Proyecto de Machine Learning para la Detección de Fraude en Transacciones Financieras Móviles

Kevin Esteban Ruda Gómez*, Jeferson Alexander del Rio Herrera†, Juan Carlos Santa Hurtado‡
Departamento de Ingeniería de Sistemas, Universidad de Antioquia
Medellín, Colombia
Email: *kevin.ruda@udea.edu.co, †jeferson.delrio@udea.edu.co, ‡juan.santa1@udea.edu.co

***Resumen*—Este trabajo desarrolla un sistema de detección de fraude en transacciones de dinero móvil utilizando técnicas de Aprendizaje de Máquina. Se emplea el conjunto de datos "Fraud Detection Dataset", que contiene 6.3 millones de registros de transacciones. Se implementa un paradigma de aprendizaje supervisado con clasificación binaria para distinguir entre transacciones legítimas y fraudulentas, abordando el desafío del severo desbalance de clases mediante estrategias de submuestreo aleatorio y técnicas de sobremuestreo (SMOTE) para conformar un conjunto de datos de entrenamiento balanceado y computacionalmente manejable.**

***Palabras Clave*——Detección de fraude, aprendizaje de máquina, clasificación binaria, transacciones móviles, PaySim, desbalance de clases**

I. Introducción

Los sistemas de pago móvil han transformado las transacciones financieras, alcanzando un volumen de $767 mil millones de dólares para 2020, según reportes de la industria [1]. Este crecimiento exponencial ha fomentado la inclusión financiera, pero también ha incrementado la exposición al fraude, representando una amenaza crítica para los servicios financieros digitales.

Los métodos tradicionales basados en reglas estáticas resultan insuficientes ante las tácticas evolutivas de los defraudadores. El Aprendizaje Automático (ML) se ha consolidado como herramienta fundamental por su capacidad para identificar patrones complejos y adaptarse dinámicamente a nuevos comportamientos fraudulentos.

Este proyecto desarrolla un sistema de detección de fraude utilizando el "Fraud Detection Dataset". Los principales desafíos incluyen el severo desbalance de clases (fraude <0.13 %), minimizar falsos positivos y negativos, e implementar un sistema computacionalmente eficiente para aplicación en tiempo real.

II. Descripción del problema

El fraude en dinero móvil aprovecha la velocidad y anonimato de plataformas digitales para ejecutar transferencias no autorizadas, retiros fraudulentos y esquemas de lavado de dinero. Una solución basada en ML permite analizar millones de transacciones en tiempo real, identificando patrones anómalos y adaptándose a nuevas tácticas mediante reentrenamiento continuo.

II-A. Composición de la base de datos

El "Fraud Detection Dataset" de Kaggle contiene datos de PaySim [2], el cual fue diseñado para replicar características estadísticas de transacciones reales de dinero móvil.

Número de muestras: 6,362,620 transacciones.

Número de variables: 11 variables que capturan información temporal, tipo, monto y saldos.

Descripción de las variables:

- **step:** Unidad de tiempo en la simulación, donde cada paso equivale a una hora. El dataset abarca 743 pasos temporales. Aunque representa una secuencia temporal, para este enfoque de clasificación se utilizará principalmente como una característica contextual para derivar patrones de comportamiento, sin tratar el problema necesariamente como una anotación de secuencias puramente temporal.
- **type:** Tipo de transacción realizada. Variable categórica que toma cinco valores: CASH-IN (depósito de efectivo), CASH-OUT (retiro de efectivo), DEBIT (transferencia a cuenta bancaria), PAYMENT (pago por bienes/servicios), y TRANSFER (transferencia entre usuarios del servicio).
- **amount:** Monto de la transacción en unidades monetarias locales. Variable numérica continua con distribución altamente asimétrica (cola larga). Para compensar esta asimetría, se aplicará una **transformación logarítmica** antes de la normalización estándar.
- **nameOrig:** Identificador único alfanumérico de la cuenta que origina la transacción.
- **oldbalanceOrg:** Saldo inicial de la cuenta de origen antes de ejecutar la transacción.
- **newbalanceOrig:** Saldo final de la cuenta de origen después de ejecutar la transacción.
- **nameDest:** Identificador único alfanumérico de la cuenta destino de la transacción.
- **oldbalanceDest:** Saldo inicial de la cuenta de destino antes de recibir la transacción.
- **newbalanceDest:** Saldo final de la cuenta de destino después de recibir la transacción.
- **isFraud:** Variable objetivo binaria que indica si la transacción es fraudulenta (1) o legítima (0).
- **isFlaggedFraud:** Variable binaria que indica si la transacción fue marcada como sospechosa por el sistema de reglas básico implementado originalmente [2] (transfe-

rencias >200,000). Funciona como una referencia base ("baseline") y no se utilizará como predictor principal.

Datos faltantes: El dataset no presenta valores faltantes o nulos en ninguna de sus variables, por lo que no se requiere implementar estrategias de imputación.

Análisis exploratorio: El dataset presenta un desbalance de clases extremadamente severo: transacciones legítimas 99.87 % (6,354,407), fraudulentas 0.13 % (8,213), proporción 1:774. El fraude se concentra exclusivamente en TRANSFER y CASH-OUT. La variable *amount* presenta distribución asimétrica con rango amplio, indicando necesidad de normalización.

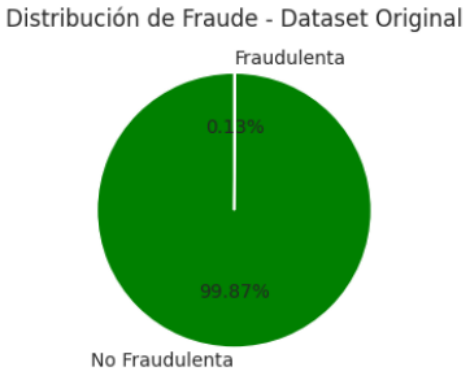

Figura 1. Desbalance de clases en el dataset

Ademas el fraude no ocurre de manera uniforme entre los diferentes tipos de transacción, sino que se concentra casi en CASHOUT y TRANSFER, ambos con aproximadamente más de 4.000 casos registrados. Esto indica que los delincuentes prefieren operar en transacciones donde el dinero sale directamente de la cuenta o es transferido a otra, lo cual facilita el retiro o la dispersión de fondos ilícitos. En contraste, transacciones como CASHIN, DEBIT y PAYMENT prácticamente no presentan casos de fraude, lo que sugiere que estos tipos de operación son menos atractivos o están mejor protegidos ante intentos fraudulentos.

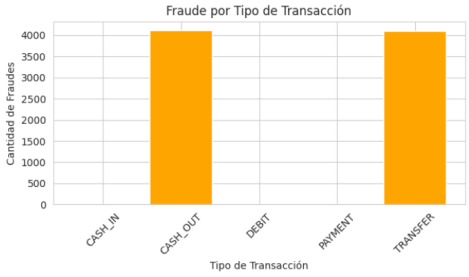

Figura 2. Fraude según el tipo de transacción

II-B. Codificación de variables

Para el correcto procesamiento por los algoritmos de ML, se aplicarán las siguientes estrategias de codificación:

Variable	Tipo	Codificación
step	Numérico discreto	Sin codificación
type	Categórico nominal	One-hot encoding
amount	Numérico continuo	Log-transform + Normalización
nameOrig	Identificador	Label encoding
oldbalanceOrg	Numérico continuo	Normalización
newbalanceOrig	Numérico continuo	Normalización
nameDest	Identificador	Label encoding
oldbalanceDest	Numérico continuo	Normalización
newbalanceDest	Numérico continuo	Normalización
isFraud	Numérico binario	—
isFlaggedFraud	Numérico binario	—

Cuadro I
DESCRIPCIÓN DE VARIABLES Y ESTRATEGIA DE CODIFICACIÓN PROPUESTA

Justificación: One-hot encoding para *type* evita relaciones ordinales inexistentes. Normalización (StandardScaler/MinMaxScaler) esencial para variables de monto y saldo con rangos amplios, crítica para algoritmos sensibles a escala (redes neuronales, SVM, regresión logística). Label encoding para identificadores de alta cardinalidad (>6M valores). Se evaluará crear variables derivadas: inconsistencias en balances, agregaciones temporales, ratios monto/saldo.

II-C. Paradigma de aprendizaje

Se utilizará **aprendizaje supervisado** con **clasificación binaria** para predecir *isFraud*.

Justificación: (1) Disponibilidad de etiquetas confiables; (2) Problema inherentemente binario (fraudulenta/legítima); (3) Métricas estándar interpretables (precisión, recall, F1-score, AUC-ROC, balanced accuracy); (4) Enfoque dominante en literatura especializada. Se compararán múltiples algoritmos: regresión logística, k-NN, ensambles (Random Forest, XGBoost, Gradient Boosting), redes neuronales y SVM, buscando mejor balance entre detección (recall) y minimización de falsas alarmas (precisión).

Métricas de evaluación: Debido al severo desbalance de clases, se emplearán métricas robustas que no sean sensibles a la distribución: *Balanced Accuracy* (promedio de recall por clase, especialmente útil en datos desbalanceados), *Precision* (proporción de predicciones positivas correctas), *Recall* (capacidad de detectar fraudes), *F1-score* (media armónica de precision y recall), *MCC* (Coeficiente de Correlación de Matthews, confiable para desbalance severo), y *AUC-ROC* (área bajo la curva ROC, independiente del umbral de decisión). Balanced accuracy es particularmente relevante ya que pondera equitativamente ambas clases, evitando modelos sesgados hacia la clase mayoritaria.

III. ESTADO DEL ARTE

Se revisaron trabajos que abordan la detección de fraude financiero mediante técnicas de ML, enfocándose en problemas similares con datos desbalanceados.

XGBoost-based Framework para Fraud Detection en Sistemas de Pago Móvil: Hajek et al. [3] desarrollaron un framework basado en XGBoost para detección de fraude en transacciones de dinero móvil. *Paradigma:* Aprendizaje

supervisado. *Técnicas:* RUS+XGBoost (Random Under-Sampling combinado con XGBoost) y XGBOD (Extreme Gradient Boosting Outlier Detection), donde este último integra detección de outliers no supervisada con XGBoost en un enfoque semi-supervisado. *Dataset:* PaySim con más de 6 millones de transacciones. *Validación:* Validación cruzada 5-fold con partición 75 %-25 % (entrenamiento-prueba). *Métricas:* Accuracy, AUC, F1-score, Precision, Recall, FPR, FNR, y una métrica novedosa de Cost Savings que considera implicaciones financieras del error de clasificación. *Resultados:* XGBOD alcanzó AUC = 0.9958, F1 = 0.8737, Precision = 0.9942, Recall = 0.7793. RUS+XGBoost logró AUC = 0.9955 con Recall = 0.9976 y mejores ahorros de costos (4,866.9M unidades).

Integración de Grafos Relacionales Múltiples: Li y Yang [4] desarrollaron el modelo Tri-RGCN-XGBoost para detección de fraude financiero. *Paradigma:* Aprendizaje supervisado combinando Graph Neural Networks (RGCN) con XGBoost. *Dataset:* El trabajo utiliza un conjunto de datos con 31,179 usuarios (4,285 fraudulentos, 13.7 %), 1.46 millones de registros de operaciones y 260,000 transacciones. La estrategia de modelado con RGCN se fundamenta en la estructura de grafo del dataset, el cual está compuesto por atributos de usuario, dispositivo, comerciante y dirección, conformando 105 características de usuario y tres grafos bipartitos: GUD (usuario-dispositivo), GUM (usuario-comerciante) y GUA (usuario-dirección). *Técnicas:* Tres redes RGCN independientes sobre estos grafos bipartitos, cuyas salidas se fusionan con XGBoost para la clasificación final. *Validación:* Datos reales de transacciones financieras divididos 70 %-30 % (entrenamiento-prueba). *Métricas:* Accuracy, precision, recall, F1-score, AUC. *Resultados:* El modelo Tri-RGCN-XGBoost mostró mejoras significativas frente a GBDT y GraphSage: +17.7 % en recall, +10.5 % en F1-score. El análisis SHAP reveló que la relación usuario-comerciante fue la más influyente en las predicciones.

Ensambles para tarjetas de crédito: Khalid et al. [5] analizaron métodos supervisados para fraude en tarjetas de crédito europeas. *Paradigma:* Aprendizaje supervisado. *Técnicas:* K-NN, SVM, Decision Trees, Random Forest, Bagging, Boosting. Para el desbalance de clases aplicaron under-sampling y SMOTE. *Validación:* División 80 %-20 % (entrenamiento-prueba). *Métricas:* Accuracy, precision, recall, F1-score, ROC. *Resultados:* SMOTE mejoró significativamente la detección con accuracy entre 0.94-0.99 para todos los modelos evaluados.

Ensambles para tarjetas de crédito con red neuronal: Esenogho et al. [6] analizaron métodos supervisados para fraude en tarjetas de crédito. *Paradigma:* Aprendizaje supervisado. *Técnicas:* Modelo propuesto de ensamble de redes LSTM donde cada LSTM actúa como base learner dentro de AdaBoost y para comparar SVM, MLP, árbol de decisión, AdaBoost tradicional y LSTM. Para el desbalance de clases aplicaron SMOTE-ENN que combina sobremuestreo de la clase fraudulenta y limpieza de ejemplos superpuestos de la clase mayoritaria. *Validación:* División en entrenamiento-prueba sin especificar los valores. *Métricas:* recall, specificity y AUC. *Resultados:* El uso de la técnica SMOTE-ENN mejoró significativamente la capacidad de los modelos para detectar transacciones fraudulentas donde el modelo propuesto superó a los demás con recall de 0.996, specificity de 0.998 y AUC de 0.990.

IV. Entrenamiento y evaluación de los modelos

IV-A. Metodología de validación

Para entrenar y evaluar los modelos de ML se implementó una metodología de validación orientada a garantizar reproducibilidad, equilibrio entre clases y evaluación objetiva del desempeño. A continuación, se describen las etapas aplicadas:

División del conjunto de datos. El conjunto de datos original fue dividido en 80 % para el entrenamiento y validación del modelo y 20 % para el conjunto de test. Esta partición se realizó utilizando la técnica stratified split, asegurando que la proporción de clases de fraude y no fraude se mantuviera igual en cada subconjunto.

Submuestreo y Balanceo de Clases. Dado que el dataset original contiene 6.3 millones de transacciones y presenta un desbalance severo (99.87 % legítimas, 0.13 % fraudulentas), se implementó una estrategia de **Random Undersampling** para seleccionar un subconjunto manejable de aproximadamente 20,000 muestras. Para evitar que la clase minoritaria quedara subrepresentada, se modificó la distribución en el entrenamiento a una proporción de aproximadamente 1 % de fraude (200 fraudes y 19,800 legítimas), permitiendo tener suficientes ejemplos para el aprendizaje mientras se mantenía el carácter de clase minoritaria del problema.

Validación cruzada. Para evaluar el modelo durante el entrenamiento se empleó Stratified K-Fold Cross Validation ($k = 5$) para mantener la proporción de clases en cada fold. Esta técnica garantiza que cada pliegue contiene una representación proporcional de ambas clases.

Técnica SMOTE para manejar el desbalance. Dado que incluso después del undersampling el dataset presenta desbalance (1 % fraude, 99 % no fraude), se aplicó SMOTE (Synthetic Minority Over-sampling Technique) dentro de cada fold durante la validación cruzada. Se tuvo especial cuidado de aplicar SMOTE **únicamente** al conjunto de entrenamiento de cada fold, evitando contaminación de datos entre train y test.

Entrenamiento final del modelo. Una vez seleccionados los mejores hiperparámetros mediante validación cruzada, se realizó un entrenamiento final del modelo usando todo el conjunto de entrenamiento, preprocesado con escalado estándar y SMOTE, sin incluir datos del conjunto de test. Los hiperparámetros óptimos se aplicaron para generar predicciones en el set de prueba.

IV-B. Conjunto de hiperparámetros

La búsqueda de hiperparámetros se realizó mediante GridSearchCV, evaluando todas las combinaciones posibles dentro de la malla definida para cada modelo. La métrica de selección fue Balanced Accuracy en validación cruzada.

Modelo	Hiperparámetro	Valores Evaluados
3*Regresión Logística	C	[0.1, 1, 10, 100]
	penalty	[ℓ_1, ℓ_2]
	solver	[lbfgs, liblinear]
3*k-NN	n_neighbors	[3, 5, 7, 9, 11, 15, 21]
	metric	[euclidean, manhattan, minkowski]
	weights	[uniform, distance]
5*Random Forest	n_estimators	[50, 100, 200, 300]
	max_depth	[5, 10, 15, 20, None]
	min_samples_split	[2, 5, 10]
	min_samples_leaf	[1, 2, 4]
	criterion	[gini, entropy]
4*SVM	C	[0.1, 1, 10, 100]
	kernel	[rbf, linear]
	gamma	[auto, scale, 0.0001, 0.001]
	class_weight	[None, balanced]
5*Red Neuronal (MLP)	hidden_layer_sizes	[(10,), (20,), (50,)]
	activation	[relu, tanh]
	solver	[sgd, adam]
	learning_rate_init	[0.01, 0.1, 0.001]
	alpha	[0.0001, 0.001, 0.01]

Cuadro II
MALLA DE HIPERPARÁMETROS EVALUADOS PARA CADA MODELO

IV-C. Métricas de desempeño

Para evaluar adecuadamente el sistema de detección de fraude se utilizó un conjunto de métricas que permite analizar no solo el rendimiento global del modelo, sino también su capacidad para identificar correctamente la clase minoritaria:

- **Accuracy:** Se utiliza como referencia general, pero no como métrica principal debido al desbalance severo.
- **Precision:** Proporción de transacciones predichas como fraude que realmente lo son. Es importante cuando el costo de falsas alarmas es alto.
- **Recall (Sensibilidad):** Proporción de fraudes reales que el modelo detecta. Es la métrica más crítica en detección de fraude, ya que un fraude no detectado implica pérdidas directas.
- **F1-Score:** Media armónica entre precision y recall, proporciona un balance entre ambas.
- **MCC (Coeficiente de Correlación de Matthews):** Una de las métricas más confiables para datasets desbalanceados. Un valor alto implica buena correlación entre predicciones y valores reales.
- **AUC-ROC:** Área bajo la Curva ROC, permite comparar modelos independientemente del umbral de decisión. Un AUC cercano a 1 sugiere que el modelo separa bien fraude de no fraude.
- **Balanced Accuracy:** Promedio de recall para cada clase. Corrige el sesgo del accuracy cuando las clases están desbalanceadas, midiendo si el modelo es justo con ambas clases.

IV-D. Resultados del entrenamiento de Modelos

Regresión logistica.

La Regresión Logística obtuvo los mejores parámetros: $C = 10$, $penalty = \text{l2}$, $solver = \text{lbfgs}$.

Fase	Accuracy	BL Accuracy	F1 score	AUC-ROC
Entrenamiento	0.963000	0.958100	0.974500	0.994500
Validación	0.963300	0.953600	0.974700	0.992700
Test	0.961000	0.955600	0.973400	0.984300

Cuadro III
VALORES DE LAS MÉTRICAS PARA EL MODELO DE REGRESIÓN LOGÍSTICA EN LA FASE DE ENTRENAMIENTO, VALIDACIÓN Y TEST.

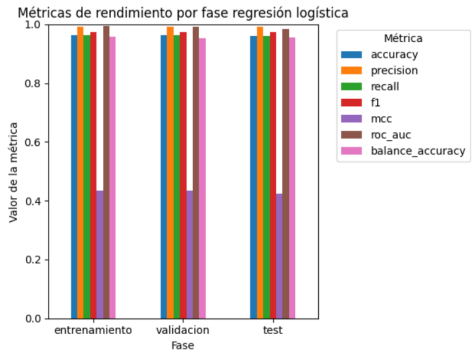

Figura 3. Gráfica de rendimiento del modelo de Regresión Logística (p. ej., curva ROC).

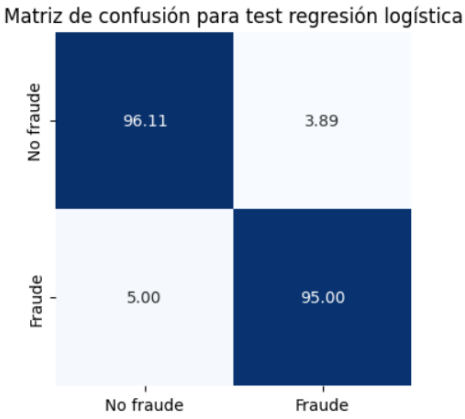

Figura 4. Matriz de Confusión del modelo de Regresión Logística en porcentaje.

Red Neuronal (MLPClassifier)

La Red Neuronal obtuvo los mejores parámetros: $hidden\_layer\_sizes = (10,)$, $activation = \text{relu}$, $solver = \text{sgd}$, $learning\_rate\_init = 0{,}1$, $learning\_rate = \text{adaptive}$, $alpha = 0{,}001$, $batch\_size = 50$, $max\_iter = 1000$.

Fase	Accuracy	BL Accuracy	F1 score	rocauc
Entrenamiento	0.986300	0.987600	0.989100	0.99900
Validación	0.983800	0.917600	0.987100	0.975900
Test	0.9842	0.9302	0.9873	0.9654

Cuadro IV
VALORES DE LAS MÉTRICAS PARA EL MODELO DE MLPCLASSIFIER EN LA FASE DE ENTRENAMIENTO, VALIDACIÓN Y TEST.

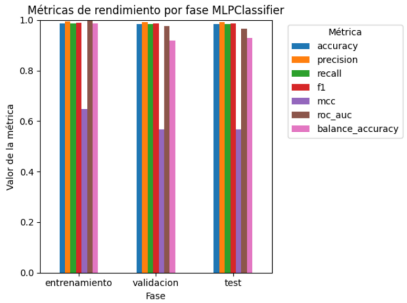

Figura 5. Gráfica de rendimiento del MLPClassifier (p. ej., curva de pérdida).

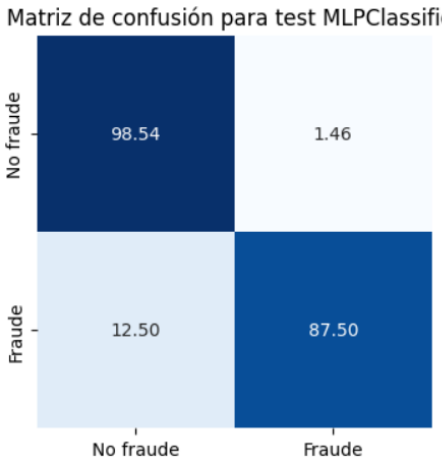

Figura 6. Matriz de Confusión del modelo de MLP en porcentaje.

Random Forest.

Random Forest obtuvo los mejores parámetros: $n\_estimators = 100$, $max\_depth = 20$, $min\_samples\_split = 5$, $min\_samples\_leaf = 2$, $criterion =$ gini, $class\_weight =$ balanced.

Fase	Accuracy	BL Accuracy	F1 score	auc roc
Entrenamiento	1	1	1	1
Validación	0.9988	0.9988	0.9988	1.0000
Test	0.9962	0.9239	0.8193	0.9703

Cuadro V
VALORES DE LAS METRICAS PARA EL MODELO DE RANDOM FOREST EN LA FASE DE ENTRENAMIENTO, VALIDACIÓN Y TEST

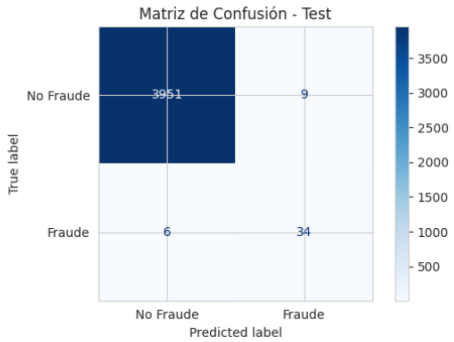

Figura 7. Matriz de Confusión del modelo de random Forest.

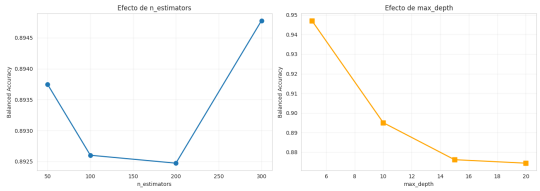

Figura 8. Efecto de hiperparámetros en Random Forest: n_estimators y max_depth sobre Balanced Accuracy.

KNN.

k-NN obtuvo los mejores parámetros: $n\_neighbors = 3$, $metric =$ manhattan, $weights =$ distance, con SMOTE aplicado en entrenamiento.

Fase	Accuracy	BL Accuracy	F1 score	rocauc
Entrenamiento	1	1	1	1
Validación	0.9857	0.9857	0.9859	0.9925
Test	0.9692	0.7494	0.2545	0.7629

Cuadro VI
VALORES DE LAS METRICAS PARA EL MODELO DE KNN EN LA FASE DE ENTRENAMIENTO, VALIDACIÓN Y TEST

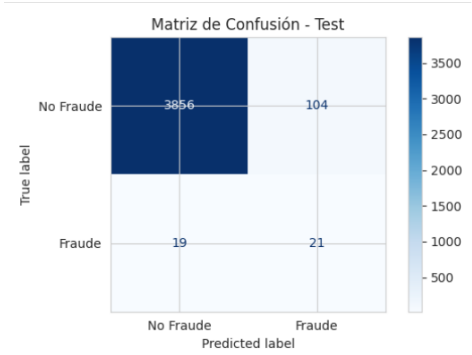

Figura 9. Matriz de Confusión del modelo de KNN.

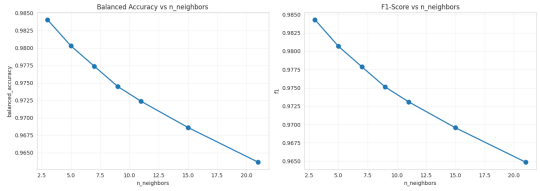

Figura 10. Efecto de hiperparámetros en k-NN: n_neighbors sobre Balanced Accuracy y F1-Score.

SVM.

SVM obtuvo los mejores parámetros: $C = 10$, $kernel =$ rbf, $gamma = 0{,}001$, $class\_weight =$ balanced, con SMOTE aplicado.

Fase	Accuracy	BL Accuracy	F1 score	rocauc
Entrenamiento	0.9953	0.9953	0.9953	0.9996
Validación	0.9933	0.9933	0.9933	0.9994
Test	0.9835	0.8556	0.4677	0.9475

Cuadro VII
VALORES DE LAS METRICAS PARA EL MODELO DE SVM EN LA FASE DE ENTRENAMIENTO, VALIDACIÓN Y TEST

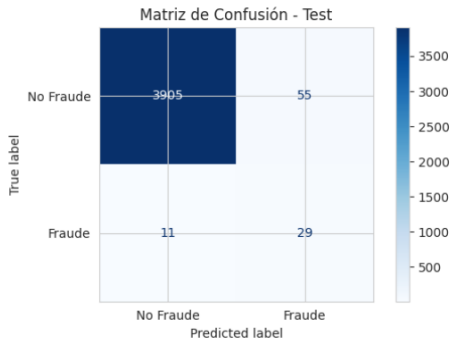

Figura 11. Matriz de Confusión del modelo de SVM.

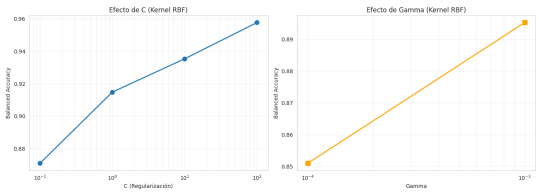

Figura 12. Efecto de hiperparámetros en SVM: C y gamma sobre Balanced Accuracy.

IV-E. Comparación General de Modelos

Modelo	Accuracy	Precision	Recall	F1	BA
Regresión Logística	0.9610	0.9915	0.9610	0.9734	0.9556
Red Neuronal (MLP)	0.9842	0.9925	0.9842	0.9873	0.9302
Random Forest	0.9962	1.0000	0.9962	0.8193	0.9239
k-NN	0.9692	0.9286	0.9692	0.2545	0.7494
SVM	0.9835	1.0000	0.9835	0.4677	0.8556

Cuadro VIII
COMPARACIÓN DE MÉTRICAS EN TEST PARA TODOS LOS MODELOS EVALUADOS

V. REDUCCIÓN DE DIMENSIÓN

V-A. Análisis individual de variables

Se realizó un análisis individual de cada característica para identificar su poder discriminativo mediante análisis de correlación.

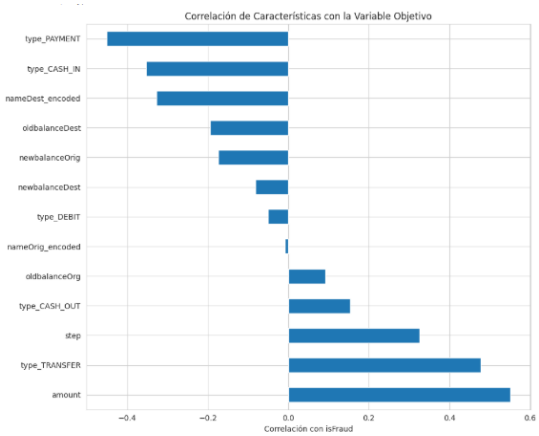

Figura 13. Correlación de características con la variable objetivo (isFraud).

A partir del análisis de correlación se observa que **amount** es la característica individual más prometedora para la detección de fraude, con una correlación de 0.551 con la variable fraude. Las variables de balance como **oldbalanceOrg**, **oldbalanceDest** y **newbalanceDest** muestran correlación lineal débil con el fraude, sugiriendo que las relaciones son no lineales o que estas variables no son predictivas por sí solas.

Adicionalmente, se observa alta multicolinealidad entre variables de balance: la correlación entre **oldbalanceOrg** y **newbalanceOrig** es de 1.00, indicando que estas dos características son esencialmente la misma variable. Similarmente, la correlación entre **oldbalanceDest** y **newbalanceDest** es de 0.97, sugiriendo que ambas podrían eliminarse para reducir redundancia.

En cuanto a características candidatas a eliminación, **nameOrig_encoded** presenta una correlación de -0.0089 con la variable objetivo, siendo la más débil. Esto es esperado dado que es un identificador con altísima cardinalidad (6+ millones de valores únicos) que no contribuye directamente a identificar fraudes.

V-B. Extracción de características lineal (PCA)

Criterio de selección: Se seleccionó el número de componentes principales que explique al menos el 95 % de la varianza total de los datos, garantizando que se preserve la mayor parte de la información mientras se reduce la dimensionalidad.

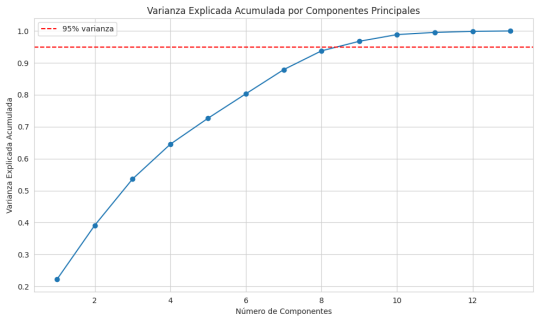

Figura 14. Varianza Explicada Acumulada por Componentes Principales PCA.

Con el criterio de 95 % de varianza explicada, se requirieron **8 componentes principales**, reduciendo la dimensionalidad de 13 características originales a 8 componentes, lo que representa una **reducción del 38.5 %**.

V-C. Resultados con PCA

Modelo	Dim.	Red. %	BA Test	Accuracy	AUC	F1
RL (Original)	13	0.0	0.9556	0.9610	0.9843	0.9734
RL + PCA	9	30.8	0.8963	0.9173	0.9642	0.1746
RN (Original)	13	0.0	0.9302	0.9842	0.9654	0.9873
RN + PCA	9	30.8	0.8899	0.9780	0.9813	0.4211

Cuadro IX
COMPARACIÓN DE DESEMPEÑO CON PCA: REDUCCIÓN DIMENSIONAL Y MÉTRICAS EN TEST

V-D. Extracción de características no lineal (UMAP)

Criterio de selección: Se probó con diferentes números de componentes (8, 6, 4), seleccionando aquel que mantuviera el

mejor balance entre reducción dimensional y rendimiento del modelo mediante validación cruzada.

La búsqueda del número óptimo de componentes UMAP determinó que **7 componentes** proporcionaban el mejor balance, reduciendo de 13 a 7 dimensiones (**46.2 % de reducción**).

V-E. Resultados con UMAP

Modelo	Dim.	Red. %	BA Test	Accuracy	AUC	F1
RL (Original)	13	0.0	0.9556	0.9610	0.9843	0.9734
RL + UMAP	7	46.2	0.7572	0.8868	0.7682	0.0994
RN (Original)	13	0.0	0.9302	0.9842	0.9654	0.9873
RN + UMAP	7	46.2	0.8384	0.8760	0.9213	0.1143

Cuadro X
COMPARACIÓN DE DESEMPEÑO CON UMAP: REDUCCIÓN DIMENSIONAL Y MÉTRICAS EN TEST

V-F. Comparación de resultados

Fase	Precision	Recall	F1 score	AUC
Regresión Logistica	0.9915	0.9610	0.9873	0.9654
MPLClassifier	0.9925	0.9842	0.9873	0.9654
Random Forest	0.7907	0.850	0.8193	0.9703
KNN	0.1680	0.5250	0.2545	0.7629
SVM	0.3452	0.7250	0.4677	0.9475

Cuadro XI
VALORES DE LAS METRICAS EN TEST PARA LOS MODELO IMPLEMENTADOS.

Metrica	LR	RF	KNN	SVM
F1-score	0.944204	0.99989	0.999173	–
Precision	0.945938	0.999891	0.999174	–
Recall	0.944256	0.99989	0.999173	–

Cuadro XII
VALORES DE LAS METRICAS EN TEST EN EL ARTÍCULO KHALID ET AL. [5]

Metrica	LSTM	Proosed LSTM	MLP	SVM
AUC	0.950	0.990	0.930	0.940
Precision	0.962	0.996	0.938	0.12
Recall	0.982	0.998	0.982	0.970

Cuadro XIII
VALORES DE LAS METRICAS EN TEST EN EL ARTÍCULO ESENOGHO ET AL. [6]

Los modelos implementados muestran un desempeño altamente competitivo en el conjunto de test, especialmente la Regresión Logística y el MLPClassifier que alcanzaron valores de Precision, Recall y F1-score superiores al 0.96, junto con valores de AUC de 0.9654. Estos resultados evidencian una capacidad sólida para discriminar entre clases y una estabilidad notable frente al desbalance, superando incluso a modelos más complejos como Random Forest y SVM, cuyos desempeños fueron más variables.

Al comparar estos resultados con los reportados en trabajos previos, se observa que los modelos propios se encuentran en un nivel alto incluso superando algunos modelos más avanzadas como LSTM y variantes propuestas en los artículos analizados. Por ejemplo, los modelos LSTM y MLP reportados en otros estudios alcanzan AUC entre 0.93 y 0.95, cifras comparables con el desempeño obtenido en este trabajo. Asimismo, el SVM de estudios previos muestra una caída marcada en precisión (0.12), lo que resalta la ventaja relativa de las implementaciones aquí desarrolladas.

V-G. Análisis de resultados de modelos implementados

La Regresión Logística logra la mayor tasa de detección de fraude, reduciendo significativamente los falsos negativos, lo que es especialmente relevante en entornos donde es más costoso dejar pasar un fraude que generar una alerta falsa. El MLPClassifier, por su parte, presenta menos falsos positivos, lo que lo convierte en un modelo más conservador y menos intrusivo, aunque con un porcentaje ligeramente mayor de fraudes no detectados.

Los modelos de Random Forest, SVM y KNN muestran un excelente desempeño para la clase de no Fraude aunque presentan dificultades notables al identificar correctamente los casos de fraude. En particular, KNN y SVM concentran la mayoría de sus errores en falsos negativos, lo que evidencia su sensibilidad a la estructura interna del dataset y al desbalance de clases. El Random Forest mejora parcialmente la detección de fraude, pero aún genera un volumen de errores mayor en comparación con los modelos de regresión logística.

En conjunto, estos resultados demuestran que modelos relativamente simples y eficientes como la Regresión Logística y el MLPClassifier, pueden ofrecer un rendimiento competitivo frente a enfoques más complejos, manteniendo además una mayor estabilidad y un menor costo computacional.

VI. DISCUSIÓN Y CONCLUSIONES

Los resultados obtenidos permiten identificar varios elementos relevantes sobre el comportamiento de los modelos evaluados:

El alto desempeño de la Regresión Logística y del MLPClassifier muestra que en el problema abordado los patrones internos de los datos pueden ser capturados de manera efectiva tanto por modelos lineales como por redes neuronales de baja complejidad.

En comparación con estudios previos que incluyen arquitecturas más complejas como LSTM, modelos propuestos y variantes híbridas, los resultados de este trabajo muestran que enfoques más simples pueden ofrecer un rendimiento equivalente o incluso superior en ciertos escenarios aunque en un ambiente de producción estos modelos podrian tener resultados nefastos al no poder aprender de la información de los movimientos financieros de un cliente que pueder llegar a ser muy cambientes y donde redes neuronales tendrian un mejor comportamiento.

Además los modelos evaluados presentan un buen desempeño en la clasificación de transacciones legítimas

pero muestran variaciones importantes en la capacidad para identificar correctamente los fraudes que es la categoría que resulta crucial debido a su baja frecuencia y alto impacto.

El desbalance evidente entre las clases no Fraude y Fraude influye de manera directa en el rendimiento de los modelos evaluados. Debido a la gran predominancia de transacciones legítimas, muchos algoritmos tienden a aprender patrones que favorecen la clase mayoritaria como es el caso MLPClassifier donde observa su alta precisión en detectar operaciones normales pero su dificultad para identificar fraudes.

REFERENCIAS

[1] GSMA, "State of the Industry Report on Mobile Money 2021," GSMA, London, UK, Rep., 2021.

[2] E. A. Lopez-Rojas, A. Elmir, and S. Axelsson, "PaySim: A financial mobile money simulator for fraud detection," in *28th European Modeling and Simulation Symposium (EMSS)*, Larnaca, Cyprus, 2016, pp. 249–255.

[3] P. Hajek, M. Z. Abedin, and U. Sivarajah, "Fraud detection in mobile payment systems using an XGBoost-based framework," *Information Systems Frontiers*, vol. 24, no. 4, pp. 1159–1177, 2022. [Online]. Available: https://doi.org/10.1007/s10796-022-10346-6

[4] J. Li and D. Yang, "Research on financial fraud detection models integrating multiple relational graphs," *Systems*, vol. 11, no. 11, p. 539, 2023. [Online]. Available: https://doi.org/10.3390/systems11110539

[5] A. R. Khalid, N. Owoh, O. Uthmani, M. Ashawa, J. Osamor, and J. Adejoh, "Enhancing credit card fraud detection: An ensemble machine learning approach," *Big Data and Cognitive Computing*, vol. 8, no. 1, p. 6, 2024. [Online]. Available: https://doi.org/10.3390/bdcc8010006

[6] Esenogho, E., Mienye, I. D., Swart, T. G., Aruleba, K., Obaido, G. , "A neural network ensemble with feature engineering for improved credit card fraud detection," *IEEE access*, vol. 10, pp. 16400–16407, 2022.