```
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("/content/creditcard.csv")
```

```
data.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V2
0	0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.11047
1	0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.10128
2	1	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.90941
3	1	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.19032
4	2	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.13745

5 rows × 31 columns

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

```
(3973, 31)
               Time           V1           V2           V3           V4  \
count  3973.000000  3973.000000  3973.000000  3973.000000  3973.000000
mean   1638.724138    -0.333723     0.308691     0.837062     0.013144
std    1016.577498     1.351976     1.180436     0.981057     1.418751
min       0.000000   -12.168192   -15.732974   -12.389545    -4.657545
25%     750.000000    -1.030747    -0.155734     0.287224    -0.908469
50%    1526.000000    -0.451819     0.409641     0.884503     0.099497
75%    2526.000000     1.070572     0.921588     1.433851     0.982710
max    3624.000000     1.685314     6.118940     4.017561     6.013346

                 V5           V6           V7           V8           V9  ...  \
count  3973.000000  3973.000000  3973.000000  3973.000000  3973.000000  ...
mean     -0.029177     0.052836     0.146895    -0.071352     0.055778  ...
std       1.207632     1.286610     1.090769     1.294647     0.919087  ...
min     -32.092129    -7.465603   -11.164794   -23.632502    -3.336805  ...
25%      -0.573740    -0.735876    -0.278697    -0.205430    -0.473894  ...
50%      -0.089809    -0.209577     0.162196     0.028639     0.032286  ...
75%       0.396675     0.449217     0.635055     0.298866     0.622704  ...
max      10.658654    21.393069    34.303177     3.877662     6.450992  ...

                V21          V22          V23          V24          V25  \
count  3973.000000  3973.000000  3972.000000  3972.000000  3972.000000
mean      0.010465    -0.082885    -0.058484     0.029106     0.095278
std       0.855829     0.635482     0.391158     0.603965     0.405989
min     -11.273890    -5.707801    -7.996811    -2.162523    -2.322906
25%      -0.211812    -0.512434    -0.210615    -0.338940    -0.141891
50%      -0.053234    -0.064435    -0.069939     0.100635     0.108219
75%       0.102520     0.345346     0.062502     0.432692     0.361309
max      15.631453     4.393846     4.095021     1.215279     1.727063

                V26          V27          V28       Amount        Class
count  3972.000000  3972.000000  3972.000000  3972.000000  3972.000000
mean     -0.012623     0.044638     0.002639    65.000101     0.000504
std       0.503818     0.352752     0.254136   213.688183     0.022437
min      -1.338556    -5.336289    -2.909294     0.000000     0.000000
25%      -0.339998    -0.041461    -0.020983     2.265000     0.000000
50%      -0.025647     0.033751     0.021264    12.990000     0.000000
75%       0.288406     0.205736     0.087058    54.990000     0.000000
max       3.463246     3.852046     4.157934  7712.430000     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])))
```

```
0.0005037783375314861
Fraud Cases: 2
Valid Transactions: 3970
```

```
print("Amount details of the fraudulent transaction")
fraud.Amount.describe()
```

```
Amount details of the fraudulent transaction
```

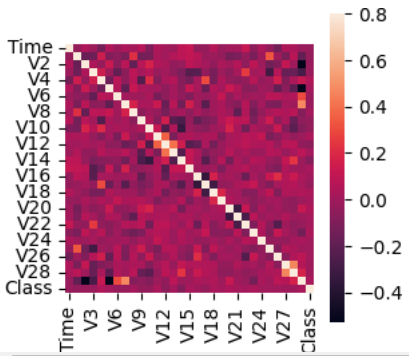
	Amount
count	2.000000
mean	264.500000
std	374.059487
min	0.000000
25%	132.250000
50%	264.500000
75%	396.750000
max	529.000000

```
print("details of valid transaction")
valid.Amount.describe()
```

```
details of valid transaction
```

	Amount
count	3970.000000
mean	64.899597
std	213.612570
min	0.000000
25%	2.270000
50%	12.990000
75%	54.990000
max	7712.430000

```
corrmat = data.corr()
fig = plt.figure(figsize = (3, 3))
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
```

```
(3973, 30)
(3973,)
```

```
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.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
import seaborn as sns
print(data['Class'].isnull().sum())
data = data.dropna(subset=['Class'])
X = data.drop(['Class'], axis=1)
Y = data["Class"]
```

```
1
```

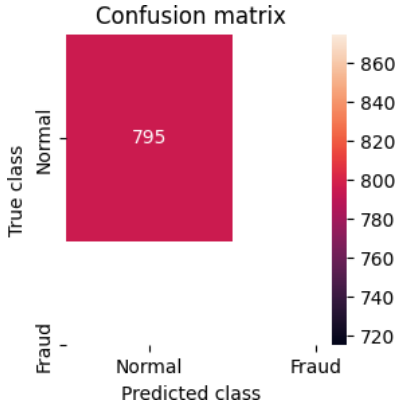
```
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score, matthews_corrcoef
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
print(data['Class'].isnull().sum())
data = data.dropna(subset=['Class'])
X = data.drop(['Class'], axis=1)
Y = data["Class"]
xTrain, xTest, yTrain, yTest = train_test_split(X, Y, test_size=0.2, random_state=42)
rfc = RandomForestClassifier()
rfc.fit(xTrain, yTrain)
yPred = rfc.predict(xTest)
print("The model used is Random Forest classifier")
acc = accuracy_score(yTest, yPred)
print("The accuracy is {}".format(acc))
prec = precision_score(yTest, yPred, zero_division=1)
print("The precision is {}".format(prec))
rec = recall_score(yTest, yPred, zero_division=1)
print("The recall is {}".format(rec))
f1 = f1_score(yTest, yPred, zero_division=1)
print("The F1-Score is {}".format(f1))
MCC = matthews_corrcoef(yTest, yPred)
print("The Matthews correlation coefficient is {}".format(MCC))
print(confusion_matrix(yTest, yPred))
print(classification_report(yTest, yPred))
```

```
0
The model used is Random Forest classifier
The accuracy is 1.0
The precision is 1.0
The recall is 1.0
The F1-Score is 1.0
The Matthews correlation coefficient is 0.0
[[795]]
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       795

    accuracy                           1.00       795
   macro avg       1.00      1.00      1.00       795
weighted avg       1.00      1.00      1.00       795
```

```
LABELS = ['Normal', 'Fraud']
conf_matrix = confusion_matrix(yTest, yPred)
plt.figure(figsize =(3, 3))
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()
```


Confusion matrix
795
True class
Normal
Fraud
Normal
Fraud
Predicted class
860
840
820
800
780
760
740
720