```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import gridspec

data = pd.read_csv("creditcard.csv")

data.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]

print(data.shape)
print(data.describe())

(284807, 31)
                Time             V1             V2             V3
V4  \
count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05
2.848070e+05
mean    94813.859575  1.168375e-15  3.416908e-16 -1.379537e-15
2.074095e-15
std     47488.145955  1.958696e+00  1.651309e+00  1.516255e+00
1.415869e+00
min         0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01 -
5.683171e+00
25%     54201.500000 -9.203734e-01 -5.985499e-01 -8.903648e-01 -
8.486401e-01
50%     84692.000000  1.810880e-02  6.548556e-02  1.798463e-01 -
1.984653e-02
75%    139320.500000  1.315642e+00  8.037239e-01  1.027196e+00
7.433413e-01
max    172792.000000  2.454930e+00  2.205773e+01  9.382558e+00
1.687534e+01

                 V5             V6             V7             V8
V9  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
2.848070e+05
mean   9.604066e-16  1.487313e-15 -5.556467e-16  1.213481e-16 -
2.406331e-15
std    1.380247e+00  1.332271e+00  1.237094e+00  1.194353e+00
```

```
1.098632e+00
min   -1.137433e+02 -2.616051e+01 -4.355724e+01 -7.321672e+01 -
1.343407e+01
25%   -6.915971e-01 -7.682956e-01 -5.540759e-01 -2.086297e-01 -
6.430976e-01
50%   -5.433583e-02 -2.741871e-01  4.010308e-02  2.235804e-02 -
5.142873e-02
75%    6.119264e-01  3.985649e-01  5.704361e-01  3.273459e-01
5.971390e-01
max    3.480167e+01  7.330163e+01  1.205895e+02  2.000721e+01
1.559499e+01

       ...            V21           V22           V23           V24  \
count  ...  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   ...  1.654067e-16 -3.568593e-16  2.578648e-16  4.473266e-15
std    ...  7.345240e-01  7.257016e-01  6.244603e-01  6.056471e-01
min    ... -3.483038e+01 -1.093314e+01 -4.480774e+01 -2.836627e+00
25%    ... -2.283949e-01 -5.423504e-01 -1.618463e-01 -3.545861e-01
50%    ... -2.945017e-02  6.781943e-03 -1.119293e-02  4.097606e-02
75%    ...  1.863772e-01  5.285536e-01  1.476421e-01  4.395266e-01
max    ...  2.720284e+01  1.050309e+01  2.252841e+01  4.584549e+00

                V25           V26           V27           V28
Amount  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
284807.000000
mean   5.340915e-16  1.683437e-15 -3.660091e-16 -1.227390e-16
88.349619
std    5.212781e-01  4.822270e-01  4.036325e-01  3.300833e-01
250.120109
min   -1.029540e+01 -2.604551e+00 -2.256568e+01 -1.543008e+01
0.000000
25%   -3.171451e-01 -3.269839e-01 -7.083953e-02 -5.295979e-02
5.600000
50%    1.659350e-02 -5.213911e-02  1.342146e-03  1.124383e-02
22.000000
75%    3.507156e-01  2.409522e-01  9.104512e-02  7.827995e-02
77.165000
max    7.519589e+00  3.517346e+00  3.161220e+01  3.384781e+01
25691.160000

              Class
count  284807.000000
mean        0.001727
std         0.041527
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000
```

```
[8 rows x 31 columns]
```

```
fraud = data[data['Class'] == 1]
valid = data[data['Class'] == 0]
outlierFraction = len(fraud)/float(len(valid))
print(outlierFraction)
print('Fraud Cases: {}'.format(len(data[data['Class'] == 1])))
print('Valid Transactions: {}'.format(len(data[data['Class'] == 0])))
```

```
0.0017304750013189597
Fraud Cases: 492
Valid Transactions: 284315
```

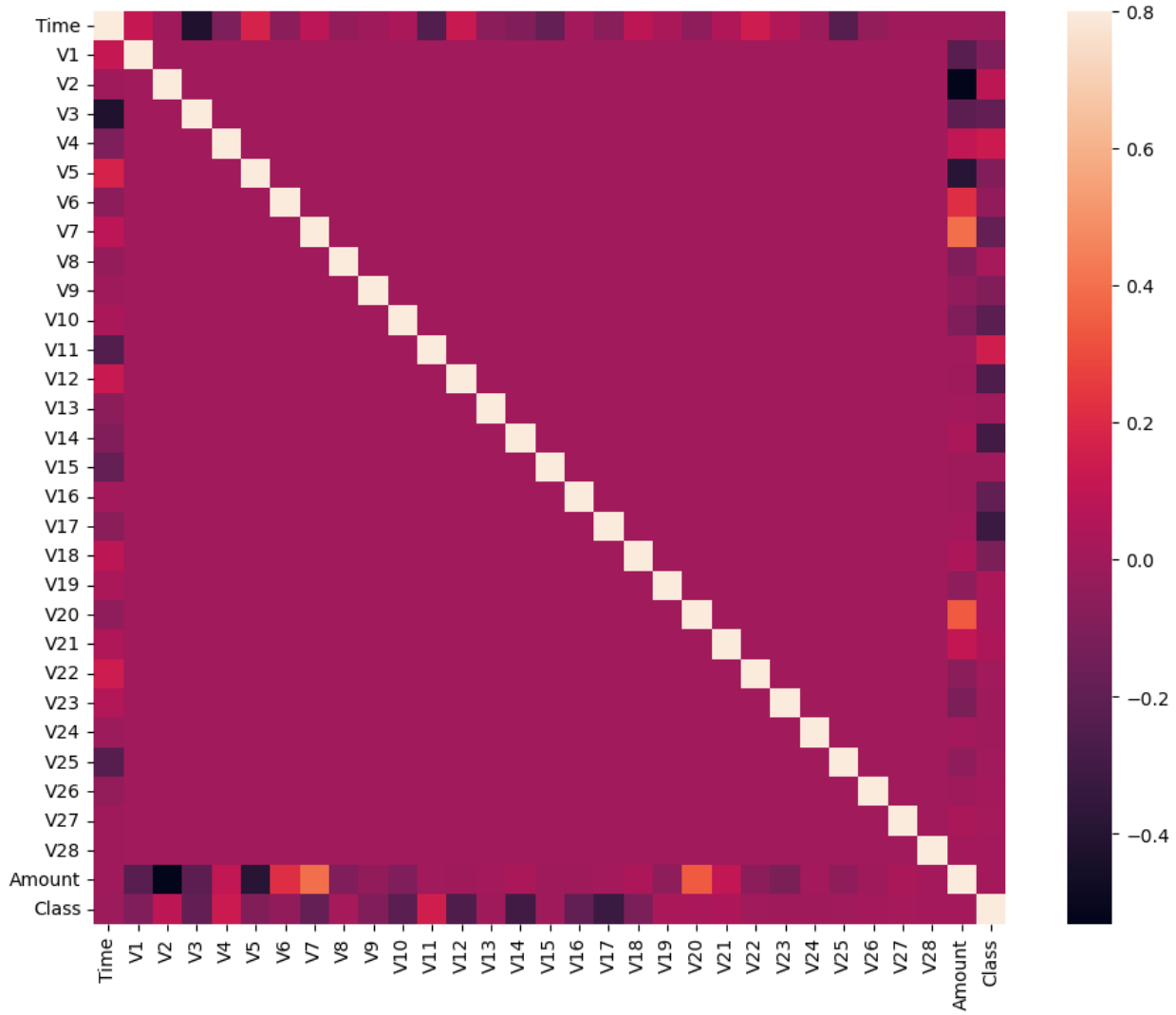
```
fraud.Amount.describe()
```

```
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

```
valid.Amount.describe()
```

```
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```

```
corrmat = data.corr()
fig = plt.figure(figsize = (12, 9))
sns.heatmap(corrmat, vmax = .8, square = True)
plt.show()
```




```
X = data.drop(['Class'], axis = 1)
Y = data["Class"]
print(X.shape)
print(Y.shape)
xData = X.values
yData = Y.values
```

```
(284807, 30)
(284807,)
```

```
from sklearn.model_selection import train_test_split

xTrain, xTest, yTrain, yTest = train_test_split(xData, yData,
test_size = 0.2, random_state = 42)

from sklearn.ensemble import RandomForestClassifier
```

```
rfc = RandomForestClassifier()
rfc.fit(xTrain, yTrain)
```

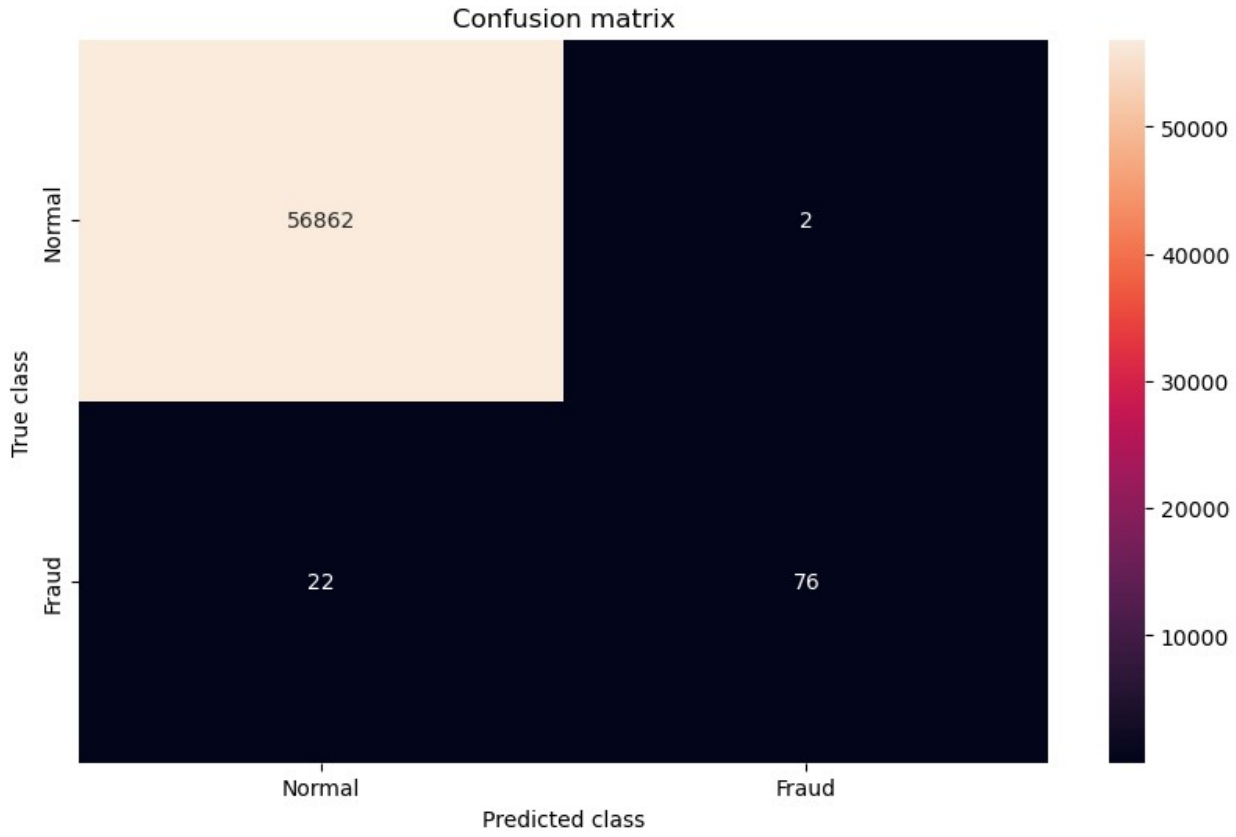
```
RandomForestClassifier()
```

```
yPred = rfc.predict(xTest)
```

```
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score, matthews_corrcoef
from sklearn.metrics import confusion_matrix

n_outliers = len(fraud)
n_errors = (yPred != yTest).sum()
print("The model used is Random Forest classifier")

acc = accuracy_score(yTest, yPred)
print("The accuracy is {}".format(acc))

prec = precision_score(yTest, yPred)
print("The precision is {}".format(prec))

rec = recall_score(yTest, yPred)
print("The recall is {}".format(rec))

f1 = f1_score(yTest, yPred)
print("The F1-Score is {}".format(f1))

MCC = matthews_corrcoef(yTest, yPred)
print("The Matthews correlation coefficient is{}".format(MCC))
```

```
The model used is Random Forest classifier
The accuracy is 0.9995786664794073
The precision is 0.9743589743589743
The recall is 0.7755102040816326
The F1-Score is 0.8636363636363636
The Matthews correlation coefficient is0.8690748763736589
```

```
LABELS = ['Normal', 'Fraud']
conf_matrix = confusion_matrix(yTest, yPred)
plt.figure(figsize =(10, 6))
sns.heatmap(conf_matrix, xticklabels = LABELS, yticklabels = LABELS,
annot = True, fmt ="d");
plt.title("Confusion matrix")
plt.ylabel('True class')
plt.xlabel('Predicted class')
plt.show()
```

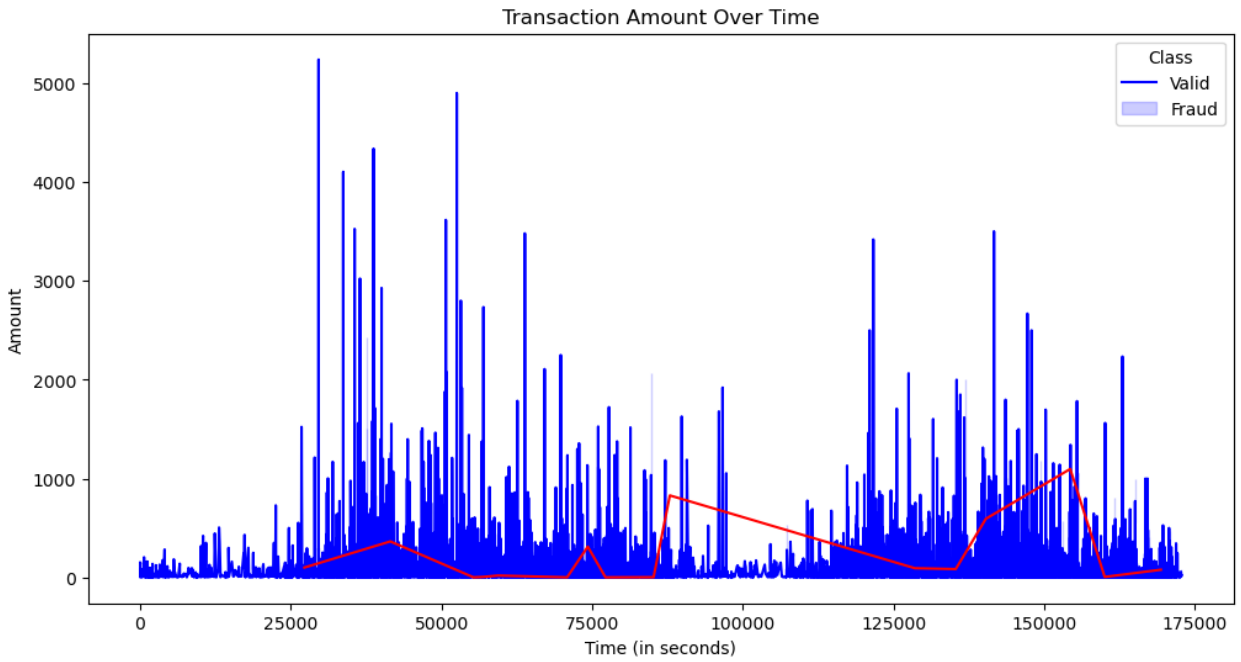



```
from sklearn.metrics import classification_report

# Generate the classification report
report = classification_report(yTest, yPred, target_names=['Valid',
'Fraud'])
print("Classification Report:")
print(report)
```

```
Classification Report:
              precision    recall  f1-score   support

       Valid       1.00      1.00      1.00     56864
       Fraud       0.97      0.78      0.86        98

    accuracy                           1.00     56962
   macro avg       0.99      0.89      0.93     56962
weighted avg       1.00      1.00      1.00     56962
```

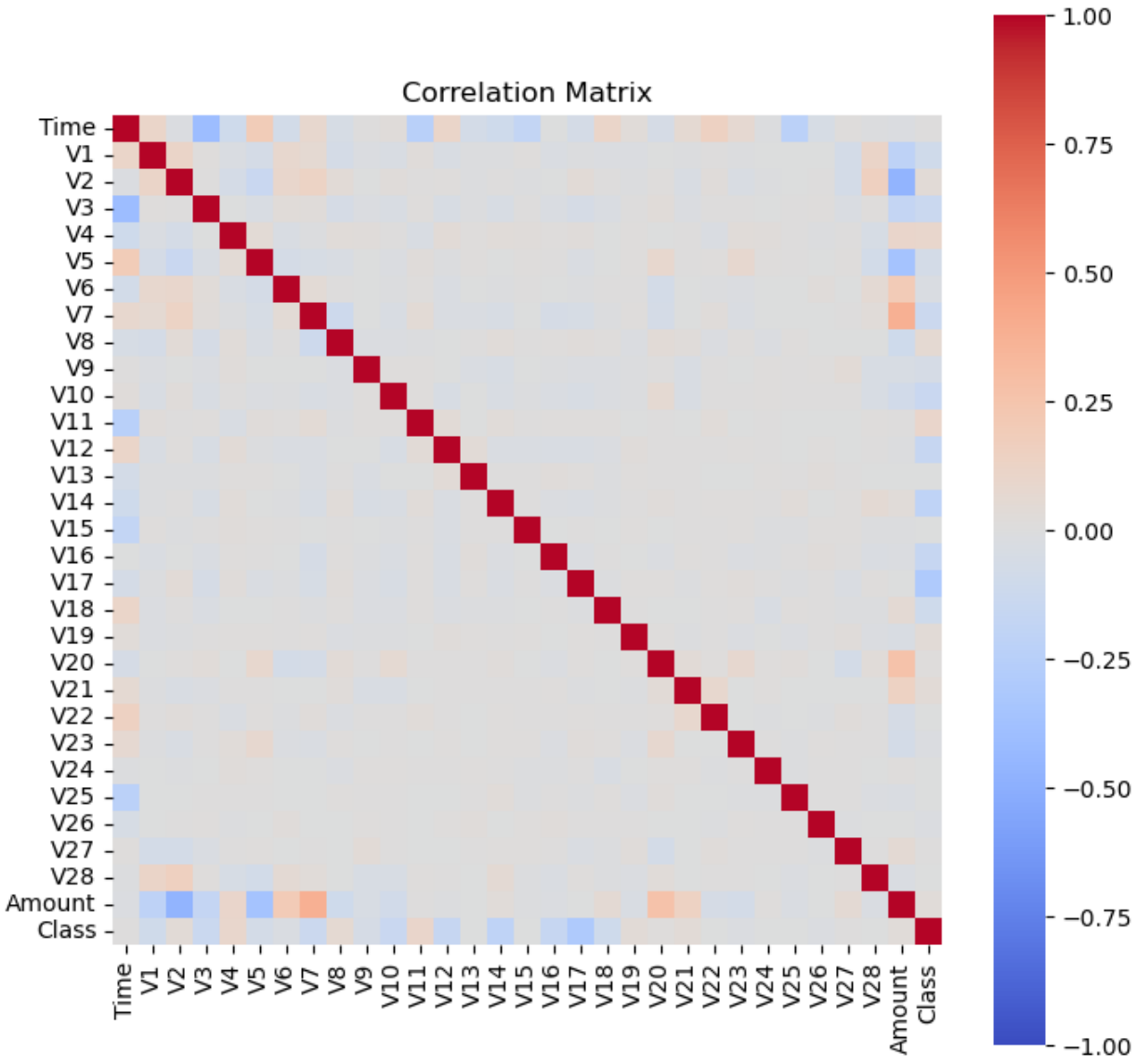
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

# Load the dataset
data = pd.read_csv('creditcard.csv')

# Reducing the dataset size by sampling 10,000 rows
data_sampled = data.sample(n=10000, random_state=42)

# Splitting data into fraud and valid transactions
fraud = data_sampled[data_sampled['Class'] == 1]
valid = data_sampled[data_sampled['Class'] == 0]

plt.figure(figsize=(12, 6))
sns.lineplot(data=data_sampled, x='Time', y='Amount', hue='Class',
palette={0: 'blue', 1: 'red'})
plt.title('Transaction Amount Over Time')
plt.xlabel('Time (in seconds)')
plt.ylabel('Amount')
plt.legend(title='Class', labels=['Valid', 'Fraud'])
plt.show()
```



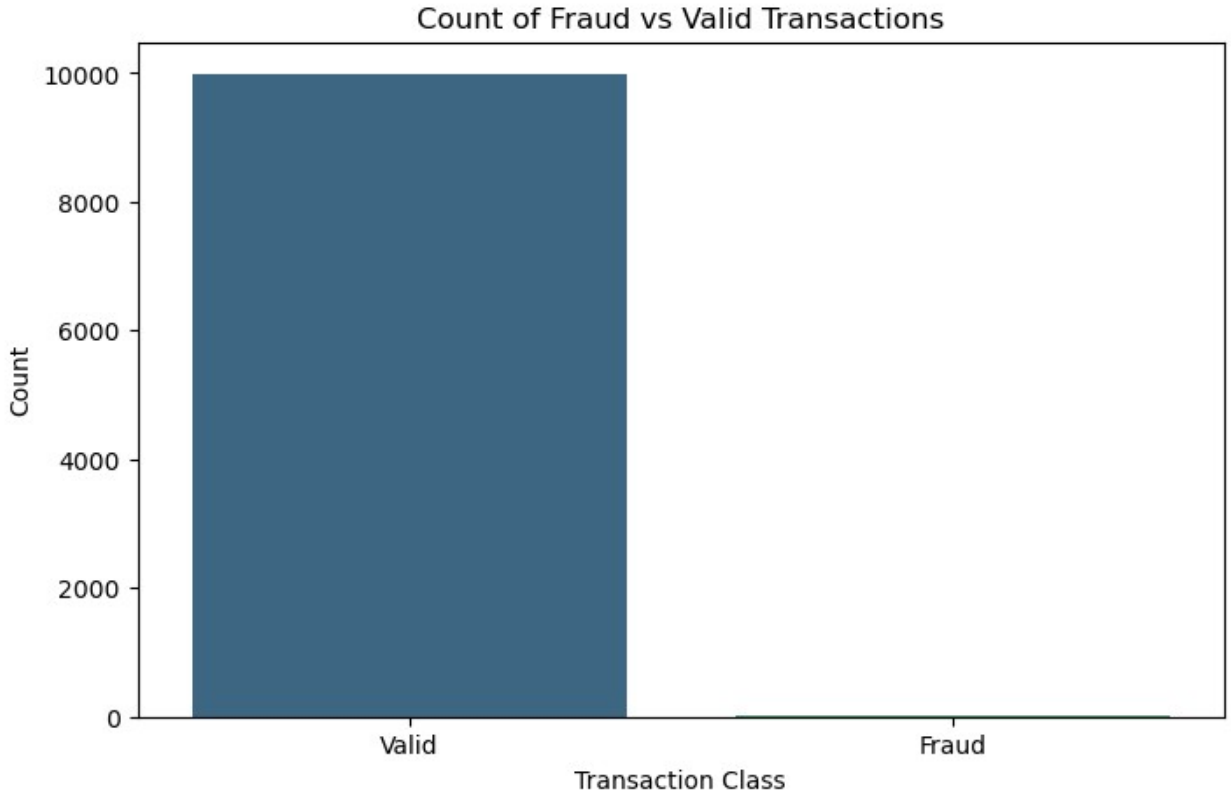

```
corr_matrix = data_sampled.corr()
plt.figure(figsize=(8, 8))
sns.heatmap(corr_matrix, annot=False, fmt='.2f', cmap='coolwarm',
square=True, vmax=1.0, vmin=-1.0)
```

```
plt.title('Correlation Matrix')
plt.show()
```

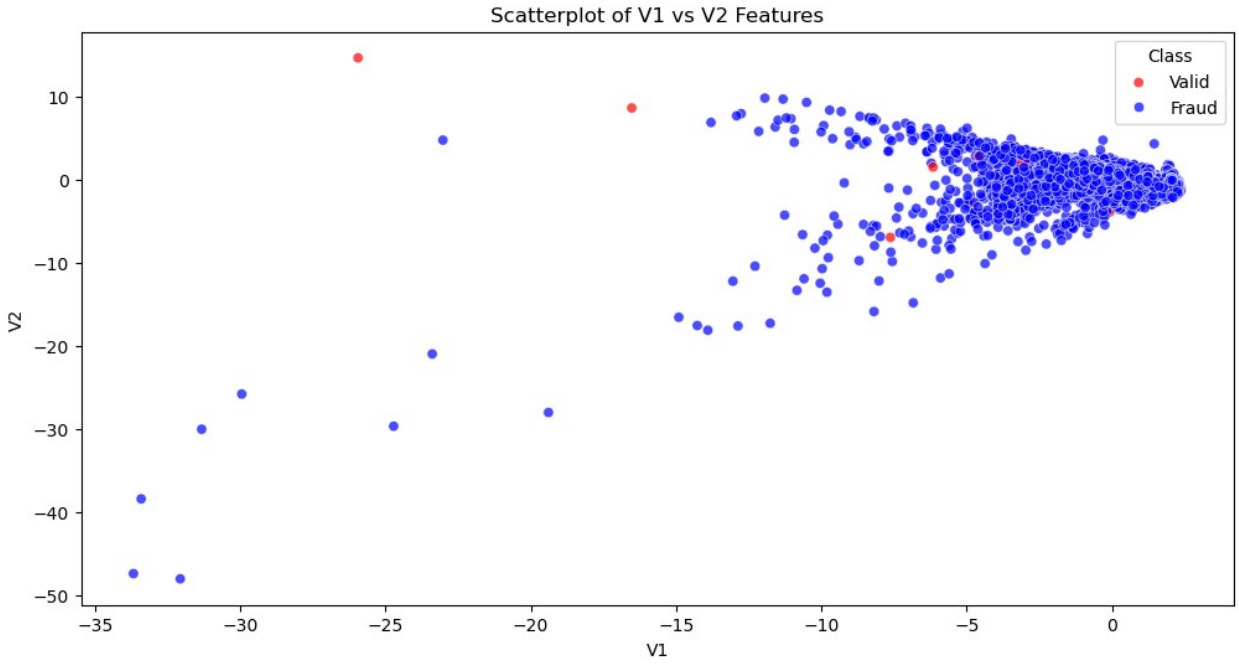



```
plt.figure(figsize=(8, 5))
sns.barplot(x=['Valid', 'Fraud'], y=[len(valid), len(fraud)],
palette='viridis')
plt.title('Count of Fraud vs Valid Transactions')
plt.xlabel('Transaction Class')
plt.ylabel('Count')
plt.show()

C:\Users\Akshay\AppData\Local\Temp\ipykernel_11660\3239519294.py:2:
FutureWarning:
```

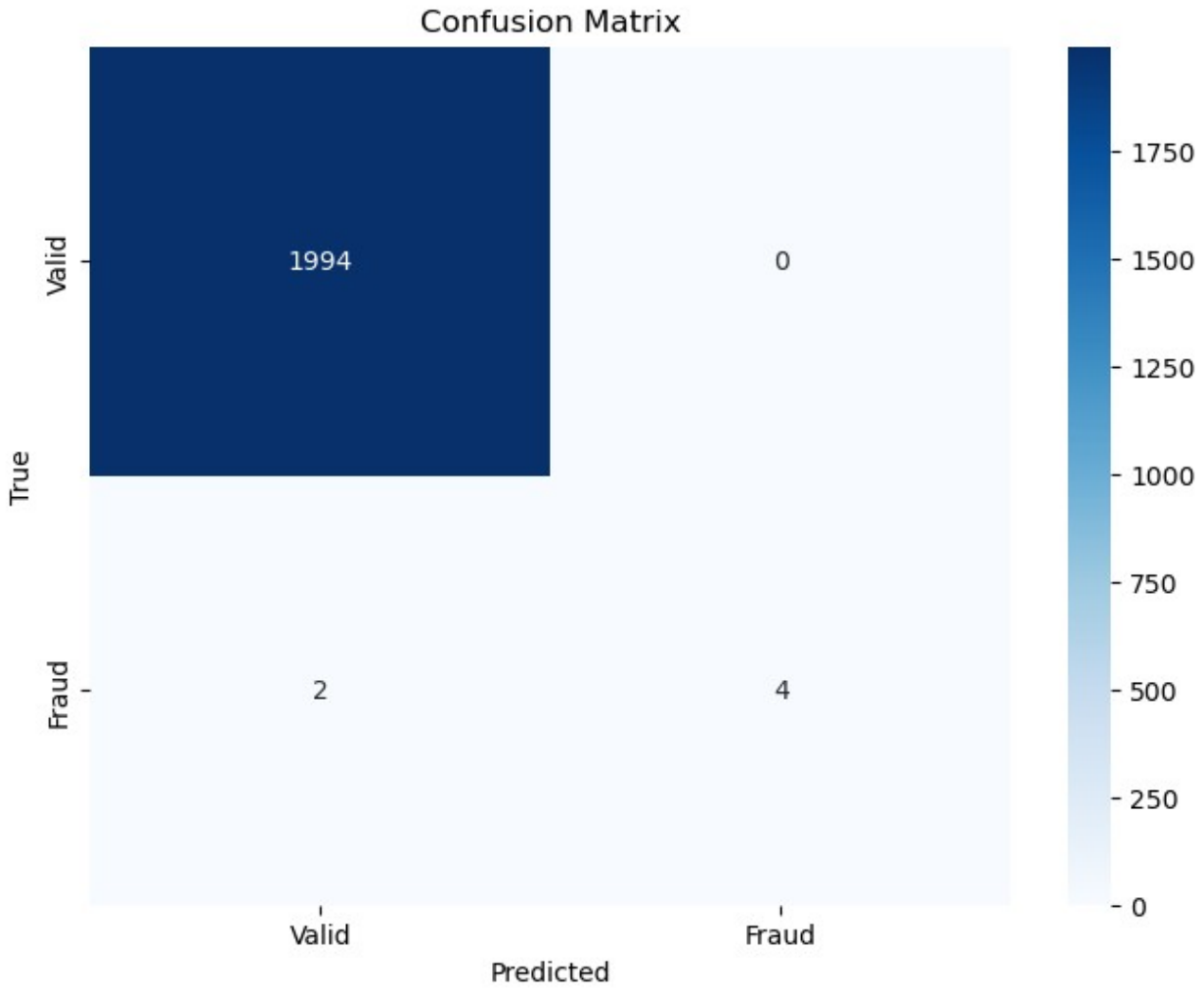
```
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=['Valid', 'Fraud'], y=[len(valid), len(fraud)], palette='viridis')
```




```
plt.figure(figsize=(12, 6))
sns.scatterplot(data=data_sampled, x='V1', y='V2', hue='Class', palette={0: 'blue', 1: 'red'}, alpha=0.7)
plt.title('Scatterplot of V1 vs V2 Features')
plt.xlabel('V1')
plt.ylabel('V2')
plt.legend(title='Class', labels=['Valid', 'Fraud'])
plt.show()
```




```
# 6. Confusion Matrix Heatmap
# Preparing data for RandomForestClassifier
X = data_sampled.drop('Class', axis=1)
y = data_sampled['Class']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
y_pred = rfc.predict(X_test)

conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=['Valid', 'Fraud'], yticklabels=['Valid', 'Fraud'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```


Confusion Matrix
1994
0
2
4
Valid
Fraud
True
Valid
Fraud
Predicted
1750
1500
1250
1000
750
500
250
0