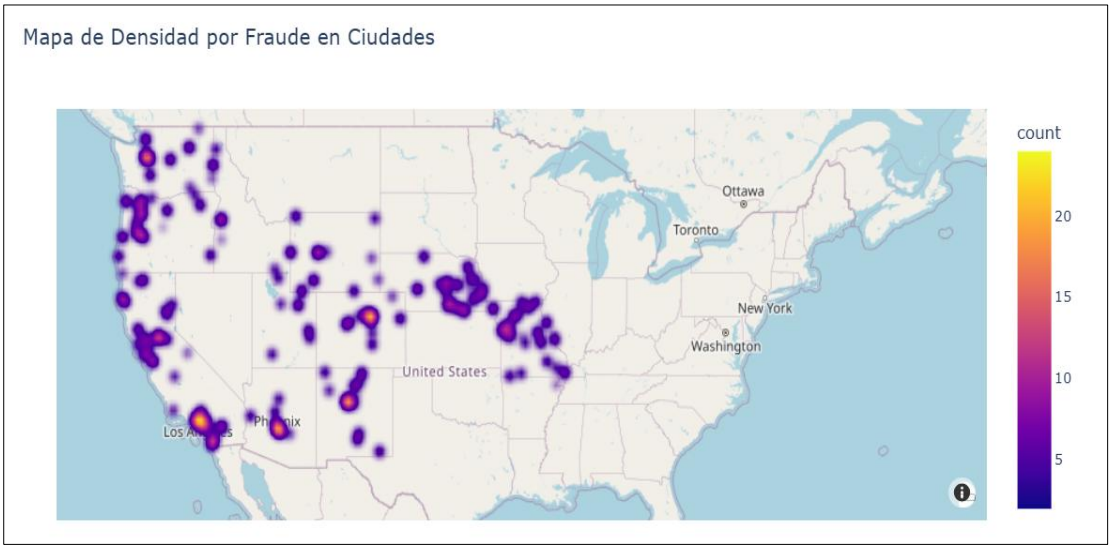


Nota: La figura describe la cantidad de fraudes por día de la semana, siendo mayor el lunes. Obtención: Elaboración propia

En la Figura 28 se observa las regiones con mayor densidad por fraude en los Estados Unidos.

Figura 28.

Mapa de densidad por fraude en ciudades

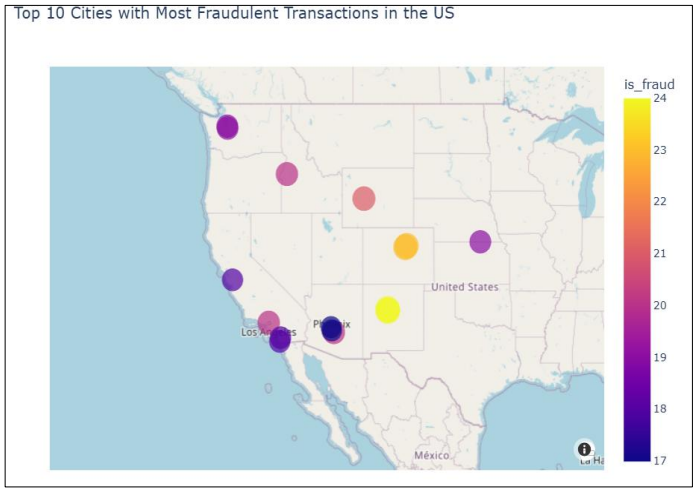


Nota: La figura muestra la densidad por fraudes en los Estados Unidos. Obtención: Elaboración propia

En la Figura 29 se observa las 10 ciudades con mayores transacciones con fraudes en los Estados Unidos.

Figura 29.

Top 10 con las ciudades con mayores transacciones de fraudes en EEUU



Nota: La figura muestra el Top 10 con las ciudades con mayores transacciones de fraudes en EEUU.

Obtención: Elaboración propia

En el Apéndice C se encuentran las figuras desde la Figura 94 a Figura 97 del resto del análisis bivariado. Y en el Apéndice D se encuentran las figuras desde la Figura 98 a Figura 99 de identificación de los outliers y tratamiento de estos.

3.2 Evaluación

Previo a la evaluación de los 8 modelos entrenados, se realiza la predicción con los datos de test y se obtiene los valores predichos como se observa la Tabla 7.

Tabla 7.

Valores de test vs Valores de predicho con los modelos entrenados

y_test	y_pred Regresión Logística	y_pred Naive Bayes	y_pred SVM	y_pred KNN	y_pred Árbol de decisión	y_pred Random Forest	y_pred Gradient Boosting	y_pred Red neuronal MLP
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
.....	.....	.....	.....	.....	.....	.....	.....	.....
0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Nota: La tabla muestra Valores de test y los Valores de predicho con los 8 modelos entrenados.

Obtención: Elaboración propia

Modelo de Regresión Logística

En la Tabla 8 se describen las condiciones de preprocesamiento de los datos escalados, codificados, las condiciones de entrenamiento con los mejores hiperparámetros y mejor score con RandomSearchCV y Validación cruzada. Se detallan las métricas obtenidas de clasificación con el modelo de Regresión Logística.

Tabla 8.

Resultados de métricas del modelo entrenado con Regresión Logística

Escalador y Codificador	Mejores hiperparámetros y Mejor score	Métricas	Tasas y Falsos
Standar scaler Label encoder Datos sin outliers	Mejores hiperparámetros encontrados: {'solver': 'liblinear', 'penalty': 'l1', 'max_iter': 100, 'class_weight': None, 'C': 0.01} Mejor score obtenido: 0.8344	Accuracy: 0.8807 Precision (sensibilidad): 0.5126 Recall: 0.7415 F1-score: 0.6061 Specificity: 0.9004 AUC: 0.8210	Tasa de falsos negativos: 0.2584 Tasa de falsos positivos: 0.0995 Falsos negativos: 92 Falsos positivos: 251

Nota: La tabla muestra los resultados de los mejores hiperparámetros y métricas del modelo entrenado con Regresión Logística. Obtención: Elaboración propia

En las Figura 30, Figura 31, Figura 32 y Figura 33, se observa que el modelo tuvo una precisión de 0.96 al predecir la clase 0 y una precisión de 0.51 al predecir la clase 1 por eso tiene un accuracy de 0.88. En la matriz de confusión se distingue en los aciertos 2270 verdaderos negativos, 264 verdaderos positivos, mientras que en los errores hay 92 casos de falsos negativos, 251 casos de falsos positivos y un AUC de 0.82 lo que no significa que clasifica medianamente los verdaderos positivos. En la gráfica de frontera de decisión se compara los datos de pruebas con las predicciones, determinando que los datos no se

clasifican adecuadamente, lo que corrobora cuando se observa las clasificaciones durante el entrenamiento que no se clasifican tampoco muy bien.

Figura 30.

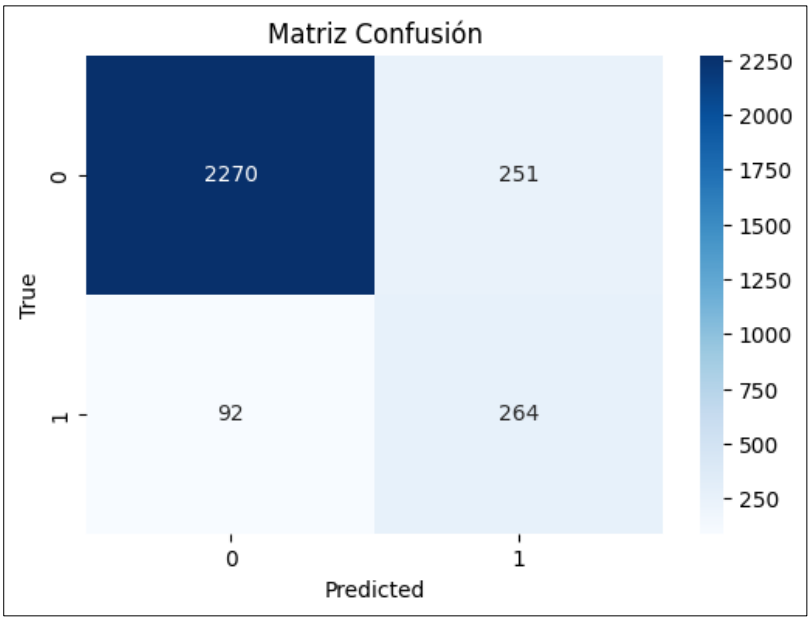
Reporte de matriz de clasificación del modelo entrenado con Regresión Logística

```
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.90      0.93      2521
           1       0.51      0.74      0.61       356

    accuracy                           0.88      2877
   macro avg       0.74      0.82      0.77      2877
weighted avg       0.91      0.88      0.89      2877
```

Nota: La figura muestra un accuracy de 0.88, con una precisión 0.96 de la clase 0 y precisión 0.51 de la clase 1 del modelo entrenado con Regresión Logística. Obtención: Elaboración propia

Figura 31.

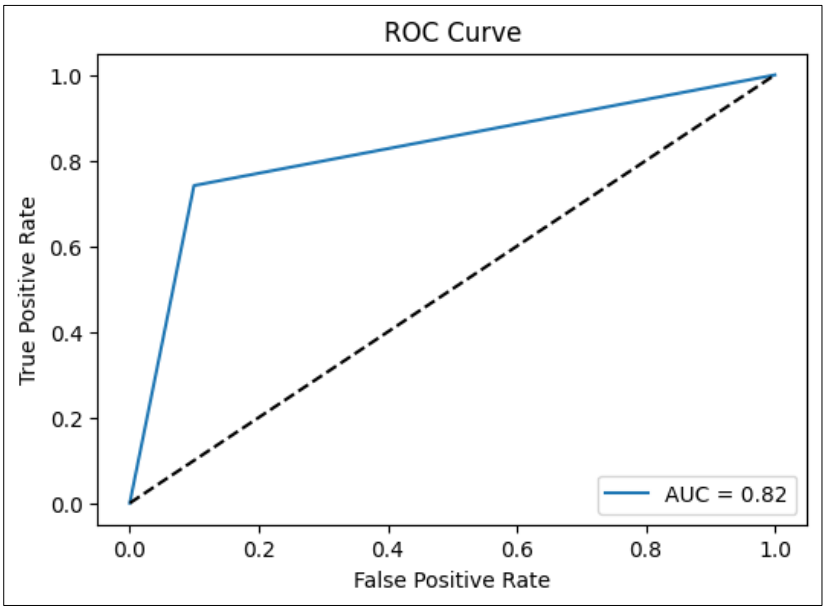
Matriz de confusión del modelo entrenado con Regresión Logística



Nota: La figura los verdaderos positivos, verdaderos negativos, falsos positivos, falsos negativos del modelo entrenado con Regresión Logística. Obtención: Elaboración propia

Figura 32.

Curva ROC del modelo entrenado con Regresión Logística

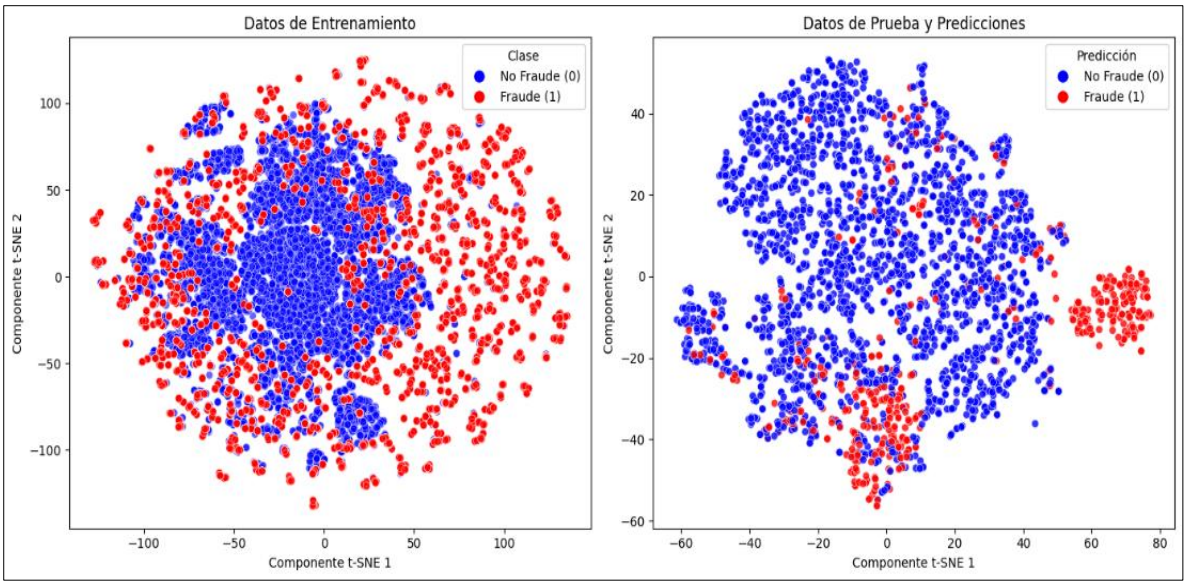


Nota: La figura muestra un valor de AUC de 0.82 del modelo entrenado con Regresión Logística.

Obtención: Elaboración propia

Figura 33.

Frontera de Decisión en entrenamiento y prueba del modelo entrenado con Regresión Logística



Nota: La figura muestra la frontera de decisión durante el entrenamiento y las pruebas del modelo entrenado con Regresión Logística. Obtención: Elaboración propia

Modelo de Naive Bayes

En la Tabla 9 se describen las condiciones de preprocesamiento de los datos escalados, codificados, las condiciones de entrenamiento con los mejores hiperparámetros y mejor score con RandomSearchCV y Validación cruzada. Se detallan las métricas obtenidas de clasificación con el modelo de Naive Bayes.

Tabla 9.

Resultados de métricas del modelo entrenado con Naive Bayes

Escalador y Codificador	Mejores hiperparámetros y Mejor score	Métricas	Tasas y Falsos
Datos sin escalar Label encoder Datos sin outliers	Mejores hiperparámetros encontrados: {'var_smoothing': 0.0001519911082952933, 'priors': None} Mejor score obtenido: 0.8627	Accuracy: 0.9440 Precision: 0.8018 Recall: 0.7275 F1-score: 0.7628 Specificity: 0.9746 AUC: 0.8510	Tasa de falsos negativos: 0.2724 Tasa de falsos positivos: 0.0253 Falsos negativos: 97 Falsos positivos: 64

Nota: La tabla muestra los resultados de los mejores hiperparámetros y métricas del modelo entrenado con Naive Bayes. Obtención: Elaboración propia

En las Figura 34, Figura 35, Figura 36 y Figura 37, se observa que el modelo tuvo una precisión de 0.96 al predecir la clase 0 y una precisión de 0.80 al predecir la clase 1 por eso tiene un accuracy de 0.94. En la matriz de confusión se distingue en los aciertos 2457 verdaderos negativos, 259 verdaderos positivos, mientras que en los errores hay 97 casos de falsos negativos, 64 casos de falsos positivos y un AUC de 0.85 lo que significa que clasifica medianamente los verdaderos positivos. En la gráfica de frontera de decisión se compara los datos de pruebas con las predicciones, determinando que los datos no se clasifican adecuadamente, lo que corrobora cuando se observa las clasificaciones durante el entrenamiento que no se clasifican tampoco muy bien.

Figura 34.

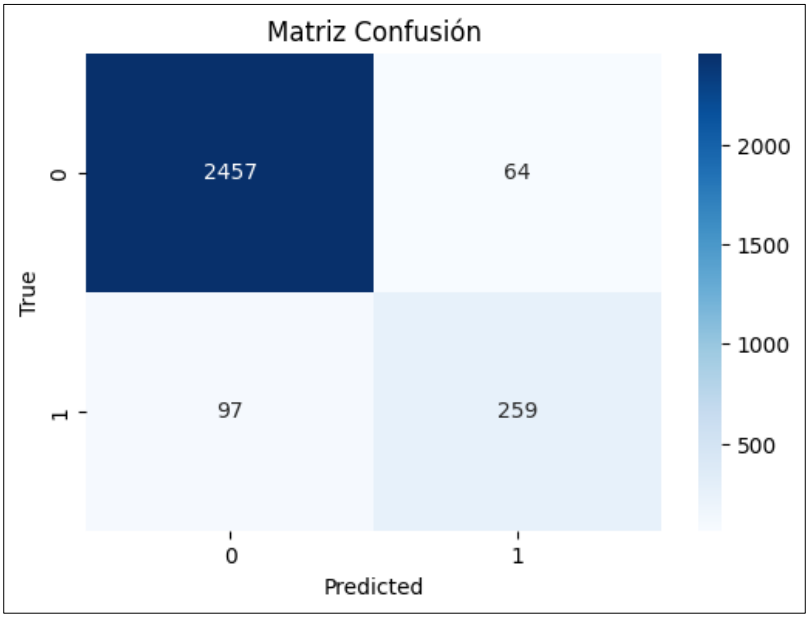
Reporte de matriz de clasificación del modelo entrenado con Naive Bayes

```
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.97      0.97      2521
           1       0.80      0.73      0.76       356

    accuracy                           0.94      2877
   macro avg       0.88      0.85      0.87      2877
weighted avg       0.94      0.94      0.94      2877
```

Nota: La figura muestra un accuracy de 0.88, con una precisión 0.96 de la clase 0 y precisión 0.51 de la clase 1 del modelo entrenado con Naive Bayes. Obtención: Elaboración propia

Figura 35.

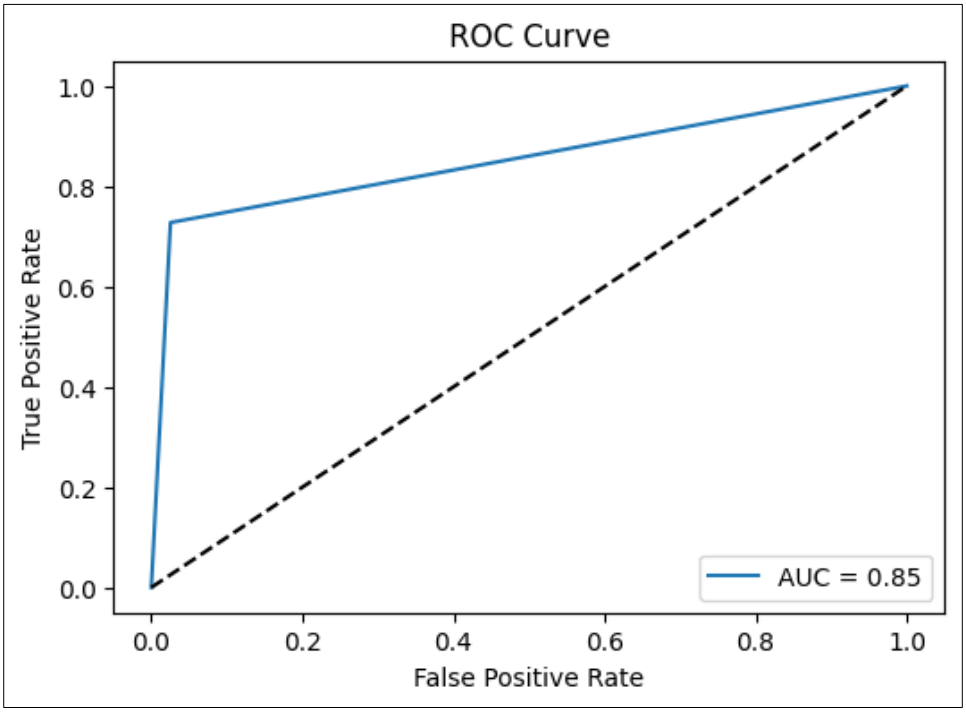
Matriz de confusión del modelo entrenado con Naive Bayes



Nota: La figura muestra los verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos del modelo entrenado con Naive Bayes. Obtención: Elaboración propia

Figura 36.

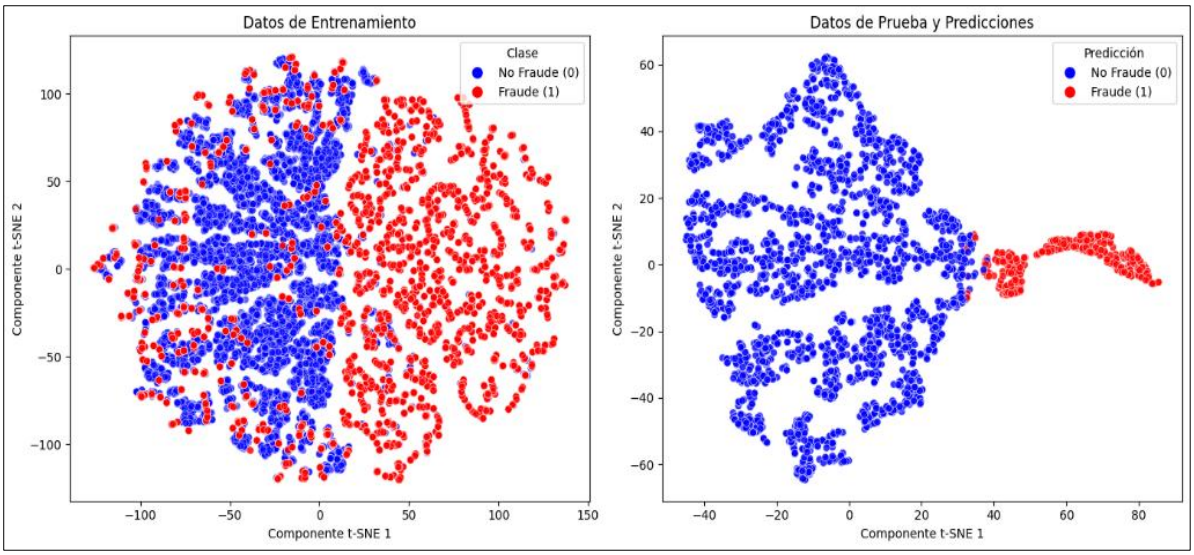
Curva ROC del modelo entrenado con Nayve Bayes



Nota: La figura muestra un valor de AUC de 0.85 del modelo entrenado con Nayve Bayes. Obtención: Elaboración propia

Figura 37.

Frontera de Decisión en entrenamiento y prueba del modelo entrenado con Naive Bayes



Nota: La figura muestra la frontera de decisión en entrenamiento y pruebas del modelo entrenado con Naive Bayes. Obtención: Elaboración propia

Modelo de Soporte de Máquina Vectorial

En la Tabla 10 se describen las condiciones de preprocesamiento de los datos escalados, codificados, las condiciones de entrenamiento con los mejores hiperparámetros y mejor score con RandomSearchCV y Validación cruzada. Se detallan las métricas obtenidas de clasificación con el modelo de Soporte de Máquina Vectorial.

Tabla 10.

Resultados de métricas del modelo entrenado con SVM

Escalador y Codificador	Mejores hiperparámetros y Mejor score	Métricas	Tasas y Falsos
Standar scaler Label encoder Datos sin outliers	Mejores hiperparámetros encontrados: {'tol': 0.001, 'kernel': 'rbf', 'gamma': 'scale', 'degree': 2, 'coef0': 0, 'C': 100} Mejor score obtenido: 0.9801	Accuracy: 0.9454 Precision: 0.7544 Recall: 0.8286 F1-score: 0.7898 Specificity: 0.9619 AUC: 0.8952	Tasa de falsos negativos: 0.1713 Tasa de falsos positivos: 0.0380 Falsos negativos: 61 Falsos positivos: 96

Nota: La tabla muestra los resultados de los mejores hiperparámetros y métricas del modelo entrenado con SVM. Obtención: Elaboración propia

En las Figura 38, Figura 39, Figura 40 y Figura 41, se observa que el modelo tuvo una precisión de 0.98 al predecir la clase 0 y una precisión de 0.75 al predecir la clase 1 por eso tiene un accuracy de 0.95. En la matriz de confusión se distingue en los aciertos 2425 verdaderos negativos, 295 verdaderos positivos, mientras que en los errores hay 61 casos de falsos negativos, 96 casos de falsos positivos y un AUC de 0.90 lo que significa que clasifica mejor los verdaderos positivos. En la gráfica de frontera de decisión se compara los datos de pruebas con las predicciones, determinando que los datos no se clasifican adecuadamente, lo que corrobora cuando se observa las clasificaciones durante el entrenamiento que no se clasifican tampoco muy bien.

Figura 38.

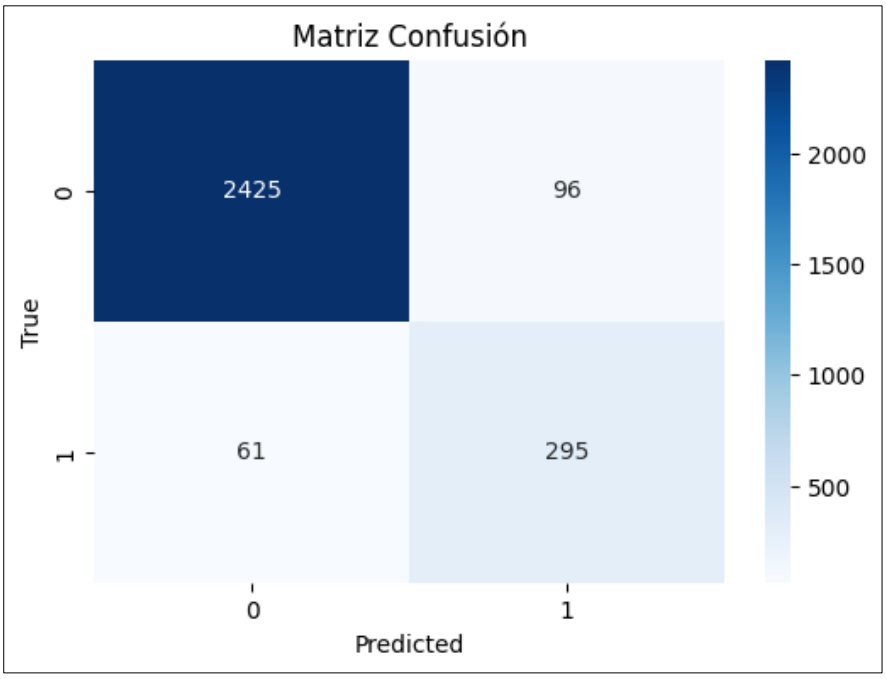
Reporte de matriz de clasificación del modelo entrenado con SVM

```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.96      0.97      2521
           1       0.75      0.83      0.79       356

    accuracy                           0.95      2877
   macro avg       0.86      0.90      0.88      2877
weighted avg       0.95      0.95      0.95      2877
```

Nota: La figura muestra un accuracy 0.95, con precisión 0.98 en Clase 0 y precisión 0.75 en Clase 1 del modelo entrenado con SVM. Obtención: Elaboración propia

Figura 39.

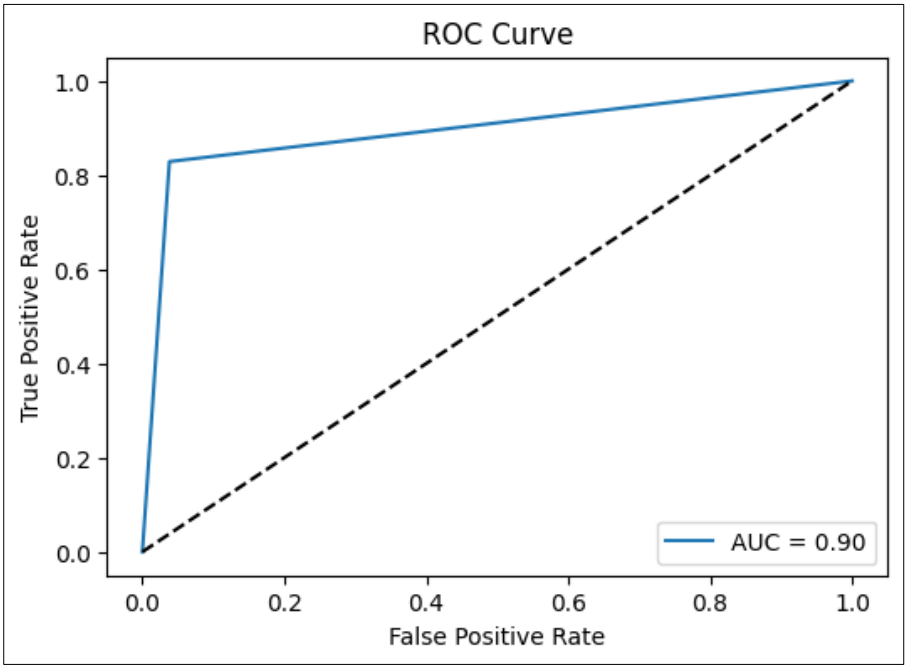
Matriz de confusión del modelo entrenado con SVM



Nota: La figura muestra los verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos del modelo entrenado con SVM. Obtención: Elaboración propia

Figura 40.

Curva ROC del modelo entrenado con SVM

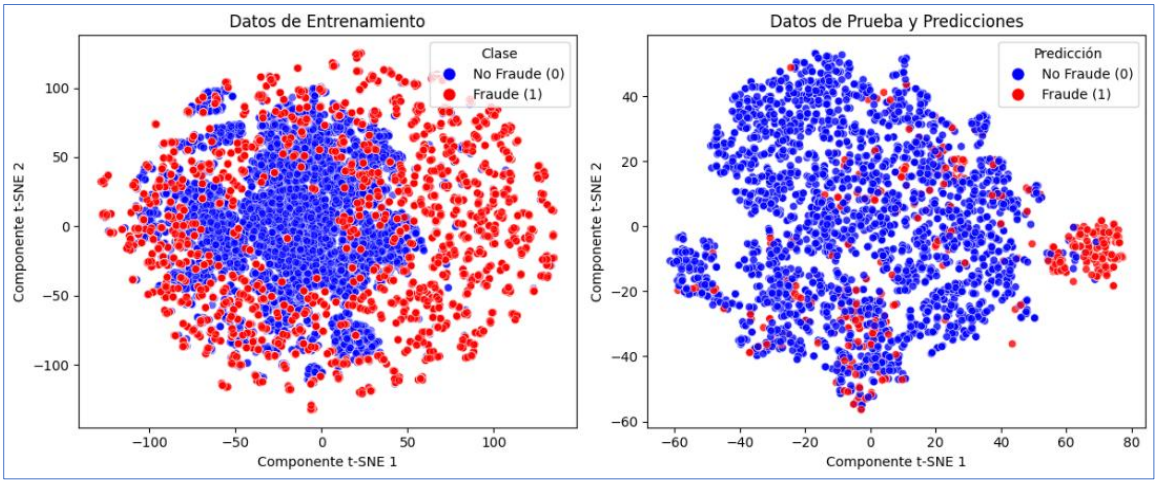


Nota: La figura muestra un valor de AUC 0.90 del modelo entrenado con SVM. Obtención:

Elaboración propia

Figura 41.

Frontera de decisión del modelo entrenado con SVM



Nota: La figura la frontera de decisión en entrenamiento y pruebas del modelo entrenado con SVM.

Obtención: Elaboración propia

Modelo de KNN (K-Nearest Neighbors)

En la Tabla 11 se describen las condiciones de preprocesamiento de los datos escalados, codificados, las condiciones de entrenamiento con los mejores hiperparámetros y mejor score con RandomSearchCV y Validación cruzada. Se detallan las métricas obtenidas de clasificación con el modelo de KNN.

Tabla 11.

Resultados de métricas del modelo entrenado con KNN

Escalador y Codificador	Mejores hiperparámetros y Mejor score	Métricas	Tasas
Standar scaler Label encoder Datos sin outliers	Mejores hiperparámetros encontrados: {'weights': 'uniform', 'p': 1, 'n_neighbors': 3, 'metric': 'manhattan'} Mejor score obtenido: 0.9700	Accuracy: 0.9416 Precision: 0.7216 Recall:0.8595 F1-score: 0.7846 Specificity: 0.9531 AUC: 0.9063	Tasa de falsos negativos: 0.1404 Tasa de falsos positivos: 0.0468 Falsos negativos: 50 Falsos positivos: 118

Nota: La tabla muestra los resultados de los mejores hiperparámetros y métricas del modelo entrenado con KNN. Obtención: Elaboración propia

En las Figura 42, Figura 43, Figura 44 y Figura 45, se observa que el modelo tuvo una precisión de 0.98 al predecir la clase 0 y una precisión de 0.72 al predecir la clase 1 por eso tiene un accuracy de 0.94. En la matriz de confusión se distingue en los aciertos 2403 verdaderos negativos, 306 verdaderos positivos, mientras que en los errores hay 50 casos de falsos negativos, 118 casos de falsos positivos y un AUC de 0.91 lo que significa que clasifica mejor los verdaderos positivos. En la gráfica de frontera de decisión se compara los datos de pruebas con las predicciones, determinando que los datos no se clasifican adecuadamente, lo que corrobora cuando se observa las clasificaciones durante el entrenamiento que no se clasifican tampoco muy bien.

Figura 42.

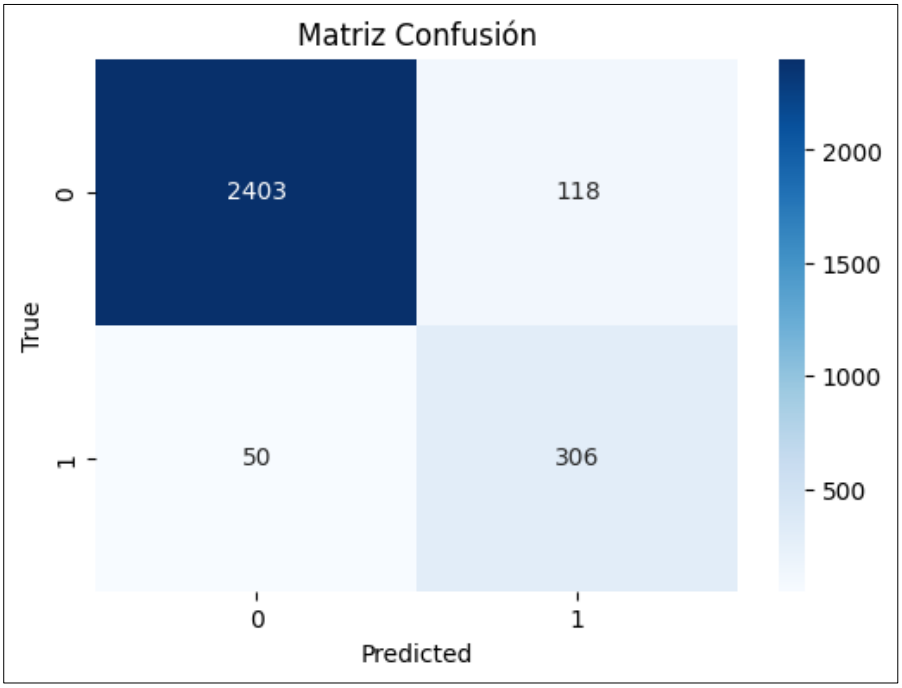
Reporte de matriz de clasificación del modelo entrenado con KNN

```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.95      0.97      2521
           1       0.72      0.86      0.78       356

    accuracy                           0.94      2877
   macro avg       0.85      0.91      0.88      2877
weighted avg       0.95      0.94      0.94      2877
```

Nota: La figura muestra un accuracy 0.94, con precisión 0.98 de Clase 0 y precisión 0.72 de Clase 1 del modelo entrenado con KNN. Obtención: Elaboración propia

Figura 43.

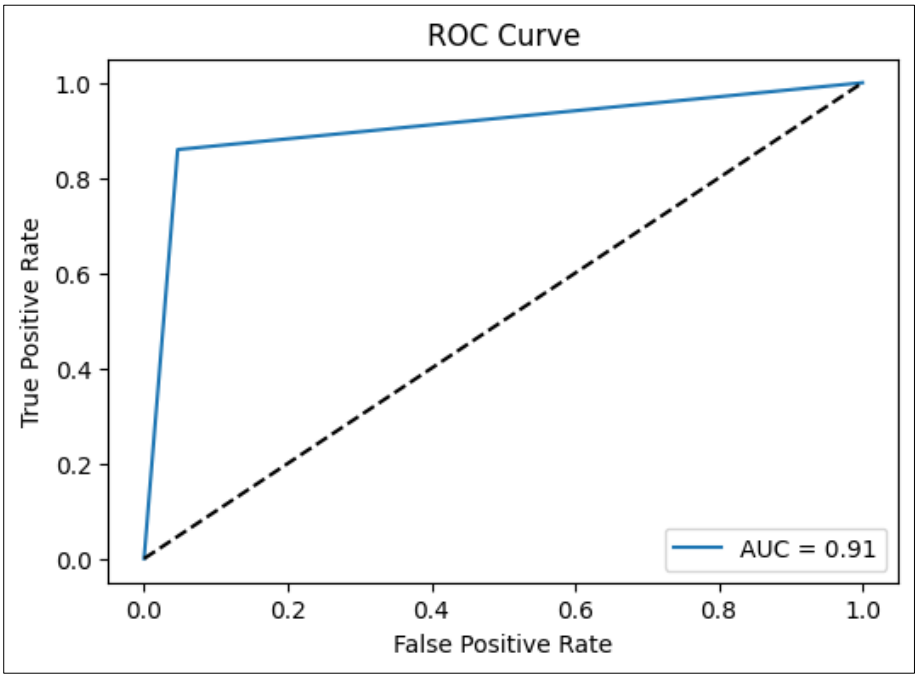
Matriz de confusión del modelo entrenado con KNN



Nota: La figura muestra los verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos del modelo entrenado con KNN. Obtención: Elaboración propia

Figura 44.

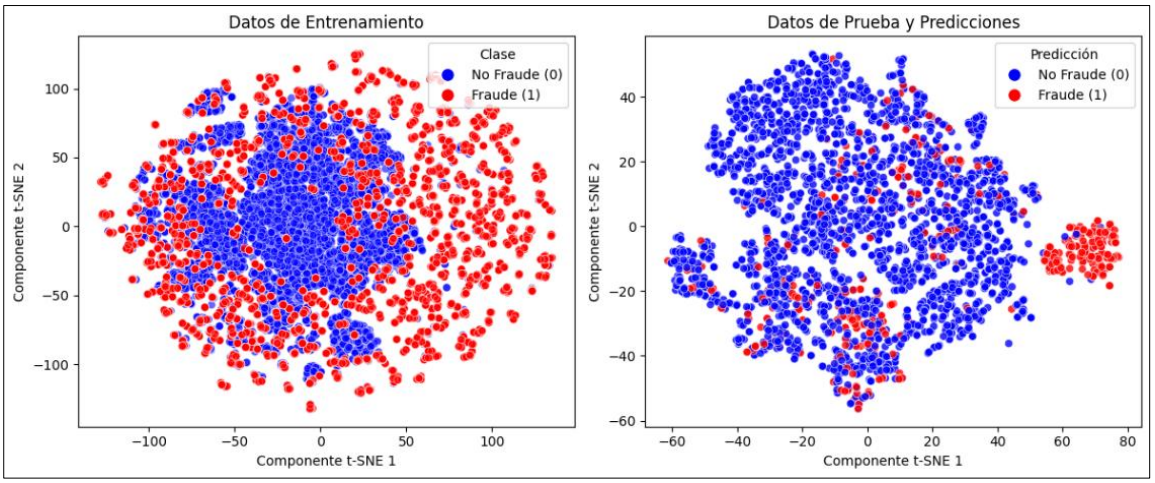
Curva ROC del modelo entrenado con KNN



Nota: La figura muestra un valor AUC 0.91 del modelo entrenado con KNN. Obtención: Elaboración propia

Figura 45.

Frontera de decisión del modelo entrenado con KNN



Nota: La figura muestra la frontera de decisión durante el entrenamiento y pruebas del modelo entrenado con KNN. Obtención: Elaboración propia

Modelo de Árbol de decisión

En la Tabla 12 se describen las condiciones de preprocesamiento de los datos escalados, codificados, las condiciones de entrenamiento con los mejores hiperparámetros y mejor score con RandomSearchCV y Validación cruzada. Se detallan las métricas obtenidas de clasificación con el modelo de Árbol de decisión.

Tabla 12.

Resultados de métricas del modelo entrenado con Árbol de decisión

Escalador y Codificador	Mejores hiperparámetros y Mejor score	Métricas	Tasas
Datos sin escalar Label encoder Datos con outliers	Mejores hiperparámetros: {'criterion': 'entropy', 'max_depth': 11, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 2, 'splitter': 'best'} Mejor score obtenido: 0.9955	Accuracy: 0.9777 Precision: 0.8705 Recall: 0.9634 F1-score: 0.9146 Specificity: 0.9797 AUC: 0.9716	Tasa de falsos negativos: 0.0365 Tasa de falsos positivos: 0.0202 Falsos negativos: 13 Falsos positivos: 51

Nota: La tabla muestra los resultados de los mejores hiperparámetros y métricas del modelo entrenado con Árbol de decisión. Obtención: Elaboración propia

En las Figura 46, Figura 47, Figura 48 y Figura 49, se observa que el modelo tuvo una precisión de 0.99 al predecir la clase 0 y una precisión de 0.87 al predecir la clase 1 por eso tiene un accuracy de 0.98. En la matriz de confusión se distingue en los aciertos 2470 verdaderos negativos, 343 verdaderos positivos, mientras que en los errores hay 13 casos de falsos negativos, 51 casos de falsos positivos y un AUC de 0.97 lo que significa que clasifica mejor los verdaderos positivos. En la gráfica de frontera de decisión se compara los datos de pruebas con las predicciones, determinando que los datos se clasifican un poco mejor, lo que corrobora cuando se observa las clasificaciones durante el entrenamiento que se clasifican mejor.

Figura 46.

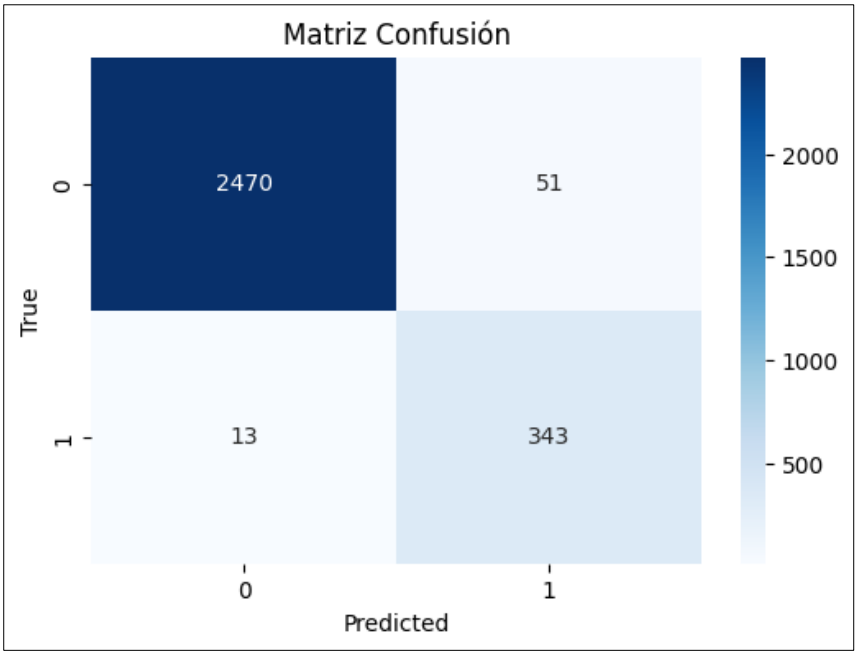
Reporte de matriz de clasificación del modelo entrenado con Árbol de decisión

Classification Report:	precision	recall	f1-score	support
0	0.99	0.98	0.99	2521
1	0.87	0.96	0.91	356
accuracy			0.98	2877
macro avg	0.93	0.97	0.95	2877
weighted avg	0.98	0.98	0.98	2877

Nota: La figura muestra un accuracy 0.98, con precisión 0.99 de Clase 0 y precisión de Clase 1 del modelo entrenado con Árbol de decisión. Obtención: Elaboración propia

Figura 47.

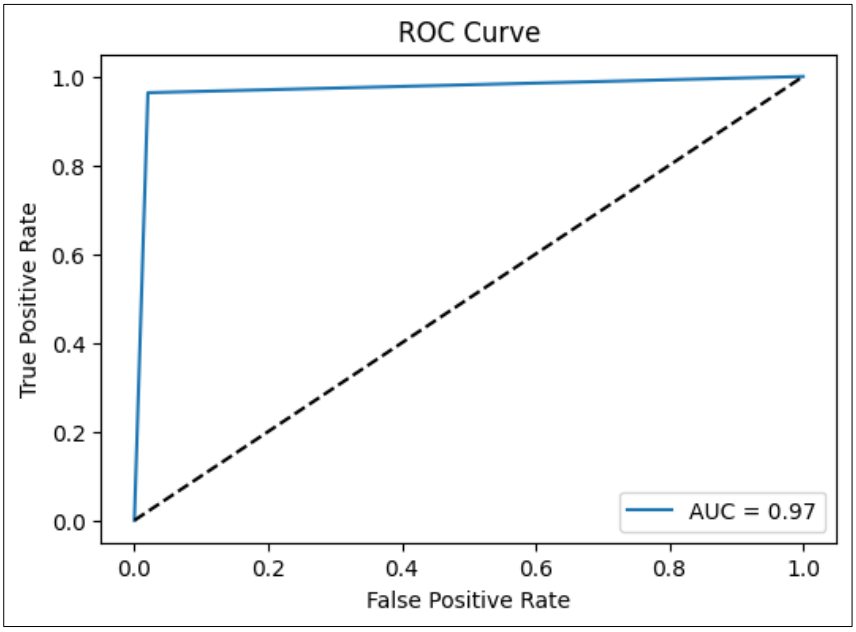
Matriz de confusión del modelo entrenado con Árbol de decisión



Nota: La figura muestra los verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos del modelo entrenado con Árbol de decisión. Obtención: Elaboración propia

Figura 48.

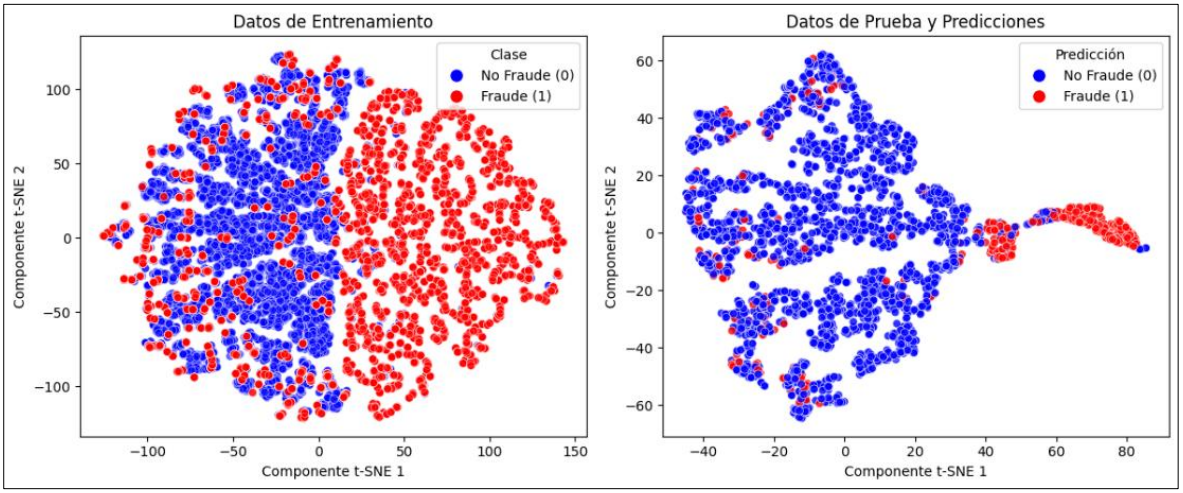
Curva ROC del modelo entrenado con Árbol de decisión



Nota: La figura muestra un valor AUC 0.97 del modelo entrenado con Árbol de decisión. Obtención: Elaboración propia

Figura 49.

Frontera de decisión del modelo entrenado con Árbol de decisión



Nota: La figura muestra la frontera de decisión en entrenamiento y prueba del modelo entrenado con Árbol de decisión. Obtención: Elaboración propia

Modelo de Random Forest

En la Tabla 13 se describen las condiciones de preprocesamiento de los datos escalados, codificados, las condiciones de entrenamiento con los mejores hiperparámetros y mejor score con RandomSearchCV y Validación cruzada. Se detallan las métricas obtenidas de clasificación con el modelo de Random Forest.

Tabla 13.

Resultados de métricas del modelo entrenado con Random Forest

Escalador y Codificador	Mejores hiperparámetros y Mejor score	Métricas	Tasas
Datos sin escalar Label encoder Datos con outliers	Mejores hiperparámetros: {'criterion': 'entropy', 'max_depth': 12, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 9, 'n_estimators': 200} Mejor score obtenido: 0.9996	Accuracy: 0.9878 Precision: 0.9572 Recall: 0.9438 F1-score: 0.9504 Specificity: 0.9940 AUC: 0.9689	Tasa de falsos negativos: 0.0561 Tasa de falsos positivos: 0.0059 Falsos negativos: 20 Falsos positivos: 15

Nota: La tabla muestra los resultados de los mejores hiperparámetros y métricas del modelo entrenado con Random Forest. Obtención: Elaboración propia

En las Figura 50, Figura 51, Figura 52 y Figura 53, se observa que el modelo tuvo una precisión de 0.99 al predecir la clase 0 y una precisión de 0.96 al predecir la clase 1 por eso tiene un accuracy de 0.99. En la matriz de confusión se distingue en los aciertos 2506 verdaderos negativos, 336 verdaderos positivos, mientras que en los errores hay 20 casos de falsos negativos, 15 casos de falsos positivos y un AUC de 0.97 lo que significa que clasifica mejor los verdaderos positivos. En la gráfica de frontera de decisión se compara los datos de pruebas con las predicciones, determinando que los datos se clasifican un poco mejor, lo que corrobora cuando se observa las clasificaciones durante el entrenamiento que se clasifican mejor.

Figura 50.

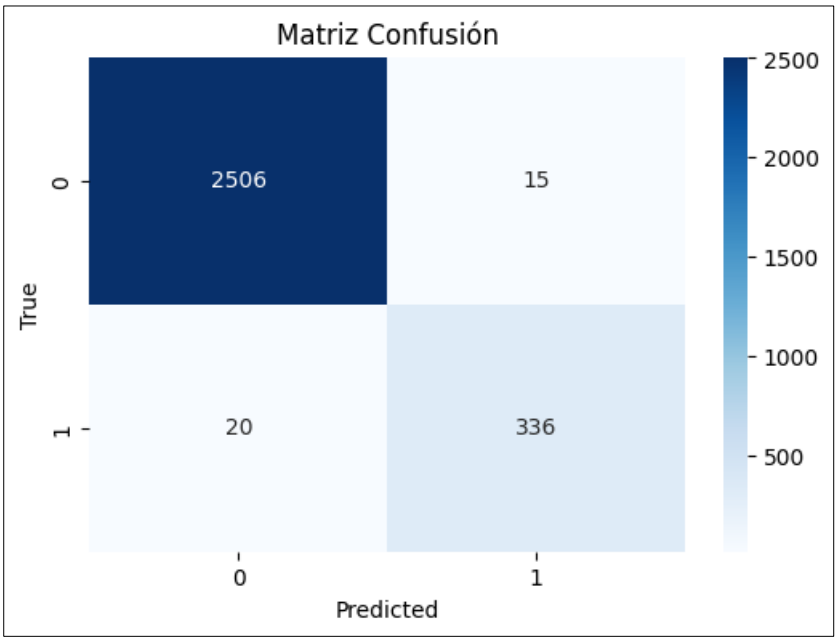
Reporte de matriz de clasificación del modelo entrenado con Random Forest

```
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      2521
           1       0.96      0.94      0.95       356

    accuracy                           0.99      2877
   macro avg       0.97      0.97      0.97      2877
weighted avg       0.99      0.99      0.99      2877
```

Nota: La figura muestra un accuracy 0.99 con una precisión 0.99 de la Clase 0 y precisión 0.96 de la Clase 1 del modelo entrenado con Random Forest. Obtención: Elaboración propia

Figura 51.

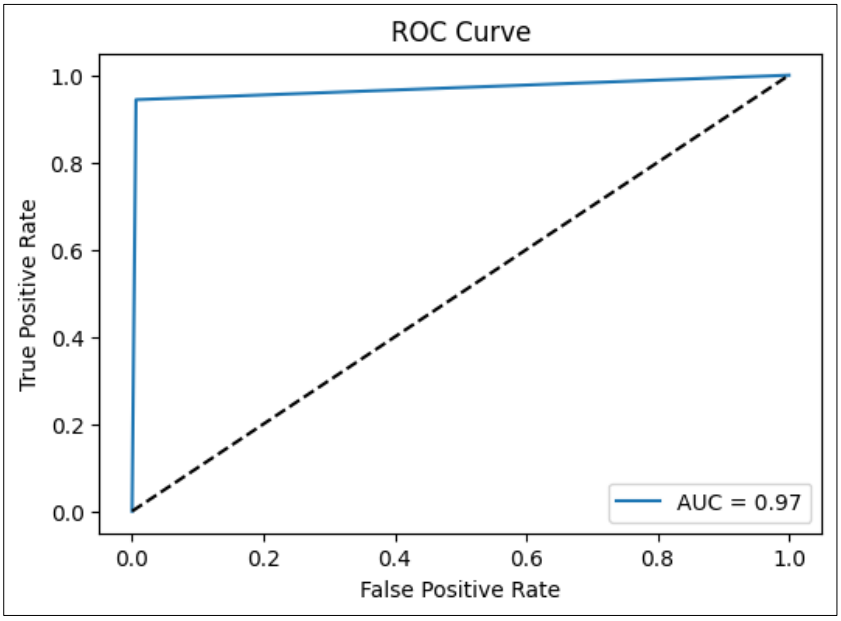
Matriz de confusión del modelo entrenado con Random Forest



Nota: La figura muestra los verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos del modelo entrenado con Random Forest. Obtención: Elaboración propia

Figura 52.

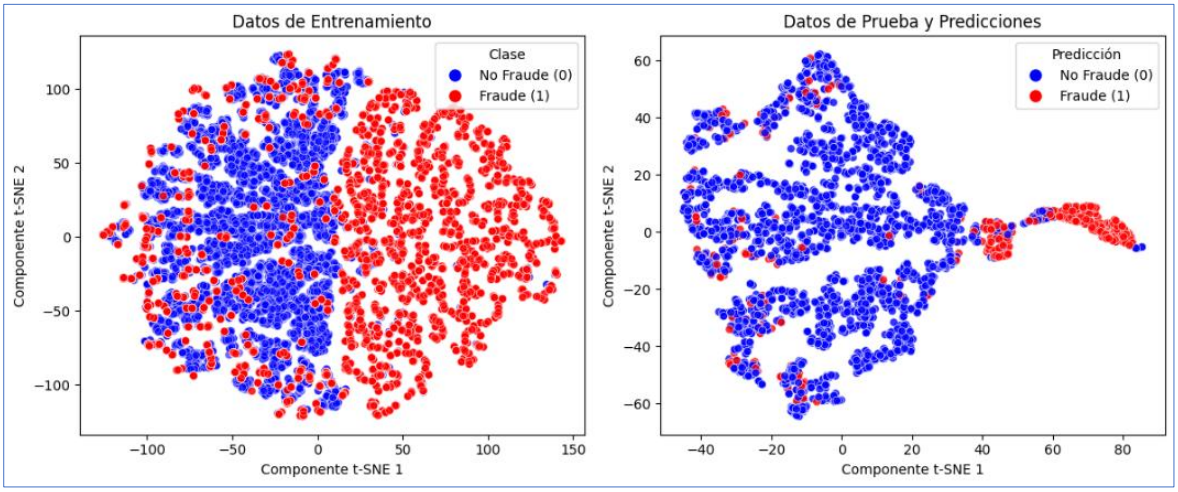
Curva ROC del modelo entrenado con Random Forest



Nota: La figura muestra un valor AUC 0.97 del modelo entrenado con Random Forest. Obtención: Elaboración propia

Figura 53.

Frontera de decisión del modelo entrenado con Random Forest



Nota: La figura muestra la frontera de decisión en entrenamiento y pruebas del modelo entrenado con Random Forest. Obtención: Elaboración propia

Modelo de Gradient Boosting

En la Tabla 14 se describen las condiciones de preprocesamiento de los datos escalados, codificados, las condiciones de entrenamiento con los mejores hiperparámetros y mejor score con RandomSearchCV y Validación cruzada. Se detallan las métricas obtenidas de clasificación con el modelo de Gradient Boosting

Tabla 14.

Resultados de métricas del modelo entrenado con Gradient Boosting

Escalador y Codificador	Mejores hiperparámetros y Mejor score	Métricas	Tasas
Datos sin escalar Label encoder Datos con outliers	Mejores hiperparámetros encontrados: {'subsample': 0.8, 'n_estimators': 300, 'min_samples_split': 5, 'min_samples_leaf': 4, 'max_depth': 4, 'loss': 'log_loss', 'learning_rate': 0.2} Mejor score obtenido: 0.9955	Accuracy: 0.9895 Precision: 0.9630 Recall: 0.9522 F1-score: 0.9576 Specificity: 0.9948 AUC: 0.9735	Tasa de falsos negativos: 0.0477 Tasa de falsos positivos: 0.0051 Falsos negativos: 17 Falsos positivos: 13

Nota: La tabla muestra los resultados de los mejores hiperparámetros y *métricas del modelo entrenado con Gradient Boosting. Obtención: Elaboración propia*

En las Figura 54, Figura 55, Figura 56 y Figura 57, se observa que el modelo tuvo una precisión de 0.99 al predecir la clase 0 y una precisión de 0.96 al predecir la clase 1 por eso tiene un accuracy de 0.99. En la matriz de confusión se distingue en los aciertos 2508 verdaderos negativos, 339 verdaderos positivos, mientras que en los errores hay 17 casos de falsos negativos, 13 casos de falsos positivos y un AUC de 0.97 lo que significa que clasifica mejor los verdaderos positivos. En la gráfica de frontera de decisión se compara los datos de pruebas con las predicciones, determinando que los datos se clasifican un poco mejor, lo que corrobora cuando se observa las clasificaciones durante el entrenamiento que se clasifican mejor.

Figura 54.

Reporte de matriz de clasificación del modelo entrenado con Gradient Boosting.

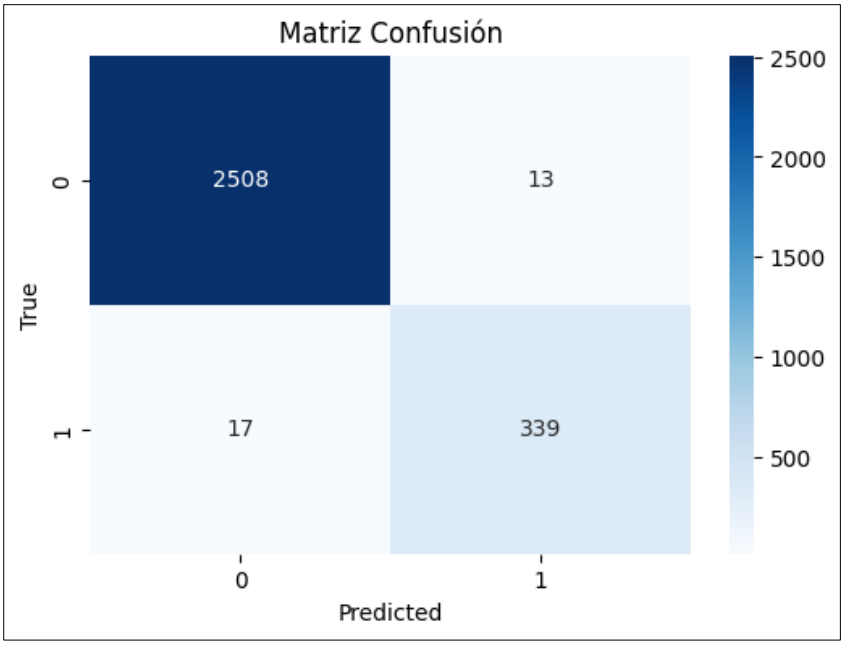
Classification Report:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	2521
1	0.96	0.95	0.96	356
accuracy			0.99	2877
macro avg	0.98	0.97	0.98	2877
weighted avg	0.99	0.99	0.99	2877

Nota: La figura muestra un accuracy de 0.99 con precisión 0.99 de la Clase 0 y precisión 0.96 de la Clase 1 del modelo entrenado con Gradient Boosting. Obtención: Elaboración propia

Figura 55.

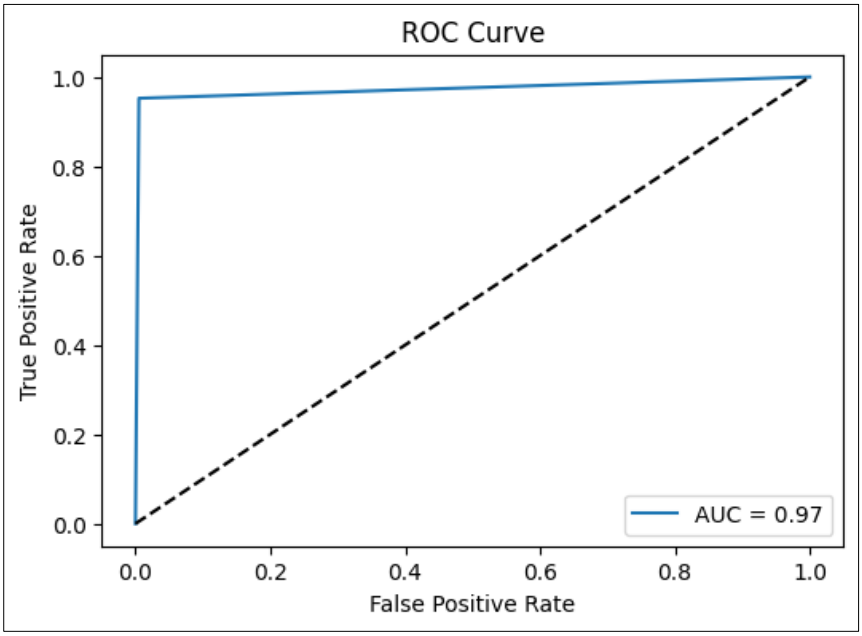
Matriz de confusión del modelo entrenado con Gradient Boosting



Nota: La figura muestra los verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos del modelo entrenado con Gradient Boosting. Obtención: Elaboración propia

Figura 56.

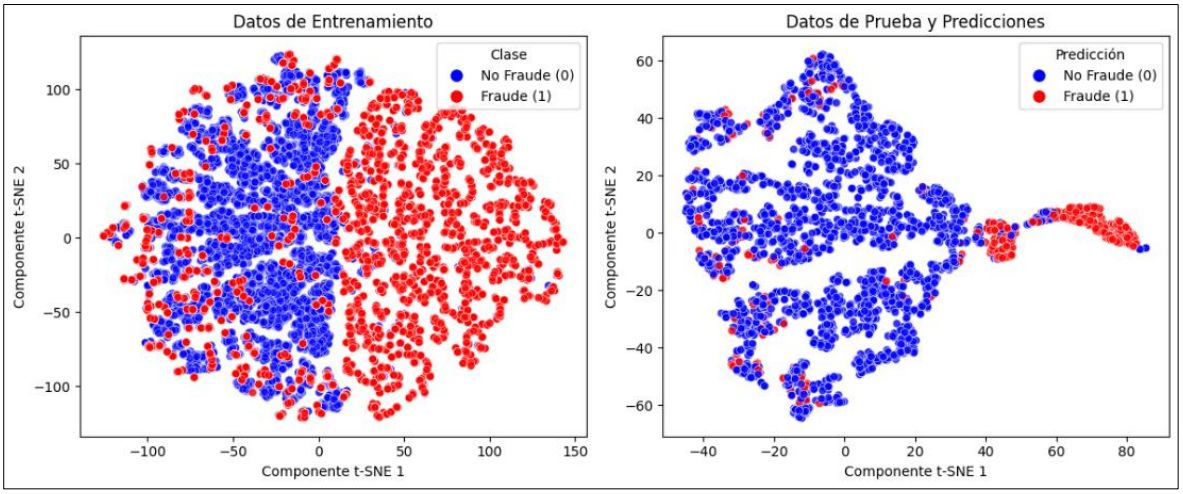
Curva ROC del modelo entrenado con Gradient Boosting



Nota: La figura muestra un valor AUC 0.97 del modelo entrenado con Gradient Boosting. Obtención: Elaboración propia

Figura 57.

Frontera de decisión del modelo entrenado con Gradient Boosting



Nota: La figura muestra la frontera de decisión en entrenamiento y pruebas del modelo entrenado con Gradient Boosting. Obtención: Elaboración propia

Modelo de Red Neuronal MLP (Perceptrón Multicapa)

En la Tabla 15 se describen las condiciones de preprocesamiento de los datos escalados, codificados, las condiciones de entrenamiento con los mejores hiperparámetros y mejor score con RandomSearchCV y Validación cruzada. Se detallan las métricas obtenidas de clasificación con el modelo de MLP.

Tabla 15.

Resultados de métricas del modelo entrenado con red neuronal MLP

Escalador y Codificador	Técnicas de regularización	Métricas	Tasas
Standar scaler Label encoder	L2, Dropout, Early Stopping Train Loss: 0.1129295825958252 Test Loss: 0.1285205483436584 Train Accuracy: 0.9694444537162781 Test Accuracy: 0.9586374759674072	Accuracy: 0.958637469586374 Precision: 0.801526717557251 Recall: 0.884831460674157 F1-score: 0.841121495327102 Specificity: 0.969059896866322 AUC: 0.92694567877024	Tasa de falsos negativos: 0.1151685393258427 Tasa de falsos positivos: 0.0309401031336771 Falsos negativos: 41 Falsos positivos: 78

Nota: La tabla muestra los resultados de los mejores hiperparámetros y métricas del modelo entrenado con red neuronal MLP. Obtención: Elaboración propia

En la Figura 58, Figura 59, Figura 60 y Figura 61, se observa que el modelo tuvo una precisión de 0.99 al predecir la clase 0 y una precisión de 0.96 al predecir la clase 1 por eso tiene un accuracy de 0.99. En la matriz de confusión se distingue en los aciertos 2508 verdaderos negativos, 339 verdaderos positivos, mientras que en los errores hay 17 casos de falsos negativos, 13 casos de falsos positivos y un AUC de 0.97 lo que significa que

clasifica mejor los verdaderos positivos. En la gráfica de frontera de decisión se compara los datos de pruebas con las predicciones, determinando que los datos se clasifican un poco mejor, lo que corrobora cuando se observa las clasificaciones durante el entrenamiento que se clasifican mejor.

Figura 58.

Reporte de matriz de clasificación del modelo entrenado con red neuronal MLP

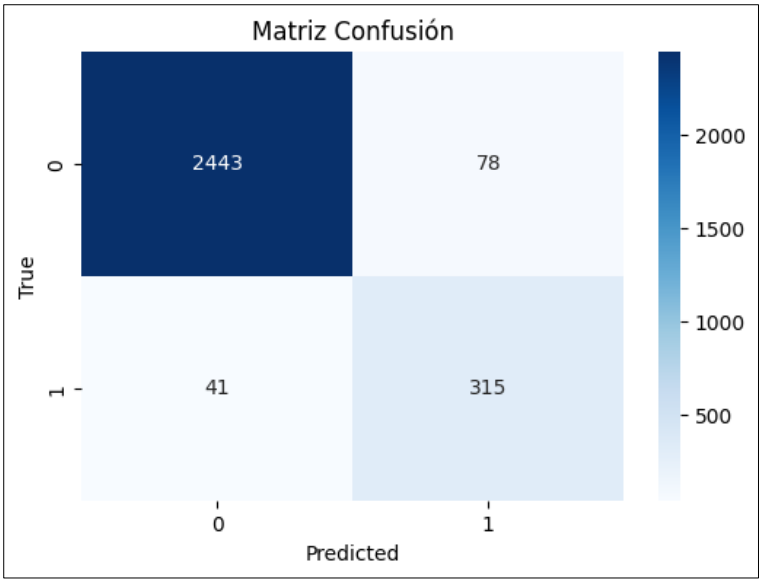
Classification Report:

	precision	recall	f1-score	support
0	0.98	0.97	0.98	2521
1	0.80	0.88	0.84	356
accuracy			0.96	2877
macro avg	0.89	0.93	0.91	2877
weighted avg	0.96	0.96	0.96	2877

Nota: La figura muestra un accuracy 0.96 con precisión 0.98 de Clase 0 y precisión 0.80 de Clase 1 del modelo entrenado con red neuronal MLP. Obtención: Elaboración propia

Figura 59.

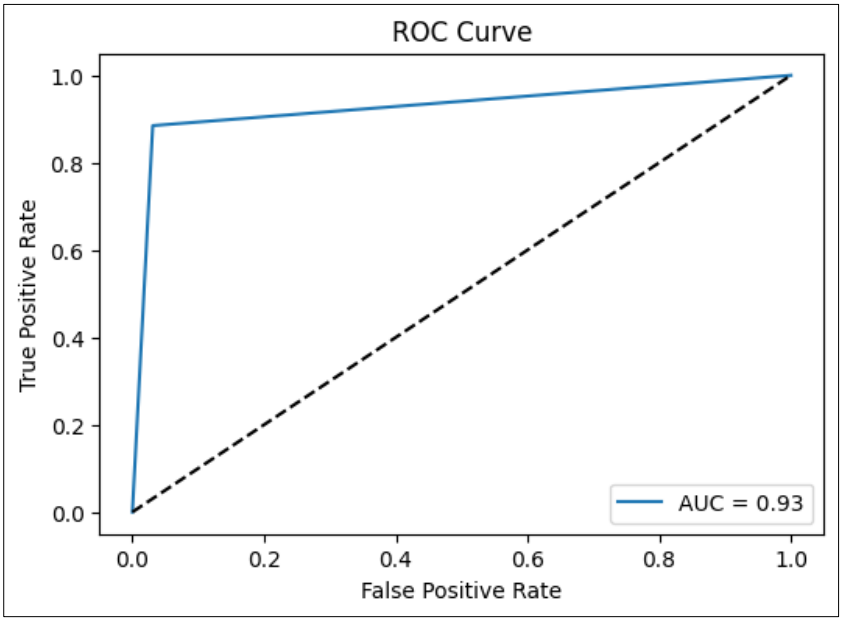
Matriz de confusión del modelo entrenado con red neuronal MLP



Nota: La figura muestra los verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos del modelo entrenado con red neuronal MLP. Obtención: Elaboración propia

Figura 60.

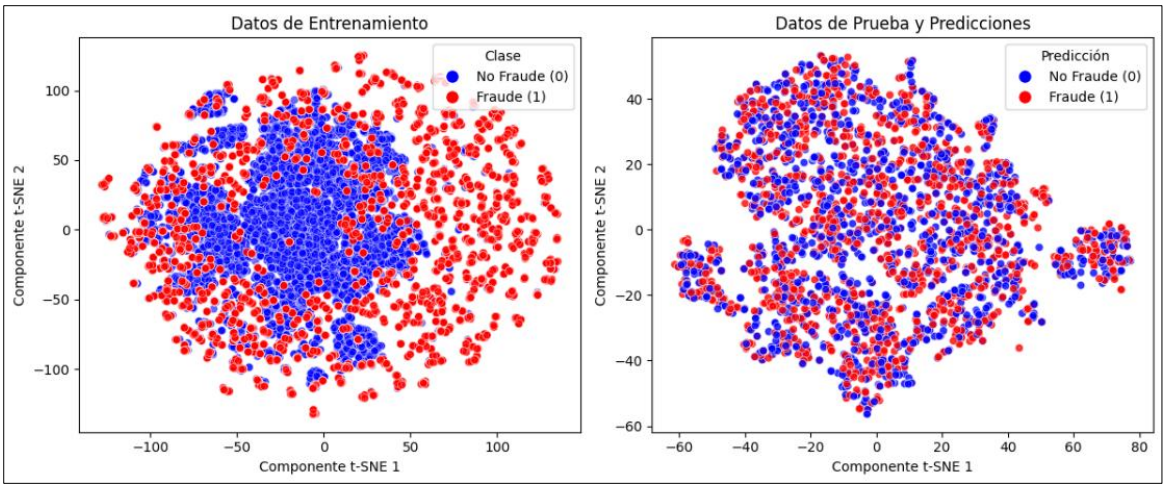
Curva ROC del modelo entrenado con red neuronal MLP



Nota: La figura muestra un valor AUC 0.93 del modelo entrenado con red neuronal MLP. Obtención: Elaboración propia

Figura 61.

Frontera de decisión del modelo entrenado con red neuronal MLP

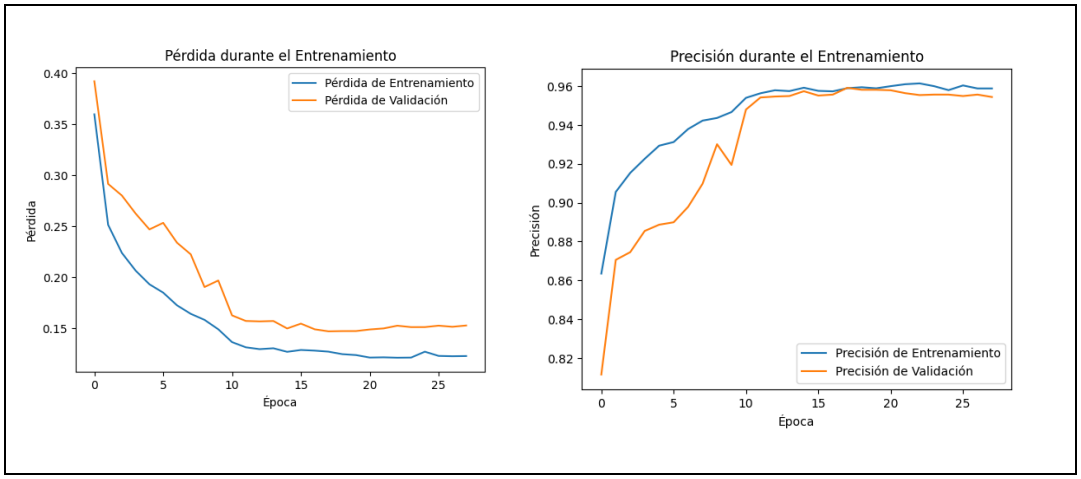


Nota: La figura muestra la frontera de decisión en entrenamiento y pruebas del modelo entrenado con red neuronal MLP. Obtención: Elaboración propia

En la Figura 62 se observa que las curvas de pérdidas y las curvas de precisión durante el entrenamiento y validación no presentan overfitting, ocurriendo una mínima perdida solamente durante el entrenamiento.

Figura 62.

Curvas de perdida y Curvas de precisión en entrenamiento de red MLP



Nota: La figura muestra Curvas de perdida y las Curvas de precisión en entrenamiento de red MLP.

Obtención: Elaboración propia

3.3 Explicabilidad de los mejores modelos

Una vez realizado el análisis de los 8 modelos se procede a escoger los mejores modelos para establecer una predicción con la moda, los modelos seleccionados son:

- Árbol de decisión
- Random Forest
- Gradient Boosting

Estos modelos fueron seleccionados por presentar los más altos valores de acurracy, recall y auc, además de obtener los valores más bajos de falsos negativos, siendo esto crítico al momento de poner en producción, es decir al desplegar en servidores de los comercios e instituciones financieras para aminorar las pérdidas económicas por predicciones realizadas con menos errores.

En la tabla 16 se describe las métricas de los mejores modelos obtenidos en el entrenamiento en la detección de fraudes de tarjetas de crédito.

Tabla 16.

Métricas de los mejores modelos obtenidos

Modelo	Métricas	Tasa y Falsos
Árbol de decisión	**Accuracy: 0.9777** Precision: 0.8705 **Recall: 0.9634** F1-score: 0.9146 Specificity: 0.9797 **AUC: 0.9716**	Tasa de falsos negativos: 0.0365 Tasa de falsos positivos: 0.0202 **Falsos negativos: 13** Falsos positivos: 51
Random Forest	**Accuracy: 0.9878** Precision: 0.9572 **Recall: 0.9438** F1-score: 0.9504 Specificity: 0.9940 **AUC: 0.9689**	Tasa de falsos negativos: 0.0561 Tasa de falsos positivos: 0.0059 **Falsos negativos: 20** Falsos positivos: 15
Gradient Boosting	**Accuracy: 0.9895** Precision: 0.9630 **Recall: 0.9522** F1-score: 0.9576 Specificity: 0.9948 **AUC: 0.9735**	Tasa de falsos negativos: 0.0477 Tasa de falsos positivos: 0.0051 **Falsos negativos: 17** Falsos positivos: 13

Nota: La tabla muestra los resultados de métricas de los mejores modelos entrenados con Árbol de Decisión, Random Forest y Gradient Boosting. Obtención: Elaboración propia

Detallando la explicabilidad de los modelos seleccionados se puede describir lo siguiente:

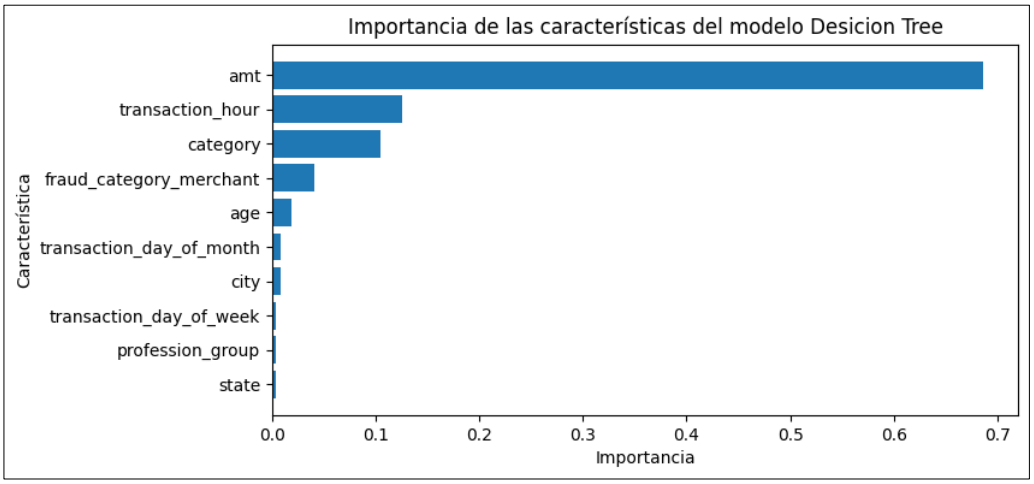
Modelo de Árbol de decisión

En la Figura 63 las variables con mayor importancia que absorbieron mayor conocimiento durante en entrenamiento son: amt, transaction_hour y category mediante el modelo de Árbol de decisión.

En la Figura 64 con la librería SHAP se observa que las variables más significativas son: amt, category y transaction_hour.

Figura 63.

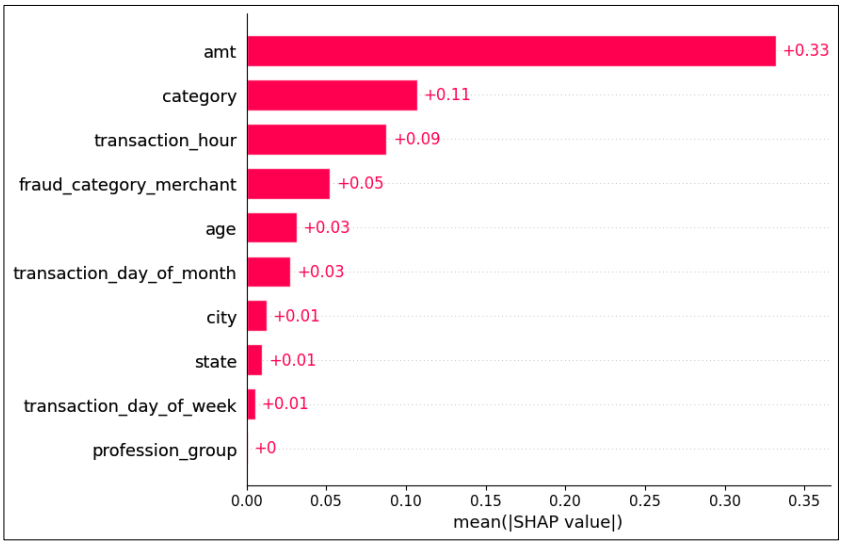
Importancia de las características del modelo Árbol de decisión



Nota: La figura muestra importancia de las variables del entrenamiento con Ärbol de decisión. Obtención: Elaboración propia

Figura 64.

Influencia de las características en la predicción del modelo Árbol de decisión con SHAP



Nota: La figura muestra influencia de las variables del entrenamiento con SHAP del modelo Árbol de decisión. Obtención: Elaboración propia

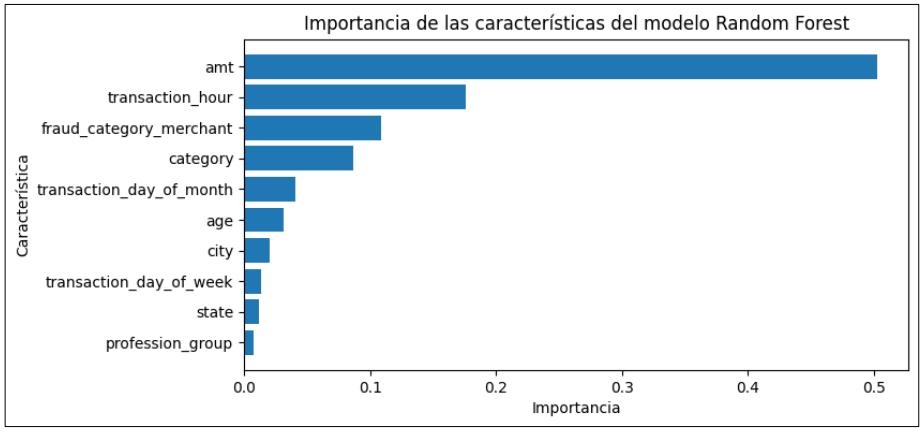
Modelo de Random Forest

En la Figura 65 las variables con mayor importancia que absorbieron mayor conocimiento durante en entrenamiento son: amt, transaction_hour y fraud_category_merchant mediante el modelo de Random Forest.

En la Figura 66 con la librería SHAP se observa que las variables más significativas son: amt, transaction_hour y fraud_category_merchant.

Figura 65.

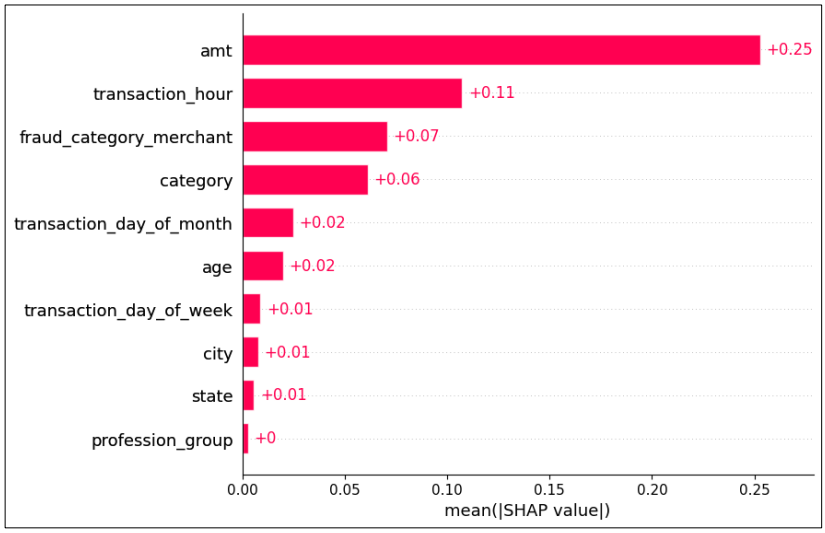
Importancia de las características del modelo Random Forest



Nota: La figura muestra importancia de las variables del entrenamiento con Random Forest. Obtención: Elaboración propia

Figura 66.

Influencia de las características en la predicción del modelo Random Forest con SHAP



Nota: La figura muestra influencia de las variables del entrenamiento con SHAP del modelo Random Forest. Obtención: Elaboración propia

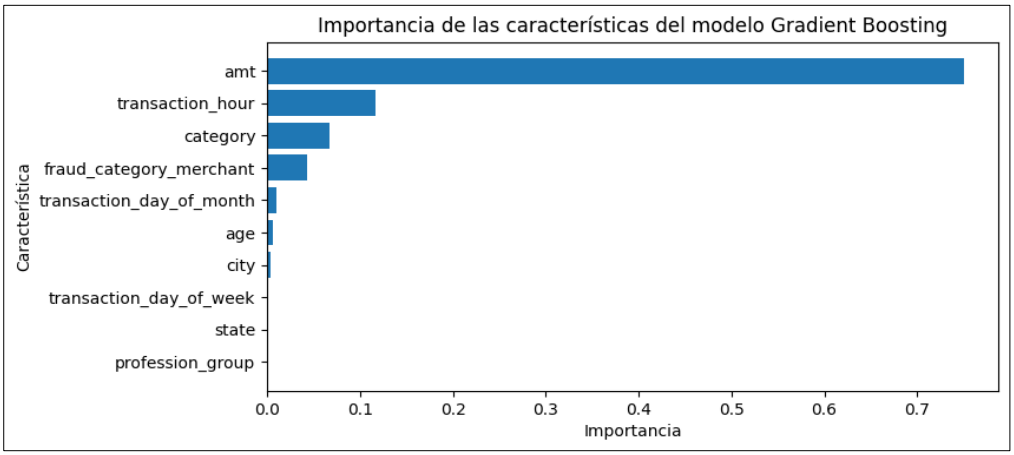
Modelo de Gradient Boosting

En la Figura 67 las variables con mayor importancia que absorbieron mayor conocimiento durante en entrenamiento son: amt, transaction_hour y category mediante el modelo de Gradient Boosting.

En la Figura 68 con la librería SHAP se observa que las variables más significativas son: amt, category y fraud_category_merchant.

Figura 67.

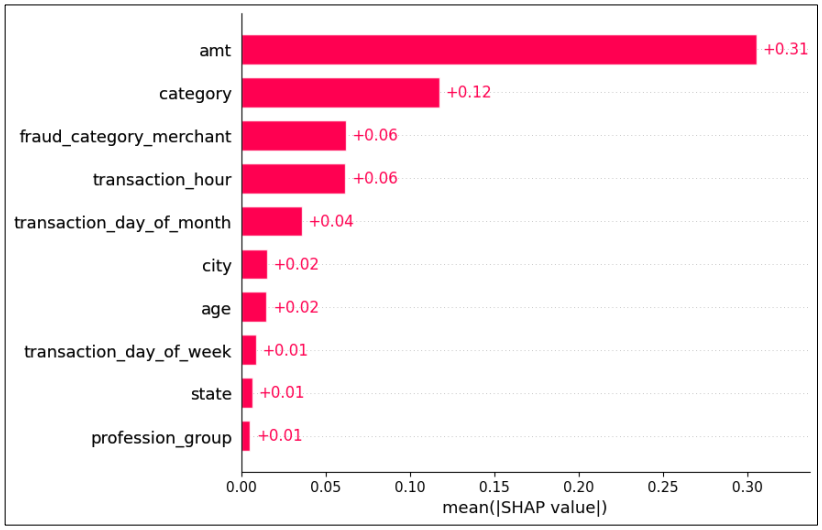
Importancia de las características del modelo Gradient Boosting.



Nota: La figura muestra importancia de las variables del entrenamiento con Gradient Boosting. Obtención: Elaboración propia

Figura 68.

Influencia de las características en la predicción del modelo Gradient Boosting con SHAP



Nota: La figura muestra influencia de las variables del entrenamiento con SHAP del modelo Gradient Boosting. Obtención: Elaboración propia

En la tabla 17 se observa las variables comunes con la importancia de acuerdo con el modelo entrenado de los 3 modelos seleccionados y mediante la librería SHAP. Se puede determinar que las 4 variables más importantes son: amt, transaction_hour, fraud_category_merchant, category.

Tabla 17.

Importancia de características con el árbol y Características significativas con SHAP

Modelo	**Importancia de características en entrenamiento**	**Características significativas con SHAP**
Árbol de decisión	amt, transaction_hour y category	amt, category y transaction_hour.
Random Forest	amt, transaction_hour y fraud_category_merchant	amt, transaction_hour y fraud_category_merchant.
Gradient Boosting	amt, transaction_hour y category	amt, category y fraud_category_merchant.

Nota: La tabla muestra Importancia de características con el árbol de cada modelo y las Características significativas con SHAP. Obtención: Elaboración propia

En el Apéndice E se encuentran las figuras desde la Figura 100 a Figura 106 donde se muestran los árboles obtenidos del entrenamiento de cada algoritmo de árbol de decisión, random forest y gradient boosting que nos permite interpretar las reglas obtenidas en el aprendizaje supervisado.

3.4 Predicción con datos nuevos usando la Aplicación web local

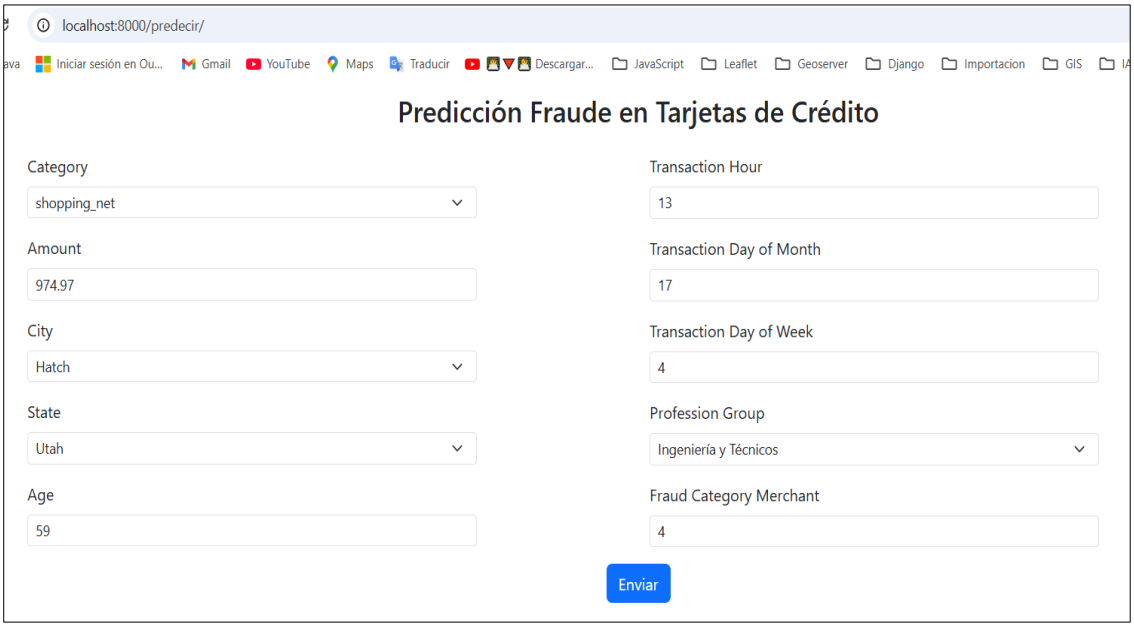
Se procede a realizar la predicción de fraudes con datos nuevos mediante la Aplicación web en un ambiente web desarrollado en un entorno local.

En la Figura 69 se observa el formulario de la aplicación web local donde se ingresan datos nuevos de la categoría del tipo de negocio, monto de la transacción, ciudad,

estado, edad del cliente, hora de la transacción, día del mes de la transacción, transacción de día de la semana, profesión del cliente y nivel de fraude del negocio del comerciante.

Figura 69.

Formulario de la Aplicación web local de predicción de fraude en tarjetas de crédito

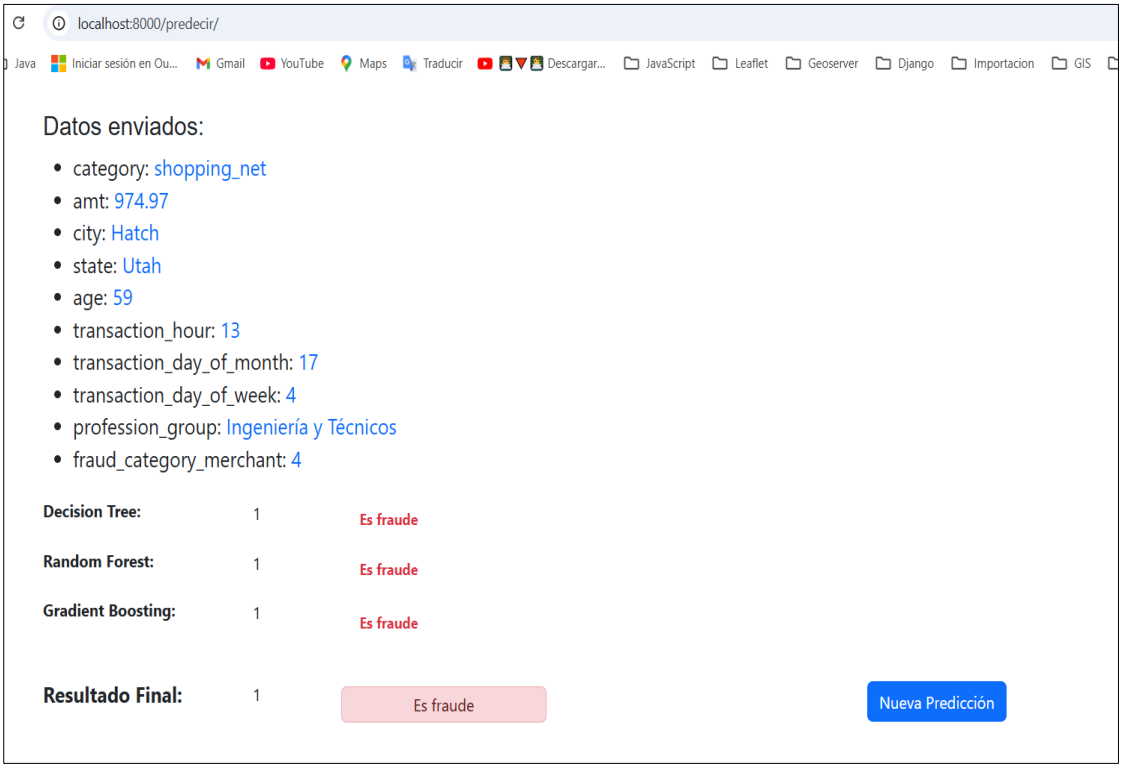


Nota: La figura muestra Formulario de la Aplicación web local de predicción de fraude en tarjetas de crédito. Obtención: Elaboración propia

En la Figura 70 se observa que se realiza la predicción de transacción con fraude de la tarjeta de crédito con los siguientes datos: categoría del negocio shopping_net, monto de transacción $974,97, ciudad Hatch, estado Utah, 59 años, 13h00, 17 día de mes, 4 (viernes), profesión del cliente Ingeniería y Técnicos y 4 nivel de fraude de establecimiento.

Figura 70.

Predicción con Transacción Fraude con Tarjeta de crédito en Aplicación web local

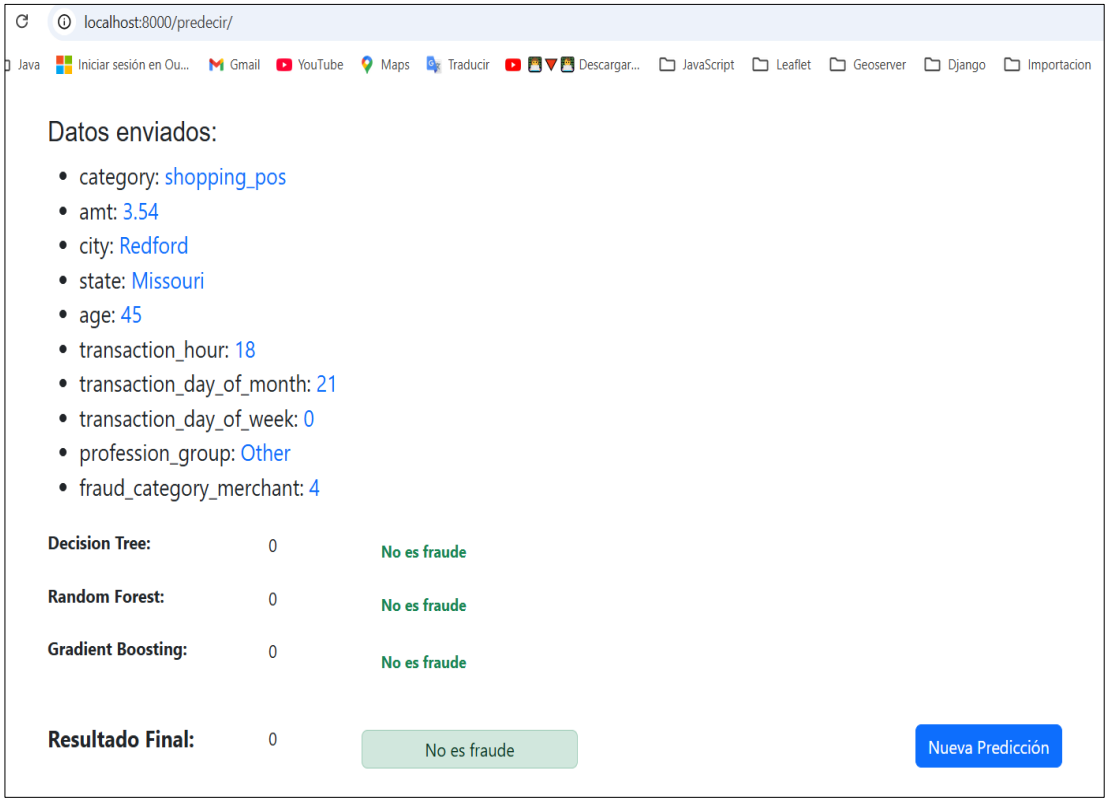


Nota: La figura muestra Predicción con Transacción Fraude con Tarjeta de crédito en Aplicación web local. Obtención: Elaboración propia

En la Figura 71 se observa que se realiza la predicción de transacción con no fraude de la tarjeta de crédito con los siguientes datos: categoría del negocio shopping_pos, monto de transacción $3,54, ciudad Redford, estado Missouri, 45 años, 18h00, 21 día de mes, 0 (lunes), profesión del cliente Other y 4 nivel de fraude de establecimiento.

Figura 71.

Predicción con Transacción No Fraude con Tarjeta de crédito en Aplicación web local



Nota: La figura muestra Predicción con Transacción No Fraude con Tarjeta de crédito en Aplicación web local. Obtención: Elaboración propia

CAPITULO 4

CONCLUSIONES

4. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones generales

Del objetivo general: Desarrollar modelos predictivos de detección de fraudes mediante técnicas de minería de datos para la identificación de transacciones fraudulentas en tarjetas de crédito con alta precisión.

- Se realizó la revisión literaria del estado de arte y marco teórico tomando como referencias artículos indexados en revistas científicas y libros de los últimos principalmente de los últimos 5 años.
- Se utilizó la metodología del proceso KDD siguiendo las fases de selección de datos, preprocesamiento de datos, minería de datos e interpretación y evaluación de datos.
- Se exploró fuentes de repositorio público, seleccionando a la plataforma de datasets científicos de Kaggle, con un dataset almacenado en 2024 con calificación de 10 sobre datos sintéticos de transacciones fraudulentas de Estados Unidos
- Se desarrollaron 8 modelos de aprendizaje supervisado de clasificación entrenados con los algoritmos de Regresión Logística, Naive Bayes, KNN, SVM, Árbol de decisión, Random Forest, Gradient Boosting y Red neuronal MLP.
- Se evaluaron los modelos mediante métricas de clasificación, encontrándose 3 mejores modelos con los algoritmos de Árbol de decisión con accuracy 0.977, AUC 0.961, falsos negativos 13; Random Forest con accuracy 0.987, AUC 0.968, falsos negativos 20 y Gradient Boosting con accuracy 0.989, AUC 0.973, falsos negativos 17.
- Se implementó una aplicación web local para realizar la predicción con datos nuevos, prediciendo tanto para una transacción con fraude y transacción no fraude de tarjeta de crédito.

4.2 Conclusiones específicas

4.2.1. Análisis del cumplimiento de los objetivos de la investigación

Del primer objetivo específico: Analizar un conjunto de datos históricos de transacciones con tarjetas de crédito para identificación de características que permitan distinguir transacciones legítimas de transacciones fraudulentas.

- Se identificó el dataset Credit Card Fraud del repositorio de Kaggle apropiado como fuente de datos para investigaciones de ciencia de datos, el tamaño comprende de 14.446 registros con 14 variables predictoras entre categóricas y numéricas y 1 variable target.
- Se ejecutó la metodología del proceso KDD, en donde el preprocesamiento fue necesario realizar una limpieza, transformación de los datos, tratamiento de los outliers, escalamiento de las variables numéricas, codificación de las variables categóricas y balanceo de los datos para obtener datos con calidad y apropiados para poder realizar el entrenamiento de los modelos según los requisitos de los algoritmos de machine learning y deep learning.

Del segundo objetivo específico: Diseñar modelos de clasificación con técnicas de minería de datos como Machine Learning y Deep Learning para la detección de fraudes con tarjetas de crédito.

- Antes del entrenamiento de los modelos, se realizó la partición de los datos en 80% de train y 20% de test, a los datos de train se realiza el escalamiento de las variables numéricas para que se ajuste a la escala de las observaciones de cada grupo de test y train, codificación de las variables categóricas de train / test, balanceo de datos y tratamiento de outliers del train, dejando los datos de test con datos desbalanceados y outliers que permita reflejar datos del mundo real.
- Se entrenó 8 modelos de aprendizaje supervisado con los mismos datos de train, 7 de ellos con los algoritmos de machine learning como Regresión Logística, Naive

Bayes, KNN, SVM, Árbol de decisión y algoritmos de ensemble como Random Forest y Gradient Boosting. El último modelo se usó al algoritmo de Deep Learning de la red neuronal MLP.

- Se aplicaron a los algoritmos de machine learning, técnicas de optimización RandomSearchCV con validación cruzada para encontrar los mejores hiperparámetros de los modelos y lograr una mejor generalización en la predicción del fraude de tarjeta de crédito.
- Se aplicó en el algoritmo de Deep learning, técnicas de regularización L2, Dropout y Early Stopping para evitar el sobreajuste de la red neuronal aminorando las pérdidas durante el entrenamiento y alcanzar una óptima generalización del modelo en la predicción de fraude de tarjeta de crédito.

Del tercer objetivo específico: Comparar el desempeño de los modelos obtenidos en términos de las métricas de accuracy, precisión, recall, F1-Score, matriz de confusión, falsos positivos y falsos negativos utilizando métricas como la AUC-ROC para selección del mejor modelo.

- Se realizó la comparación de los 8 modelos entrenados de aprendizaje supervisado una vez realizado la predicción con los mismos datos de test, tomando en cuenta las métricas de clasificación como reporte de matriz de clasificación, matriz de confusión, acurracy, precisión, recall (sensibilidad), f1-score, especificidad y curva ROC,
- Se seleccionaron los 3 mejores modelos con los algoritmos de Árbol de decisión con accuracy 0.977, AUC 0.961, falsos negativos 13; Random Forest con accuracy 0.987, AUC 0.968, falsos negativos 20 y Gradient Boosting con accuracy 0.989, AUC 0.973, falsos negativos 17.

Del cuarto objetivo específico: Desarrollar una API como medio de comunicación entre la interfaz visual y el mejor modelo predictivo para realizar predicciones con nuevos datos.

- Se implementó una aplicación web local que permite realizar la predicción de fraude de tarjeta de crédito con los 3 mejores modelos seleccionados, la aplicación recoge las entradas para posteriormente realizar la predicción con cada modelo, luego la aplicación analiza los resultados de predicción de cada modelo, escogiendo la moda de los resultados de la predicción para mostrar por pantalla: “Transacción Fraude” o “Transacción No Fraude”.
- Al tener la aplicación web en un entorno local, el tiempo de respuesta es menor a 1 minuto, pero esto podría verse afectado cuando ya se tenga la aplicación en un entorno real con un servidor y conexión con datos en base de datos o en línea, es decir en tiempo real.

4.2.2. Contribución a la gestión empresarial

Algunos de los aspectos que se pueden considerar como contribución a la gestión empresarial como la reducción de pérdidas financieras, mejora de la experiencia del cliente, optimización de recursos, entre otros, se describen a continuación:

- Reducción de pérdidas financieras: Identificar fraudes de forma más eficaz contribuye a disminuir las pérdidas económicas vinculadas a operaciones fraudulentas, salvaguardando tanto a los clientes como a la compañía.
- Mejora de la experiencia del cliente: Al reducir las operaciones fraudulentas, los clientes sienten más seguridad y confianza al utilizar sus tarjetas, lo que incrementa la lealtad y la satisfacción del cliente.
- Optimización de recursos: Un modelo para identificar fraudes posibilita que las compañías mejoren sus recursos de seguridad, centrando los esfuerzos de investigación y mitigación en los casos más dudosos.

- Cumplimiento normativo: Promueve el acatamiento de normativas y regulaciones financieras que demandan acciones apropiadas para evitar fraudes, eludiendo penalizaciones y potenciando la imagen de la compañía.

4.2.3. Contribución a nivel académico

Entre los puntos que se pueden considerar como contribución a nivel académico como los avances en la investigación, innovación tecnológica, formación y educación entre otros, se describen a continuación:

- Avances en la investigación: El desarrollo de modelos para la predicción de fraudes favorece el progreso del saber en campos como el aprendizaje automático, la ciencia de datos y la ciberseguridad.
- Innovación tecnológica: La elaboración y perfeccionamiento de estos modelos fomenta la innovación en algoritmos y métodos de análisis de datos, los cuales pueden ser utilizados en otras áreas, mediante la predicción tomando como referencia los 3 mejores modelos seleccionados de Árbol de decisión, Random Forest y Gradient Boosting para establecer una predicción híbrida con la moda de las predicciones obtenidas de cada modelo.
- Formación y educación: Ofrece ejemplos prácticos y auténticos para la instrucción en materias de ciencia de datos, finanzas y ciberseguridad, capacitando a los alumnos para afrontar retos de la vida real.

4.2.4. Contribución a nivel personal

Entre los aspectos que se pueden considerar como contribución a nivel personal como la conciencia y educación, adquisición de habilidades técnicas, solución a problemas reales, mejora en la comunicación escrita, desarrollo profesional, crecimiento personal, se describen a continuación:

- Conciencia y educación: Un incremento en la comprensión y el entendimiento acerca

de la relevancia de la seguridad económica y los métodos para identificar fraudes. Fomentando un mayor entendimiento de los retos en la seguridad financiera, incentivando una postura proactiva hacia la educación constante y la autoeducación en estos asuntos.

- Adquisición de habilidades técnicas: Adquiriendo una mejora en competencias avanzadas en aprendizaje automático, análisis de datos y programación. Esto permitió mejorar las habilidades técnicas para ofrecer el conocimiento y saberes muy apreciados en el ámbito laboral y en la investigación científica.
- Resolución de problemas reales: Resolviendo problemas prácticos y significativos, utilizando métodos y técnicas de machine learning y deep learning para descubrir soluciones eficaces de modelos predictivos. Esto permite fomentar capacidades para solucionar problemas complejos, lo que promueve la confianza y la habilidad para afrontar retos en diferentes situaciones de la vida real aplicados a todo tipo de organizaciones e instituciones públicas y privadas.
- Mejora en habilidades de comunicación escrita: Mejora en la habilidad para transmitir conceptos técnicos de forma precisa y eficaz en los reportes escritos de machine learning y deep learning.
- Desarrollo profesional: Adquisición de experiencia útil que se puede aplicar directamente en puestos profesionales en finanzas, ciberseguridad y análisis de información. Incrementando la posibilidad de empleo para puestos especializados en diferentes sectores como científico de datos.
- Crecimiento personal: Promoción de formación de una mentalidad de desarrollo y la dedicación al aprendizaje constante, potenciando la resistencia y la capacidad de adaptación frente al aprendizaje del uso de las técnicas de la Inteligencia Artificial en el campo de machine learning.

4.2.5. Limitaciones y problemas durante la Investigación

Entre las limitaciones y problemas que se encontraron durante el proyecto de titulación, tuvieron que ver con la búsqueda de dataset, preprocesamiento de los datos, entre otros, se describen a continuación:

- En la búsqueda de datos del dataset solo fue posible encontrar datos sintéticos puesto información real de transacciones fraudulentas, es información sensible que no se publica con normalidad como otros tipos de datasets que no se requiera de privacidad de datos.
- La fecha de la transacción y fecha de nacimiento del cliente fue preprocesada para encontrar día se la semana, día de mes, mes, año de la transacción y la edad del cliente.
- De los datos de las profesiones del cliente como habían numerosas profesiones fue necesario categorizarlas por profesiones afines para aminorar los resultados de la codificación de la variable categórica.
- Los datos presentaban outliers por lo que fue necesario realizar tratamiento de los datos mediante la técnica de rango intercuartil para poder tener los datos con valores aceptables y evitar el sobreajuste en el entrenamiento de algoritmos que son sensibles a los datos atípicos.
- La clasificación binaria de los datos de la variable objetivo se encontraba desbalanceada por lo que fue obligatorio realizar el balanceo de los datos para aminorar el sesgo en el entrenamiento de los algoritmos de aprendizaje supervisado.

4.3 Recomendaciones

Entre las recomendaciones al proyecto de titulación, se analiza la mitigación de riesgos, usos de datos en tiempo real, consideraciones éticas y privacidad, entre otros, se describen a continuación:

- Estrategias de mitigación de riesgos: Desarrollar estrategias complementarias para mitigar los riesgos asociados con falsos positivos y falsos negativos de los modelos seleccionados para la predicción del fraude.
- Uso de datos en tiempo real: Integrar otras fuentes de datos en tiempo real para mejorar la detección de fraudes con mayor precisión que beneficie a las organizaciones e instituciones públicas y privadas.
- Consideración éticas y privacidad: Asegurar el cumplimiento de normativas de privacidad y ética en la gestión de datos en entornos reales mediante la anonimización de los datos siguiendo las recomendaciones de la ley de protección de datos del Ecuador.
- Publicaciones y difusión: Publicar los hallazgos y contribuciones del proyecto en conferencias y revistas académicas para compartir los resultados obtenidos para contribuir al conocimiento académico y recibir feedback de la comunidad científica mediante la participación en congresos y eventos de ciberseguridad y ciencia de datos.

4.4 Trabajos futuros

Entre los trabajos futuros del proyecto de titulación, se proponen los siguientes aspectos como amplitud de los datos a entrenar y el desarrollo de otros modelos para el entrenamiento en la predicción de fraude de tarjetas de crédito:

- Integración con datos adicionales: Investigación de técnicas avanzadas como el aprendizaje profundo y el análisis de redes para detectar patrones de fraude más complejos, explorando la integración de datos adicionales, como análisis de comportamiento y biometría que permita mejorar los modelos.
- Aplicación de otros algoritmos de aprendizaje profundo: La implementación de redes neuronales profundas, tales como las redes neuronales recurrentes (RNN) y las neuronales convolucionales (CNN), que pueden también capturar patrones de datos

tabulados. Luego, analizar y cotejar el desempeño de estos modelos con el modelo vigente en cuanto a exactitud, precisión, recall y la tasa de falsos negativos.

- Sistemas de detección en tiempo real: Crear una infraestructura capaz de identificar fraudes en tiempo real mediante el streaming de datos y métodos de procesamiento en tiempo real. Luego, analizar la habilidad del sistema para gestionar grandes cantidades de información y su eficacia en la identificación de operaciones fraudulentas en tiempo real.

BIBLIOGRAFÍA

Ameijeiras, D., Valdés, O., & González, H. (2021). Algoritmos de detección de anomalías con redes profundas. Revisión para detección de fraudes bancarios. *Revista Cubana de Ciencias Informáticas*, *15*(4), 244–264. https://doi.org/10.3115/v1/d14-1179

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Xplore Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798–1828. https://doi.org/10.1109/TPAMI.2013.50

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. En: M. Jordan, J. Kleinberg, & B. Scholkopf, Eds.; 1st ed., Vol. 1. https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf

Bottou, L. (2010). Large-Scale Machine Learning with Stochastic Gradient Descent. *Proceedings of COMPSTAT 2010 - 19th International Conference on Computational Statistics, Keynote, Invited and Contributed Papers*, 177–186. https://doi.org/10.1007/978-3-7908-2604-3_16

Bovet, D. P., & Cesati, M. (2020). *Understanding the Linux Kernel (4th ed.)*. Editorial Oxford University Press.

Burkov, A. (2020). *The hundred-page machine learning book*. Editorial Andriy Burkov.

Byrapu, S. R., Kanagala, P., Ravichandran, P., Pulimamidi, D. R., Sivarambabu, P. V., & Polireddi, N. S. A. (2024). Effective fraud detection in e-commerce: Leveraging machine learning and big data analytics. *Revista Measurement: Sensors*, *33*, 1–6. https://doi.org/10.1016/J.MEASEN.2024.101138

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *Revista BMC Genomics*, *21*(1), 1–13. https://doi.org/10.1186/S12864-019-6413-7/TABLES/5

Domor, I., & Jere, N. (2024). Deep Learning for Credit Card Fraud Detection: A Review of Algorithms, Challenges, and Solutions. *IEEE Access*, *12*, 96893–96910. https://doi.org/10.1109/ACCESS.2024.3426955

Elhusseny, S., Shimaa, O., & Amira, M. (2022). Credit Card Fraud Detection Using Machine Learning Techniques. *Future Computing and Informatics Journal*, *7*(1), 13–31. https://doi.org/https://doi.org/10.54623/fue.fcij.7.1.2

Espinosa-Hurtado, R. (2021). Análisis comparativo para la evaluación de frameworks usados en el desarrollo de aplicaciones web. *Revista CEDAMAZ*, *11*(2), 133–141. https://doi.org/10.54753/cedamaz.v11i2.1182

Espinosa-Zúñiga, J. J. (2020). Aplicación de algoritmos Random Forest y XGBoost en una base de solicitudes de tarjetas de crédito. *Revista Ingeniería, Investigación y Tecnología*, *21*(3), 1–16. https://doi.org/10.22201/FI.25940732E.2020.21.3.022

Gaikwad, V., Meher, K., Dass, R., Jonista, A. S., D'Souza, J., & Victor, R. (2023). Fraud Detection Using Machine Learning and Blockchain. *International Journal on Recent and Innovation Trends in Computing and Communication*, *11*(6s), 584–590. https://doi.org/10.17762/IJRITCC.V11I6S.6970

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* En: MIT Press, Editor. https://www.deeplearningbook.org/

Gourley, D., Totty, B., Sayer, M., Aggarwal, A., & Reddy, S. (2022). *HTTP: The Definitive Guide (2nd ed.)*. Editorial O'Reilly Media.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning* (Vol. 2). Editorial Springer New York. https://doi.org/10.1007/978-0-387-84858-7

Haykin, S. S. (2009). *Neural networks and learning machines: Vol. I* (3rd ed.). Editorial Prentice Hall/Pearson. https://dai.fmph.uniba.sk/courses/NN/haykin.neural-networks.3ed.2009.pdf

Ileberi, E., Sun, Y., & Wang, Z. (2022). A machine learning based credit card fraud detection using the GA algorithm for feature selection. *Journal of Big Data*, *9*(24), 1–17. https://doi.org/10.1186/s40537-022-00573-8

Intan, N., Nenda, B., & Ramadhan, D. (2021). Machine Learning Algorithms in Fraud Detection: Case Study on Retail Consumer Financing Company. *Asia Pacific Fraud Journal*, *6*(2), 213–221. https://doi.org/10.21532/apfjournal.v6i2.216

Johnson, P. (2024). *Modern API design: REST, GraphQL, and beyond*. Editorial HiTeX Press.

Joshi, A. V. (2023). Machine Learning and Artificial Intelligence, Second Edition. In *Machine Learning and Artificial Intelligence, Second Edition* (Vol. 2). Editorial Springer International Publishing. https://link.springer.com/book/10.1007/978-3-031-12282-8

Kingma, D. P., & Ba, J. L. (2014). Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. (pp. 1-15). https://arxiv.org/abs/1412.6980v9

Li, H., Cui, Y., Liu, Y., Li, W., Shi, Y., Fang, C., Li, H., Gao, T., Hu, L., & Lu, Y. (2018). Ensemble learning for overall power conversion efficiency of the all-organic dye-sensitized solar cells. *IEEE Access*, *6(1)*, 34118–34126. https://ieeexplore.ieee.org/document/8395263

Mabdeh, A. N., Al-Fugara, A., Ahmadlou, M., Pradhan, B., Nouh Mabdeh, A., & Al-Fugara, K. (2021). Novel Ensemble-Based Machine Learning Models Based on The Bagging, Boosting and Random Subspace Methods for Landslide Susceptibility Mapping. *Revista Research Square*, 1(1),1–31. https://doi.org/10.21203/RS.3.RS-649364/V1

Majumdar, P. (2023). *Mastering Classification Algorithms for Machine Learning* (Vol. 1). Editorial BPB PUBLICATIONS. https://dokumen.pub/mastering-classification-algorithms-for-machine-learning-learn-how-to-apply-classification-algorithms-for-effective-machine-learning-solutions.html

Murillo, D., Saavedra, D., & Quintero, E. (2018). Extracción de datos de perfiles en Google Scholar utilizando un algoritmo en el lenguaje R para hacer minería de datos. *Revista I+D Tecnológico*, *14*(1), 94–104. https://doi.org/10.33412/IDT.V14.1.1807

Nguyen, Q. (2023). *Bayesian optimization in action*. Editorial Manning Publications.

Nilson Report. (2023). *Home - Nilson Report*. Card Fraud Worldwide. Recuperado el 16 de septiembre de 2024. https://nilsonreport.com/

Prateeksha, M., Swetha, N., & Patil, M. (2023). Credit Card Fraud Detection Using Machine-learning. *International Journal of Advanced Research (IJAR)*, *11*(4), 1559–1563. https://doi.org/http://dx.doi.org/10.21474/IJAR01/16824

Registro Oficial Suplemento. (2021). *Código Orgánico Integral Penal, COIP*. Recuperado el 14 de septiembre de 2024. https://www.defensa.gob.ec/

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Revista Nature 323* (1), 533–536. https://doi.org/10.1038/323533a0

Saia, S. M., Nelson, N. G., Young, S. N., Parham, S., & Vandegrift, M. (2022). Ten simple rules for researchers who want to develop web apps. *Revista PLoS Computational Biology*, *18*(1), 1-18. https://doi.org/10.1371/JOURNAL.PCBI.1009663

Shah, M., Kantawala, H., Gandhi, K., Patel, R., Patel, K. A., & Kothari, A. (2023). Theoretical Evaluation of Ensemble Machine Learning Techniques. *Proceedings - 5th International Conference on Smart Systems and Inventive Technology, ICSSIT 2023*, (pp. 829–837). https://doi.org/10.1109/ICSSIT55814.2023.10061139

Shekhar, S., Bansode, A., & Salim, A. (2022). A Comparative study of Hyper-Parameter Optimization Tools. *Proceedings of IEEE CSDE 2021.* (pp. 1-6). http://arxiv.org/abs/2201.06433

Shu, X., & Ye, Y. (2023). Knowledge Discovery: Methods from data mining and machine learning. *Revista Social Science Research*, *110(1)*, 1–16. https://doi.org/10.1016/J.SSRESEARCH.2022.102817

Smith, Q.-J., & Valverde, R. (2021). A perceptron based neural network data analytics architecture for the detection of fraud in credit card transactions in financial legacy systems. *Revista WSEAS TRANSACTIONS on SYSTEMS and CONTROL*, *16(1)*, 358–374. https://doi.org/10.37394/23203.2021.16.31

Sosa, M. (2007). Inteligencia artificial en la gestión financiera empresarial. *Revista Pensamiento & Gestión*, *23(1)*, 153–186. https://www.redalyc.org/pdf/646/64602307.pdf

Superintendencia de Economía Popular y Solidaria. (2021). *Situación de los Servicios Financieros Digitales y Seguridad de la Información en el SFPS*. Recuperado el 14 de octubre de 2024. https://www.seps.gob.ec/

Taye, M. M. (2023). Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions. *Revista Computers 2023, 12*(5), 91-117. https://doi.org/10.3390/COMPUTERS12050091

Tiwari, P., Mehta, S., Sakhuja, N., Kumar, J., & Kumar, A. (2021). Credit Card Fraud Detection using Machine Learning: A Study. *Technical Report*, 1–10. https://arxiv.org/abs/2108.10005v1

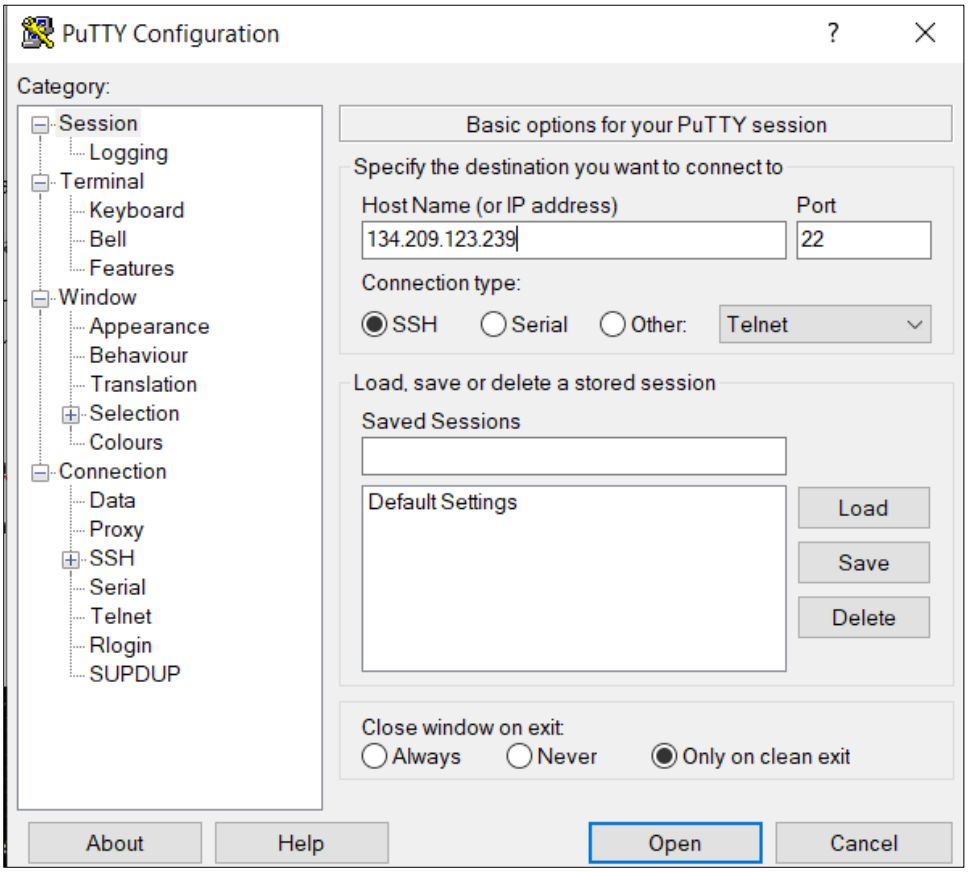
APÉNDICES

Apéndice A. Manual de instalación de la aplicación web de detección de fraude de tarjetas de crédito.

La instalación de la aplicación web se realizará en un servidor Ubuntu versión 24.10 Se ingresa a la aplicación a través de SSH, en este ejemplo se lo realiza a través de la herramienta PuTTY., el servidor tiene asignada la siguiente dirección ip: 134.209.123.239, como se observa en la Figura 72.

Figura 72.

Configuración del terminal de PuTTY

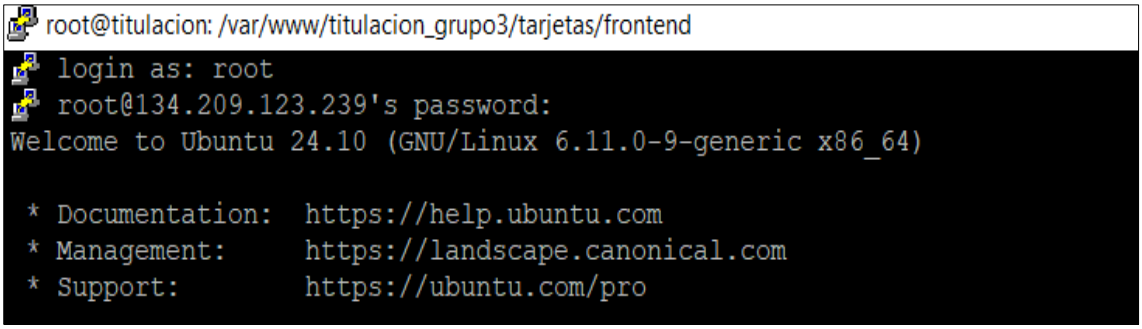


Nota: La figura muestra los parámetros que deben configurarse del terminal de PuTTY como el Host Name, puerto de conexión, tipo de conexión. Obtención: Elaboración propia.

Se ingresa el usuario y la clave como se muestra en la Figura 73:

Figura 73.

Terminal de Linux



Nota: La figura muestra el terminal de Linux para ingresar el usuario y la clave. Obtención:

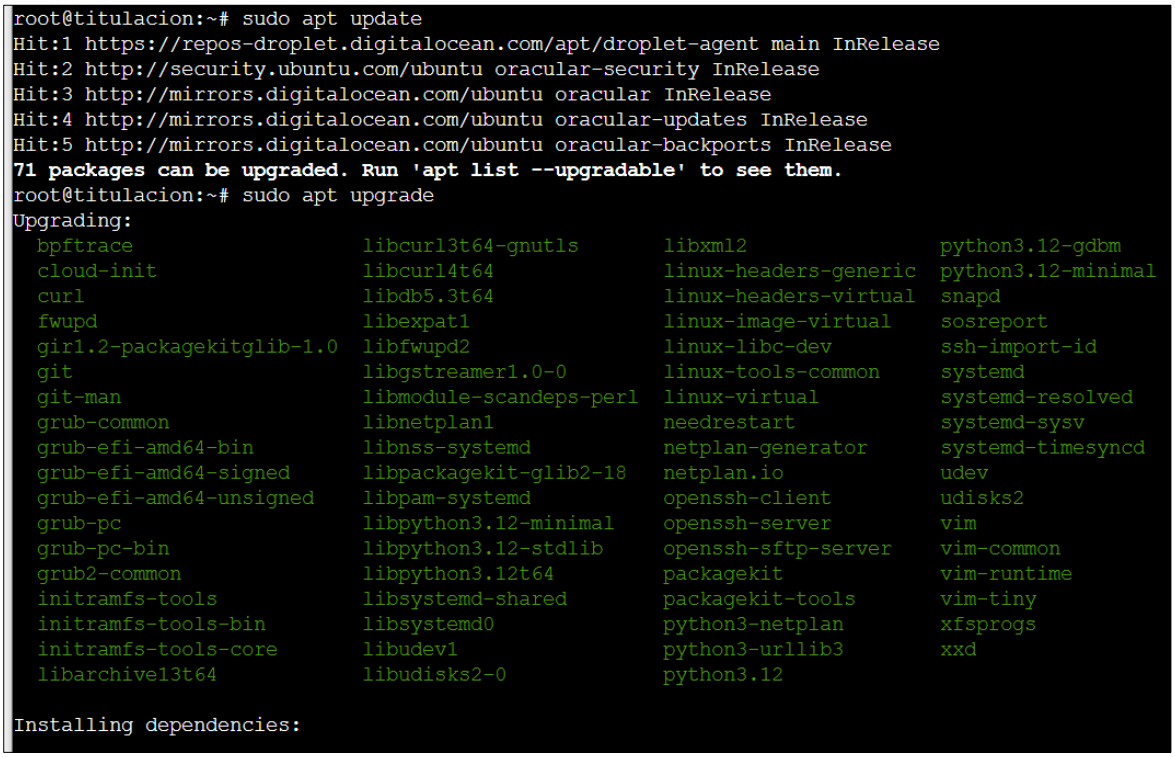
Elaboración propia.

Se realizan las actualizaciones del sistema como se visualiza en la Figura 74:

```
sudo apt update

sudo apt upgrade
```

Figura 74.

Actualizaciones del sistema



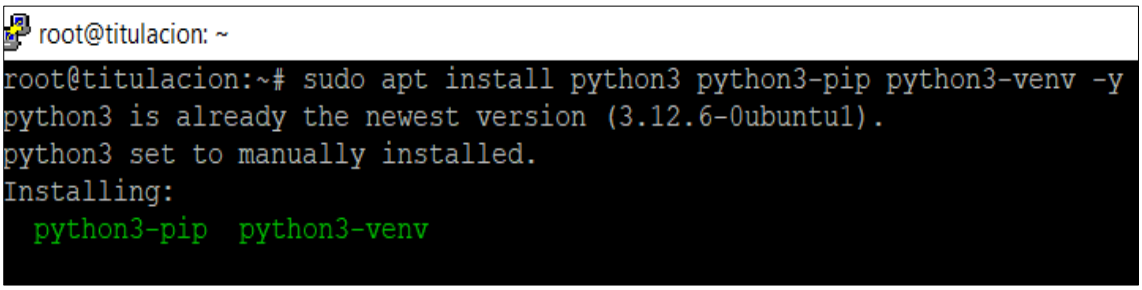
Nota: La figura muestra las actualizaciones del sistema. Obtención: Elaboración propia.

Se procede a instalar Python, pip y venv, como se observa en la Figura 75.

```
sudo apt install python3 python3-pip python3-venv -y
```

Figura 75.

Instalación de Python, pip y veny.



Nota: La figura muestra la instalación de las herramientas necesarias como Python, pip y veny.

Obtención: Elaboración propia.

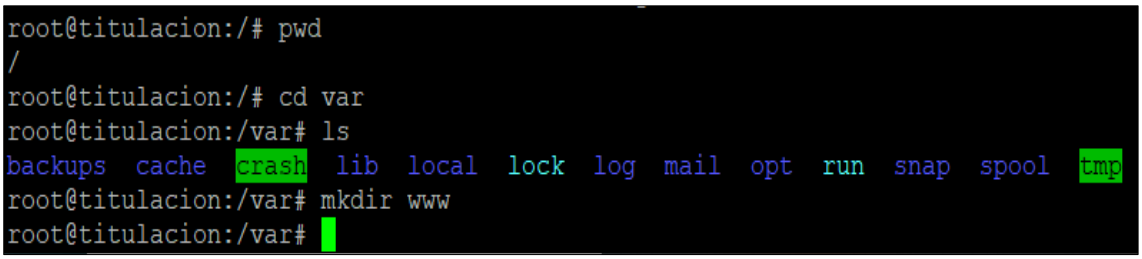
El proyecto será descargado en la ruta del servidor /var/www para lo cual se ingresa al directorio var y se realiza la creación de la carpeta www, como se ve en la Figura 76.

```
cd var
```

```
mkdir www
```

Figura 76.

Descarga del proyecto



Nota: La figura muestra la descarga del proyecto al directorio var y se crea la carpeta www

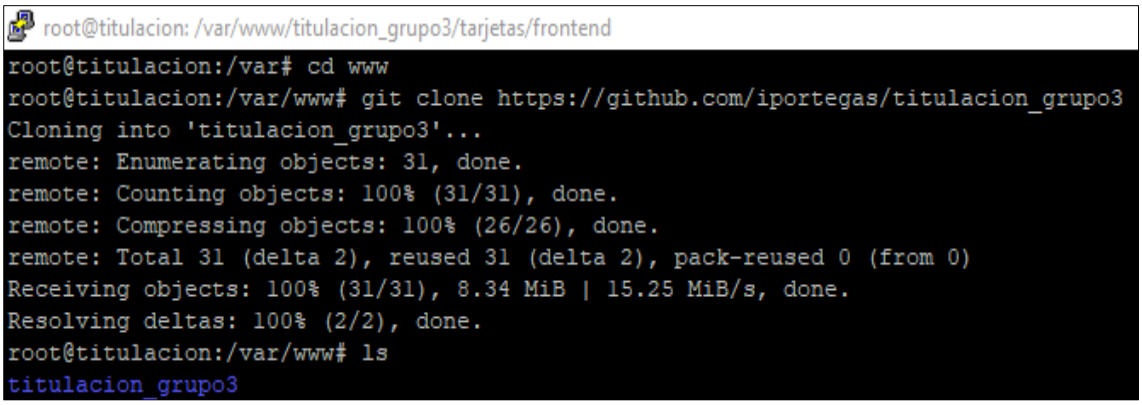
Obtención: Elaboración propia.

Se ingresa al directorio creado www y se realiza la descarga del proyecto desde el repositorio de github, como se observa en la Figura 77:

```
git clone https://github.com/iportegas/titulacion_grupo3
```

Figura 77.

Ingreso al directorio y Descarga del proyecto



Nota: La figura muestra el ingreso al directorio www y descarga del proyecto en la carpeta www.

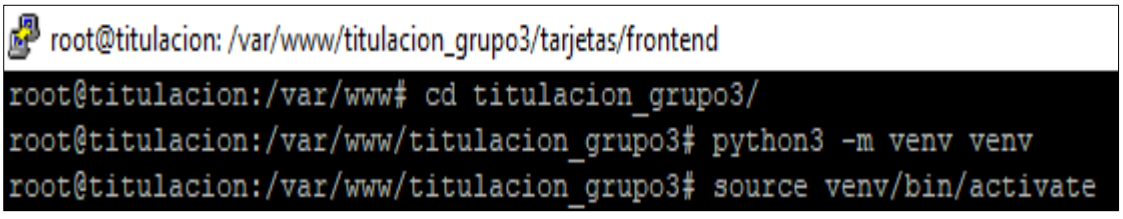
Obtención: Elaboración propia.

Luego, se procede a crear un entorno virtual y activarlo, como se observa en la Figura 78.

```
cd titulacion_grupo3

python3 -m venv venv

source venv/bin/activate
```

Figura 78.

Creación de entorno virtual



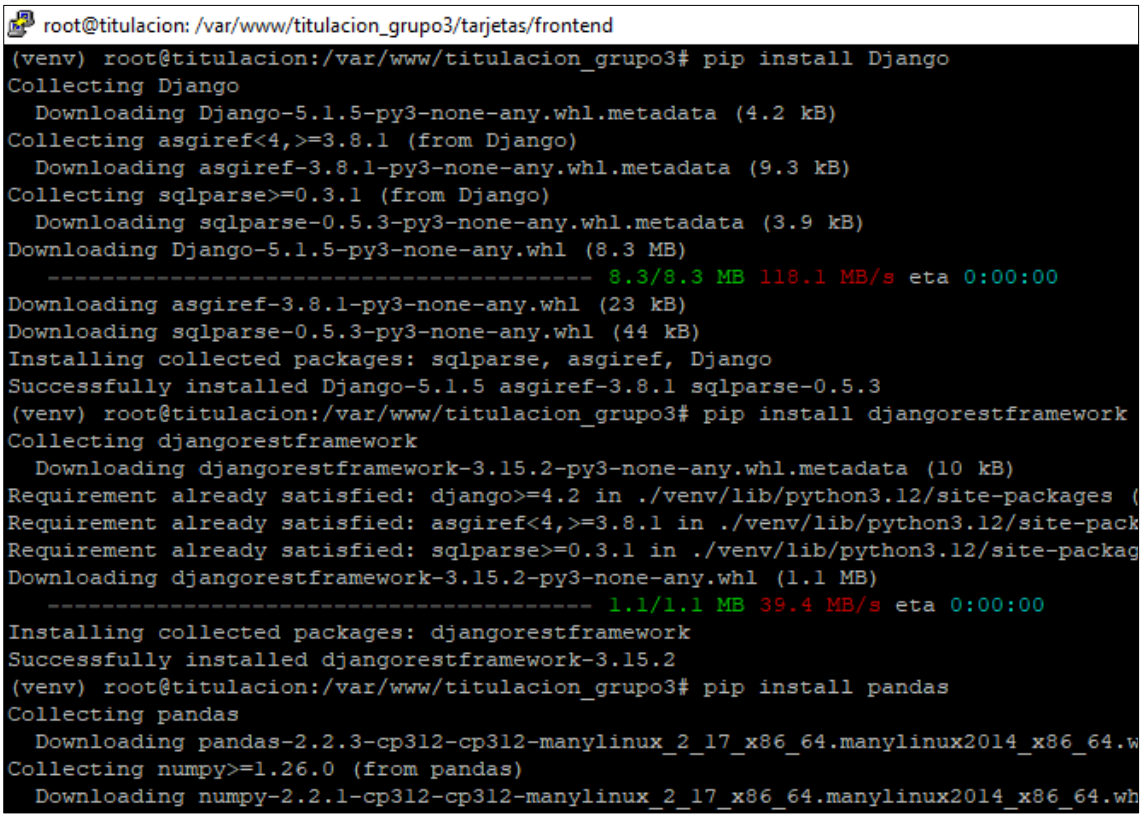
Nota: La figura muestra la creación del entorno virtual y la activación. Obtención: Elaboración propia

Posteriormente, se debe realizar la instalación de los paquetes de python necesarios para que la aplicación pueda funcionar como se observa en la Figura 79:

```
pip install Django

pip install djangorestframework

pip install pandas

pip install scikit-learn

pip install requests
```

Figura 79.

Instalación de los paquetes de Python



Nota: La figura muestra la instalación de los paquetes de Python Django, djangorestframework, pandas, scikit-learn y requests. Obtención: Elaboración propia

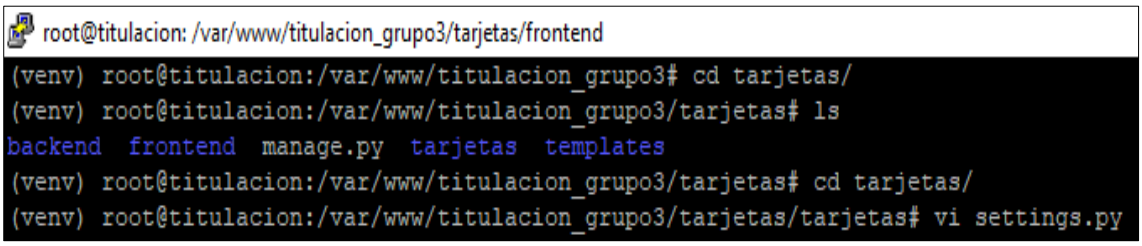
Ahora, realizar la configuración necesaria en el archivo settings.py desactivando el modo de depuración DEBUG=False y permitiendo que se puede acceder a la IP en `ALLOWED_HOSTS = ['*']`.

```
cd tarjetas/

cd tarjetas/

vi settings.py
```

Como se observa en la Figura 80 y Figura 81.

Figura 80.

Configuración de archivo settings.py



Nota: La figura muestra la configuración del archivo settings.py. Obtención: Elaboración propia

Figura 81.

Desactivando el modo de depuración

Nota: La figura muestra la desactivación en modo de depuración DEBUG=False para acceder a la IP. Obtención: Elaboración propia

Se procede a instalar el módulo Supervisor la que permitirá que el sistema operativo reinicie la aplicación de forma automática manteniéndola activa como se observa en la Figura 82.

```
sudo apt install supervisor -y
```

Figura 82.

Instalación del módulo Supervisor

```
root@titulacion:/var/www/titulacion_grupo3/tarjetas# sudo apt install supervisor -y
Installing:
  supervisor

Suggested packages:
  supervisor-doc

Summary:
  Upgrading: 0, Installing: 1, Removing: 0, Not Upgrading: 0
  Download size: 285 kB
  Space needed: 1717 kB / 6095 MB available

Get:1 http://mirrors.digitalocean.com/ubuntu oracular/universe amd64 supervisor all
Fetched 285 kB in 6s (45.5 kB/s)
Selecting previously unselected package supervisor.
(Reading database ... 111668 files and directories currently installed.)
Preparing to unpack .../supervisor_4.2.5-2_all.deb ...
Unpacking supervisor (4.2.5-2) ...
Setting up supervisor (4.2.5-2) ...
Created symlink '/etc/systemd/system/multi-user.target.wants/supervisor.service' → '
Processing triggers for man-db (2.12.1-3) ...
Scanning processes...
Scanning candidates...
Scanning linux images...
```

Nota: La figura muestra la instalación del módulo Supervisor para que el sistema operativo se reinicie. Obtención: Elaboración propia

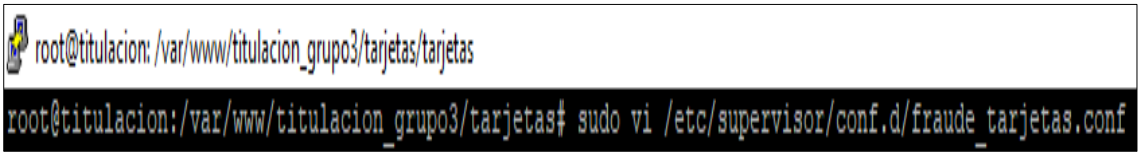
Se necesita crear un archivo de configuración para supervisor en la siguiente rut: `/etc/supervisor/conf.d/ fraude_tarjetas.conf` para lo cual se utiliza el comando vi.

`sudo vi /etc/supervisor/conf.d/fraude_tarjetas.conf`

Como se observa en la Figura 83.

Figura 83.

Configuración de módulo Supervisor



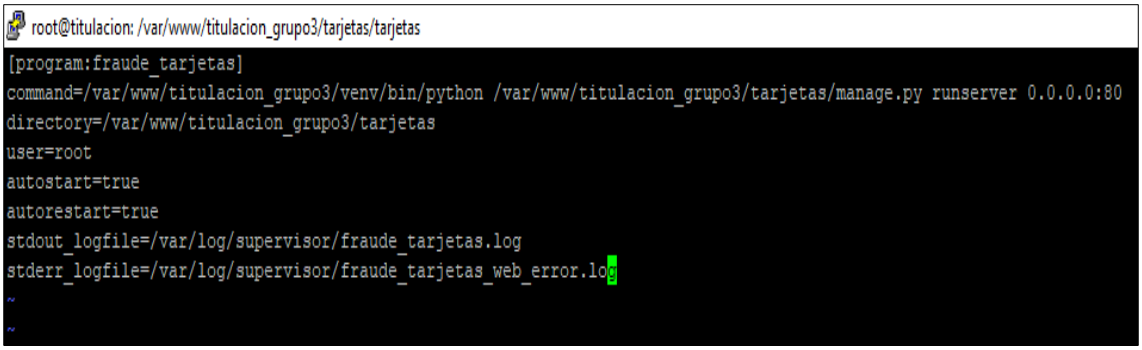
Nota: La figura muestra la configuración del módulo Supervisor. Obtención: Elaboración propia

La configuración es la siguiente como se muestra en la Figura 84:

```
[program:fraude_tarjetas]
command=/var/www/titulacion_grupo3/venv/bin/python
/var/www/titulacion_grupo3/tarjetas/manage.py runserver 0.0.0.0:80
directory=/var/www/titulacion_grupo3/tarjetas
user=root
autostart=true
autorestart=true
stdout_logfile=/var/log/supervisor/fraude_tarjetas.log
stderr_logfile=/var/log/supervisor/fraude_tarjetas_web_error.log
```

Figura 84.

Descripción de la configuración del módulo Supervisor



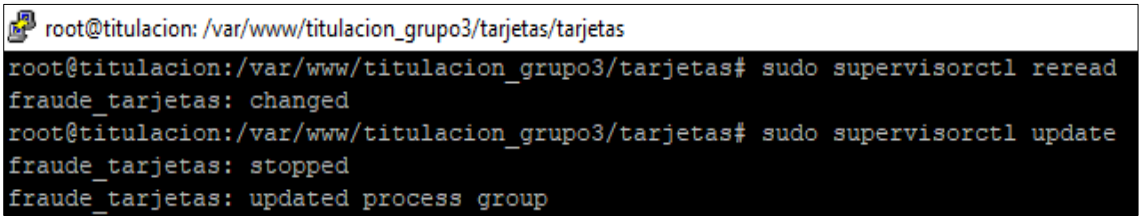
Nota: La figura muestra la descripción de la configuración del módulo Supervisor. Obtención: Elaboración propia

Ahora se procede a recargar y actualizar supervisor para que las configuraciones del archivo tengan efecto, como se observa en la Figura 85.

```
sudo supervisorctl reread
sudo supervisorctl update
```

Figura 85.

Recarga y actualización del módulo Supervisor



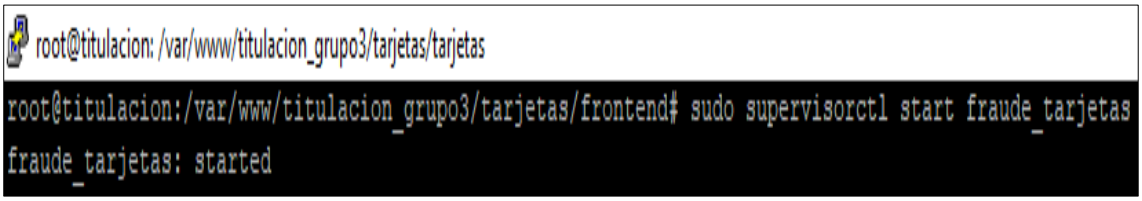
Nota: La figura muestra la recarga y actualización del módulo Supervisor. Obtención: Elaboración propia

Se procede arrancar supervisor, como se observa en la Figura 86.

```
sudo supervisorctl start fraude_tarjetas
```

Figura 86.

Arrancado del módulo Supervisor



Nota: La figura muestra el arrancado del módulo Supervisor. Obtención: Elaboración propia

Al ser un trabajo de investigación y titulación no se tiene contemplado en el alcance instalar servidores más potentes para ambientes de producción como, por ejemplo: Gunicorn o Ngix los cuales son útiles cuando se va a tener una gran carga transaccional; por lo cual se utiliza el propio servidor embebido de Django.

Se debe verificar que funcione correctamente el API: /api/predecir_fraude/

url con la dirección IP asignada: http://134.209.123.239/api/predecir_fraude/

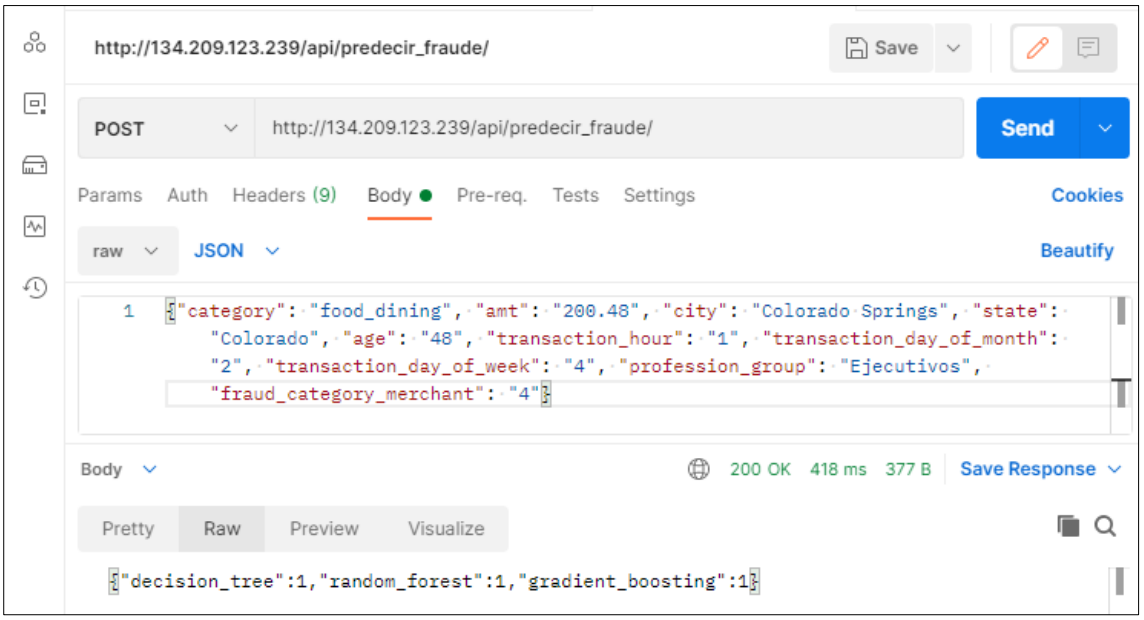
Para realizar la prueba del funcionamiento del API se va a utilizar el programa de postman con la siguiente cadena json de ejemplo:

{"category": "food_dining", "amt": "200.48", "city": "Colorado Springs", "state": "Colorado", "age": "48", "transaction_hour": "1", "transaction_day_of_month": "2", "transaction_day_of_week": "4", "profession_group": "Ejecutivos", "fraud_category_merchant": "4"}

Ver Figura 87.

Figura 87.

Prueba de funcionamiento de la API



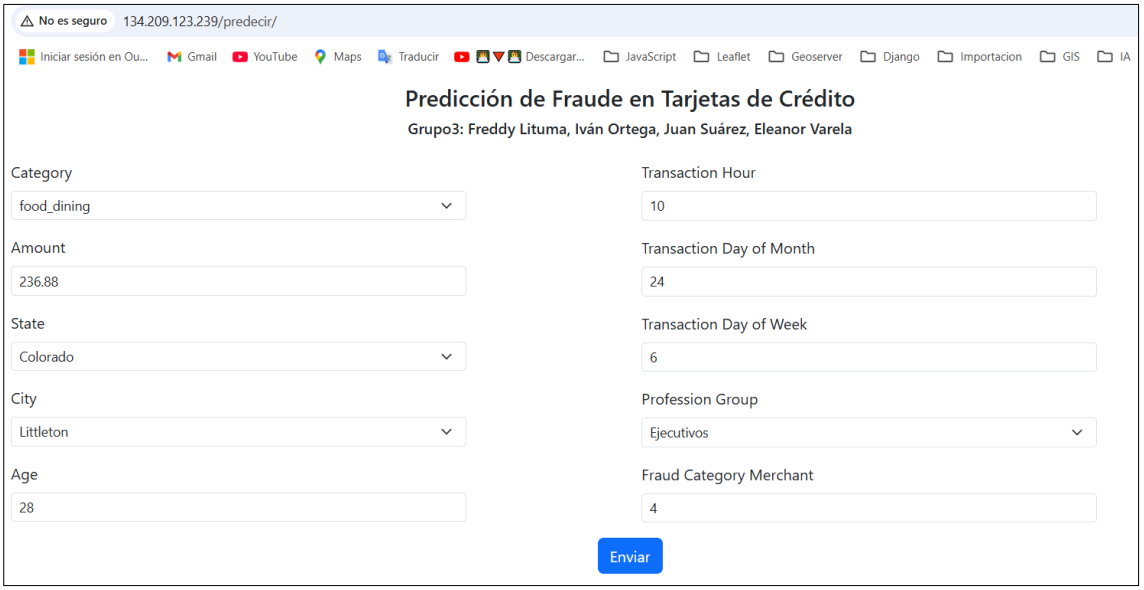
Nota: La figura muestra el funcionamiento de la API. Obtención: Elaboración propia

Se puede verificar que la API está funcionando correctamente ya que devuelve el siguiente resultado: {"decision_tree":1,"random_forest":1,"gradient_boosting":1}

Ahora se necesita verificar el funcionamiento de la aplicación con el front end, ver Figura 88 y Figura 89:

Figura 88.

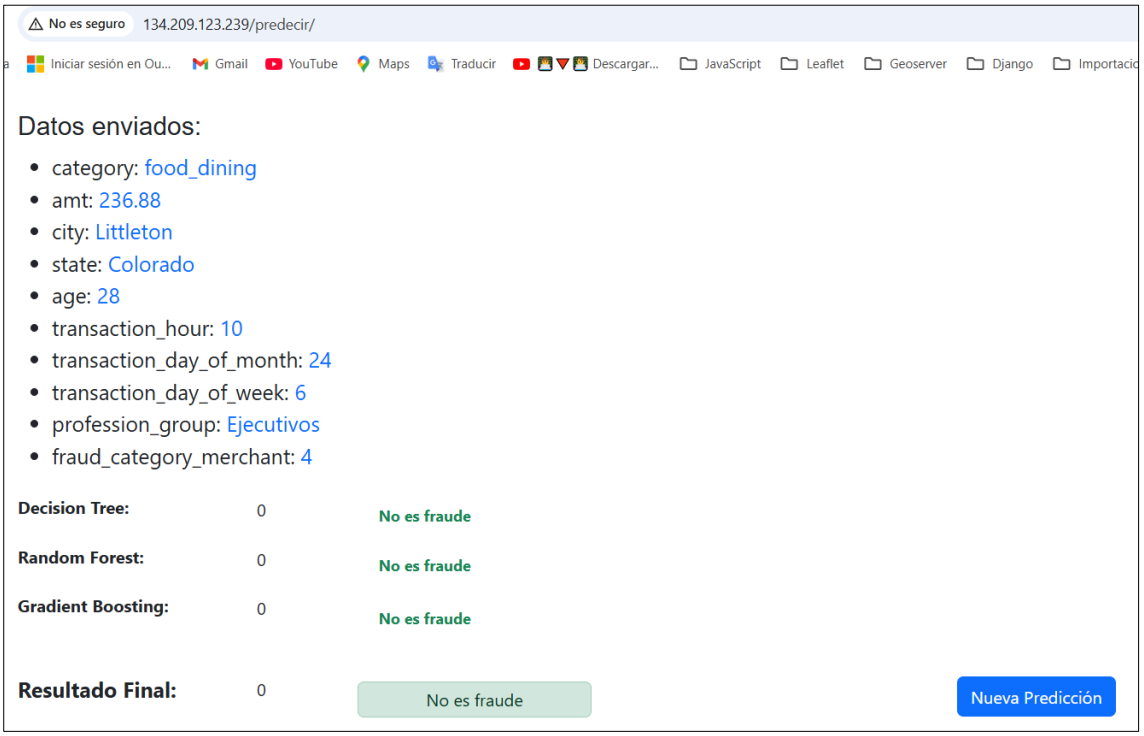
Funcionamiento de la aplicación en el front end



Nota: La figura muestra el funcionamiento de la aplicación en el front end. Obtención: Elaboración propia

Figura 89.

Resultados de predicción en la aplicación web

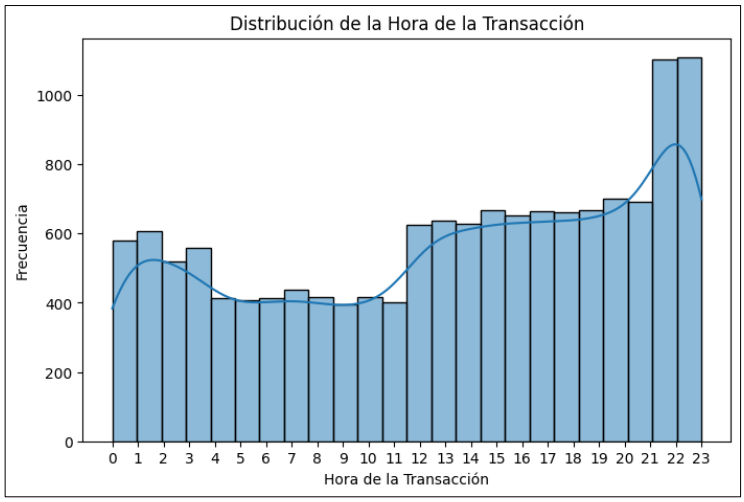


Nota: La figura muestra los resultados de la predicción de la aplicación en el front end. Obtención: Elaboración propia

Apéndice B. Análisis Univariado

Figura 90.

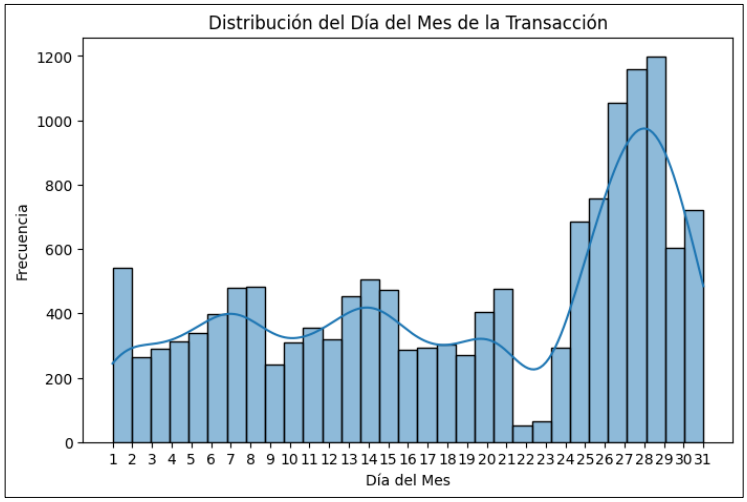
Distribución de hora de transacciones



Nota: La figura muestra la distribución de la hora de las transacciones. Obtención: Elaboración propia

Figura 91.

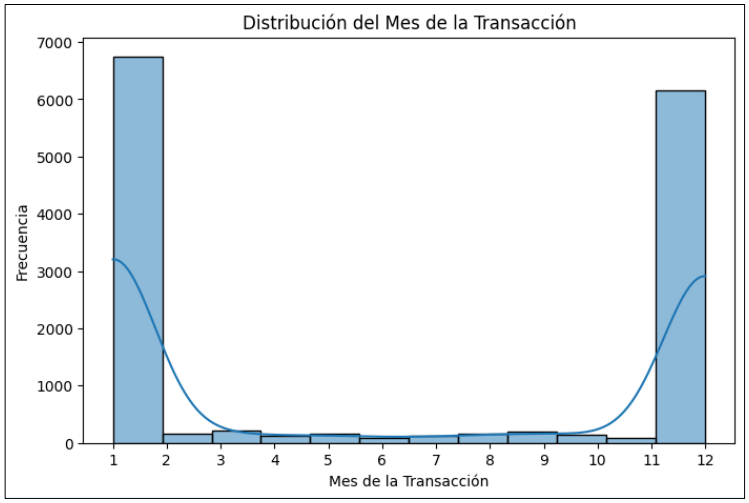
Distribución de día de mes por transacción



Nota: La figura muestra la distribución de día de mes por transacción. Obtención: Elaboración propia

Figura 92.

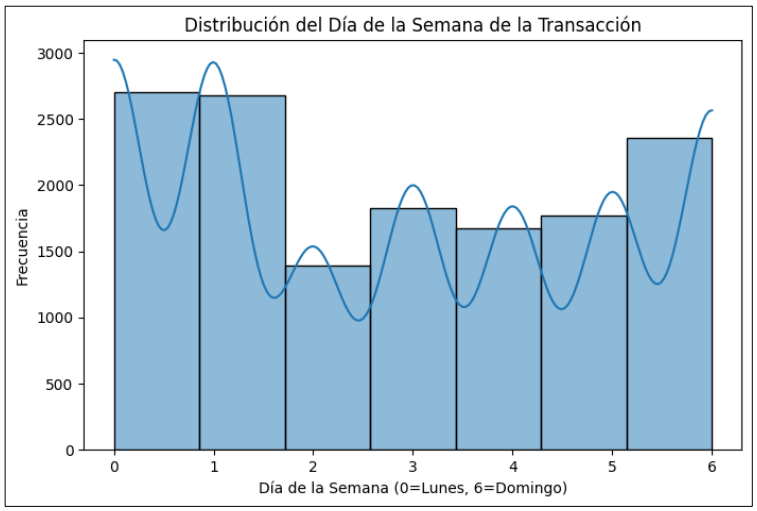
Distribución de mes de transacciones



Nota: La figura muestra la distribución de mes de transacciones. Obtención: Elaboración propia

Figura 93.

Distribución del día de semana de la transacción

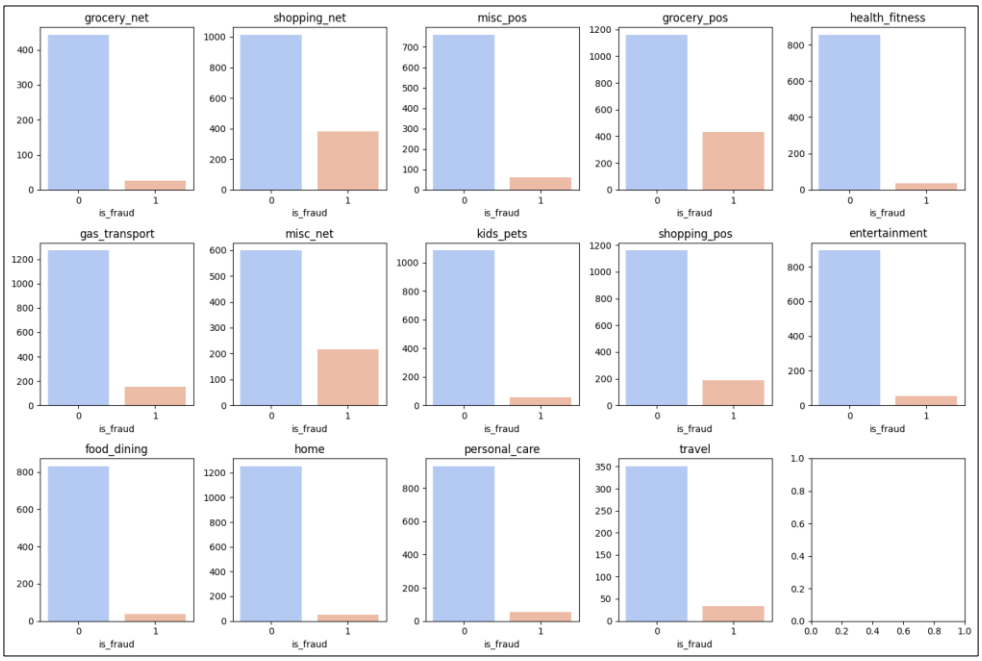


Nota: La figura muestra la distribución del día de semana de la transacción. Obtención: Elaboración propia

Apéndice C. Análisis Bivariado

Figura 94.

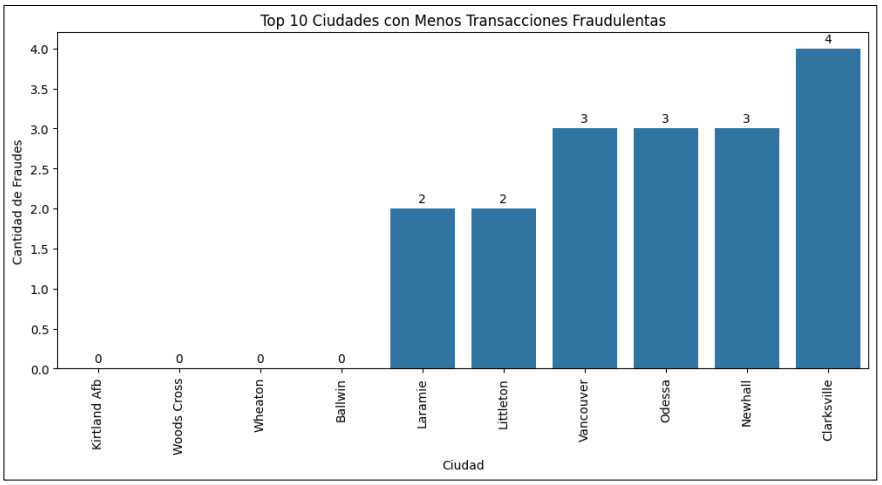
Productos con transacciones de fraudes



Nota: La figura muestra Productos con transacciones de fraudes. Obtención: Elaboración propia

Figura 95.

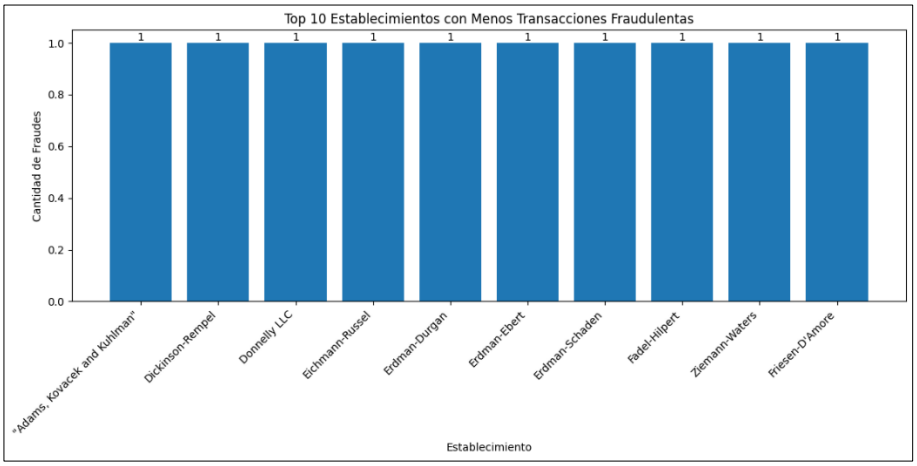
Top 10 ciudades con menos transacciones fraudulentas



Nota: La figura muestra Top 10 ciudades con menos transacciones fraudulentas. Obtención: Elaboración propia

Figura 96.

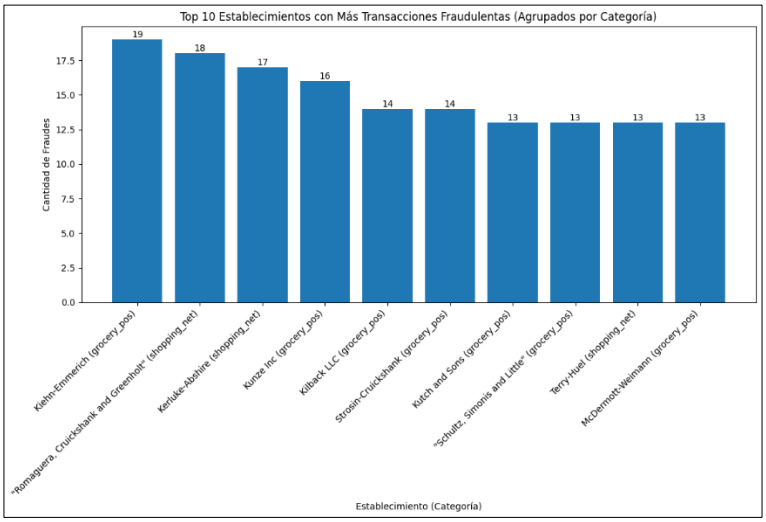
Top 10 de establecimientos con menos transacciones de fraudes



Nota: La figura muestra Top 10 de establecimientos con menos transacciones de fraudes. Obtención: Elaboración propia

Figura 97.

Top 10 establecimientos con más fraudes agrupados por categoría de productos

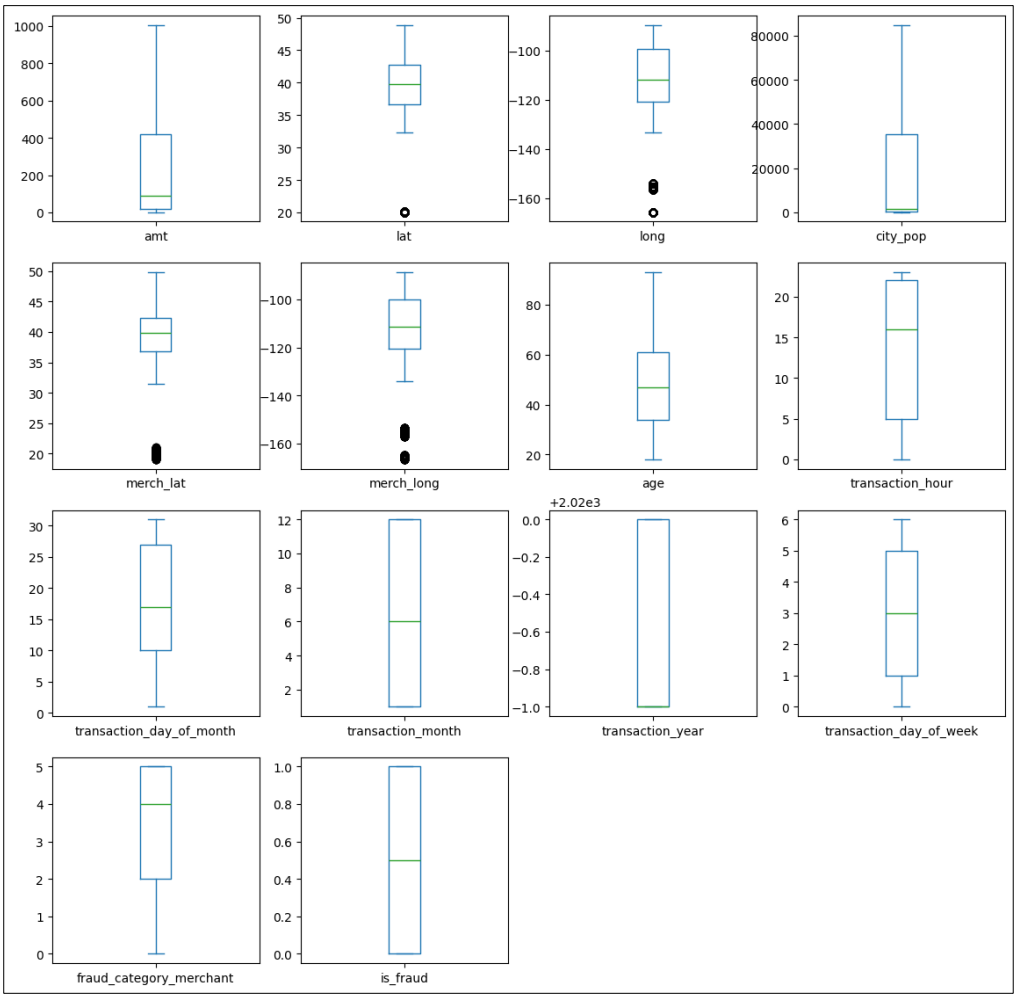


Nota: La figura muestra Top 10 establecimientos con más fraudes agrupados por categoría de productos. Obtención: Elaboración propia

Apéndice D. Outliers

Figura 98.

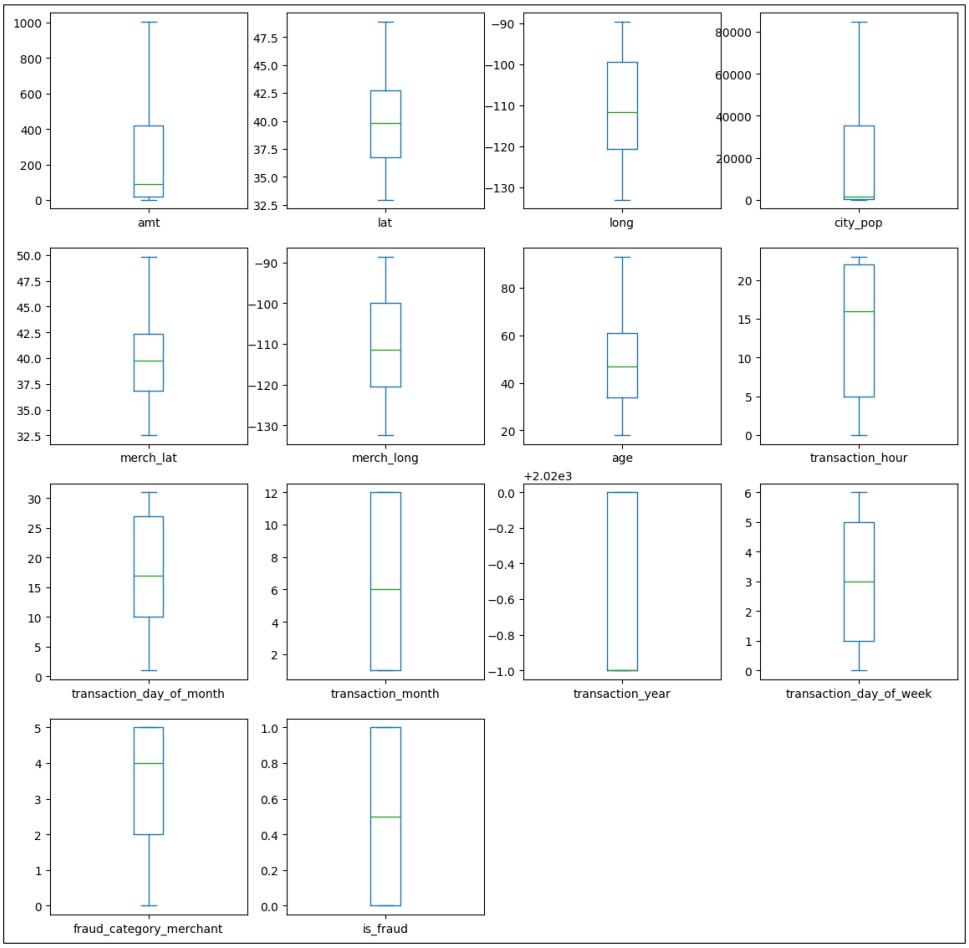
Datos con outliers



Nota: La figura muestra Datos con outliersl. Obtención: Elaboración propia

Figura 99.

Datos sin outliers tratados con rango Intercuartil



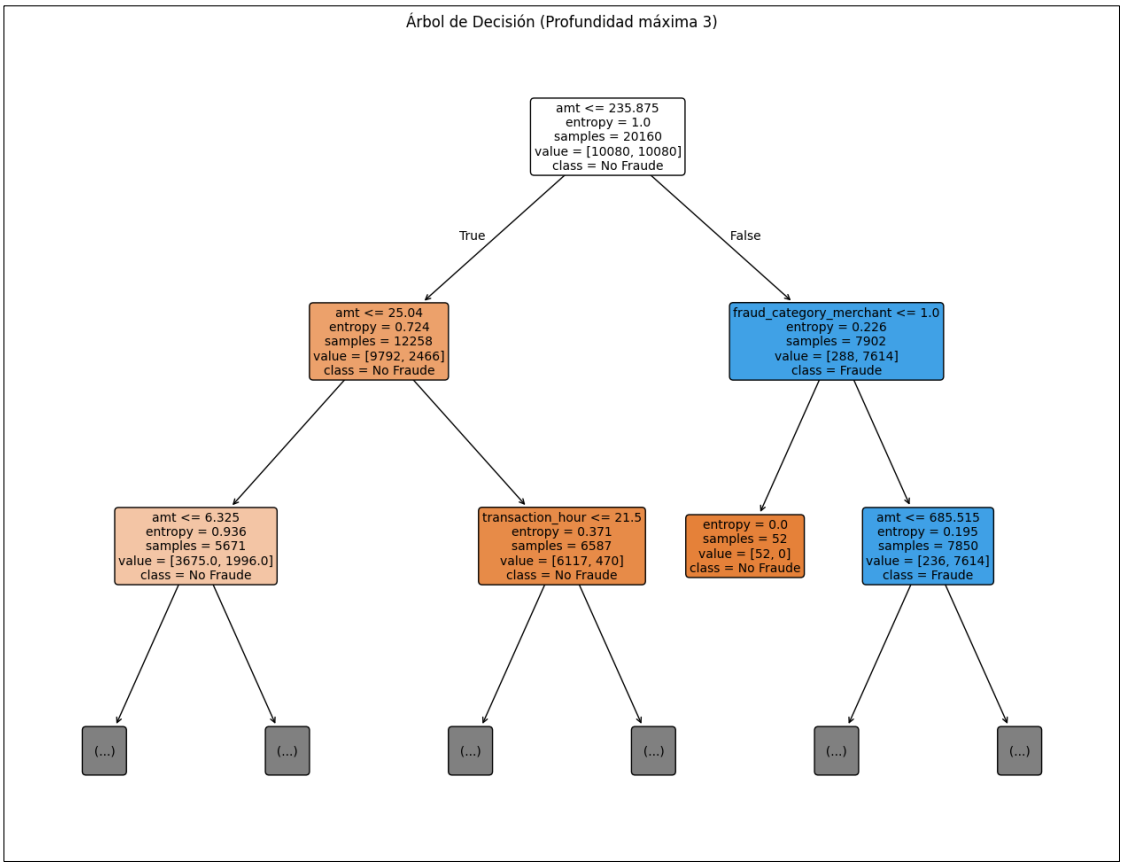
Nota: La figura muestra Datos sin outliers tratados con rango Intercuartil. Obtención: Elaboración propia

Apéndice E. Árboles

Árbol de decisión

Figura 100.

Árbol de decisión con profundidad de 3

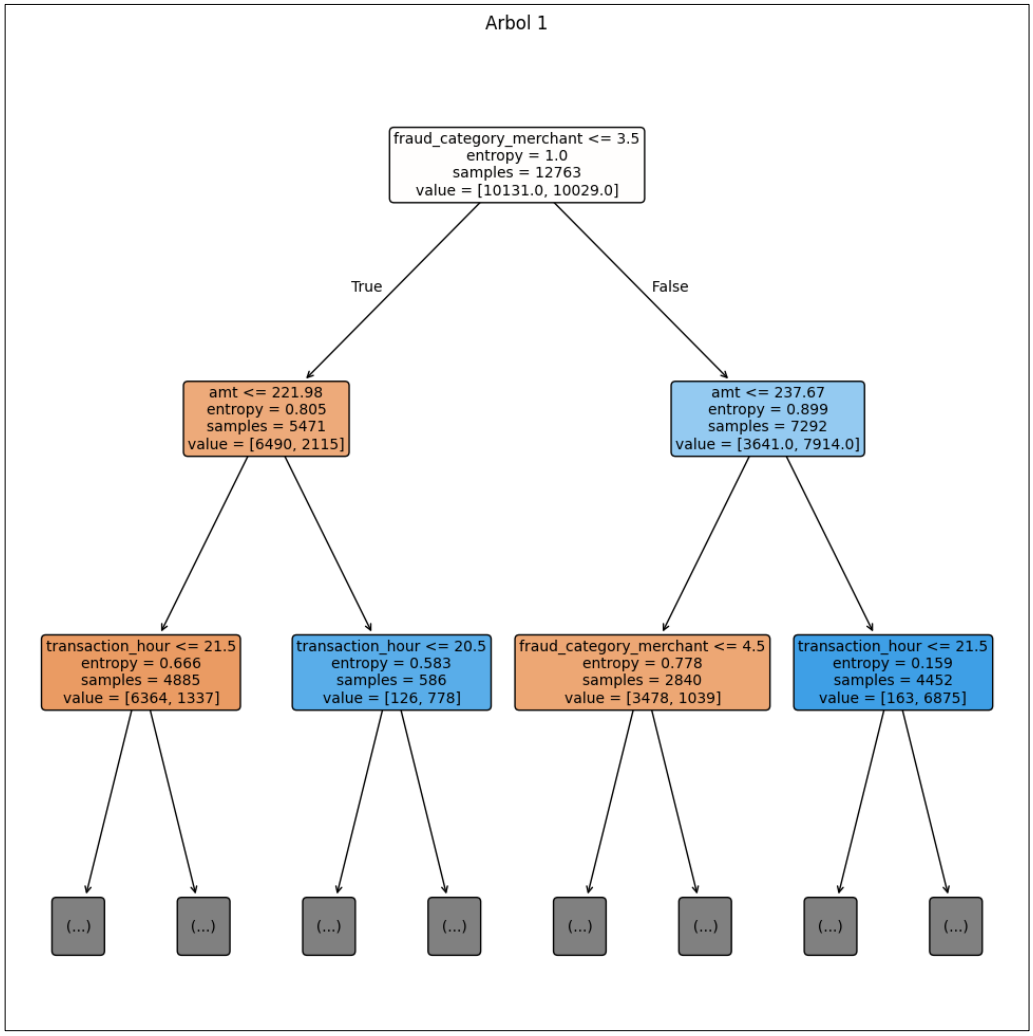


Nota: La figura muestra Árbol de decisión con profundidad de 3. Obtención: Elaboración propia

Random forest

Figura 101.

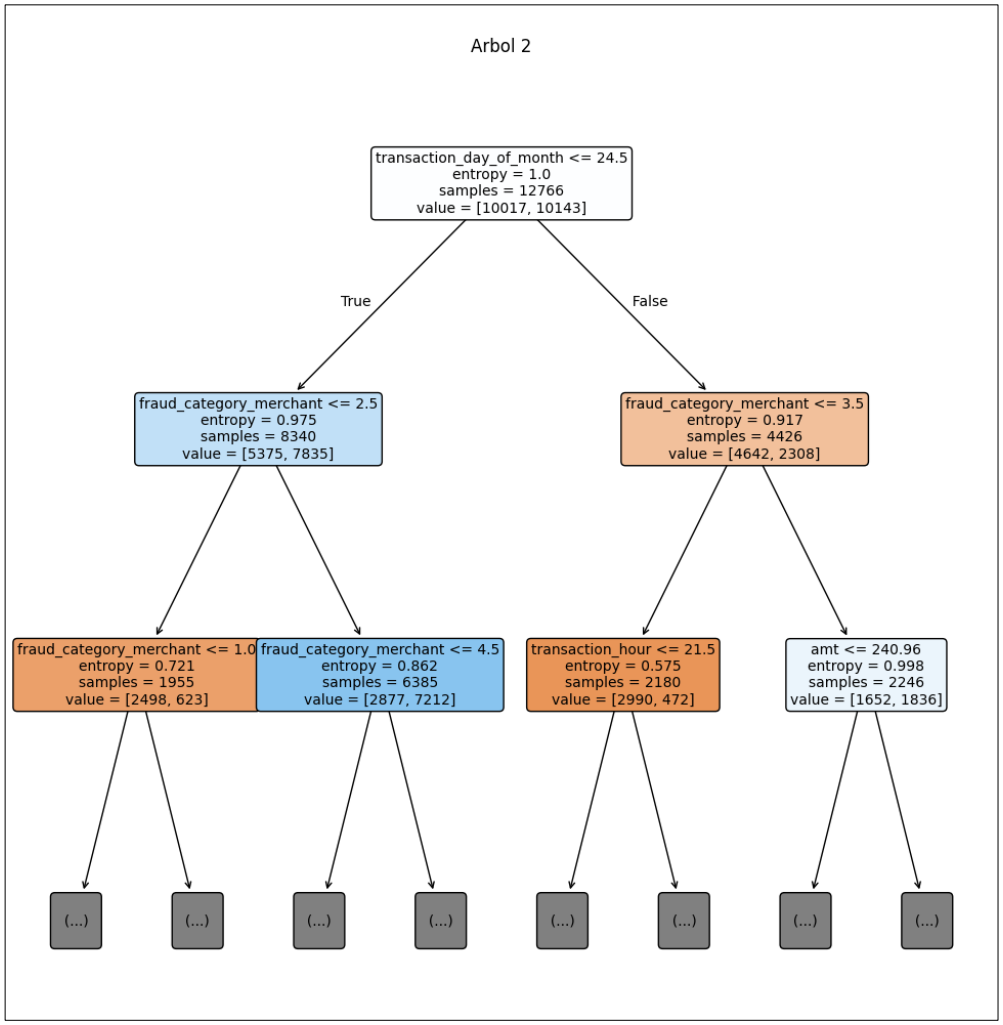
Árbol de decisión #1 de Random Forest



Nota: La figura muestra Árbol de decisión #1 de Random Forest. Obtención: Elaboración propia

Figura 102.

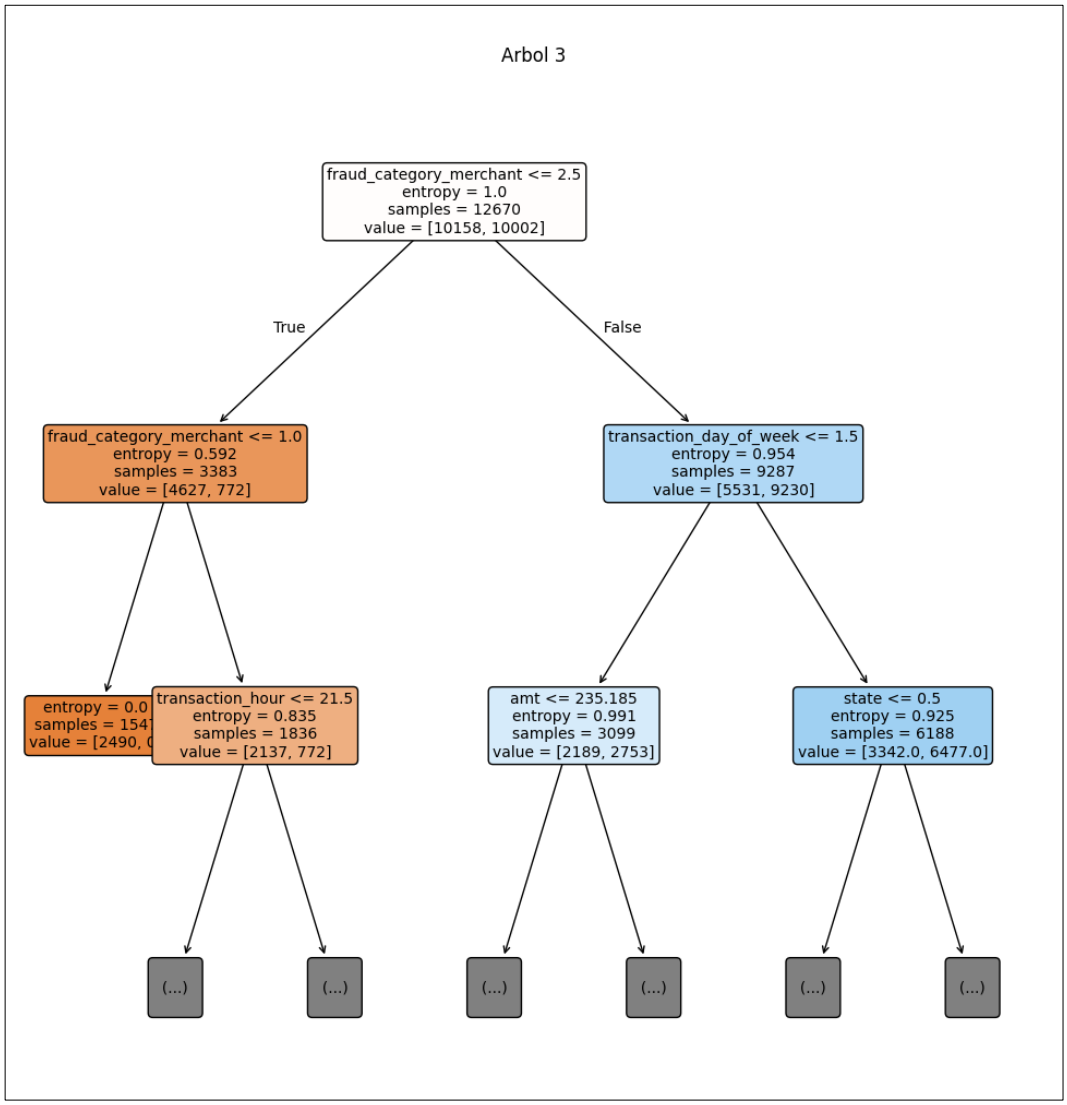
Árbol de decisión #2 de Random Forest



Nota: La figura muestra Árbol de decisión #2 de Random Forest. Obtención: Elaboración propia

Figura 103.

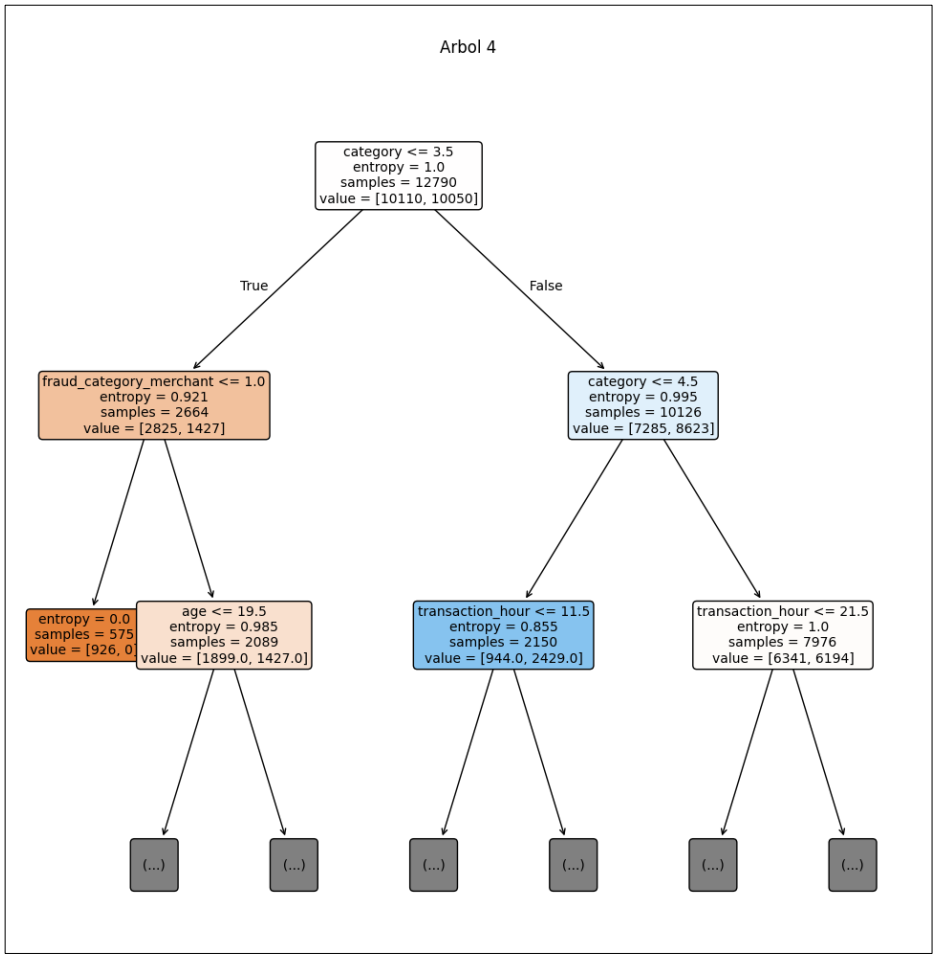
Árbol de decisión #3 de Random Forest



Nota: La figura muestra Árbol de decisión #3 de Random Forest. Obtención: Elaboración propia

Figura 104.

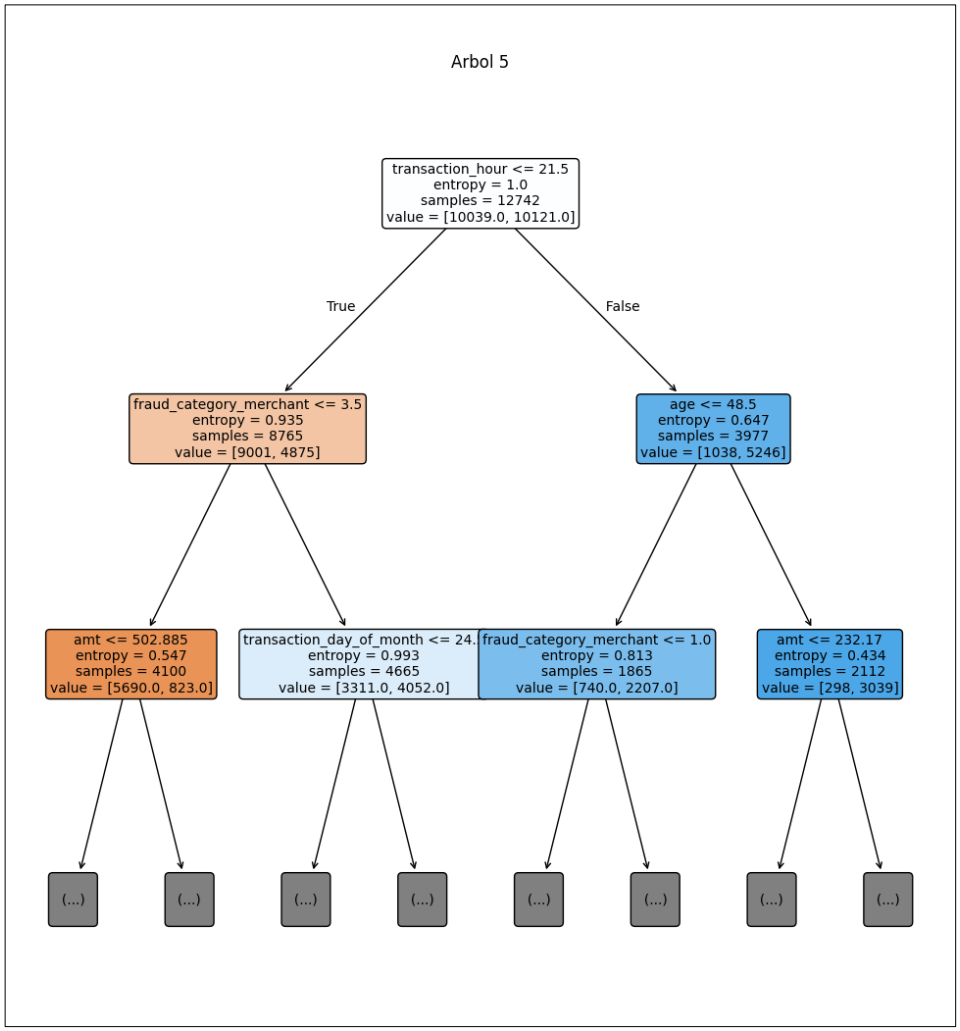
Árbol de decisión #4 de Random Forest



Nota: La figura muestra Árbol de decisión #4 de Random Forest. Obtención: Elaboración propia

Figura 105.

Árbol de decisión #5 de Random Forest

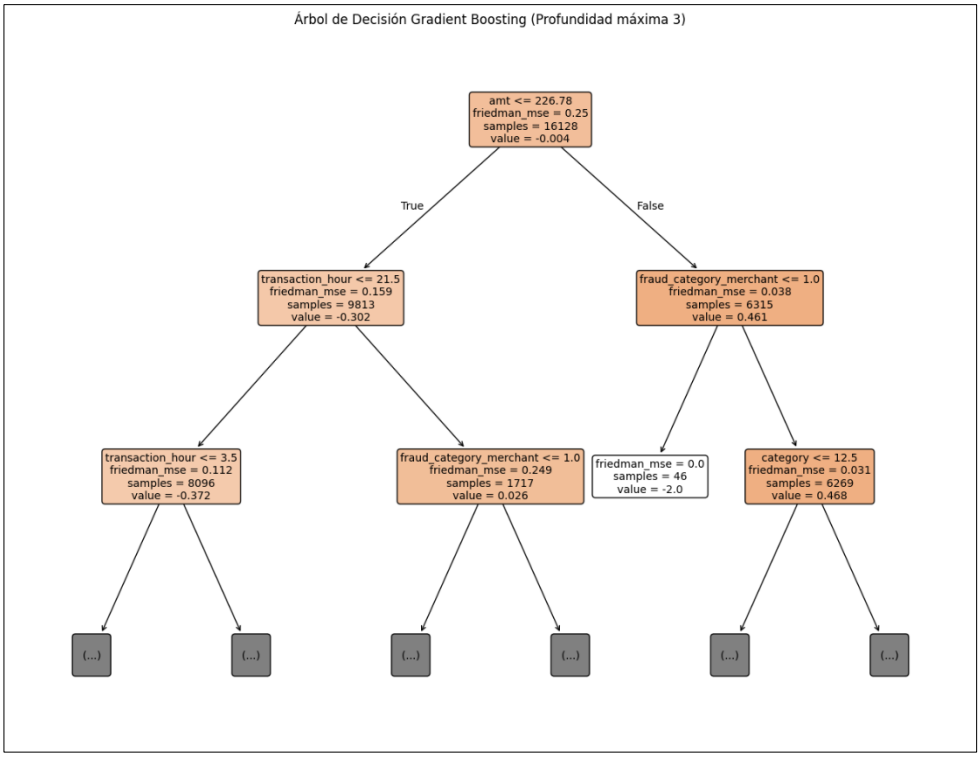


Nota: La figura muestra Árbol de decisión #5 de Random Forest. Obtención: Elaboración propia

Gradient Boosting

Figura 106.

Árbol de decisión con Gradient Boosting



Nota: La figura muestra Árbol de decisión con Gradient Boosting. Obtención: Elaboración propia