

Digital Fraud and Cybersecurity: Artificial Intelligence-Based Strategies for Anomaly Detection

Agustín Lujan, Roxana Martínez

Centro de Altos Estudios en Tecnología Informática (CAETI)
Universidad Abierta Interamericana (UAI)
Buenos Aires, Argentina
roxana.martinez@uai.edu.ar;
AgustinEzequiel.LujanBidondo@alumnos.uai.edu.ar

Abstract. Fraud in electronic transactions represents a growing challenge in the financial sector, driven by digitalization and the rise of online commerce. Cybercriminals have developed increasingly sophisticated strategies to exploit vulnerabilities in payment systems, resulting in significant financial losses and compromising the security of consumers, businesses, and banking institutions. Traditionally, fraud detection has relied on predefined rules and supervised models, which require large volumes of labeled data. However, the rapid evolution of fraudulent tactics limits the effectiveness of these approaches. In this context, anomaly detection-based machine learning emerges as an alternative for the early identification of suspicious transactions without the need for prior fraud data. This study focuses on developing a model based on anomaly detection techniques to identify real-time fraudulent transactions. Various approaches will be evaluated, including autoencoders, isolation forests, and One-Class SVM. Autoencoders, as neural networks designed to reconstruct normal data, can detect suspicious transactions when reconstruction errors are high. Isolation forests identify anomalies by isolating outlier observations in a dataset, enabling efficient fraud detection. Finally, One-Class SVM creates a decision boundary that separates normal transactions from potentially fraudulent ones, making it particularly useful in scenarios where fraud cases represent a small proportion of total transactions. The implementation of these techniques will allow the analysis of large volumes of data with greater accuracy and speed, facilitating more effective fraud pattern detection. The results obtained will contribute to the development of more efficient solutions for protecting electronic transactions in financial and commercial environments.

Keywords: *Cybersecurity, fraud detection, machine learning, anomaly detection.*

Fraude Digital y Ciberseguridad: Estrategias Basadas en Inteligencia Artificial para la Detección de Anomalías

Resumen. El fraude en transacciones electrónicas representa un desafío creciente en el sector financiero, impulsado por la digitalización y el auge del comercio en línea. Los ciberdelincuentes han desarrollado estrategias cada vez más sofisticadas para explotar vulnerabilidades en los sistemas de pago, lo que ha generado pérdidas millonarias y comprometido la seguridad de consumidores, empresas y entidades bancarias. Por lo general, la detección de fraudes ha dependido de reglas predefinidas y modelos supervisados, los cuales requieren grandes volúmenes de datos etiquetados. Sin embargo, la rápida evolución de las tácticas fraudulentas limita la eficacia de estos enfoques. En este contexto, el aprendizaje automático basado en detección de anomalías surge como una alternativa para la identificación temprana de transacciones sospechosas sin necesidad de datos previos de fraudes. Este estudio se enfoca en el desarrollo de un modelo basado en técnicas de detección de anomalías para identificar transacciones fraudulentas en tiempo real. Se evaluarán distintos enfoques, entre ellos autoencoders, isolation forests y One-Class SVM. Los autoencoders, son redes neuronales diseñadas para reconstruir datos normales, pueden detectar transacciones sospechosas cuando el error de reconstrucción es elevado. Por su parte, los isolation forests identifican anomalías al aislar observaciones atípicas en un conjunto de datos, permitiendo una detección eficiente de fraudes. Finalmente, One-Class SVM genera una frontera de decisión que separa las transacciones normales de las potencialmente fraudulentas, lo que resulta útil en escenarios donde los fraudes representan una pequeña proporción del total de transacciones. La implementación de estas técnicas permitirá analizar grandes volúmenes de datos con mayor precisión y rapidez, facilitando la detección de patrones de fraude de manera más efectiva. Los resultados obtenidos contribuirán al desarrollo de soluciones más eficaces para la protección de transacciones electrónicas en entornos financieros y comerciales.

Palabras Clave: *Ciberseguridad; detección de fraudes; aprendizaje automático; detección de anomalías.*

1 Introducción

El incremento acelerado de las transacciones digitales ha abierto nuevas oportunidades en el sector financiero, pero también ha incrementado los riesgos asociados al fraude en línea. Este fenómeno impacta significativamente tanto a instituciones financieras como a sus clientes, esto ha generado pérdidas millonarias y ha deteriorado la confianza en los sistemas de pago.

Tradicionalmente, la detección de fraudes se ha basado en reglas predefinidas y heurísticas, por ejemplo, la identificación de compras en ubicaciones inusuales o transacciones de alto valor en cortos periodos de tiempo han sido indicios de fraude electrónico. Estos métodos si bien resultaron útiles en un inicio, han quedado rezaga-

dos ante la sofisticación y dinamismo de las tácticas actuales con la imposibilidad de adaptarse a nuevos tipos de fraude y además genera falsos positivos.

Ante estos desafíos, la inteligencia artificial (IA) se posiciona como una herramienta clave en el fortalecimiento de la ciberseguridad. En particular, los enfoques basados en aprendizaje automático, especialmente en la detección de anomalías, para identificar comportamientos atípicos sin depender exclusivamente de grandes volúmenes de datos etiquetados. Este punto es importante, ya que, si bien es posible acceder a ciertos datasets, muchos contienen información sensible, por lo que se deben tomar precauciones al utilizarlos para entrenar modelos de IA. Diversos estudios han abordado la aplicación de técnicas avanzadas de machine learning para la detección de transacciones fraudulentas en tarjetas de crédito, destacando una alta efectividad y precisión en comparación a métodos tradicionales basados en reglas [1]. Por otro lado, investigaciones recientes han explorado enfoques más sofisticados, como la detección de anomalías transaccionales utilizando técnicas de aprendizaje automático basadas en grafos, obteniendo así resultados prometedores para detectar patrones fraudulentos complejos mediante el análisis de relaciones entre transacciones [2].

En este sentido, la técnica de los autoencoders permite reconstruir datos considerados “normales” y detectar anomalías cuando el error de reconstrucción es elevado, validaron el uso de esta técnica en la detección de transacciones fraudulentas, Mohammed Abdulhameed Al-Shabi de la Taibah University a evidenciando una mejora significativa en la identificación de patrones atípicos en transacciones de tarjeta de crédito utilizando esta técnica [3]. Otra técnica complementaria es la de isolation forests la cual aíslan observaciones atípicas de forma eficiente, Julien Lesouple y otros investigadores han evidenciado la robustez de esta técnica especialmente combinada con Autoencoders y así logrando un enfoque híbrido aumentando la precisión de la detección de anomalías [4][6]. Por último, el modelo One-Class SVM establece una frontera de decisión que separa de forma precisa las transacciones legítimas de las potencialmente fraudulentas. Esta frontera puede ser comprometida cuando la cantidad de datos a explorar es muy elevada [7].

Este trabajo se enfoca en el ámbito de la ciberseguridad y tiene como objetivo comparar el desempeño de tres técnicas de detección de anomalías —Autoencoders, Isolation Forest y One-Class SVM— en la identificación de fraudes digitales. La investigación se basa en un desarrollo propio de un script en Python que implementa estas técnicas sobre un conjunto de datos reales extraído de la plataforma Kaggle, publicado por la IEEE [5]. El script automatiza todo el flujo de procesamiento: desde la carga y fusión de los archivos de transacciones e identidades, hasta el pre-procesamiento, entrenamiento de modelos, generación de predicciones y exportación de resultados. Los datos obtenidos se utilizarán para realizar un análisis preliminar del rendimiento de cada técnica en la detección de transacciones fraudulentas y comparárlas entre sí.

2 Propuesta

Actualmente, existen algunas falencias en los sistemas tradicionales de detección de fraudes. Muchos de ellos se basan en reglas fijas o en modelos supervisados que, si bien resultan útiles en contextos controlados, no logran adaptarse con eficacia ante

nuevas estrategias de fraude que evolucionan de forma constante. Además, requieren grandes volúmenes de datos previamente clasificados y procesados, lo cual representa una limitación significativa en entornos reales. Este tipo de enfoques tiende a generar un número elevado de falsos positivos o no logra detectar operaciones verdaderamente sospechosas. Según un informe de Juniper Research, se estima que las pérdidas por fraude en pagos en línea superarán los 48 mil millones de dólares a nivel mundial para 2023 [8], lo que evidencia la magnitud del problema y la necesidad de contar con mecanismos más eficaces. Por eso, es necesario pensar alternativas más flexibles y eficientes, que permitan actuar en contextos inesperados y ofrecer una mejor capacidad de adaptación. La propuesta que se desarrolla en este trabajo surge justamente de esa necesidad: explorar y comparar métodos de detección de anomalías que no dependen de datos etiquetados, y que puedan contribuir a mejorar la seguridad en las transacciones digitales.

Este trabajo de investigación se centra en el desarrollo y evaluación de un sistema de detección de fraude en transacciones de tarjetas de crédito basado en técnicas avanzadas de aprendizaje automático. Se trabajarán en tres modelos complementarios: autoencoders, isolation forests y One-Class SVM. Los autoencoders, basados en redes neuronales, se entrenarán para reconstruir datos considerados normales, permitiendo identificar anomalías cuando el error de reconstrucción supera un umbral determinado. Por su parte, el algoritmo de isolation forests se encargará de aislar observaciones atípicas en grandes volúmenes de datos, mientras que el modelo One-Class SVM establecerá una frontera de decisión para separar las transacciones legítimas de aquellas potencialmente fraudulentas.

La primera etapa del proyecto consistirá en el preprocesamiento y la preparación del dataset, utilizando la base de datos IEEE Fraud Detection disponible en Kaggle [5]. Se realizó una limpieza de los datos utilizando Pandas y Scikit-learn, convirtiendo valores nulos y reemplazándolo por 0 en variables numéricas, luego se aplicará técnicas de escalado y normalización con StandardScaler. De esta forma se garantiza que los modelos operen sobre datos consistentes y evitando así que el código tenga procesos de error al momento de ejecutarse. Mejorar la calidad de los datos es un punto crítico en la detección de fraudes financieros, donde el desbalance de clases puede impactar significativamente en el rendimiento de los algoritmos.

En la siguiente fase, se implementaron modelos de detección de anomalías en Python, en el cual se entrenará un Autoencoder utilizando TensorFlow y Keras, basado en una estructura de dos capas con codificación y decodificación. Por otro lado, se empleó Isolation Forest, un modelo basado en árboles para identificar transacciones anómalas. Finalmente, se implementó One-Class SVM, tomando una muestra de 50,000 registros para evitar bloqueos y mejorar la eficiencia del entrenamiento.

Una vez obtenidos los resultados, se procederá a analizarlos. El resultado y análisis de este se enfocará en las distintas técnicas y en la eficacia que tiene cada una en la detección de fraudes, comparando estas entre sí.

3 Código y Resultados de datos

3.1 Construcción y funcionamiento del código

En este proyecto se utilizó una base de datos de la IEEE Fraud Detection disponible en Kaggle [3], el cual simula operaciones reales en un entorno bancario con el objetivo de identificar transacciones fraudulentas. El mismo Dataset cuenta con cinco csv de los cuales se trabajaron con dos: `train_transaction.csv` y `train_identity.csv`, por ser los que ofrecen información directamente relevante para el problema de detección de fraude.

El archivo `train_transaction.csv` constituye la base del análisis. En él se encuentra el registro de cada transacción realizada, identificada de manera única por la columna `TransactionID`. Por otro lado, se incluye una columna llamada `isFraud` la cual varía entre dos valores, 1 para fraude y 0 para transacción legítima, la columna `TransactionDT` indica el tiempo y `TransactionAmt` indica el monto de la transacción. La columna `ProductCD` indica la categoría del producto, por último el csv finaliza con variables auxiliares desde la columna `v1` hasta la `v339`.

El segundo csv que se tomó en cuenta es el de `train_identity.csv`, el cual complementa los datos de las transacciones con información sobre la identidad y el entorno del usuario que realizó la operación. Entre las variables más relevantes se encuentran `DeviceType` y `DeviceInfo`, que hacen referencia al tipo y modelo del dispositivo utilizado. Estos datos son valiosos para la detección de patrones inusuales que puedan estar vinculados a situaciones de fraude. Ambos archivos fueron combinados utilizando la columna `TransactionID` como clave. No obstante, es importante mencionar que no todas las transacciones de la tabla principal tienen información asociada en el archivo de identidad, por lo que tras la fusión quedaron registros con datos faltantes.

El equipo utilizado para ejecutar el script, creado con Python, cuenta con una tarjeta gráfica nvidia con soporte CUDA, dicho equipo fue configurado para que TensorFlow no utilice optimizaciones de oneDNN ni GPU (Fig. 1.), y el proceso completo se realice únicamente en CPU. El código fue configurado para que se utilice 6 núcleos y 12 hilos del procesador (Fig. 2.). Además, se utilizaron las librerías, como Pandas para manipular los datos, Scikit-learn para el preprocesamiento y modelos de aprendizaje, y TensorFlow/Keras para la implementación del modelo de autoencoder (Fig. 3.).

```
# Desactivar las optimizaciones de oneDNN antes de importar TensorFlow
os.environ['TF_ENABLE_ONEDNN_OPTS'] = '0'
os.environ["CUDA_VISIBLE_DEVICES"] = "-1" # Usar solo CPU
```

Fig. 1. Desactivar las optimizaciones de oneDNN antes de importar TensorFlow.

```
# Configurar la CPU para usar múltiples hilos
physical_devices = tf.config.list_physical_devices('CPU')
if len(physical_devices) > 0:
    tf.config.set_visible_devices(physical_devices, 'CPU')
    tf.config.threading.set_intra_op_parallelism_threads(12)
    tf.config.threading.set_inter_op_parallelism_threads(12)

print("Configuración de CPU completada.")
```

Fig. 2. Configurar la CPU para usar múltiples hilos.

```
import tensorflow as tf
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import IsolationForest
from sklearn.svm import OneClassSVM
from keras.layers import Input, DenseS
from keras.models import Model
import numpy as np
```

Fig. 3. Librerías utilizadas.

En primera instancia, se cargaron los archivos `train_transaction.csv` y `train_identity.csv`. Se utiliza la columna `TransactionID` para combinar ambos archivos y para formar un solo `DataFrame`. En el preprocesamiento del `DataFrame`, se eliminan las columnas `TransactionID` y `isFraud` del conjunto de datos, ya que la primera es solo un identificador único de cada registro y la segunda columna se utilizará como etiqueta de referencia. Luego del preprocesamiento, se seleccionan únicamente las columnas numéricas y se completan los valores nulos con ceros, esto es para evitar problemas en el entrenamiento. Para asegurar una mejor convergencia de los modelos, los datos son normalizados utilizando `StandardScaler`, esta normalización de datos es un punto crucial para mejorar el rendimiento de los modelos de clasificación [9], transformando así todas las variables a una escala estándar.

Finalmente, el conjunto de datos preprocesado fue dividido en dos subconjuntos: uno de entrenamiento, compuesto por aproximadamente 255.000 registros, y otro de prueba, con cerca de 63.750 registros. Esta proporción del 80/20 se escogió para asegurar un volumen adecuado de datos tanto para el aprendizaje de los modelos como para su evaluación.

3.2 Modelos de aprendizaje

Se entrenaron tres modelos de aprendizaje automático, orientados a la detección de anomalías: Autoencoder, Isolation Forest y One-Class SVM (OCSVM). La elección

de estas técnicas corresponde a la necesidad de aplicar enfoques no supervisados, con capacidad de detectar comportamientos anómalos sin depender de un volumen significativo de datos etiquetados, este punto es importante ya que es una característica habitual en escenarios de fraude financiero, donde los casos positivos son escasos y desbalanceados.

El primer modelo fue un autoencoder funciona mediante una red neuronal, la cual está diseñada para reconstruir los datos de entrada y reconocer anomalías teniendo en cuenta la diferencia entre la entrada y la salida reconstruida. Este modelo es entrenado con 10 épocas con un lote del tamaño de 256 (Fig. 4.). La eficacia de los autoencoders radica en la capacidad para aprender representaciones comprimidas de datos normales, lo que permite identificar anomalías a través de errores de reconstrucción significativos [10].

```
print("Entrenando Autoencoder...")
input_dim = X_train.shape[1]
input_layer = Input(shape=(input_dim,))
encoded = Dense(64, activation='relu')(input_layer)
decoded = Dense(input_dim, activation='sigmoid')(encoded)

autoencoder = Model(input_layer, decoded)
autoencoder.compile(optimizer='adam', loss='mean_squared_error')

# Entrenar con medición de tiempo
start_time = time.time()
autoencoder.fit(X_train, X_train, epochs=10, batch_size=256, validation_data=(X_test, X_test), verbose=1)
end_time = time.time()
print(f"Autoencoder entrenado en {end_time - start_time:.2f} segundos.")
```

Fig. 4. Fragmento del código para entrenar el Autoencoder.

A continuación, se entrena un Isolation Forest. Esta técnica funciona mediante la creación de árboles de decisión que fraccionan aleatoriamente los datos, logrando aislar las observaciones anómalas en ramas más cortas debido a su naturaleza distinta. En este trabajo, el modelo fue entrenado sobre el conjunto completo de entrenamiento, ajustando una tasa de contaminación del 10%, en concordancia con el grado de desbalance de los datos (Fig. 5.). La capacidad de Isolation Forest para manejar datos de alta dimensión y la eficiencia computacional que tiene esta técnica, la hace altamente útil en escenarios en donde las anomalías son raras y diferentes de los datos normales [11].

```
print("Δ Entrenando Isolation Forest...")
start_time = time.time()
iso_forest = IsolationForest(contamination=0.1, n_jobs=-1)
iso_forest.fit(X_train)
end_time = time.time()
print(f"Isolation Forest entrenado en {end_time - start_time:.2f} segundos.")
```

Fig. 5. Fragmento del código para entrenar el Isolation Forest.

Finalmente, se implementó un modelo One-Class SVM, una técnica que aprende la distribución de la clase mayoritaria para detectar desviaciones en los datos. Se limita el conjunto de entrenamiento a 50,000 muestras, esto es para evitar problemas de rendimiento (Fig. 6.). Un ejemplo de la capacidad de detectar anomalías de esta técnica se presenta en el estudio One Class Support Vector Machine for Anomaly Detection

in the Communication Network Performance Data [12], donde se implementó un detector basado en OCSVM para identificar comportamientos anómalos en datos de rendimiento de redes.

```
print("Entrenando One-Class SVM...")

# Tomar una muestra de 50,000 filas para evitar bloqueos
sample_size = min(50000, len(X_train))
X_train_sample = X_train[:sample_size]

start_time = time.time()
one_class_svm = OneClassSVM(nu=0.1, kernel="rbf", gamma="scale", verbose=True)
one_class_svm.fit(X_train_sample)
end_time = time.time()

print(f"One-Class SVM entrenado en {end_time - start_time:.2f} segundos.")
```

Fig. 6. Fragmento del código para entrenar el OneClassSVM.

El código completo se encuentra disponible para su consulta en el siguiente repositorio de GitHub: <https://github.com/Debugzin/Fraude-Digital-y-Ciberseguridad>.

3.3 Resultados finales

Una vez entrenados los modelos, se generan predicciones sobre los datos de prueba para cada tipo de técnica. En principio para el autoencoder, se reconstruyen los datos y se calcula el error cuadrático medio (MSE) entre la entrada y la salida, definiendo un umbral para clasificar transacciones fraudulentas, los datos de salida se clasifican como normales (0) o anómalas (1).

Para los casos de Isolation Forest y el One-Class SVM se devuelven etiquetas indicando si una transacción es normal o sospechosa. Estas etiquetas se representan con el valor -1 indica una transacción sospechosa y 1 si es normal.

Finalmente, los resultados fueron almacenados en un archivo de tipo Excel, incluyendo el TransactionID, y la columna isFraud original, estas columnas acompañan a las predicciones de cada modelo como se muestra en la tabla 1.

Tabla 1. Tabla de resultados obtenidos después del entrenamiento de cada técnica.

TransactionID	isFraud	Autoencoder Prediction	Isolation Forest Prediction	One-Class SVM Prediction
3459432	0	0	1	1
3459433	0	0	1	1
3459434	0	0	1	1
3459435	0	0	1	1
3459436	0	0	1	1

Este archivo tiene un total de 118,109 registros, los cuales se utilizarán para el análisis final y así comparar el desempeño de cada técnica para determinar cuál de ellas logró una detección más efectiva de transacciones.

4 Análisis de los resultados

El código que se utilizó para procesar los resultados obtenidos en el punto anterior se encuentra disponible en el siguiente repositorio de GitHub: <https://github.com/Debugzin/Fraude-Digital-y-Ciberseguridad/blob/58446c58c00c40b20b460ab49072486c4992d5f5/Metricas.py>.

Este script permite generar métricas comparativas a partir de las predicciones realizadas por cada modelo. Para evaluar el desempeño de los modelos, se generaron cuatro gráficos principales: una matriz de confusión por modelo, curvas ROC junto con sus respectivos valores AUC y un gráfico de barras apiladas con los valores de TP, TN, FP y FN.

La matriz de confusión (Fig. 7.) permite evaluar la capacidad de cada modelo para clasificar correctamente transacciones legítimas y fraudulentas. En estos gráficos, el Autoencoder mostró un desempeño equilibrado, con una cantidad de verdaderos negativos de 108,984, esto significa que la técnica identifica correctamente transacciones legítimas en su mayoría. Sin embargo, tuvo ciertas dificultades al detectar algunos fraudes, mostrando un total de 3,218 sobre falsos negativos, fraudes que no fueron detectados por el modelo, aunque mantuvo una baja cantidad los falsos positivos 4,882. Por otro lado, Isolation Forest y One-Class SVM tuvieron más problemas al etiquetar demasiadas transacciones legítimas como fraudulentas (103,956 y 103,736 falsos positivos respectivamente), esto quiere decir que comparando las distintas técnicas la de Autoencoder tuvo una mejor performance.

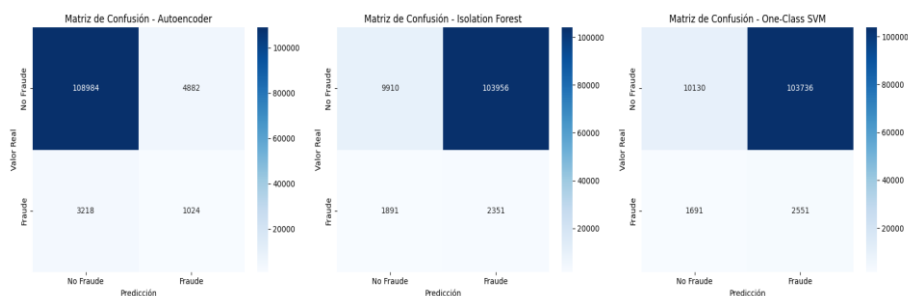


Fig. 7. Matriz de confusión de las distintas técnicas.

Otro gráfico representativo de los resultados es el de barras (Fig. 8.). Este gráfico contempla 4 puntos importantes, los cuales son los verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN). Se observa claramente que el modelo Autoencoder domina en predicciones correctas para casos legítimos (TN), pero por otro lado las barras de Isolation Forest y One-Class SVM son altamente dominadas por falsos positivos (FP), esto indica un exceso de alertas incorrectas. Este resultado resalta nuevamente la ventaja del Autoencoder en mantener un equilibrio entre precisión y sensibilidad, frente a los otros modelos que presentan un desempeño menos eficaz debido al alto número de errores.

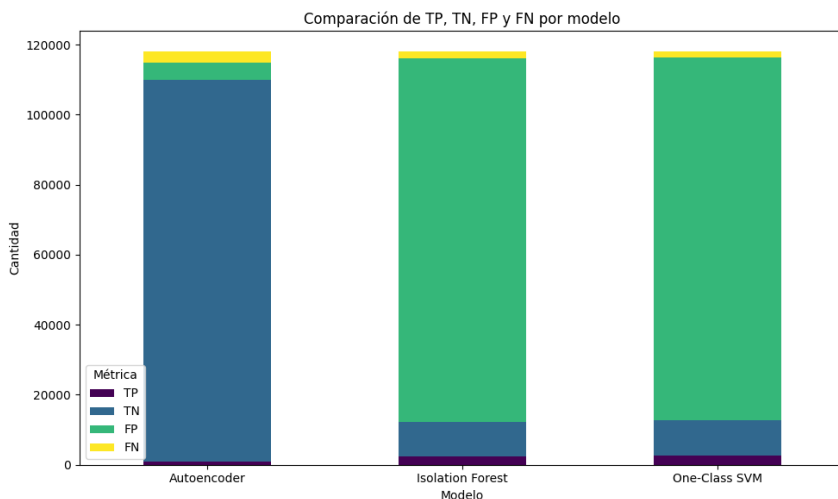


Fig. 8. Gráfico de barras comparando TP, TN, FP y FN.

Por último, la curva ROC, junto con la métrica AUC (Área Bajo la Curva) (Fig. 9), evalúa la capacidad de los modelos para diferenciar correctamente entre transacciones fraudulentas y no fraudulentas.

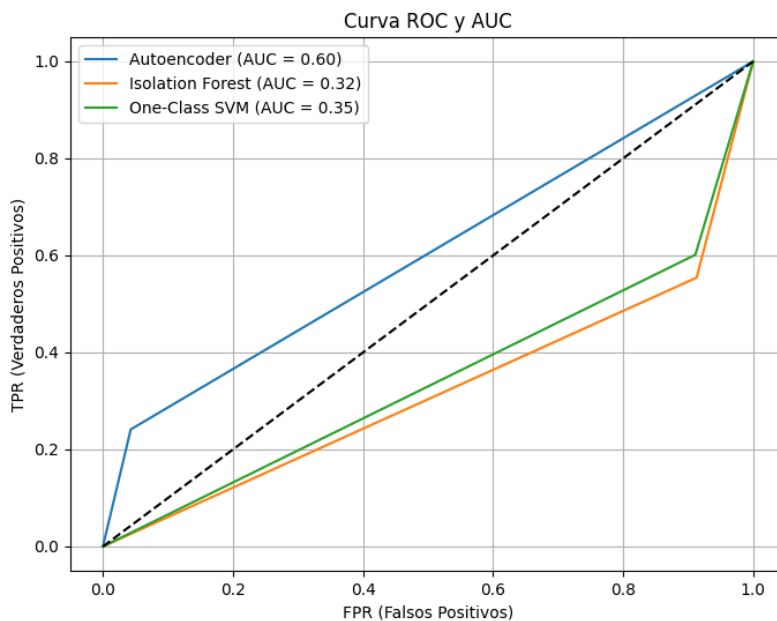


Fig. 9. Curva ROC y AUC.

En la Figura 9, el Autoencoder obtuvo el valor moderado de AUC (0.60), el cual es el más alto entre las 3 técnicas, esto sugiere una capacidad aceptable del modelo para detectar fraudes, y con espacio para mejorar. En caso contrario, Isolation Forest (0.32) y One-Class SVM (0.35) mostraron valores muy pobres en comparación al Autoencoder e inferior al valor de referencia (0.5), esto significa que la capacidad de detectar fraudes de estas técnicas es peor al que podría obtenerse simplemente mediante clasificación aleatoria.

Los resultados obtenidos a lo largo de la investigación permiten establecer algunos puntos relevantes sobre el comportamiento de cada técnica frente al problema de la detección de fraudes. En primer lugar, queda en evidencia que los enfoques de detección de anomalías no supervisados tienen un enorme potencial para este tipo de escenarios, donde los casos etiquetados de fraude suelen ser escasos, desbalanceados o directamente inexistentes. Sin embargo, también se confirma que no todas las técnicas responden de igual manera a esta problemática, siendo importante evaluar sus características operativas y su capacidad para discriminar entre patrones válidos y anómalos.

Un hallazgo significativo de esta investigación es que el modelo Autoencoder mostró una capacidad más estable para mantener bajo control el número de falsos positivos, lo cual es un factor imprescindible en la aplicación práctica de cualquier sistema de detección de fraudes. En contextos reales, un modelo que genera demasiadas alertas incorrectas puede saturar los sistemas y afectar negativamente la experiencia del usuario final. La efectividad del Autoencoder para generalizar el comportamiento normal y su baja tasa de errores en la clasificación de transacciones legítimas lo posicionan como una alternativa sólida, especialmente en fases preliminares o como sistema complementario dentro de un pipeline híbrido. Por otro lado, se deja en evidencia la necesidad de perfeccionar las estrategias de detección de fraudes mediante la exploración de modelos mixtos, ajustados a cada entorno de aplicación. Incorporar una etapa de calibración automática de umbrales, aplicar técnicas de selección de características, o incluso fusionar la salida de distintos modelos a través de enfoques de ensamblado podrían representar caminos valiosos a futuro.

Por lo que se propone un modelo de referencia cuyo núcleo sea un Autoencoder entrenado exclusivamente con transacciones legítimas, capaz de aprender el comportamiento habitual de los datos y detectar desviaciones mediante el error de reconstrucción. La salida del modelo se complementaría con un umbral ajustado con datos históricos, y podría integrarse en un esquema modular que incluya una etapa de filtrado automático seguida de una verificación manual o semiautomática. Esta estructura también permitiría incorporar modelos adicionales, como Isolation Forest, en paralelo o en cascada para contextos más complejos. Este modelo sería óptimo por su bajo costo, escalabilidad y capacidad de adaptarse a datos no etiquetados.

6 Discussion

Los resultados confirman que las técnicas no supervisadas, en particular el modelo Autoencoder, ofrecen una alternativa viable y eficiente para detectar fraudes electrónicos en contextos con datos escasos o no etiquetados. Esta técnica logra una alta precisión y baja tasa de falsos positivos, posicionándose como una herramienta pro-

metedora frente a desafíos reales. ¿Qué implicaría para una organización reducir significativamente las alertas falsas sin perder efectividad en la detección? ¿Cómo cambiaría la gestión del fraude si se pudiera automatizar la identificación de patrones anómalos sin necesidad de grandes volúmenes de datos históricos etiquetados? En contraste, técnicas como Isolation Forest y One-Class SVM demostraron mayor sensibilidad a la configuración de sus parámetros y una tendencia a generar alertas excesivas, lo que limita su aplicabilidad directa, aunque no descarta su utilidad dentro de esquemas híbridos o como filtros secundarios.

Este trabajo evidencia sobre la necesidad de calibrar correctamente los modelos y adaptar sus salidas a las particularidades del dominio. La utilización de un dataset real y la comparación entre tres enfoques diferentes, junto a la creación del código son fortalezas de la investigación. Sin embargo, también se detectan limitaciones: la falta de ajuste de hiperparámetros, la no inclusión de variables categóricas enriquecidas y la necesidad de explorar otros métodos de evaluación. De cara al futuro, ¿sería posible mejorar aún más la precisión del Autoencoder incorporando arquitectura profunda o aprendizaje semisupervisado? ¿Qué impacto tendría en entornos altamente dinámicos como los pagos en línea o los sistemas bancarios automatizados? Estas preguntas abren nuevas líneas de investigación para seguir perfeccionando la detección de fraudes digitales en entornos reales y cambiantes.

5 Conclusiones

Los resultados obtenidos en esta investigación demuestran la potencia de cada modelo, demostrando que la técnica del Autoencoder es una de las más poderosas para las detecciones de fraudes electrónicos. Logrando una alta probabilidad de detección y una baja probabilidad de falsos positivos. En cambio, los modelos Isolation Forest y One-Class SVM mostraron un desempeño más limitado, especialmente debido al elevado número de falsos positivos que generaron. Este comportamiento, demuestra la necesidad de aplicar ajustes de parámetros más robustos, incluir selección de variables o modificar la interpretación de las salidas del modelo. Aun así, ambos modelos pueden resultar útiles como componentes de verificación dentro de sistemas más amplios y no deben descartarse completamente.

Es importante remarcar que la mejora continua de estas técnicas es importante, dado que los ciberdelincuentes se encuentran en constante evolución, desarrollando nuevas formas de vulnerar sistemas y explotar fallas que aún no han sido abordadas. En este sentido, el aporte de este trabajo se enmarca en ofrecer recomendaciones como, priorizar modelos capaces de adaptarse a datos no etiquetados, en este caso serían los Autoencoders; evaluar cuidadosamente las tasas de falsos positivos; y desarrollar soluciones híbridas que puedan compensar las debilidades individuales de cada técnica. Este último enfoque, en particular, tiene un alto potencial para convertirse en una herramienta eficaz de prevención de fraudes, al combinar la precisión de modelos como el Autoencoder con el respaldo adicional de métodos complementarios, logrando así una detección más robusta y adaptable a la actualidad.

Referencias

- [1] Pérez González, G. A. (2021). Detección de transacciones fraudulentas en tarjetas de crédito mediante el uso de modelos de Machine Learning [Trabajo de grado, Universidad de los Andes]. Repositorio Institucional Uniandes. <http://hdl.handle.net/1992/53571>
- [2] Sánchez, J. S. (2022). Detección de anomalías transaccionales usando técnicas de machine learning con grafos [Trabajo de grado, Universidad del Rosario]. Repositorio Institucional. https://doi.org/10.48713/10336_40983
- [3] Al-Shabi, M. A. (2019). Credit card fraud detection using autoencoder model in unbalanced datasets. *Journal of Advances in Mathematics and Computer Science*, 33(5), 1–16. <https://doi.org/10.9734/jamcs/2019/v33i530192>
- [4] Mohammed, M. K., & Telek, M. (2023). Anomaly detection using combination of autoencoder and isolation forest. *Proceedings of the 2023 World Intelligent Systems Conference (WINS)*. <https://doi.org/10.33111/WINS2023-005>
- [5] Kaggle. (s.f.). IEEE-CIS Fraud Detection Competition. <https://www.kaggle.com/competitions/ieec-fraud-detection>
- [6] Lesouple, J., Baudoin, D., Sousa, J., & Remy, G. (2021). Generalized isolation forest for anomaly detection. *Pattern Recognition Letters*, 149, 109–119. <https://doi.org/10.1016/j.patrec.2021.05.022>
- [7] Qiao, Y., Wu, K., & Jin, P. (2023). Efficient anomaly detection for high-dimensional sensing data with One-Class support vector machine. *IEEE Transactions on Knowledge and Data Engineering*, 35(1), 404–417. <https://doi.org/10.1109/TKDE.2021.3077046>
- [8] Juniper Research. (2023). Online payment fraud: Market forecasts, emerging threats & segment analysis 2023–2028. <https://www.juniperresearch.com/research/fintech-payments/fraud-security/online-payment-fraud-research-report/>
- [9] Raju, V. N. G., Lakshmi, K. P., Jain, V. M., Kalidindi, A., & Padma, V. (2020). Study the influence of normalization/transformation process on the accuracy of supervised classification. In *Proceedings of the 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)* (pp. [número de páginas si lo tenés]). <https://doi.org/10.1109/ICSSIT48917.2020.9214160>
- [10] Ciencia de Datos. (s.f.). Detección de anomalías con autoencoders y Python. <https://cienciadedatos.net/documentos/py32-deteccion-anomalias-autoencoder-python>
- [11] DataCamp. (s.f.). Guía del bosque del aislamiento: Explicación e implementación en Python. <https://www.datacamp.com/es/tutorial/isolation-forest>
- [12] Zhang, R., Zhang, S., Muthuraman, S., & Jianmin, J. (2007). One class support vector machine for anomaly detection in the communication network performance data. *Proceedings of the 5th Conference on Applied Electromagnetics, Wireless and Optical Communications*, 31–37. <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=79bcceb0166a19bc82e07fbdf6cc7ff026550bd0>