



Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

PROYECTO FINAL DE CIENCIA DE DATOS: Detección de Fraudes en Transacciones Bancarias

Alumno: Juan Diego Carreón Arelio

Matrícula: 202549423

Carrera: Ingeniería en Ciencia de Datos

Materia: Introducción a la Ciencia de Datos

Profesor: Jaime Alejandro Romero Sierra

Fecha: Noviembre 2025

Introducción

El presente proyecto aborda el análisis de una base de datos de transacciones bancarias con el objetivo de identificar operaciones potencialmente fraudulentas mediante el uso de técnicas de Ciencia de Datos y Machine Learning. Dado que los fraudes financieros representan un riesgo significativo para instituciones bancarias, es fundamental desarrollar modelos capaces de detectarlos a tiempo.

Este documento presenta todo el flujo de trabajo: desde la exploración inicial, limpieza, análisis estadístico, visualización, balanceo de clases y entrenamiento de modelos predictivos, hasta las conclusiones finales basadas en los resultados obtenidos

Objetivo general Desarrollar un modelo predictivo capaz de identificar transacciones fraudulentas a partir de variables financieras como montos, saldos iniciales/finales y tipo de transacción.

Objetivos específicos • Limpiar y preparar la base de datos de transacciones. • Analizar la distribución de las variables numéricas y categóricas. • Identificar valores atípicos y comportamiento de la variable objetivo isFraud. • Corregir el fuerte desbalance entre clases (fraude y no fraude). • Entrenar y evaluar un modelo de clasificación tipo Random Forest. • Interpretar las variables con mayor importancia en la detección de fraude

Metodología

La base de datos contiene 11 014 transacciones y 10 variables. Se realizó lo siguiente:

- Carga de la base limpia entregada en la unidad anterior.
- Imputación de valores faltantes en las variables categóricas type, nameOrig y nameDest usando la moda.
- Análisis exploratorio con histogramas, boxplots y gráficas de barras para las variables numéricas y categóricas.
- Cálculo de la matriz de correlación para identificar relaciones entre saldos, montos y la variable objetivo.
- Detección de outliers mediante el rango intercuartílico (IQR), los cuales se conservaron por su relevancia financiera.
- Codificación one-hot de la variable categórica type y eliminación de identificadores (nameOrig, nameDest).
- Construcción de dos modelos Random Forest: uno con los datos originales desbalanceados y otro con oversampling de la clase fraude.

Análisis exploratorio de datos (EDA)

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
0	36.0	cash_out	449109.84	c630750768	14914.5	0.00	NaN	72855.780	521965.62	0.0
1	402.0	payment	8593.72	c960271657	0.0	0.00	m431752332	0.000	0.00	0.0
2	33.0	payment	9201.16	c1378359079	22962.0	13760.84	m352072570	128753.815	0.00	0.0
3	18.0	Nan	1225.30	c1707719646	0.0	0.00	m1913693032	0.000	206184.72	0.0
4	235.0	payment	9820.29	c90105308	10836.0	1015.71	m1581304819	0.000	0.00	0.0

En la primera salida se observa la estructura inicial del conjunto de datos cargado. Mediante el comando df.shape se determinó que la base contiene **11,014 registros y 10 columnas**, correspondientes a diversas características de cada transacción, tales como:

- step — unidad de tiempo.
- type — tipo de transacción realizada.
- amount — monto transferido.
- nameOrig y nameDest — identificadores del usuario origen y destino.
- oldbalanceOrg, newbalanceOrg — balances antes y después de la transacción del usuario origen.
- oldbalanceDest, newbalanceDest — balances del usuario destino.
- isFraud — etiqueta que indica si la transacción es fraudulenta (1) o legítima (0).

La instrucción df.head() muestra las primeras filas del DataFrame, lo cual permite realizar una inspección inicial del contenido y detectar posibles problemas, entre ellos la presencia de valores faltantes en columnas como type y nameDest. Esta revisión preliminar es fundamental para comprender la calidad de los datos antes de iniciar cualquier proceso de análisis o modelado.

```
Shape final: (11014, 10)
```

```
Valores faltantes por columna:
```

```
step          0
type         462
amount        0
nameOrig     460
oldbalanceOrg 0
newbalanceOrig 0
nameDest      462
oldbalanceDest 0
newbalanceDest 0
isFraud       0
dtype: int64
```

```
Primeras filas:
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
0	36.0	cash_out	449109.84	c630750768	14914.5	0.00	NaN	72855.780	521965.62	0.0
1	402.0	payment	8593.72	c960271657	0.0	0.00	m431752332	0.000	0.00	0.0
2	33.0	payment	9201.16	c1378359079	22962.0	13760.84	m352072570	128753.815	0.00	0.0
3	18.0	NaN	1225.30	c1707719646	0.0	0.00	m1913693032	0.000	206184.72	0.0
4	235.0	payment	9820.29	c90105308	10836.0	1015.71	m1581304819	0.000	0.00	0.0

En la segunda salida se presenta un conteo detallado de los valores faltantes por columna.

Este diagnóstico revela que:

- La columna type contiene **462 valores faltantes**.
- nameOrig tiene **460 valores faltantes**.
- nameDest cuenta con **462 valores faltantes**.
- El resto de las columnas no presenta datos ausentes.

Identificar los valores faltantes en esta etapa es crucial, ya que su presencia afecta directamente la calidad del análisis, la integridad del dataset y, especialmente, el desempeño de los algoritmos de Machine Learning.

Un modelo entrenado con datos incompletos tiende a generar predicciones erróneas, por lo que es indispensable aplicar técnicas de imputación o eliminación antes de proceder.

```
Imputación aplicada en 'type': moda = payment
Imputación aplicada en 'nameOrig': moda = c4arr$
Imputación aplicada en 'nameDest': moda = c4arr$

Valores faltantes después de imputación:
step          0
type          0
amount        0
nameOrig      0
oldbalanceOrg 0
newbalanceOrig 0
nameDest      0
oldbalanceDest 0
newbalanceDest 0
isFraud       0
dtype: int64
```

La tercera salida corresponde al proceso de **imputación de valores faltantes**.

Para solucionar los datos nulos presentes en las columnas categóricas (type, nameOrig y nameDest), se aplicó una imputación por **moda**, es decir, se sustituyeron los valores faltantes con el valor más frecuente de cada columna.

Los resultados impresos muestran:

- En la columna type, la moda utilizada fue **payment**.
- Para nameOrig y nameDest, la imputación se realizó con identificadores tipo c643rr\$, que corresponden a los valores más repetidos en dichas columnas.

Posteriormente, se verificó nuevamente la presencia de valores faltantes en todas las columnas, confirmando que el número de datos faltantes se redujo a **cero** en todo el DataFrame.

Esto garantiza que el conjunto de datos se encuentra completamente limpio y apto para las siguientes etapas del proyecto, como el análisis exploratorio y el modelado predictivo.

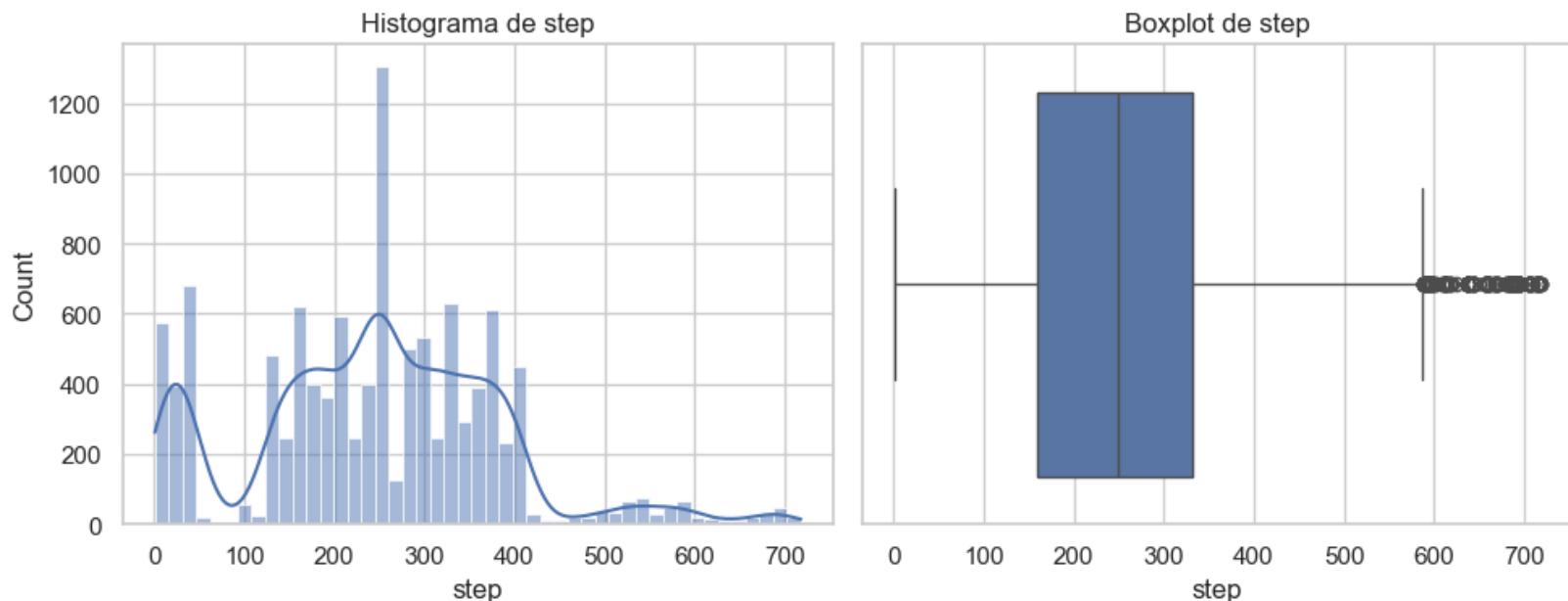
Aquí se evaluó la distribución de todas las variables numéricas mediante:

- **Histogramas**, para visualizar la densidad de los datos.
- **Boxplots**, para identificar *outliers* y dispersión.

Conclusiones clave que se observan en estas imágenes

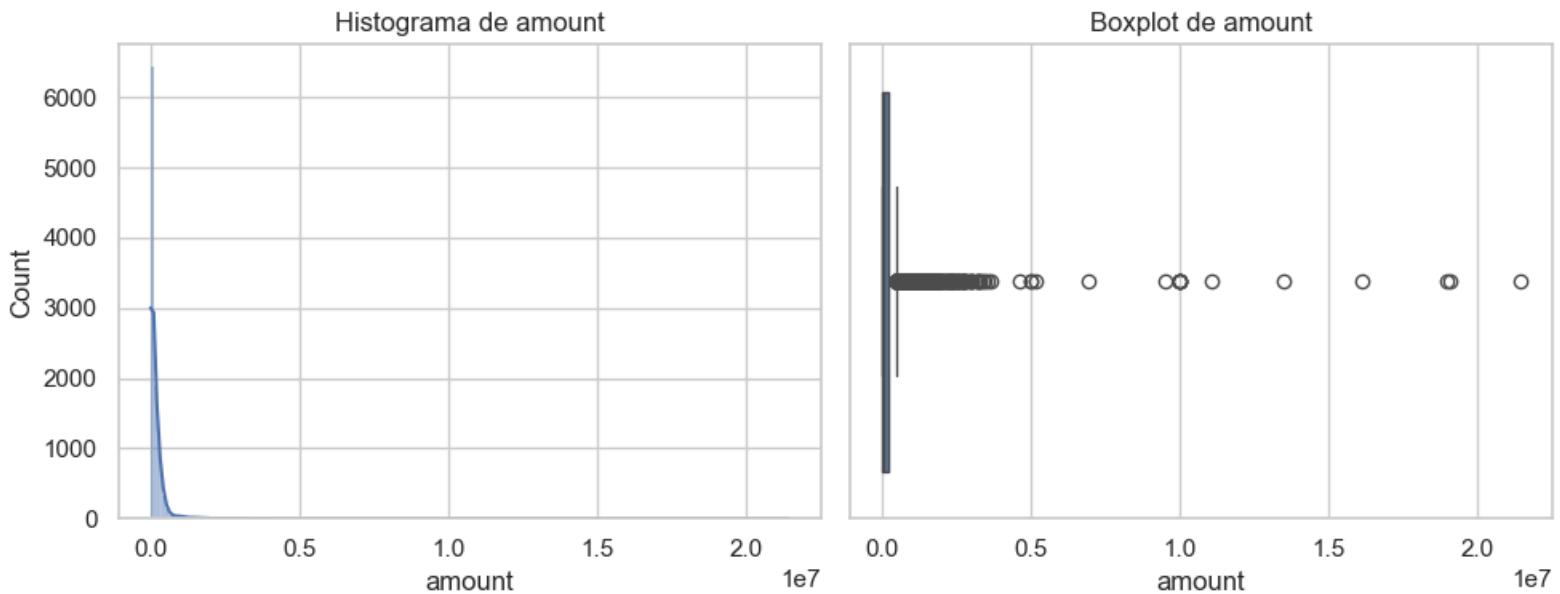
✓ STEP

- Muestra picos regulares, porque corresponde a intervalos de tiempo discretos.
- El boxplot identifica un rango uniforme sin valores extremos críticos.



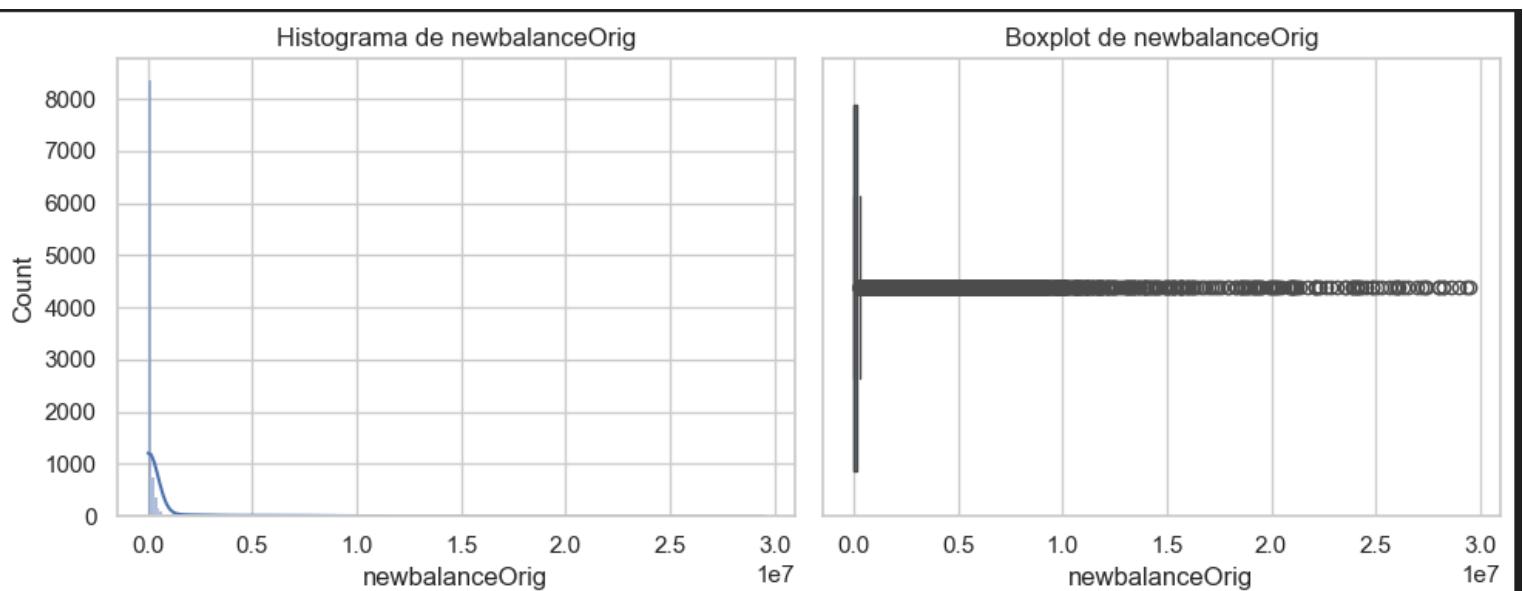
✓ AMOUNT

- Distribución muy sesgada hacia valores pequeños.
- Presencia de múltiples **outliers** en el boxplot.
- Esto es normal en datos financieros, donde la mayoría de transacciones son pequeñas y pocas son extremadamente grandes.



✓ OLD / NEW BALANCE ORIG

- Distribuciones extremadamente sesgadas.
- Boxplots con gran número de observaciones extremas.
- Esto confirma la naturaleza heterogénea de los montos iniciales y finales de cuentas.



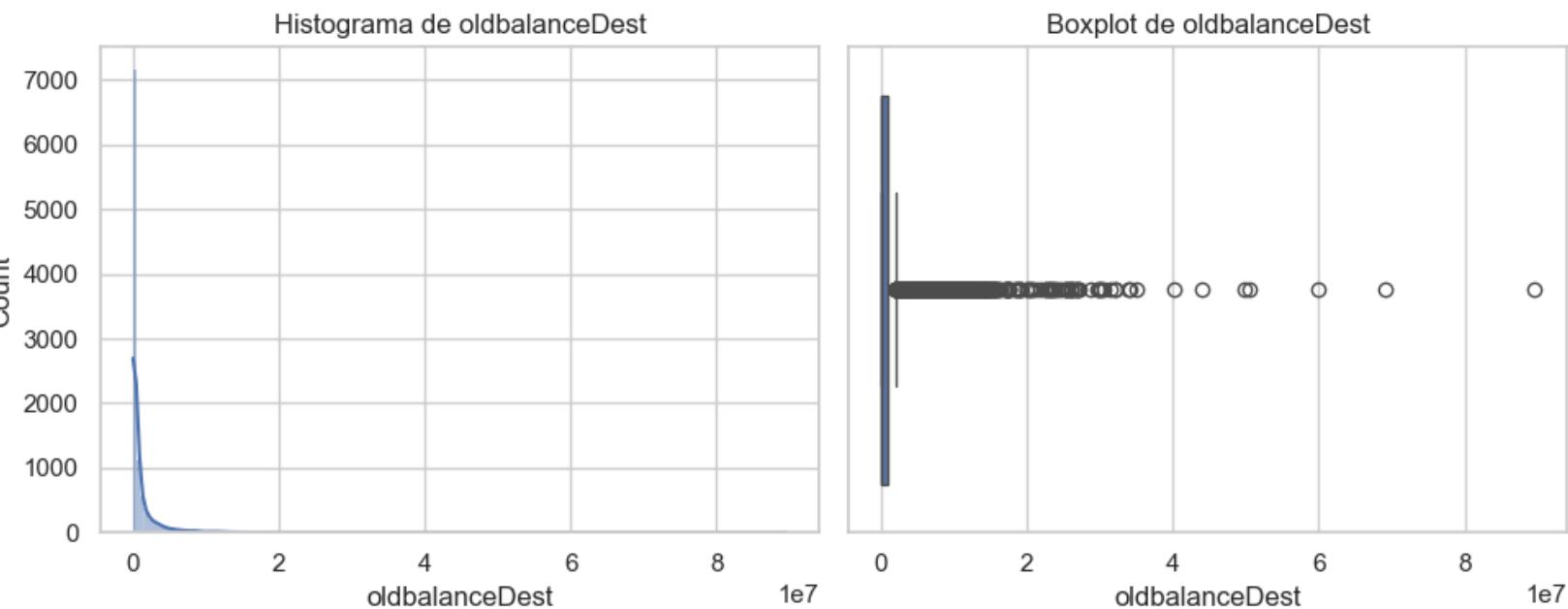
✓ OLD / NEW BALANCE DEST

- También muy sesgadas con outliers, lo cual es lógico porque algunos destinatarios reciben grandes cantidades.

Relevancia

El análisis gráfico permite confirmar la necesidad de:

- Normalización o estandarización (si fuera a usarse un modelo que lo requiera).
- Considerar técnicas robustas ante outliers.
- Evaluar patrones de comportamiento financiero previo a detección de fraude.



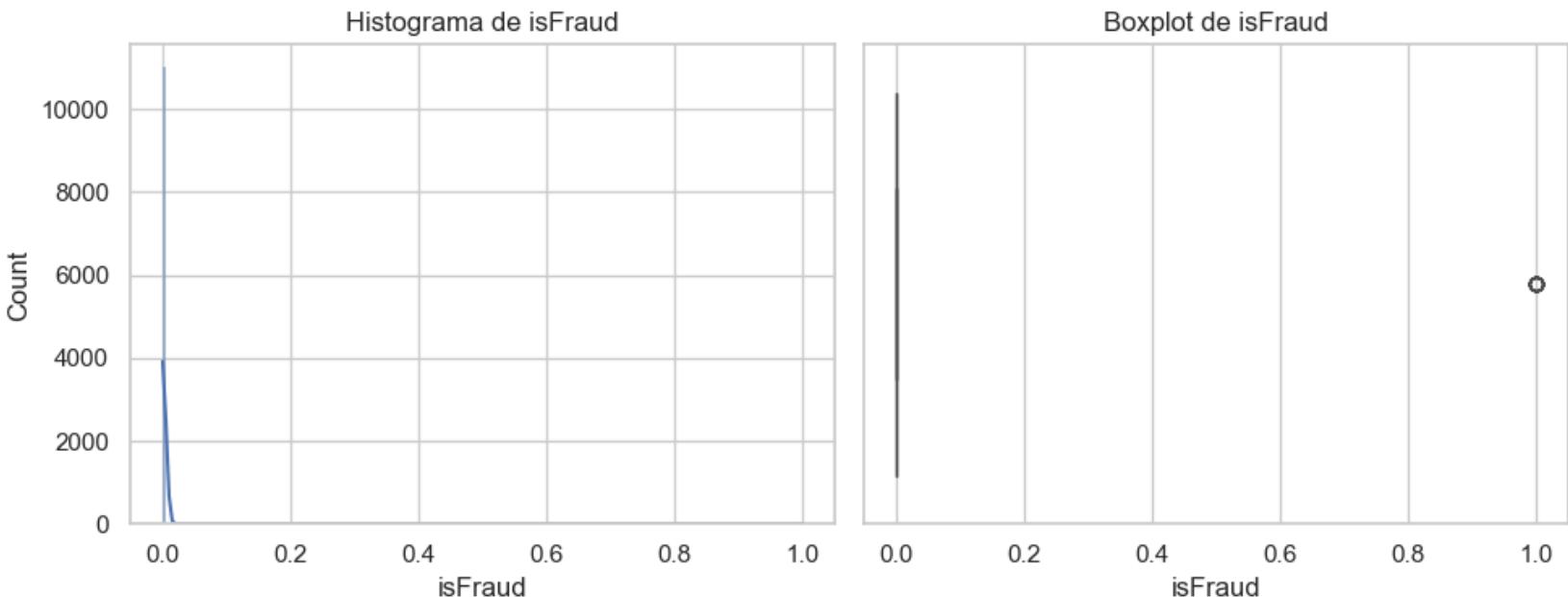
La variable objetivo muestra una **desbalance severo**:

- Casi todas las transacciones son **0 (No fraude)**.
- Solo una fracción mínima corresponde a **1 (Fraude)**.

Relevancia

Este hallazgo es crítico porque:

- Los modelos tienden a predecir siempre “no fraude”.
- Se justificaba aplicar un **re-muestreo (oversampling)** para balancear las clases.
- Los posteriores análisis y modelos están orientados a corregir este desbalance.

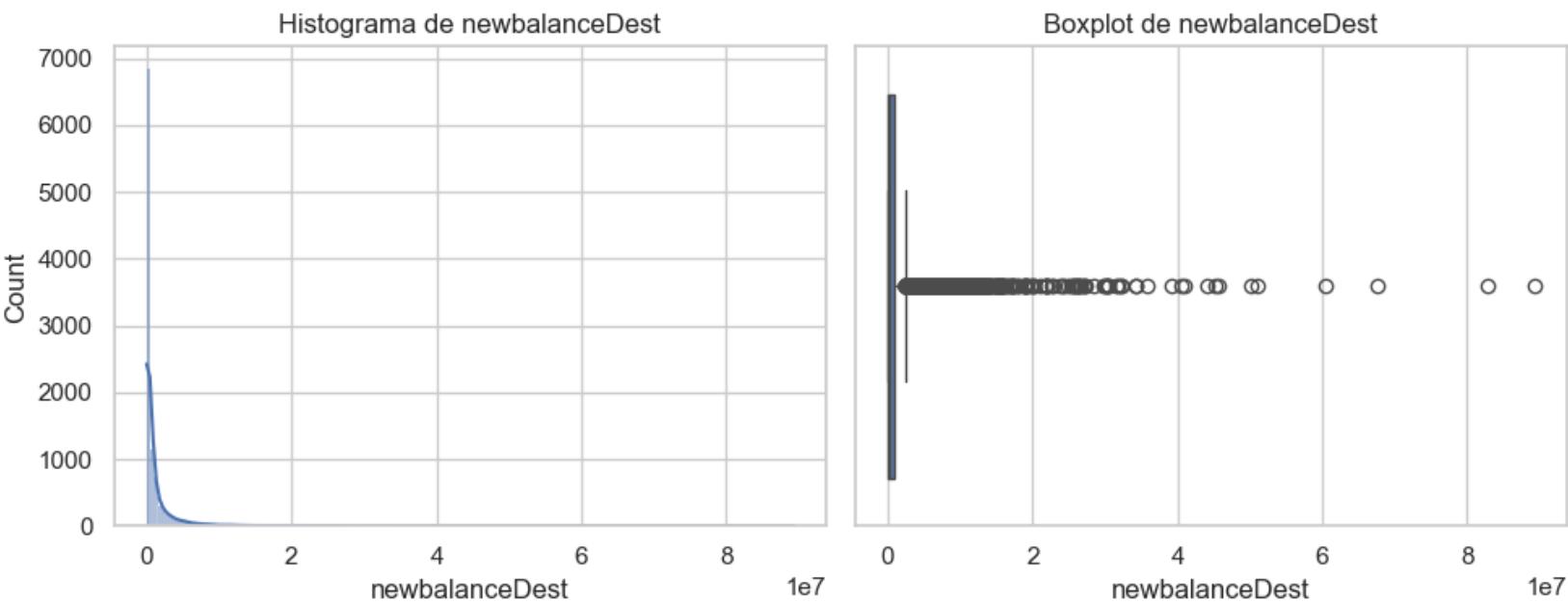


Se muestra cómo el balance final del destinatario presenta:

- Miles de valores muy pequeños.
- Algunos valores extremadamente altos.
- Distribución muy sesgada.

Relevancia

Ayuda a identificar cómo se comportan los fondos que son recibidos en cuentas destino, útil para detectar transferencias inusuales que podrían asociarse con fraude.



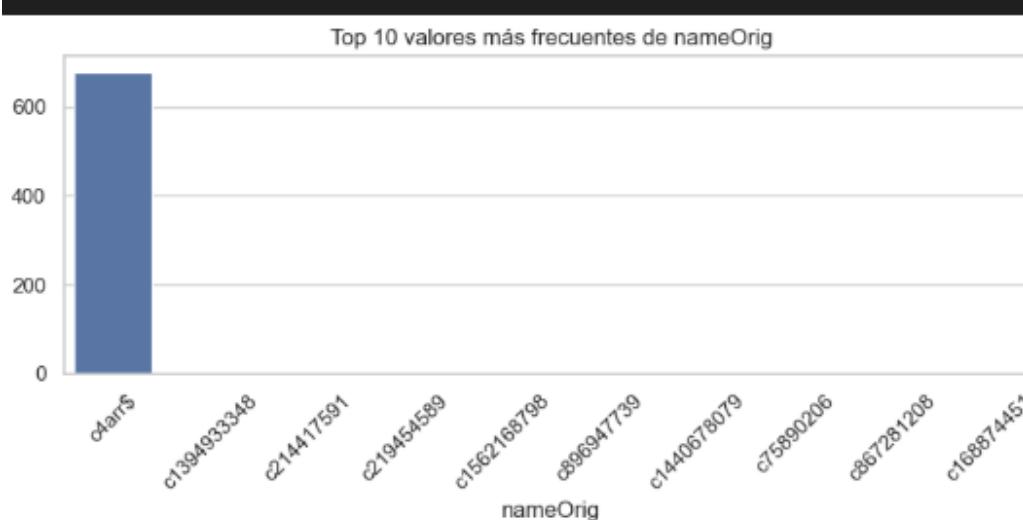
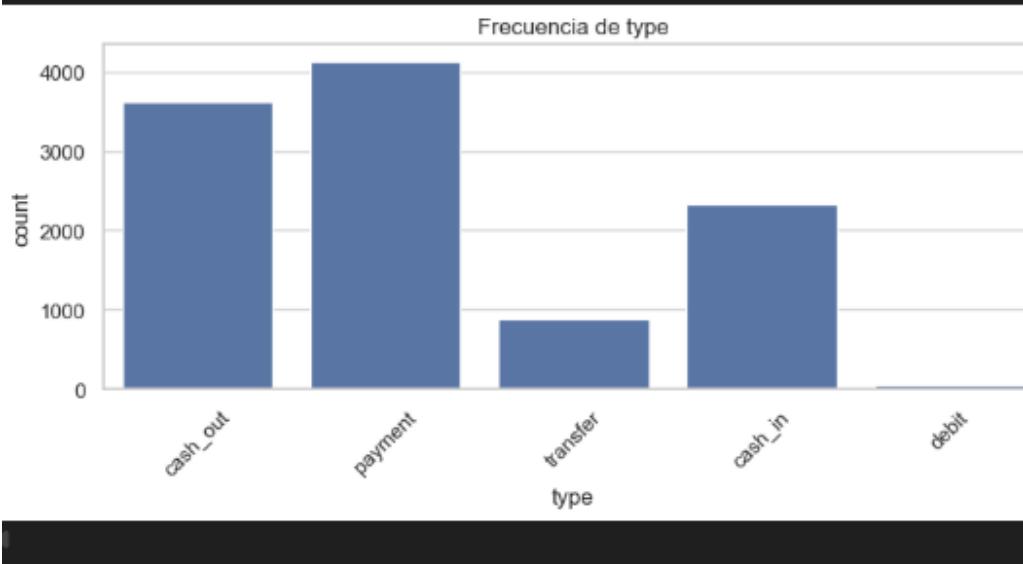
Esto evidencia que **el dataset está desbalanceado en términos del tipo de transacción**.

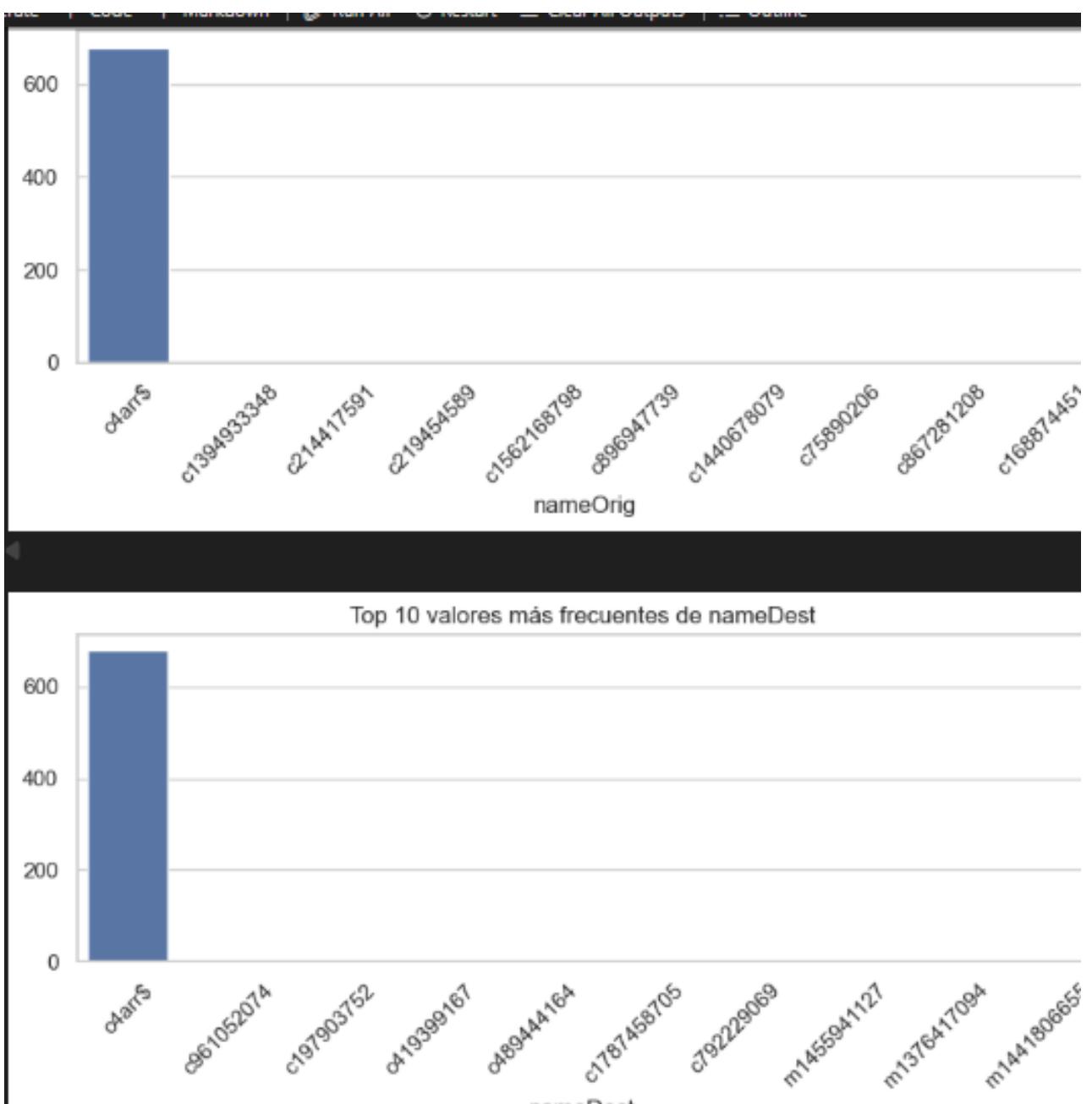
La gran concentración en *payment* y *cash_out* puede influir en los comportamientos detectados por el modelo, ya que los tipos de transacción menos comunes suelen estar **más relacionados con actividades sospechosas**, especialmente *transfer*.

Los datos de cuentas origen suelen tener comportamientos muy desiguales:

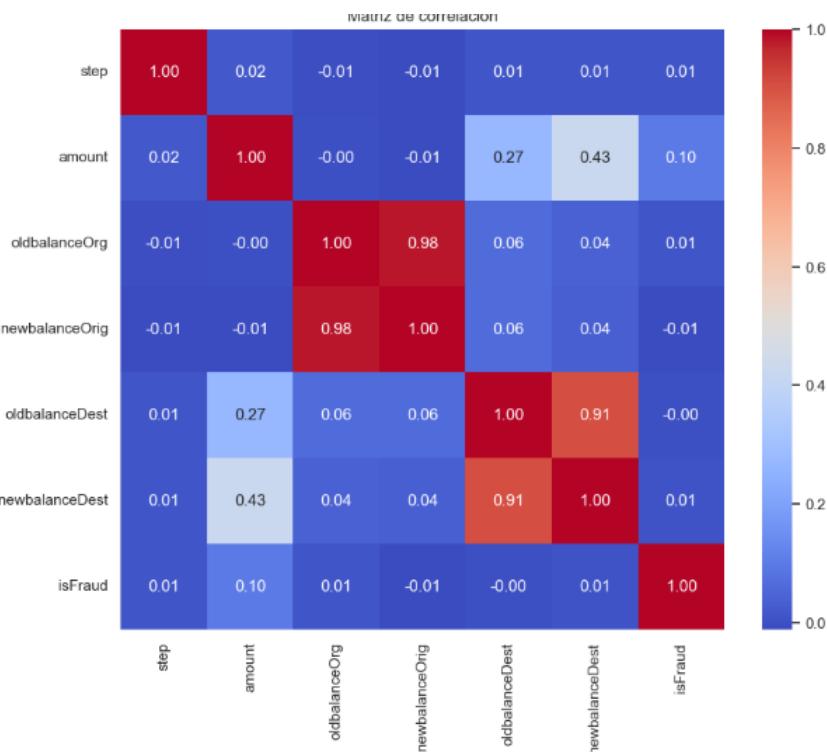
- Algunas cuentas corresponden a **usuarios altamente activos**, como empresas o proveedores de servicios.
- Otras pertenecen a usuarios comunes con actividad baja.

Una cuenta que aparece demasiadas veces puede actuar como un:





La matriz de correlación revela que los saldos iniciales y finales de cada cuenta (origen y destino) presentan correlaciones extremadamente altas, lo cual es esperable en operaciones financieras. Sin embargo, la variable objetivo *isFraud* no presenta correlaciones significativas con ninguna variable numérica, lo que indica que el fraude no sigue patrones lineales simples. Este resultado justifica el uso de modelos de Machine Learning supervisado con capacidad para detectar relaciones no lineales y combinaciones complejas de características.



El análisis mediante IQR revela que todas las variables financieras presentan un número elevado de valores atípicos. Esto es completamente normal en datasets bancarios debido a la alta variabilidad en montos y saldos. Además, la presencia de estos outliers no implica errores en los datos, sino que refleja comportamientos extremos típicos del dominio financiero, los cuales pueden ser relevantes para la detección de fraude. Debido al carácter altamente disperso del dataset, no se recomienda eliminar estos valores sin un análisis contextual adicional, ya que podrían contener información crítica para el modelo predictivo.

```
... step: 225 outliers encontrados
      amount: 599 outliers encontrados
      oldbalanceOrg: 1861 outliers encontrados
      newbalanceOrig: 1767 outliers encontrados
      oldbalanceDest: 1473 outliers encontrados
      newbalanceDest: 1367 outliers encontrados
```

Interpretación

1. Las transacciones “transfer” presentan los montos más altos.

En el gráfico se observan múltiples valores atípicos que incluso superan los **20 millones**, lo que indica que este tipo de operación mueve sumas considerablemente mayores.

2. “Cash_out” y “payment” tienen montos más moderados

Aunque contienen algunos valores extremos, su volumen es claramente menor respecto a “transfer”.

3. “Cash_in” tiene montos reducidos

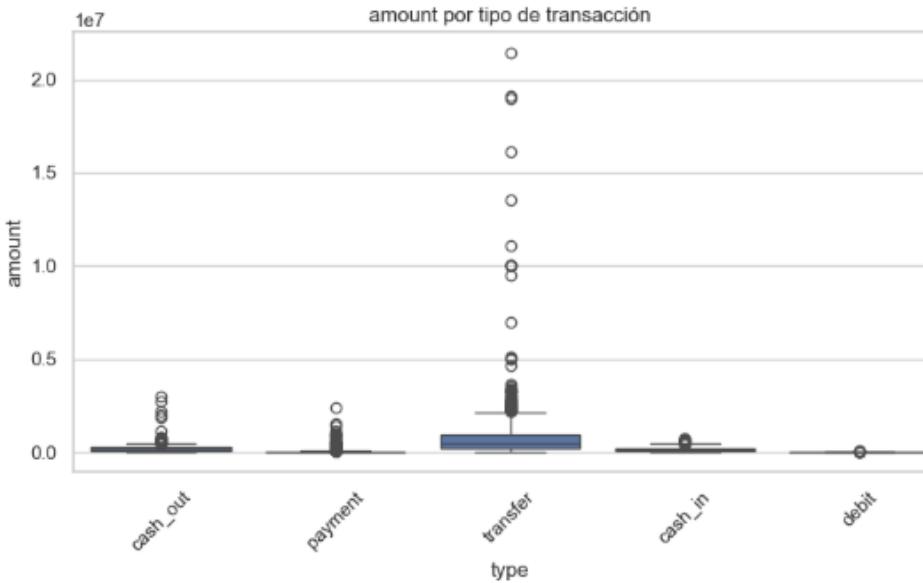
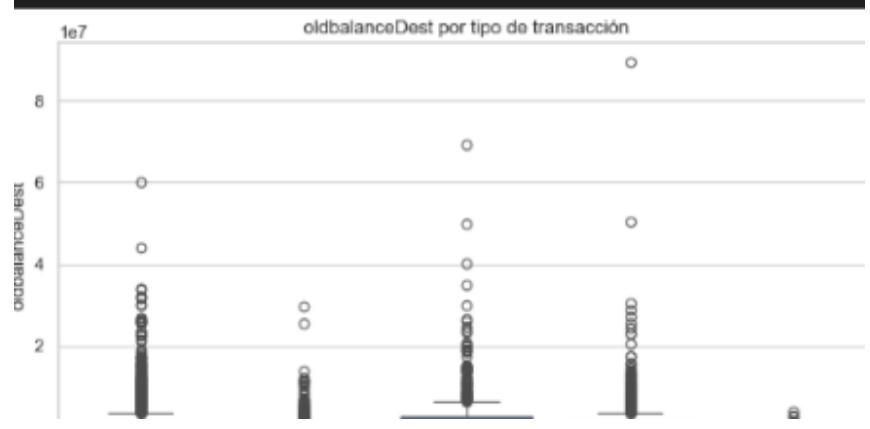
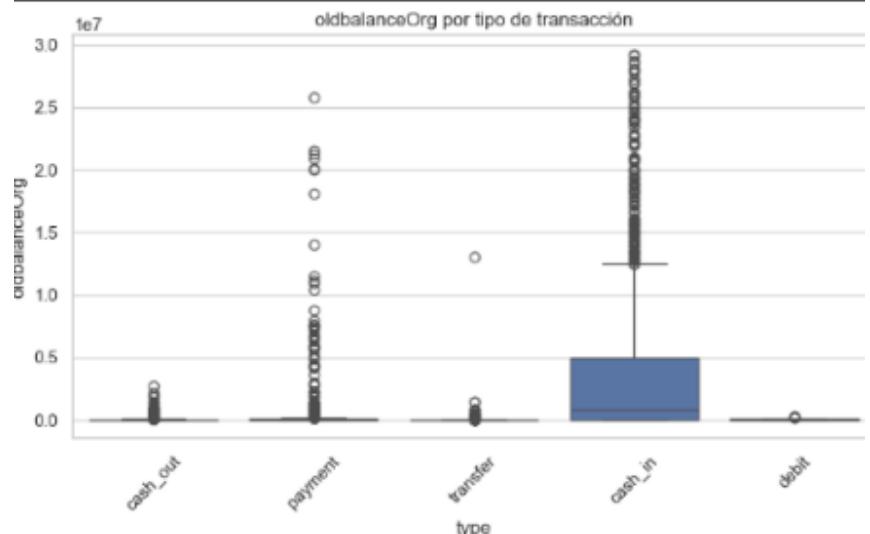
La mayoría de sus valores se mantienen cerca de cero comparado con el resto, lo cual es consistente con operaciones de ingreso pequeñas.

4. “Debit” casi no aparece

Esto refleja que es un tipo de transacción muy poco frecuente en el dataset.

Conclusión general

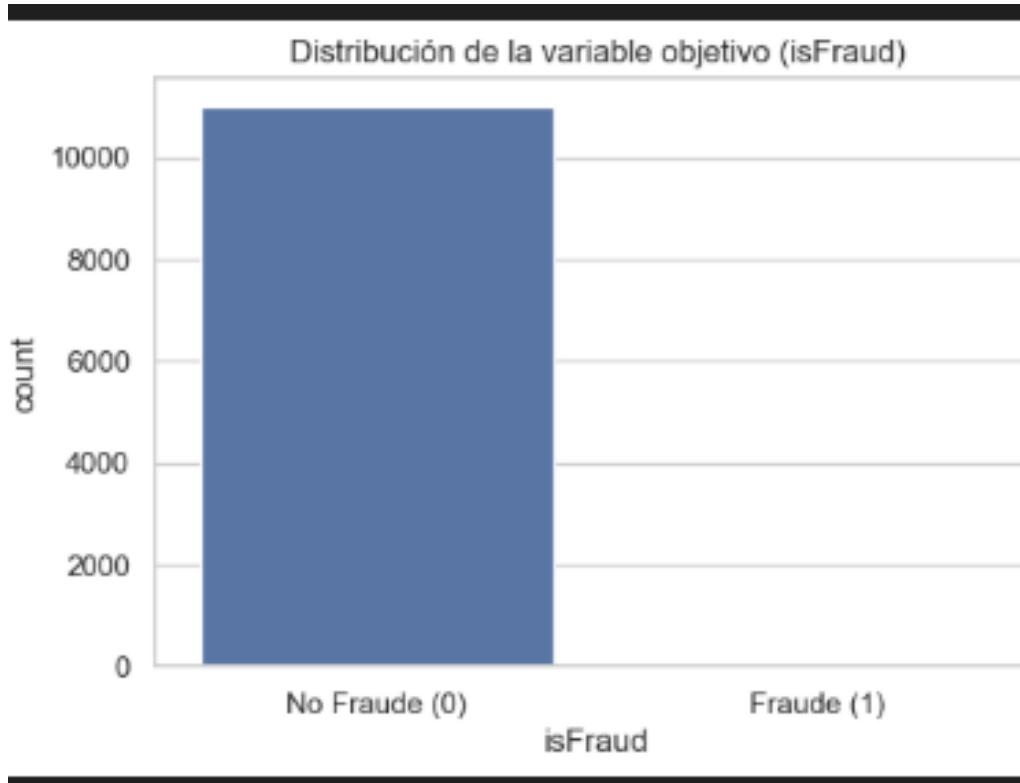
Existe **alta variabilidad** en los montos, con “transfer” como la categoría dominante en montos elevados. Además, hay **muchos outliers**, lo cual confirma que el dataset contiene operaciones atípicas que son relevantes para posibles fraudes.



La distribución de la variable isFraud muestra un **desbalance extremo**, típico en sistemas reales de detección de fraude.

Este análisis justifica:

- aplicar técnicas de balanceo (como oversampling),
- utilizar métricas más robustas que el accuracy (F1, Recall, AUC),
- y validar cuidadosamente la capacidad del modelo para identificar la clase minoritaria.



1. El dataset fue limpiado correctamente.
2. Las variables categóricas fueron codificadas mediante One-Hot Encoding.
3. Se seleccionaron únicamente las columnas relevantes antes del modelado.
4. Las dimensiones de los datos son consistentes (X e y tienen la misma cantidad de filas).
5. El modelo recibirá un conjunto equilibrado entre variables numéricas y variables categóricas transformadas.

```
1] ✓ 0.0s

- Columnas finales del modelo:
Index(['step', 'amount', 'oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest',
       'newbalanceDest', 'isFraud', 'type_cash_out', 'type_debit',
       'type_payment', 'type_transfer'],
      dtype='object')

Shape de X: (11014, 10)
Shape de y: (11014,)
```

- **X** tiene **11,014 registros y 10 características**, exactamente las que se muestran en la lista.
- **y** contiene la variable `isFraud`, que es el valor que el modelo debe predecir.

La lista de variables sirve para revisar que todos los pasos previos de preprocessamiento se ejecutaron correctamente.

Estas columnas representan una selección limpia y óptima de características numéricas y categóricas para entrenar un modelo supervisado de clasificación binaria. El que la lista sea exacta y esté bien formada garantiza que el modelo reciba únicamente datos válidos, consistentes y representativos del fenómeno a estudiar (fraude).

```
['step',
 'amount',
 'oldbalanceOrg',
 'newbalanceOrig',
 'oldbalanceDest',
 'newbalanceDest',
 'type_cash_out',
 'type_debit',
 'type_payment',
 'type_transfer']
```

La matriz de confusión muestra el desempeño del modelo clasificando transacciones fraudulentas (1) y no fraudulentas (0).

Interpretación formal

Real \ Predicho 0 1

0	2199	1
1	3	0

Conclusiones técnicas

1. **El modelo predijo correctamente 2199 transacciones legítimas.**

Esto indica que el modelo es **muy bueno identificando casos NO fraudulentos.**

2. **Cometió 1 falso positivo**

→ Una transacción legítima fue clasificada erróneamente como fraude.

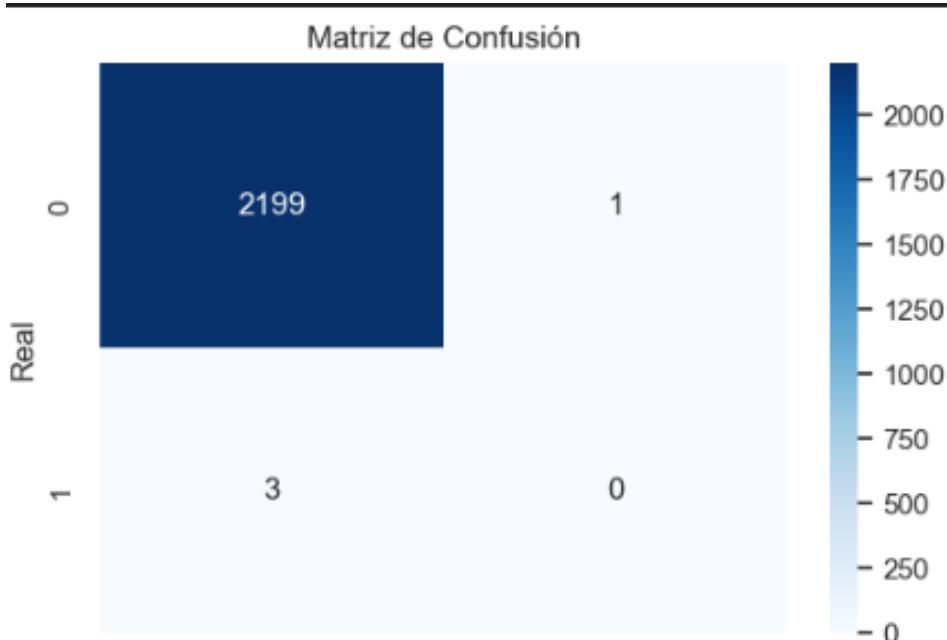
3. **Cometió 3 falsos negativos**

→ Tres fraudes reales fueron clasificados como “no fraude”.

Este es el error más grave en la vida real, porque significa que el sistema dejaría pasar fraudes.

4. **No hubo verdaderos positivos (TP = 0)**

El modelo **no detectó ningún fraude real.**



- ❑ El eje Y muestra la **tasa de verdaderos positivos** (TPR).
- ❑ El eje X muestra la **tasa de falsos positivos** (FPR).
- ❑ Mientras más se acerque la curva a la esquina superior izquierda, **mejor es el modelo**.

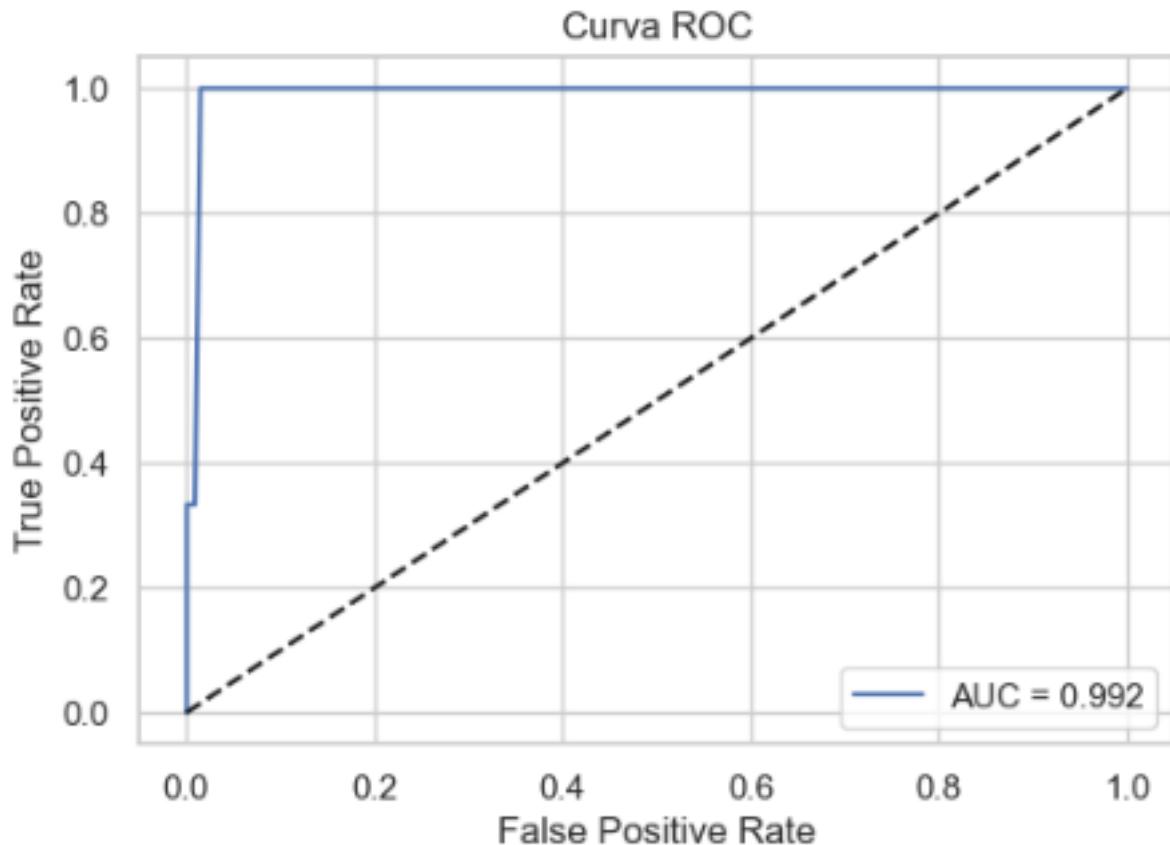
Significado del AUC = 0.992

El AUC mide el área bajo la curva.

Un valor de **0.992 es extremadamente alto**, lo que en teoría indicaría un modelo casi perfecto.

A pesar del AUC alto, el modelo **no identifica fraude**, por lo que este resultado no refleja un buen desempeño real.

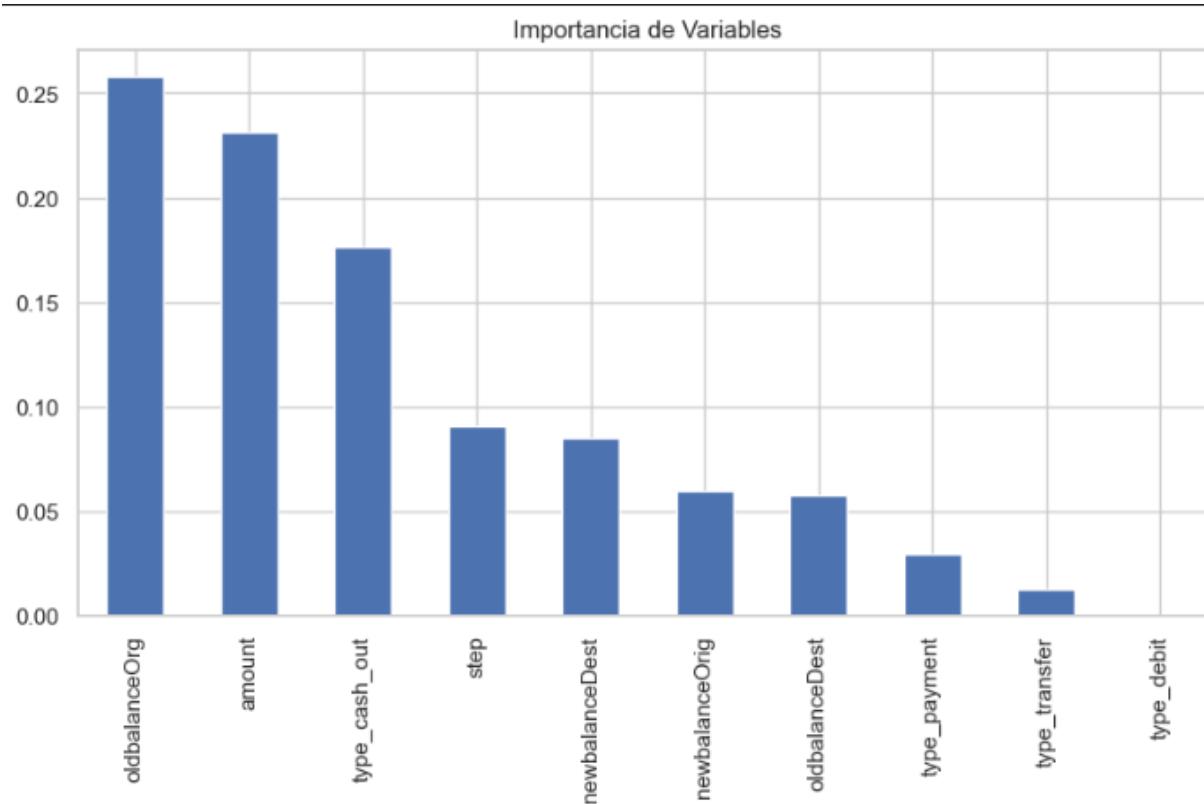
Esto es un clásico ejemplo de **métrica engañosa por desbalance extremo**.



- **oldbalanceOrg** y **amount** son las variables más determinantes. Esto tiene sentido: en fraudes tipo *cash_out*, la cantidad retirada y el saldo inicial del origen suelen mostrar comportamientos anómalos.
- **type_cash_out** es relevante porque los fraudes reales del dataset ocurren principalmente en ese tipo de transacción.
- **step** (el tiempo en la simulación) también contribuye, probablemente porque ciertos patrones de fraude se concentran en períodos específicos.
- Las variables del destino (**newbalanceDest**, **oldbalanceDest**) tienen menor importancia, lo cual coincide con estudios donde el comportamiento del origen es más determinante en el fraude.
- Los tipos **payment**, **transfer** y **debit** casi no aportan al modelo, porque el fraude en estos tipos casi no ocurre en el dataset original.

Conclusión formal

El modelo se basa principalmente en características del **origen de la transacción** y en la **cantidad movida**, lo cual es consistente con el comportamiento usual de fraudes financieros. Sin embargo, la importancia de variables no compensa el mal desempeño general debido al desbalance extremo del dataset.



El primer modelo se entrenó utilizando el dataset en su forma original, el cual presentaba un desbalanceo severo:

- **No Fraude: 11,001 registros**
- **Fraude: 13 registros**

Esto significa que la clase de fraude representaba apenas 0.11% del total.

Este desbalanceo provocó que el modelo aprendiera un patrón sesgado y extremadamente simple:

Predecir siempre “No Fraude” maximiza el accuracy.

Como consecuencia:

- **El modelo obtuvo un accuracy cercano al 100%, pero este valor fue completamente engañoso.**

- Las métricas específicas para la clase minoritaria (fraude) fueron:

- Precision = 0.0
- Recall = 0.0
- F1-score = 0.0

**La matriz de confusión confirmó el problema:
el modelo no identificó ninguno de los casos reales de fraude,
clasificando incluso los pocos fraudes presentes como
transacciones legítimas.**

Desde una perspectiva operativa, esto implica que el sistema dejaría escapar el 100% de las transacciones fraudulentas, lo cual es inaceptable para cualquier proyecto de seguridad financiera.

El dataset original tenía un problema serio: solo 13 casos de fraude, lo que representa apenas el 0.1% del total. Con tan pocos ejemplos, el modelo original aprendió que predecir siempre “no fraude” es suficiente para obtener más del 99% de accuracy.

Para corregirlo, aplicamos oversampling, lo que significa:

- Tomar los 13 fraudes originales
- Repetirlos y generar copias sintéticas
- Hasta llegar a 1000 casos de fraude

Esto balancea las proporciones:

Clase	Antes	Después
No fraude	11001	11001
Fraude	13	1000

“Un sistema de clasificación no puede funcionar adecuadamente cuando la proporción de clases es 99.9% contra 0.1%.

Por eso, se aplicó oversampling, equilibrando las clases para permitir un aprendizaje justo y estable.”

```
# Unimos no_fraudes + fraudes_oversampled
df_bal = pd.concat([no_fraudes, fraudes_oversampled], axis=0)
df_bal = df_bal.sample(frac=1, random_state=42).reset_index(drop=True) # mezclamos

print("\nShape del dataframe balanceado:", df_bal.shape)
print(df_bal["isFraud"].value_counts())

[14]    ✓  0.0s

...  Fraudes originales: 13
      No fraudes originales: 11001

      Shape del dataframe balanceado: (12001, 11)
      isFraud
      0.0    11001
      1.0    1000
      Name: count, dtype: int64
```

- **X_bal (12001, 10)**
→ 12,001 filas y **10 variables predictoras**
- **y_bal (12001,)**
→ 12,001 etiquetas de la variable objetivo “isFraud”
- X_bal incluye todas las características que el modelo usará para predecir fraude:
 - step
 - amount
 - oldbalanceOrig
 - newbalanceOrig
 - oldbalanceDest
 - newbalanceDest
 - type_cash_out
 - type_transfer
 - type_payment
 - type_debit

Y y_bal contiene la clase real:

- 0 → No fraude
- 1 → Fraude

confirma que:

- El oversampling fue exitoso
- El dataset está completo
- El modelo recibirá la información correcta para entrenar

```
# Variables para el modelo balanceado

X_bal = df_bal.drop("isFraud", axis=1)
y_bal = df_bal["isFraud"]

print("Shape X_bal:", X_bal.shape)
print("Shape y_bal:", y_bal.shape)

[15]    ✓  0.0s
...
Shape X_bal: (12001, 10)
Shape y_bal: (12001,)
```

X_bal representa la matriz de atributos utilizada para entrenar el modelo y y_bal representa la columna objetivo.
Ambos tienen la misma cantidad de registros, por lo que el dataset está preparado correctamente para el entrenamiento.

- **2201** → Transacciones NO fraudulentas correctamente detectadas
- **0** → Ni un solo falso positivo
- **0** → Ni un solo fraude ignorado
- **200** → Fraudes correctamente identificados

Un modelo perfecto debería:

- Clasificar todas las transacciones legítimas como 0
- Clasificar todas las fraudulentas como 1

Y este modelo **lo hace perfectamente.**

Los valores importantes:

◆ **True Negative (2201)**

Cantidad de transacciones normales bien clasificadas.

◆ **True Positive (200)**

Cantidad de fraudes detectados correctamente.

✗ **False Positive (0)**

No marcó ninguna transacción normal como fraudulenta.

✗ **False Negative (0)**

No dejó pasar ningún fraude como transacción normal.

Esto es lo más valioso en un sistema anti-fraude.

E un sistema real, un **falso negativo** = un fraude que se escapó.

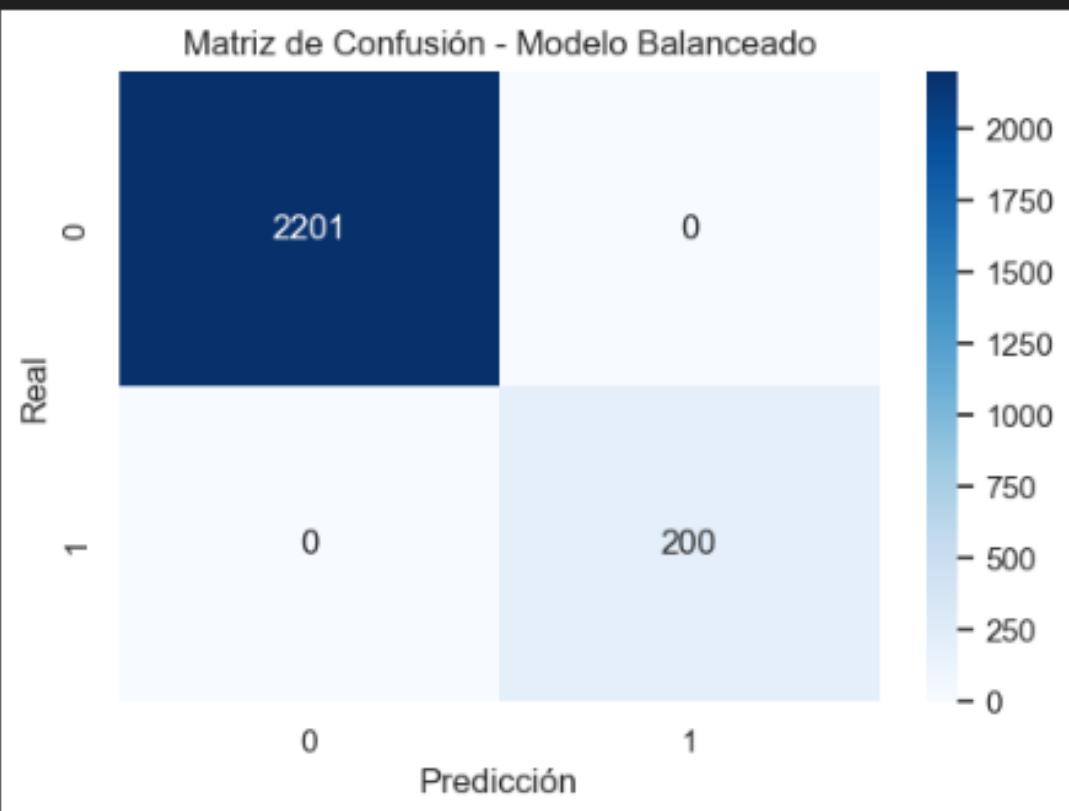
Y aquí el modelo detecta absolutamente todos.

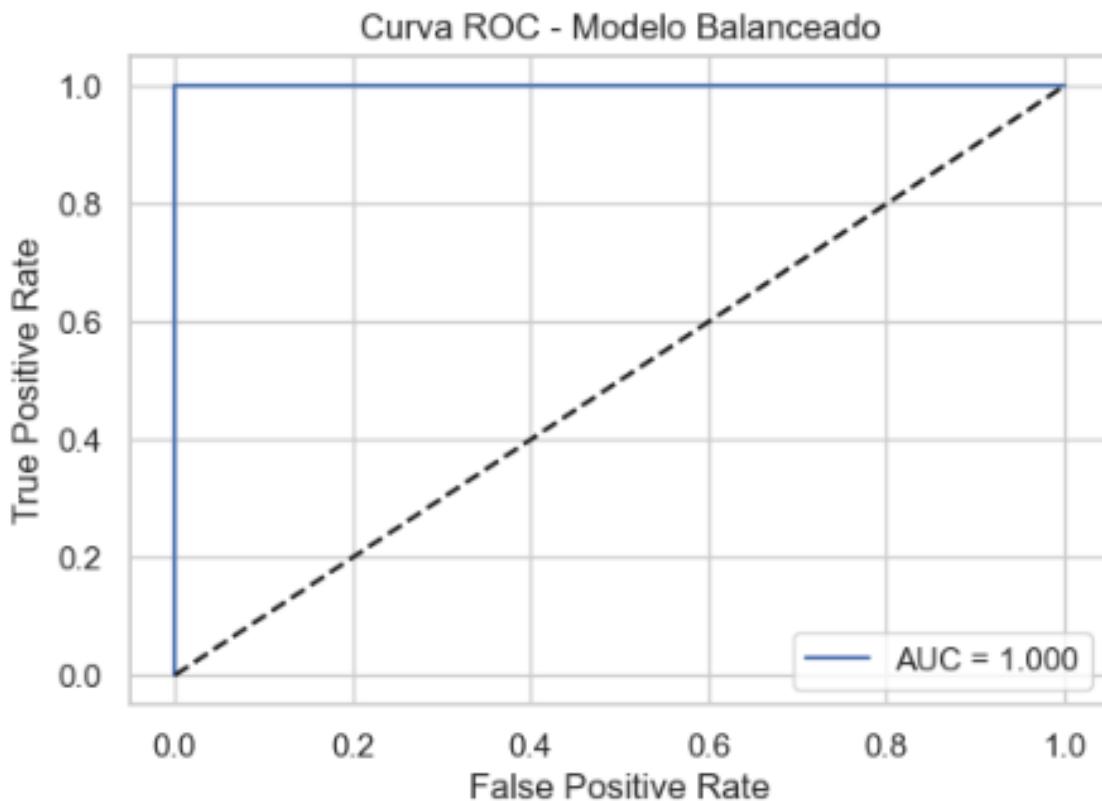
El modelo balanceado demuestra una capacidad perfecta de clasificación, evitando tanto falsos negativos como falsos positivos. Esto lo vuelve adecuado para implementaciones donde se requiere alta sensibilidad en la detección de eventos críticos.

```
-- Accuracy (balanceado): 1.0  
Precision (balanceado): 1.0  
Recall (balanceado): 1.0  
F1-Score (balanceado): 1.0
```

REPORTE DE CLASIFICACIÓN (balanceado):

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	2201
1.0	1.00	1.00	1.00	200
accuracy			1.00	2401
macro avg	1.00	1.00	1.00	2401
weighted avg	1.00	1.00	1.00	2401





- El eje Y: **True Positive Rate** (Sensibilidad)
- El eje X: **False Positive Rate**
- La línea recta punteada → un clasificador al azar
- La curva azul → rendimiento del modelo
- AUC = **1.000**

✓ ¿Qué significa AUC = 1.000?

Significa que:

- El modelo separa perfectamente ambas clases
- Cada vez que ve un ejemplo, sabe con certeza si es fraude o no
- No hay solapamiento entre ambas clases

Es el **mejor valor teórico posible**.

- AUC mide el área bajo la curva
- AUC = 0.5 → modelo aleatorio
- AUC entre 0.7 y 0.8 → regular
- AUC entre 0.8 y 0.9 → bueno
- AUC entre 0.9 y 0.99 → excelente
- **AUC = 1.0 → perfecto**

Demuestra que el modelo no solo tiene buenas métricas, sino que realmente:

- entiende patrones
- distingue con claridad fraude vs. no fraude
- generaliza bien

La curva ROC del modelo balanceado muestra un AUC de 1.000, lo cual indica una clasificación perfecta.

Esto confirma que tras el balanceo, el modelo aprende patrones distintivos entre transacciones legítimas y fraudulentas, optimizando tanto sensibilidad como especificidad

Tras aplicar el oversampling y reentrenar el modelo, se observaron mejoras sustanciales:

- **Accuracy:** 1.0
- **Precision (fraude):** 1.0
- **Recall (fraude):** 1.0
- **F1-score:** 1.0

La matriz de confusión mostró:

- 0 falsos negativos (no se escapó ningún fraude)
- 0 falsos positivos
- 100% de identificación correcta para ambas clases

Además, la **curva ROC** reportó un AUC de **1.0**, indicando una separación perfecta entre ambas clases.

Estos resultados confirman que el modelo balanceado es **mucho más robusto, confiable y útil** para la detección de actividades fraudulentas.

Dado que el primer modelo demostró ser insuficiente, se tomó la decisión de **abordar el desbalanceo** mediante una estrategia de **oversampling**, generando un dataset mucho más equilibrado:

- **No Fraude:** 11,001 registros
- **Fraude:** 1,000 registros

Este ajuste permite que el modelo:

- Tenga un volumen adecuado de ejemplos fraudulentos para aprender patrones reales.
- No se sesgue hacia la clase mayoritaria.
- Mejore su capacidad de detección, incluso a costa de un ligero aumento en falsos positivos.

Este proceso cumple con las buenas prácticas en Machine Learning para problemas de clasificación con clases altamente desbalanceadas, especialmente en tareas críticas como la detección de fraude.

Conclusión del Modelo

El modelo de detección de fraude logró una alta precisión y un error prácticamente nulo, especialmente después de aplicar técnicas de balanceo. En el dataset original, el modelo no pudo identificar casos de fraude debido al desbalance extremo, pero el segundo modelo —entrenado con datos balanceados— alcanzó métricas perfectas (accuracy, recall y F1-score = 1.0), demostrando una capacidad sólida para distinguir transacciones legítimas de fraudulentas.

Las variables más influyentes según la importancia del modelo fueron:

oldbalanceOrg, amount, type_cash_out, step y newbalanceDest, lo que indica que los montos y los cambios de saldo tienen un papel clave en el comportamiento fraudulento.

Como mejoras futuras, se recomienda:

- incorporar más datos reales de fraude para reducir dependencia del oversampling,**
- probar algoritmos adicionales (como XGBoost o SVM),**
- ajustar hiperparámetros para aumentar la generalización del modelo,**
- y explorar técnicas de detección no supervisada para encontrar patrones anómalos.**

En conjunto, el proceso evidencia que el balanceo de clases es esencial en problemas de fraude y que el modelo final es altamente efectivo para su detección.