```
# Importar las 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.preprocessing import StandardScaler
from sklearn.svm import
SVC
from sklearn.metrics import classification_report, confusion_matrix

# Ignoraramos
advertencias
import warnings
warnings.filterwarnings("ignore")

# Cargo el
dataset desde el archivo subido
df = pd.read_csv(r'C:\Users\Alberto\Desktop\Proyecto -
Ciencia de datos\creditcard.csv')

# Mostraremos las primeras filas del
dataframe
print("Primeras filas del dataframe:")
print(df.head())

# Limpiamos
los datos
# Eliminaremos valores nulos
df.dropna(inplace=True)

# Comprobamos los valores
duplicados
duplicados = df.duplicated().sum()
print(f"Duplicados antes de eliminar:
{duplicados}")
df.drop_duplicates(inplace=True)
duplicados =
df.duplicated().sum()
print(f"Duplicados después de eliminar: {duplicados}")

#
Análisis de datos

#Comenzamos a resolver las cuestiones

# Porcentaje de transacciones
fraudulentas
fraud_percentage = (df['Class'].sum() / len(df)) * 100
print(f"El
porcentaje de transacciones fraudulentas es del {fraud_percentage:.2f}%")

# Importe
medio de las transacciones fraudulentas
fraud_data = df[df['Class'] ==
1]
average_fraud_amount = fraud_data['Amount'].mean()
print(f"El importe medio
de las transacciones fraudulentas es de {average_fraud_amount:.2f}")

# Visualización
de datos
# Transacciones fraudulentas vs no fraudulentas
fraud_counts =
df['Class'].value_counts()
plt.bar(['No fraude', 'Fraude'],
fraud_counts, color=['blue', 'red'])
plt.xlabel('Tipo de
transacción')
plt.ylabel('Cantidad de
transacciones')
plt.title('Transacciones fraudulentas vs no
```

```
fraudulentas')
plt.show()

# Distribución de los importes de las transacciones
fraudulentas
plt.hist(fraud_data['Amount'], bins=50, edgecolor='black',
color='red')
plt.xlabel('Importe de transacciones
fraudulentas')
plt.ylabel('Frecuencia')
plt.title('Distribución de
importes de transacciones fraudulentas')
plt.show()

# Separación del dataset para
entrenamiento y prueba
X = df.drop('Class', axis=1)
y =
df['Class']
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train,
X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=1)

#
Creación y evaluación del modelo SVC
model_svc = SVC()
model_svc.fit(X_train,
y_train)
train_score = model_svc.score(X_train, y_train)
test_score = model_svc.score(X_test,
y_test)

print(f"Puntuación del modelo en entrenamiento:
{train_score:.2f}")
print(f"Puntuación del modelo en prueba:
{test_score:.2f}")

# Matriz de confusión y reporte de clasificación
y_pred =
model_svc.predict(X_test)
cm = confusion_matrix(y_test, y_pred, labels=[1, 0])
confusion_df =
pd.DataFrame(cm, index=['Es fraudulenta', 'Es normal'],

         columns=['Predicción fraudulenta', 'Predicción
normal'])
print("Matriz de
confusión:")
print(confusion_df)
print("Reporte de
clasificación:")
print(classification_report(y_test, y_pred))
```