Detección de fraudes con tarjetas de crédito

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

Importa las bibliotecas necesarias

```
# Importa la bibliotecas necesarias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
```

Importa y organiza el dataset

```
# Organizar los datos en un dataframe
datos = pd.read_csv("creditcard.csv")
datos.head(10)
```

```
   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
5   2.0 -0.425966  0.960523  1.141109 -0.168252  0.420987 -0.029728
0.476201
6   4.0  1.229658  0.141004  0.045371  1.202613  0.191881  0.272708 -
0.005159
7   7.0 -0.644269  1.417964  1.074380 -0.492199  0.948934  0.428118
1.120631
8   7.0 -0.894286  0.286157 -0.113192 -0.271526  2.669599  3.721818
0.370145
9   9.0 -0.338262  1.119593  1.044367 -0.222187  0.499361 -0.246761
0.651583

         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
5  0.260314 -0.568671  ... -0.208254 -0.559825 -0.026398 -0.371427 -
0.232794
6  0.081213  0.464960  ... -0.167716 -0.270710 -0.154104 -0.780055
0.750137
```

```
7 -3.807864  0.615375  ...  1.943465 -1.015455  0.057504 -0.649709 -
0.415267
8  0.851084 -0.392048  ... -0.073425 -0.268092 -0.204233  1.011592
0.373205
9  0.069539 -0.736727  ... -0.246914 -0.633753 -0.120794 -0.385050 -
0.069733

         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  0.105915  0.253844  0.081080    3.67      0
6 -0.257237  0.034507  0.005168    4.99      0
7 -0.051634 -1.206921 -1.085339   40.80      0
8 -0.384157  0.011747  0.142404   93.20      0
9  0.094199  0.246219  0.083076    3.68      0

[10 rows x 31 columns]
```

Limpia los datos

a. Valores perdidos

```
#Observamos que no hay elementos nulos
datos.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
```

b. Datos duplicados

```
#Nº de valores duplicados
datos.duplicated().sum()
print(f'Los datos duplicados son: {datos.duplicated().sum()}')
#Borramos las filas duplicadas
datos_limpios=datos.drop_duplicates()

#Comprobamos que esté correcto
datos_limpios.duplicated().sum()
print(f'Los datos duplicados tras la limpieza son:
{datos_limpios.duplicated().sum()}')
```

```
Los datos duplicados son: 1081
Los datos duplicados tras la limpieza son: 0
```

Analiza los datos

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

```
# Calcula el porcentaje de transacciones fraudulentas
total_transacciones = len(datos_limpios)
transacciones_fraudulentas = len(datos_limpios[datos_limpios['Class']
== 1])
porcentaje = ((transacciones_fraudulentas/total_transacciones)*100)
# Muestra el porcentaje de transacciones fraudulentas
print(f'El porcentaje de transacciones fraudulentas es:
{porcentaje:.3} %')
```

```
El porcentaje de transacciones fraudulentas es: 0.167 %
```

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

```
# Calcula el importe medio de las transacciones fraudulentas
trans_fraudulentas = datos_limpios[datos_limpios['Class']==1]
media_fraud=round(trans_fraudulentas['Amount'].mean(),3)
# Muestra el importe medio de las transacciones fraudulentas
```

```
print (f'El importe medio de las transacciones fraudulentas es:
{media_fraud}')
```

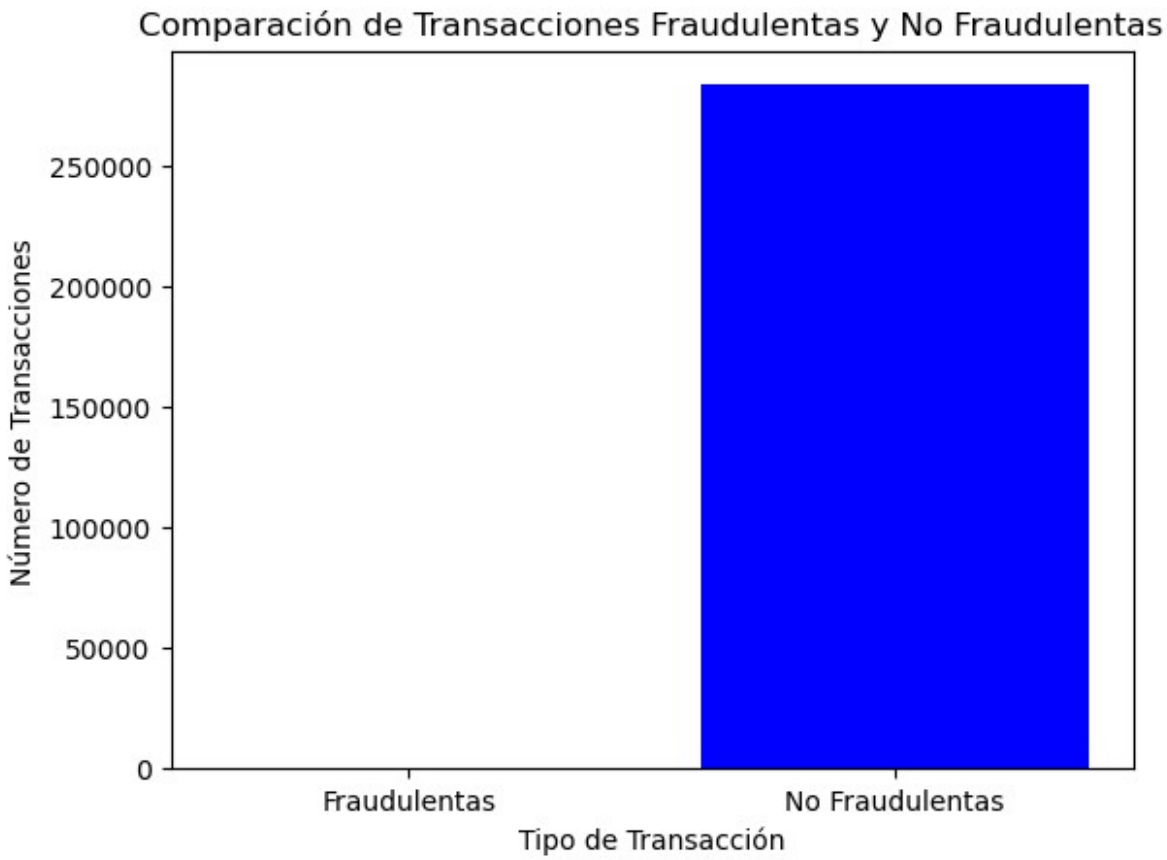
```
El importe medio de las transacciones fraudulentas es: 123.872
```

Visualiza los datos

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

```
# Cuenta el número de transacciones fraudulentas y no fraudulentes
trans_fraud=(datos_limpios['Class']==1).sum()
print(f'El número de transacciones fraudulentas es de {trans_fraud}')
trans_nofraud= (datos_limpios['Class']==0).sum()
print(f'El número de transacciones no fraudulentas es de
{trans_nofraud}')
# Muestra la distribución de las traducciones fraudulentas con
respecto de las no fraudulentas
#Crear los datos para el gráfico
labels = ['Fraudulentas', 'No Fraudulentas']
values = [trans_fraud, trans_nofraud]
# Crear el gráfico de barras
plt.bar(labels, values, color=['red', 'blue'])
plt.xlabel('Tipo de Transacción')
plt.ylabel('Número de Transacciones')
plt.title('Comparación de Transacciones Fraudulentas y No
Fraudulentas')
plt.show()
```

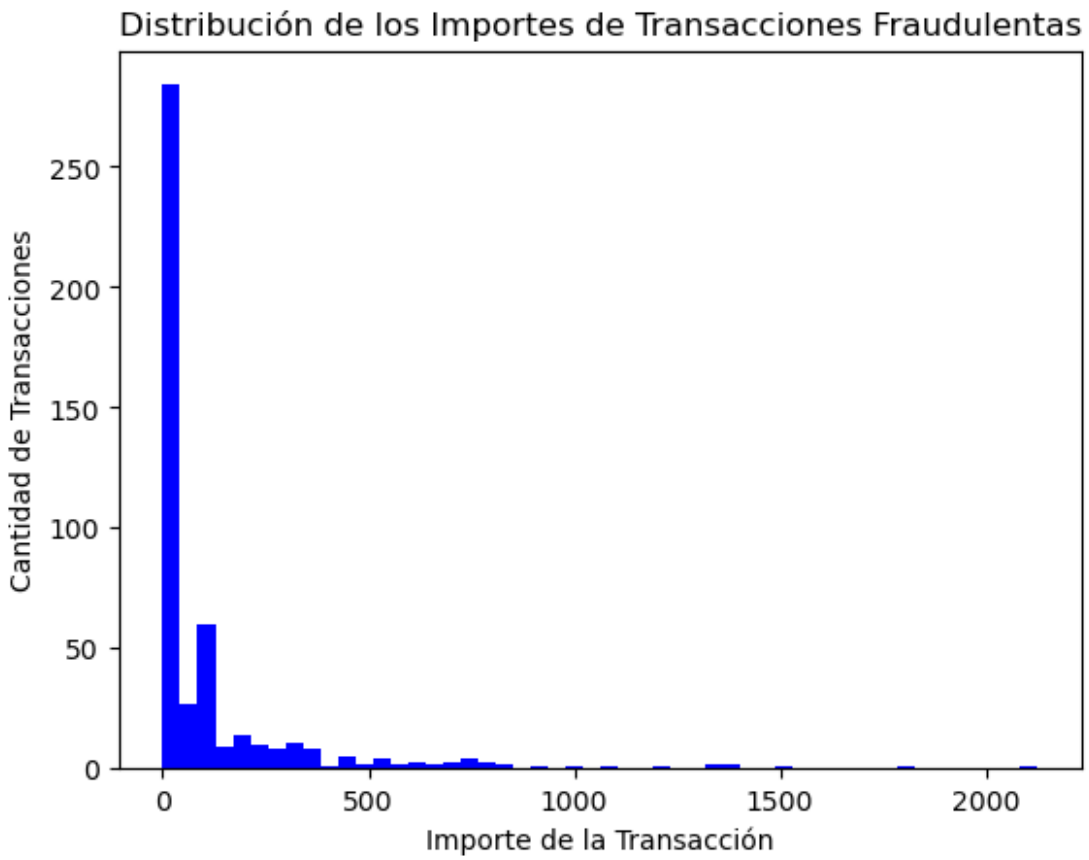
```
El número de transacciones fraudulentas es de 473
El número de transacciones no fraudulentas es de 283253
```



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

```
# Separa los datos de transacciones fraudulentas
trans_fraud=datos_limpios[datos_limpios['Class']==1]
# Muestra la distribución de los importes de las transacciones
fraudulentas
plt.hist(trans_fraud['Amount'], bins=50, color='blue')
plt.xlabel('Importe de la Transacción')
plt.ylabel('Cantidad de Transacciones')
plt.title('Distribución de los Importes de Transacciones
Fraudulentas')
plt.show
print ('Observamos que la mayoría de las transacciones fraudulentas se
producen en transacciones en torno a los 50€')
```

```
Observamos que la mayoría de las transacciones fraudulentas se
producen en transacciones en torno a los 50€
```



Desarrollo y evaluación de modelos

Separa del dataset

```
# Separa los datos de entrenamiento y evaluación
#Primero importamos el módulo
from sklearn.model_selection import train_test_split

#Creamos los dataframes X e y
X= datos_limpios.drop('Class', axis=1)
y= datos_limpios['Class']

X_train, X_test, y_train, y_test= train_test_split(X,y, test_size=0.2,
random_state=45)
```

Crea y evalúa los modelos

```
#Escribe tu código aquí
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score

# Creamos y entrenamos el modelo
```

```python
rf = RandomForestClassifier(n_estimators=80, max_depth=150,
random_state=42)
rf.fit(X_train, y_train)

#Hacemos las predicciones
predicciones = rf.predict(X_test)

#Evaluamos el modelo
metricas = classification_report(y_test, predicciones)
print(f'Métricas de clasificación:\n {metricas}')

exactitud = round(accuracy_score(y_test, predicciones)*100, 3)
print(f'El porcentaje de exactitud del modelo es del {exactitud}%')
```

```
Métricas de clasificación:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56665
           1       0.88      0.70      0.78        81

    accuracy                           1.00     56746
   macro avg       0.94      0.85      0.89     56746
weighted avg       1.00      1.00      1.00     56746

El porcentaje de exactitud del modelo es del 99.944%
```