



Detección de fraudes con tarjetas de crédito

Enlace al dataset: <https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>

Importa las bibliotecas necesarias

```
In [1]: # Importa la bibliotecas necesarias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings("ignore")
```

Importa y organiza el dataset

```
In [13]: # Organizar los datos en un dataframe
dataset = pd.read_csv('creditcard.csv')
print(dataset.head())
```

	Time	V1	V2	V3	V4	V5	V6	V7	\
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	

	V8	V9	...	V21	V22	V23	V24	V25	\
0	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	
1	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	
2	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	
3	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	
4	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	

	V26	V27	V28	Amount	Class
0	-0.189115	0.133558	-0.021053	149.62	0
1	0.125895	-0.008983	0.014724	2.69	0
2	-0.139097	-0.055353	-0.059752	378.66	0
3	-0.221929	0.062723	0.061458	123.50	0
4	0.502292	0.219422	0.215153	69.99	0

[5 rows x 31 columns]

► [Haz clic aquí para obtener una pista](#)

Limpia los datos

a. Valores perdidos

```
In [15]: print(dataset.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column      Non-Null Count  Dtype  
---  --
 0   Time        284807 non-null  float64
 1   V1          284807 non-null  float64
 2   V2          284807 non-null  float64
 3   V3          284807 non-null  float64
 4   V4          284807 non-null  float64
 5   V5          284807 non-null  float64
 6   V6          284807 non-null  float64
 7   V7          284807 non-null  float64
 8   V8          284807 non-null  float64
 9   V9          284807 non-null  float64
10  V10         284807 non-null  float64
11  V11         284807 non-null  float64
12  V12         284807 non-null  float64
13  V13         284807 non-null  float64
14  V14         284807 non-null  float64
15  V15         284807 non-null  float64
16  V16         284807 non-null  float64
17  V17         284807 non-null  float64
18  V18         284807 non-null  float64
19  V19         284807 non-null  float64
20  V20         284807 non-null  float64
21  V21         284807 non-null  float64
22  V22         284807 non-null  float64
23  V23         284807 non-null  float64
24  V24         284807 non-null  float64
25  V25         284807 non-null  float64
26  V26         284807 non-null  float64
27  V27         284807 non-null  float64
28  V28         284807 non-null  float64
29  Amount      284807 non-null  float64
30  Class       284807 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
None
```

```
In [17]: print(dataset.isnull().sum())
```

```
Time      0
V1         0
V2         0
V3         0
V4         0
V5         0
V6         0
V7         0
V8         0
V9         0
V10        0
V11        0
V12        0
V13        0
V14        0
V15        0
V16        0
V17        0
V18        0
V19        0
V20        0
V21        0
V22        0
V23        0
V24        0
V25        0
V26        0
V27        0
V28        0
Amount     0
Class      0
dtype: int64
```

```
In [19]: dataset.fillna(dataset.mean(), inplace=True) #Eliminar nulos
```

► Haz clic aquí para obtener una pista

b. Datos duplicados

```
In [23]: dataset.duplicated().sum()
```

```
Out[23]: 1081
```

```
In [25]: #Elimino Los duplicados
dataset = dataset.drop_duplicates()
```

► Haz clic aquí para obtener una pista

Analiza los datos

Pregunta 1: ¿Cuál es el porcentaje de transacciones fraudulentas en el dataset?

```
In [27]: #Calcula el porcentaje de transacciones fraudulentas
transacciones = len(dataset)
numero_transacciones_fraudulentas = dataset['Class'].sum()
#No hace falta usar un condicional para class=1 lo hayo directamente con la suma ya que Class tiene valores 1 para fraude y 0 para no fraude.
fraude = (numero_transacciones_fraudulentas / transacciones) * 100
#Muestra el porcentaje de transacciones fraudulentas
print(fraude)
```

```
0.1667101358352777
```

► Haz clic aquí para obtener una pista

Pregunta 2: ¿Cuál es el importe medio de las transacciones fraudulentas?

```
In [29]: #Seleccionando solo las filas donde la columna Class es igual a 1
transacciones_fraudulentas = dataset[dataset['Class'] == 1]
#Este método calcula la media (promedio) de los valores en la columna Amount
importe_medio_fraude = transacciones_fraudulentas['Amount'].mean()
#Muestra el importe medio
print(importe_medio_fraude)
```

```
123.87186046511628
```

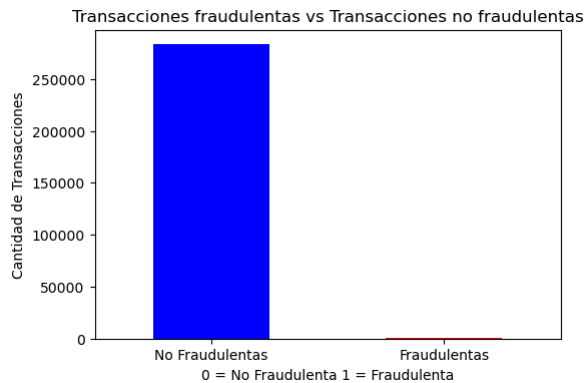
► Haz clic aquí para obtener una pista

Visualiza los datos

Pregunta 1: ¿Cuántas transacciones fraudulentas hay en comparación con las no fraudulentas? (Utiliza un gráfico de barras)

```
In [34]: #Cuenta la cantidad de 0s y 1s en la columna Class, devolviendo una Series de transacciones fraudulentas y no fraudulentas
numero_de_fraude = dataset['Class'].value_counts()

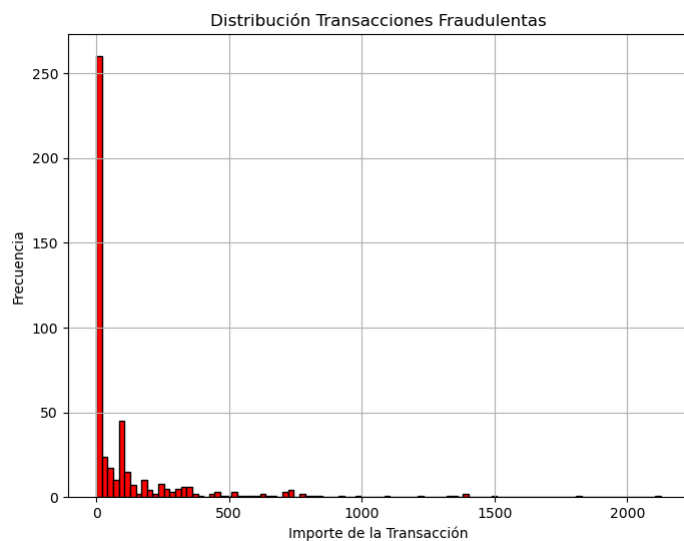
#Crear el gráfico de barras
#Configura el tamaño, tipo de gráfico y sus colores
plt.figure(figsize=(6,4))
numero_de_fraude.plot(kind='bar', color=['blue', 'red'])
#titulos
plt.title('Transacciones fraudulentas vs Transacciones no fraudulentas')
plt.xlabel('0 = No Fraudulenta 1 = Fraudulenta')
plt.ylabel('Cantidad de Transacciones')
#Asigno las etiquetas a los valores 0 y 1, incluyo rotation=0 para usar el eje x.
plt.xticks([0, 1], ['No Fraudulentas', 'Fraudulentas'], rotation=0)
plt.show()
```



► Haz clic aquí para obtener una pista

Pregunta 2: ¿Cuál es la distribución de los importes de las transacciones fraudulentas? (Utiliza un histograma)

```
In [38]: plt.figure(figsize=(8,6)) #Tamaño
plt.hist(transacciones_fraudulentas['Amount'], bins=100, color='red', edgecolor='black') #columna y color
plt.title('Distribución Transacciones Fraudulentas') #Titulo del gráfico
plt.xlabel('Importe de la Transacción') #Titulo del eje X
plt.ylabel('Frecuencia') #Titulo del eje Y
plt.grid(True) #Añadir una cuadrícula para la lectura
plt.show()
```



► Haz clic aquí para obtener una pista

Desarrollo y evaluación de modelos

Separa del dataset

```
In [43]: # Crear el dataframe
X = dataset.drop('Class', axis=1) # Todas Las columnas excepto 'Class'
y = dataset['Class'] # Solo la columna 'Class'

# Dividir los datos en entrenamiento (80 %) y evaluación (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Imprimir Las dimensiones de Los datos divididos
print(f'Datos de entrenamiento - X_train: {X_train.shape}, y_train: {y_train.shape}')
print(f'Datos de prueba - X_test: {X_test.shape}, y_test: {y_test.shape}')
#Usa print(f"...")
#El método shape de pandas se utiliza para verificar Las dimensiones de Los conjuntos resultantes (número de filas y columnas).

Datos de entrenamiento - X_train: (226980, 30), y_train: (226980,)
Datos de prueba - X_test: (56746, 30), y_test: (56746,)
```

► Haz clic aquí para obtener una pista

Crea y evalúa los modelos

In [47]: `#Transformar Los datos para mejorar el rendimiento de ciertos modelos como La Regresión Logística y SVM`

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
#Crear una instancia de RandomForestClassifier
random_forest_model = RandomForestClassifier(max_depth=150, random_state=42)
#Entrenar el modelo
random_forest_model.fit(X_train_scaled, y_train)
#Predicciones sobre Prueba
y_pred = random_forest_model.predict(X_test_scaled)
#Evaluar el rendimiento
classification_rep = classification_report(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
# Mostrar el informe de clasificación y la precisión
print("Informe de clasificación:")
print(classification_rep)
print(f"\nExactitud del modelo: {accuracy * 100:.2f}%")
#accuracy por 100 convierte este valor decimal en un porcentaje.
#.2f significa que el número será mostrado como un número de punto flotante con 2 decimales.
```

Informe de clasificación:

	precisión	recall	f1-score	support
0	1.00	1.00	1.00	56656
1	0.97	0.73	0.84	90
accuracy			1.00	56746
macro avg	0.99	0.87	0.92	56746
weighted avg	1.00	1.00	1.00	56746

Exactitud del modelo: 99.95%

► [Haz clic aquí para obtener una pista](#)