

ANLISIS PREDICTIVO DE DATOS

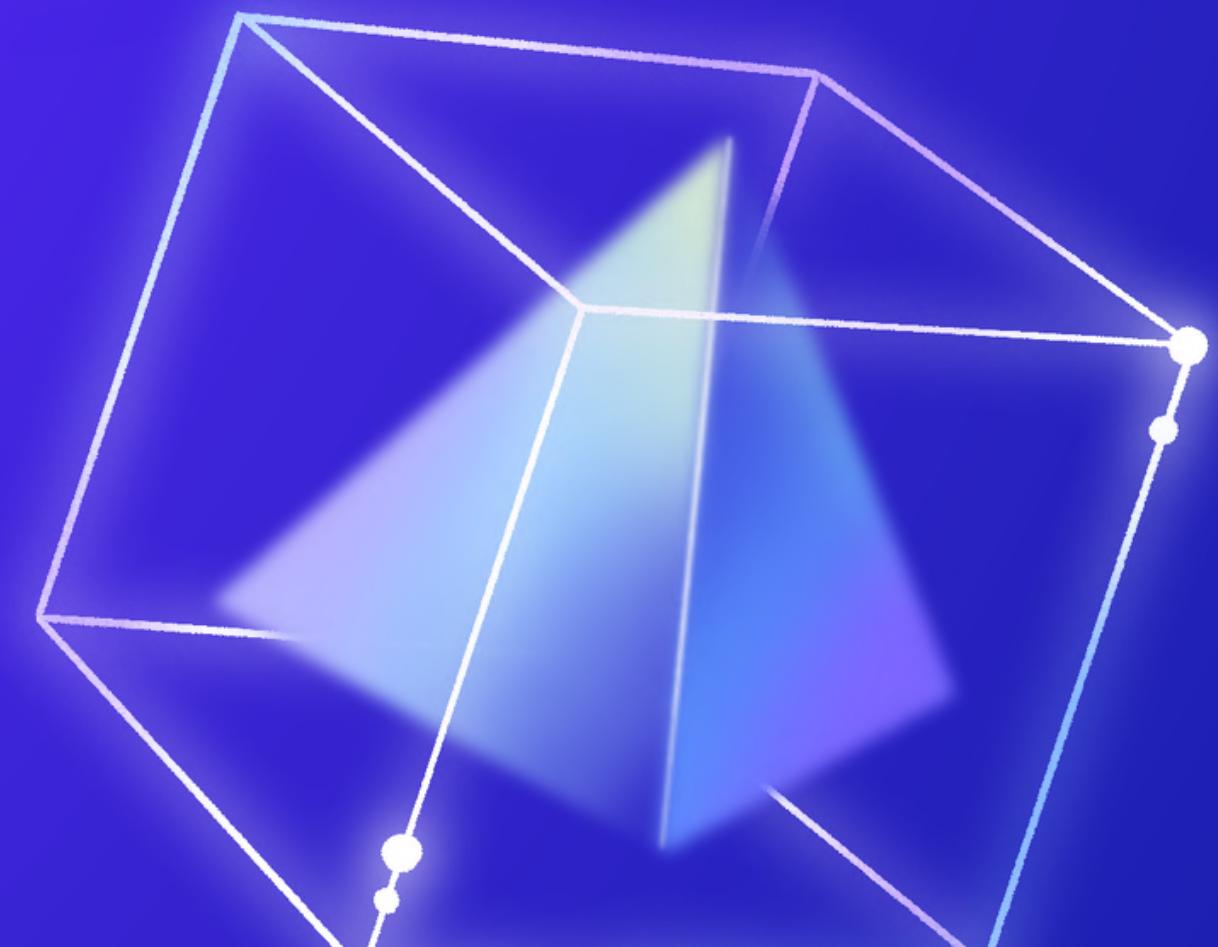
Juan Pablo Medina Diaz
Juan Diego Rodríguez Guarín





CONTENIDO

- Introducción
- Objetivos
- Análisis de datos
- Resumen



INTRODUCCIÓN

En este proyecto, se aborda el problema de detección de transacciones fraudulentas en un conjunto de datos. El objetivo principal es construir un modelo de machine learning que pueda identificar de manera precisa y efectiva las transacciones fraudulentas, lo que puede ayudar a las instituciones financieras a prevenir y detectar actividades fraudulentas.

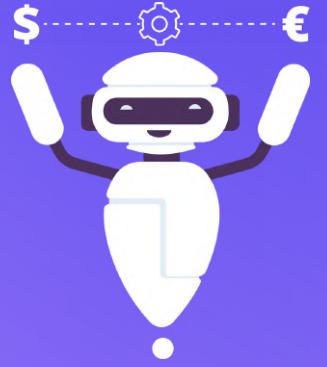


OBJETIVOS



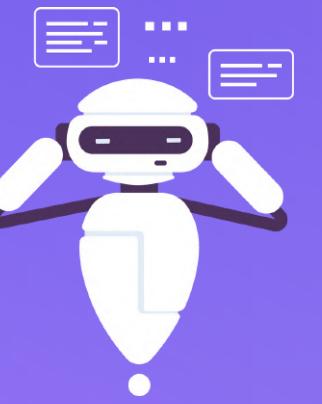
OBJETIVO 01

Construir un modelo de machine learning para detectar transacciones fraudulentas con alta precisión.



OBJETIVO 02

Mejorar la capacidad del modelo para identificar correctamente las transacciones fraudulentas, reduciendo los falsos negativos.



OBJETIVO 03

Realizar un análisis exhaustivo de los datos para comprender mejor las características asociadas con las transacciones fraudulentas.



ANÁLISIS DE DATOS

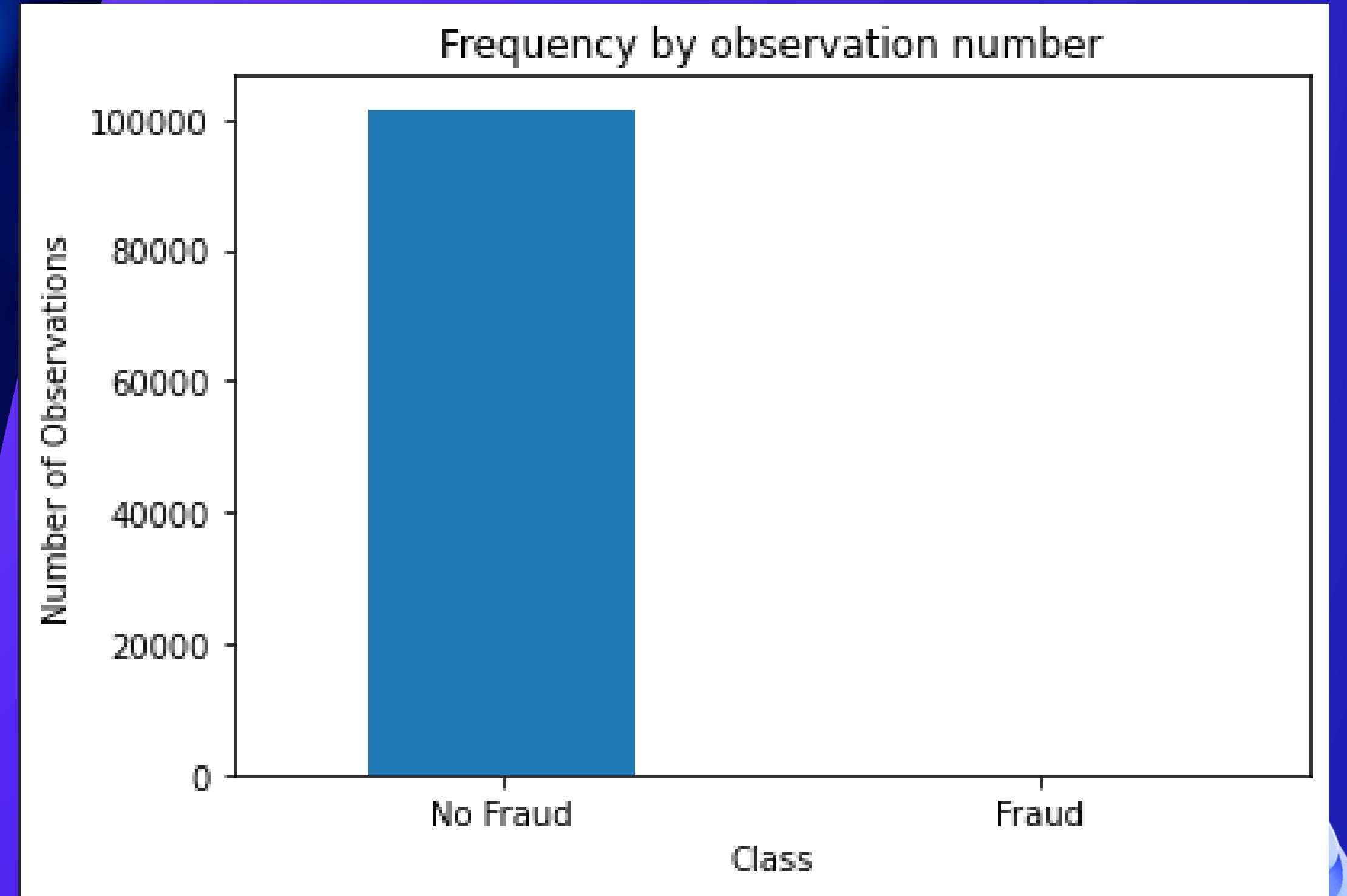
1. IMPORTAR LAS BIBLIOTECAS NECESARIAS

- Se importa la biblioteca pandas
- Se importan las funciones y clases necesarias de scikit-learn

2. CARGAR LOS DATOS

- Se utiliza la función read_csv de pandas para cargar los datos desde el archivo CSV llamado 'fraud.csv' y se almacenan en el objeto data.







3. SEPARAR LAS CARACTERÍSTICAS Y LA VARIABLE OBJETIVO:

- Se separan las características del conjunto de datos y se asignan a la variable X utilizando el método drop de pandas, excluyendo la columna 'isFraud'. Se importan las funciones y clases necesarias de scikit-learn
- Se asigna la variable objetivo 'isFraud' a la variable y.

4. DIVIDIR LOS DATOS EN CONJUNTOS DE ENTRENAMIENTO Y PRUEBA:

- Se utiliza la función train_test_split para dividir las características (X) y la variable objetivo (y) en conjuntos de entrenamiento y prueba.
- El parámetro test_size=0.2 indica que se destinará el 20% de los datos para el conjunto de prueba, mientras que el 80% restante se utilizará para el conjunto de entrenamiento.
- El parámetro random_state=42 se utiliza para establecer una semilla y asegurar la reproducibilidad de la división de datos.





5. DEFINIR LAS COLUMNAS NUMÉRICAS Y CATEGÓRICAS

- Se definen las columnas numéricas como una lista de nombres de columnas: ['Step', 'Amount', 'OldBalanceOrg', 'NewBalanceOrig', 'OldBalanceDest', 'NewBalanceDest'].
- Se define la columna categórica como una lista de un solo nombre de columna: ['Type'].

6. CREAR EL PREPROCESADOR

- Se crea un procesador utilizando ColumnTransformer para aplicar transformaciones específicas a las columnas numéricas y categóricas.
- El procesador tiene dos transformadores: uno para estandarizar las características numéricas utilizando StandardScaler y otro para codificar las características categóricas utilizando OneHotEncoder.



7. APlicar la transformación a los datos de entrenamiento y prueba

- Se utiliza el método `fit_transform` del preprocesador para aplicar la transformación a las características numéricas y categóricas del conjunto de entrenamiento (`X_train`), y se almacena en `X_train_scaled`.
- Se utiliza el método `transform` del preprocesador para aplicar la misma transformación al conjunto de prueba (`X_test`), y se almacena en `X_test_scaled`.

8. Entrenar un modelo de Regresión Logística

- Se crea un objeto `LogisticRegression`.
- Se entrena el modelo utilizando el método `fit` con las características de entrenamiento escaladas (`X_train_scaled`) y la variable objetivo de entrenamiento (`y_train`).



8. REALIZAR PREDICCIONES EN EL CONJUNTO DE PRUEBA:

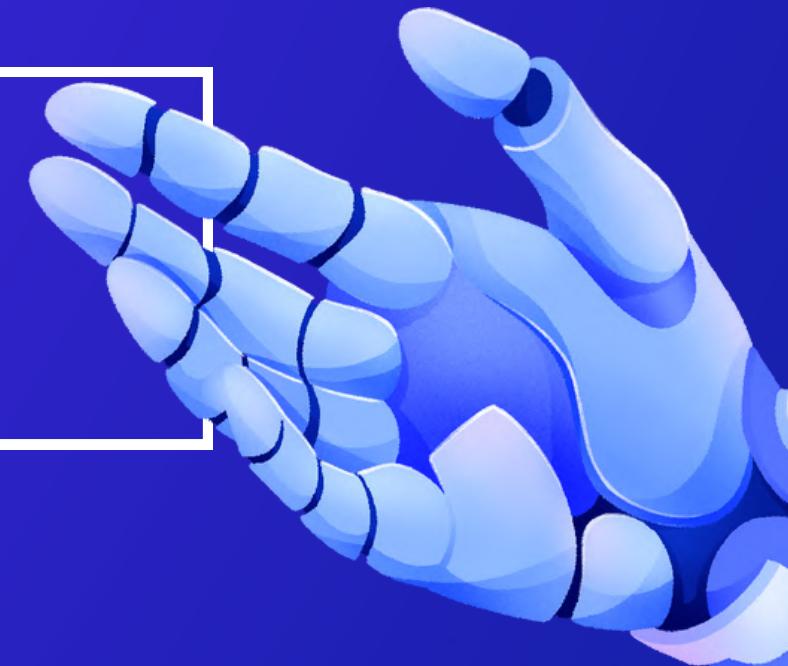
- Se utilizan las características de prueba escaladas (`x_test_scaled`) para realizar predicciones utilizando el método `predict` del modelo.
- Las predicciones se almacenan en la variable `y_pred`.

9. EVALUAR EL RENDIMIENTO DEL MODELO

- Se calcula la precisión del modelo comparando las predicciones (`y_pred`) con la variable objetivo de prueba (`y_test`) utilizando la función `accuracy_score`.
- Se calcula la matriz de confusión utilizando la función `confusion_matrix`.
- Se genera un informe de clasificación utilizando la función `classification_report`.

10. IMPRIMIR LOS RESULTADOS

- Se imprimen en la consola la precisión, la matriz de confusión y el informe de clasificación obtenidos.



RESULTADOS OBTENIDOS

Accuracy: 0.9990650986566944

Confusion Matrix:

```
[[20303     0]
 [ 19      1]]
```

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	20303
1	1.00	0.05	0.10	20
accuracy			1.00	20323
macro avg	1.00	0.53	0.55	20323
weighted avg	1.00	1.00	1.00	20323

RESUMEN

- El código proporcionado carga los datos, divide los conjuntos de entrenamiento y prueba, aplica el preprocessamiento, entrena un modelo de regresión logística y evalúa su rendimiento en el conjunto de prueba. Sin embargo, los resultados muestran que el modelo tiene dificultades para detectar de manera precisa las transacciones fraudulentas.
- Para mejorar la precisión en la detección de fraudes, se pueden considerar diferentes técnicas, como el uso de técnicas de balanceo de clases (como SMOTE-ENN o SMOTE-Tomek) para abordar el desequilibrio de clases en el conjunto de datos. Además, es posible ajustar los hiperparámetros del modelo de regresión logística, probar diferentes algoritmos de clasificación o incluso explorar enfoques de modelado más avanzados, como el uso de algoritmos de aprendizaje automático no lineales o modelos basados en árboles.
- Es importante experimentar con diferentes enfoques y técnicas para encontrar la mejor configuración que permita mejorar la detección de fraudes en tu conjunto de datos específico.

MUCHAS GRACIAS
POR VER ESTA PRESENTACIÓN

