Credit Card Fraud Detection Using Python.

In [1]:
```python
import warnings
warnings.filterwarnings('ignore')

# importing libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, cross_val_score, KFold, GridSearchCV, RandomizedSearchCV
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import GradientBoostingClassifier
```

Fetching the dataset

In [2]:
```python
import os
working_directory = os.getcwd()
print (working_directory)
```

```
/Users/ishu
```

In [8]:
```python
path = working_directory + '/Desktop/CODSOFT/creditcard.csv'
data = pd.read_csv (path)
data.head(15)
```

Out[8]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	69.99	0
5	2.0	-0.425966	0.960523	1.141109	-0.168252	0.420987	-0.029728	0.476201	0.260314	-0.568671	...	-0.208254	-0.559825	-0.026398	-0.371427	-0.232794	0.105915	0.253844	0.081080	3.67	0
6	4.0	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.005159	0.081213	0.464960	...	-0.167716	-0.270710	-0.154104	-0.780055	0.750137	-0.257237	0.034507	0.005168	4.99	0
7	7.0	-0.644269	1.417964	1.074380	-0.492199	0.948934	0.428118	1.120631	-3.807864	0.615375	...	1.943465	-1.015455	0.057504	-0.649709	-0.415267	-0.051634	-1.206921	-1.085339	40.80	0
8	7.0	-0.894286	0.286157	-0.113192	-0.271526	2.669599	3.721818	0.370145	0.851084	-0.392048	...	-0.073425	-0.268092	-0.204233	1.011592	0.373205	-0.384157	0.011747	0.142404	93.20	0
9	9.0	-0.338262	1.119593	1.044367	-0.222187	0.499361	-0.246761	0.651583	0.069539	-0.736727	...	-0.246914	-0.633753	-0.120794	-0.385050	-0.069733	0.094199	0.246219	0.083076	3.68	0
10	10.0	1.449044	-1.176339	0.913860	-1.375667	-1.971383	-0.629152	-1.423236	0.048456	-1.720408	...	-0.009302	0.313894	0.027740	0.500512	0.251367	-0.129478	0.042850	0.016253	7.80	0
11	10.0	0.384978	0.616109	-0.874300	-0.094019	2.924584	3.317027	0.470455	0.538247	-0.558895	...	0.049924	0.238422	0.009130	0.996710	-0.767315	-0.492208	0.042472	-0.054337	9.99	0
12	10.0	1.249999	-1.221637	0.383930	-1.234899	-1.485419	-0.753230	-0.689405	-0.227487	-2.094011	...	-0.231809	-0.483285	0.084668	0.392831	0.161135	-0.354990	0.026416	0.042422	121.50	0
13	11.0	1.069374	0.287722	0.828613	2.712520	-0.178398	0.337544	-0.096717	0.115982	-0.221083	...	-0.036876	0.074412	-0.071407	0.104744	0.548265	0.104094	0.021491	0.021293	27.50	0
14	12.0	-2.791855	-0.327771	1.641750	1.767473	-0.136588	0.807596	-0.422911	-1.907107	0.755713	...	1.151663	0.222182	1.020586	0.028317	-0.232746	-0.235557	-0.164778	-0.030154	58.80	0

15 rows × 31 columns

In [10]:
```python
data.shape
```

Out[10]:
```
(284807, 31)
```

Information of the data

In [11]:
```python
data.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
```

In [12]:
```python
data.dtypes
```

Out[12]:
```
Time      float64
V1        float64
V2        float64
V3        float64
V4        float64
V5        float64
V6        float64
V7        float64
V8        float64
V9        float64
V10       float64
V11       float64
V12       float64
V13       float64
V14       float64
V15       float64
V16       float64
V17       float64
V18       float64
V19       float64
V20       float64
V21       float64
V22       float64
V23       float64
V24       float64
V25       float64
V26       float64
V27       float64
V28       float64
Amount    float64
Class       int64
dtype: object
```

Checking for Null Values

In [13]:
```python
data.isna().sum().sort_values()
```

Out[13]:
```
Time      0
V28       0
V27       0
V26       0
V25       0
V24       0
V23       0
V22       0
V21       0
V20       0
V19       0
V18       0
V17       0
V16       0
Amount    0
V15       0
V13       0
V12       0
V11       0
V10       0
V9        0
V8        0
V7        0
V6        0
V5        0
V4        0
V3        0
V2        0
V1        0
V14       0
Class     0
dtype: int64
```

In [15]:
```python
# description of the data
data.describe()
```

Out[15]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	...	2.848070e+05	2.848070e+05	2.848070e+05	2.848
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.604066e-16	1.487313e-15	-5.556467e-16	1.213481e-16	-2.406331e-15	...	1.654067e-16	-3.568593e-16	2.578648e-16	4.473
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.098632e+00	...	7.345240e-01	7.257016e-01	6.244603e-01	6.056
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.343407e+01	...	-3.483038e+01	-1.093314e+01	-4.480774e+01	-2.836
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.430976e-01	...	-2.283949e-01	-5.423504e-01	-1.618463e-01	-3.545
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5.142873e-02	...	-2.945017e-02	6.781943e-03	-1.119293e-02	4.097
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01	5.971390e-01	...	1.863772e-01	5.285536e-01	1.476421e-01	4.395
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01	1.559499e+01	...	2.720284e+01	1.050309e+01	2.252841e+01	4.584

8 rows × 31 columns

In [43]:
```python
data['Class'].value_counts()
```

Out[43]:
```
0    284315
1       492
Name: Class, dtype: int64
```

In [32]:
```python
print("\n-----------fraud and valid Transaction details-----------")
f = data[data['Class'] == 1]
v = data[data['Class'] == 0]
#Print counts
print('\n 🔹There are {} fraud transactions'.format(f.shape[0]))
print(' 🔹There are {} valid transcations'.format(v.shape[0]))
```

```
-----------fraud and valid Transaction details-----------

 🔹There are 492 fraud transactions
 🔹There are 284315 valid transcations
```

In [21]:
```python
#Summary statistics for fraud transcaction
f.Amount.describe()
```

Out[21]:
```
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
```

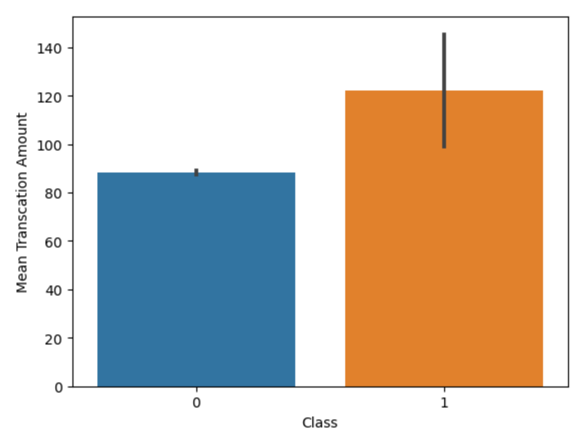
In [22]:
```python
#Summary statistics for valid transcations
v.Amount.describe()
```

Out[22]:
```
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
```

Data visualisation of amount

In [24]:
```python
sns.barplot(data=data,x='Class', y='Amount')
plt.ylabel('Mean Transcation Amount')
```

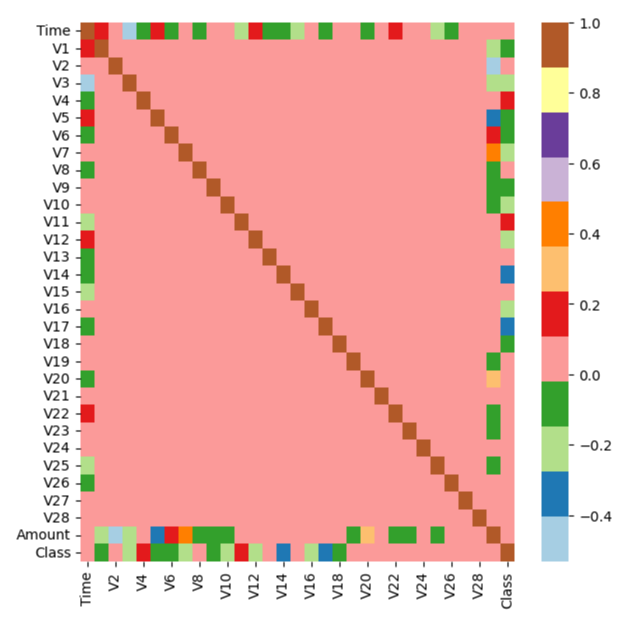
Out[24]:
```
Text(0, 0.5, 'Mean Transcation Amount')
```



In [31]:
```python
#Correlation Matrix of the DATA

corr = data.corr()
fig = plt.figure(figsize = (7, 7))
sns.heatmap(corr, cmap='Paired')
```

Out[31]:
```
<Axes: >
```



Building the Model

In [45]:
```python
X=data.drop(columns="Class",axis=1)
Y=data['Class']
print(X)
print(Y)
```

```
            Time         V1         V2        V3        V4        V5  \
0            0.0  -1.359807  -0.072781  2.536347  1.378155 -0.338321   
1            0.0   1.191857   0.266151  0.166480  0.448154  0.060018   
2            1.0  -1.358354  -1.340163  1.773209  0.379780 -0.503198   
3            1.0  -0.966272  -0.185226  1.792993 -0.863291 -0.010309   
4            2.0  -1.158233   0.877737  1.548718  0.403034 -0.407193   
...          ...        ...        ...       ...       ...       ...   
284802  172786.0 -11.881118  10.071785 -9.834783 -2.066656 -5.364473   
284803  172787.0  -0.732789  -0.055080  2.035030 -0.738589  0.868229   
284804  172788.0   1.919565  -0.301254 -3.249640 -0.557828  2.630515   
284805  172788.0  -0.240440   0.530483  0.702510  0.689799 -0.377961   
284806  172792.0  -0.533413  -0.189733  0.703337 -0.506271 -0.012546   

              V6        V7        V8        V9  ...       V20       V21  \
0       0.462388  0.239599  0.098698  0.363787  ...  0.251412 -0.018307   
1      -0.082361 -0.078803  0.085102 -0.255425  ... -0.069083 -0.225775   
2       1.800499  0.791461  0.247676 -1.514654  ...  0.524980  0.247998   
3       1.247203  0.237609  0.377436 -1.387024  ... -0.208038 -0.108300   
4       0.095921  0.592941 -0.270533  0.817739  ...  0.408542 -0.009431   
...          ...       ...       ...       ...  ...       ...       ...   
284802 -2.606837 -4.918215  7.305334  1.914428  ...  1.475829  0.213454   
284803  1.058415  0.024330  0.294869  0.584800  ...  0.059616  0.214205   
284804  3.031260 -0.296827  0.708417  0.432454  ...  0.001396  0.232045   
284805  0.623708 -0.686180  0.679145  0.392087  ...  0.127434  0.265245   
284806 -0.649617  1.577006 -0.414650  0.486180  ...  0.382948  0.261057   

             V22       V23       V24       V25       V26       V27       V28  \
0       0.277838 -0.110474  0.066928  0.128539 -0.189115  0.133558 -0.021053   
1      -0.638672  0.101288 -0.339846  0.167170  0.125895 -0.008983  0.014724   
2       0.771679  0.909412 -0.689281 -0.327642 -0.139097 -0.055353 -0.059752   
3       0.005274 -0.190321 -1.175575  0.647376 -0.221929  0.062723  0.061458   
4       0.798278 -0.137458  0.141267 -0.206010  0.502292  0.219422  0.215153   
...          ...       ...       ...       ...       ...       ...       ...   
284802  0.111864  1.014480 -0.509348  1.436807  0.250034  0.943651  0.823731   
284803  0.924384  0.012463 -1.016226 -0.606624 -0.395255  0.068472 -0.053527   
284804  0.578229 -0.037501  0.640134  0.265745 -0.087371  0.004455 -0.026561   
284805  0.800049 -0.163298  0.123205 -0.569159  0.546668  0.108821  0.104533   
284806  0.643078  0.376777  0.008797 -0.473649 -0.818267 -0.002415  0.013649   

          Amount  
0       0.244964  
1      -0.342475  
2       1.160686  
3       0.140534  
4      -0.073403  
...          ...  
284802 -0.350151  
284803 -0.254117  
284804 -0.081839  
284805 -0.313249  
284806  0.514355  

[284807 rows x 30 columns]
0         0
1         0
2         0
3         0
4         0
         ..
284802    0
284803    0
284804    0
284805    0
284806    0
Name: Class, Length: 284807, dtype: int64
```

In [33]:
```python
scaler=StandardScaler()
#Scale data
data['Amount']=scaler.fit_transform(data[['Amount']])
```

Defining x and y variables

In [35]:
```python
X = data.drop('Class', axis=1).values
y= data['Class'].values
```

In [36]:
```python
# Data splitting
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2, random_state=50)
```

In [38]:
```python
models ={'KNN': KNeighborsClassifier(),
         'LogReg': LogisticRegression(),
         'dtree': DecisionTreeClassifier(),
         'rforest': RandomForestClassifier(),
         'GB': GradientBoostingClassifier()
}
#Name and Score list
names=[]
scores=[]
#Loop over name and model
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    score = accuracy_score(y_pred, y_test)
    names.append(name)
    scores.append(score)
```

In [39]:
```python
#Create a DataFrame to Compare Scores

results = pd.DataFrame({'model': names, 'score': scores})
#Sort Results

results.sort_values(by='score', ascending=False)
```

Out[39]:

	model	score
3	rforest	0.999596
2	dtree	0.999368
1	LogReg	0.999034
4	GB	0.998947
0	KNN	0.998560

In [46]:
```python
#Define
rforest = RandomForestClassifier()
#Fit model
rforest.fit(X_train,y_train)
#predict
y_pred = rforest.predict(X_test)
#score
score = accuracy_score(y_pred, y_test)
#print
print(score)
```

```
0.9995611109160493
```

In [47]:
```python
#Create report
report=classification_report(y_test,y_pred)
#Print report
print(report)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56874
           1       0.94      0.76      0.84        88

    accuracy                           1.00     56962
   macro avg       0.97      0.88      0.92     56962
weighted avg       1.00      1.00      1.00     56962
```

In [ ]: