Credit Card Fraud Detection

April 27, 2023

[3]:
```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

[4]:
```python
#loading the data
creditdata=pd.read_csv("creditcard.csv")
```

[5]:
```python
#Viweing the whole data
creditdata
```

[5]:
```
            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  ...       V21       V22  \
0       0.462388  0.239599  0.098698  0.363787  ... -0.018307  0.277838
1      -0.082361 -0.078803  0.085102 -0.255425  ... -0.225775 -0.638672
2       1.800499  0.791461  0.247676 -1.514654  ...  0.247998  0.771679
3       1.247203  0.237609  0.377436 -1.387024  ... -0.108300  0.005274
4       0.095921  0.592941 -0.270533  0.817739  ... -0.009431  0.798278
...          ...       ...       ...       ...  ...       ...       ...
284802 -2.606837 -4.918215  7.305334  1.914428  ...  0.213454  0.111864
284803  1.058415  0.024330  0.294869  0.584800  ...  0.214205  0.924384
284804  3.031260 -0.296827  0.708417  0.432454  ...  0.232045  0.578229
284805  0.623708 -0.686180  0.679145  0.392087  ...  0.265245  0.800049
284806 -0.649617  1.577006 -0.414650  0.486180  ...  0.261057  0.643078
```

```
             V23       V24       V25       V26       V27       V28  Amount  \
0      -0.110474  0.066928  0.128539 -0.189115  0.133558 -0.021053  149.62
1       0.101288 -0.339846  0.167170  0.125895 -0.008983  0.014724    2.69
2       0.909412 -0.689281 -0.327642 -0.139097 -0.055353 -0.059752  378.66
3      -0.190321 -1.175575  0.647376 -0.221929  0.062723  0.061458  123.50
4      -0.137458  0.141267 -0.206010  0.502292  0.219422  0.215153   69.99
...          ...       ...       ...       ...       ...       ...     ...
284802  1.014480 -0.509348  1.436807  0.250034  0.943651  0.823731    0.77
284803  0.012463 -1.016226 -0.606624 -0.395255  0.068472 -0.053527   24.79
284804 -0.037501  0.640134  0.265745 -0.087371  0.004455 -0.026561   67.88
284805 -0.163298  0.123205 -0.569159  0.546668  0.108821  0.104533   10.00
284806  0.376777  0.008797 -0.473649 -0.818267 -0.002415  0.013649  217.00

        Class
0           0
1           0
2           0
3           0
4           0
...       ...
284802      0
284803      0
284804      0
284805      0
284806      0

[284807 rows x 31 columns]
```

[6]:
```
#printing 1st 5 rows
creditdata.head()
```

[6]:
```
   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

         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

        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 rows x 31 columns]
```

[7]:
```python
#the class 0-> Normal/Legit Tranaction
#the class 1->Fraud Transaction
```

[8]:
```python
#last 5 data of the dataset
creditdata.tail()
```

[8]:
```
            Time         V1         V2        V3        V4        V5  \
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  ...       V21       V22  \
284802 -2.606837 -4.918215  7.305334  1.914428  ...  0.213454  0.111864
284803  1.058415  0.024330  0.294869  0.584800  ...  0.214205  0.924384
284804  3.031260 -0.296827  0.708417  0.432454  ...  0.232045  0.578229
284805  0.623708 -0.686180  0.679145  0.392087  ...  0.265245  0.800049
284806 -0.649617  1.577006 -0.414650  0.486180  ...  0.261057  0.643078

             V23       V24       V25       V26       V27       V28  Amount  \
284802  1.014480 -0.509348  1.436807  0.250034  0.943651  0.823731    0.77
284803  0.012463 -1.016226 -0.606624 -0.395255  0.068472 -0.053527   24.79
284804 -0.037501  0.640134  0.265745 -0.087371  0.004455 -0.026561   67.88
284805 -0.163298  0.123205 -0.569159  0.546668  0.108821  0.104533   10.00
284806  0.376777  0.008797 -0.473649 -0.818267 -0.002415  0.013649  217.00

        Class
284802      0
284803      0
284804      0
284805      0
284806      0

[5 rows x 31 columns]
```

[9]:
```python
#information about the data-set
creditdata.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
```

[10]:
```
#checking the no. of missing values in each column
creditdata.isnull().sum()
```

[10]:
```
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
```

[11]:

```
#legit transactions and fraud trnsactions
creditdata['Class'].value_counts()
```

[11]:

```
0    284315
1       492
Name: Class, dtype: int64
```

[12]:

```
#legit Transaction represented by 0 where as
#fraud Transaction represeneted by 1
#Highly unbalanced data-set
```

[13]:

```
#analysis of data
#storing the legit data
legit=creditdata[creditdata.Class==0]
fraud=creditdata[creditdata.Class==1]
```

[14]:

```
print(legit.shape)
print(fraud.shape)
```

```
(284315, 31)
(492, 31)
```

[15]:
```
#stats measure of legit trans.Data
legit.Amount.describe()
```

[15]:
```
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
```

[16]:
```
#stats measure of Fraud trans.Data
fraud.Amount.describe()
```

[16]:
```
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
```

[17]:
```
#comparing the values for both type f Transactions
creditdata.groupby('Class').mean()
```

[17]:
```
               Time        V1        V2        V3        V4        V5  \
Class
0      94838.202258  0.008258 -0.006271  0.012171 -0.007860  0.005453
1      80746.806911 -4.771948  3.623778 -7.033281  4.542029 -3.151225

             V6        V7        V8        V9  ...       V20       V21  \
Class                                          ...
0      0.002419  0.009637 -0.000987  0.004467  ... -0.000644 -0.001235
1     -1.397737 -5.568731  0.570636 -2.581123  ...  0.372319  0.713588

            V22       V23       V24       V25       V26       V27       V28  \
Class
0     -0.000024  0.000070  0.000182 -0.000072 -0.000089 -0.000295 -0.000131
1      0.014049 -0.040308 -0.105130  0.041449  0.051648  0.170575  0.075667

           Amount
Class
0       88.291022
```

```
1      122.211321

[2 rows x 30 columns]
```

[18]:
```
#Dealing with Unbalanced Data[Under-Sampling]
#Building a sample dataset from orginal dataset
#Containg Legit and Fraud Trans.
```

[19]:
```
#Fraud Trans - 492
#We will be taking randomly 492 Legit Transaction and
#then Join it with 492 Fraud Transaction to create a sample dataset.
```

[20]:
```
legit_sample=legit.sample(n=492)
```

[21]:
```
#adding 2 data frames (legit sample+fraud sample) [492+492]
new_dataset = pd.concat([legit_sample,fraud],axis =0)
```

[31]:
```
#partially viewing the summation of dataset(1st 5)
new_dataset.head()
```

[31]:
```
            Time        V1        V2        V3        V4        V5        V6  \
200774  133569.0 -1.140835  0.892298  0.712443 -0.836903  0.056951  0.015829
117125   74549.0 -0.505129  0.932076  1.370205  0.127153  0.057924 -0.718603
209001  137363.0 -1.136466  0.993651 -0.227320 -1.084306  1.752104  1.879297
256070  157526.0  2.047938  0.143272 -1.703821  0.757984  0.164830 -1.159670
62189    50170.0 -1.123386 -3.714546 -1.130166  1.647322 -1.595505 -0.627075

              V7        V8        V9  ...       V21       V22       V23  \
200774 -0.033031  0.558319  0.466986  ...  0.147437  0.669019 -0.269699
117125  0.658266  0.050057 -0.407778  ... -0.195017 -0.530140  0.128338
209001  0.520433  0.777343 -0.561185  ... -0.032542 -0.071493 -0.015945
256070  0.106794 -0.293809  0.795575  ... -0.040658  0.225180  0.018667
62189   1.809276 -0.581791 -0.142721  ...  0.751071 -0.614360 -1.159144

             V24       V25       V26       V27       V28   Amount  Class
200774 -0.939491  0.169019  0.705460  0.342493  0.167799    11.99      0
117125  0.342867 -0.267032  0.080999  0.250565  0.101375    26.99      0
209001 -1.269778 -0.268364  0.394606 -0.277641  0.062518    19.91      0
256070 -0.171552  0.120334  0.645225 -0.038508 -0.031551     1.69      0
62189   0.484825  0.076043  0.265392 -0.275332  0.225451  1244.42      0

[5 rows x 31 columns]
```

[32]:
```
new_dataset['Class'].value_counts()
```

[32]:
```
0    492
1    492
```

```
Name: Class, dtype: int64
```

[33]:
```
new_dataset.groupby('Class').mean()
```

[33]:
```
               Time        V1        V2        V3        V4        V5  \
Class
0      96198.282520  0.051937  0.042002 -0.018694  0.038479 -0.051409
1      80746.806911 -4.771948  3.623778 -7.033281  4.542029 -3.151225

             V6        V7        V8        V9  ...       V20       V21  \
Class                                          ...
0      0.051192  0.077361 -0.002449 -0.021249  ... -0.031350  0.030389
1     -1.397737 -5.568731  0.570636 -2.581123  ...  0.372319  0.713588

            V22       V23       V24       V25       V26       V27       V28  \
Class
0     -0.038629 -0.006503 -0.020373 -0.003605  0.010490 -0.004008  0.006831
1      0.014049 -0.040308 -0.105130  0.041449  0.051648  0.170575  0.075667

           Amount
Class
0      101.453923
1      122.211321

[2 rows x 30 columns]
```

[34]:
```
X = new_dataset.drop(columns='Class', axis=1)
Y = new_dataset['Class']
```

[35]:
```
print(X)
```

```
            Time        V1        V2        V3        V4        V5        V6  \
200774  133569.0 -1.140835  0.892298  0.712443 -0.836903  0.056951  0.015829
117125   74549.0 -0.505129  0.932076  1.370205  0.127153  0.057924 -0.718603
209001  137363.0 -1.136466  0.993651 -0.227320 -1.084306  1.752104  1.879297
256070  157526.0  2.047938  0.143272 -1.703821  0.757984  0.164830 -1.159670
62189    50170.0 -1.123386 -3.714546 -1.130166  1.647322 -1.595505 -0.627075
...          ...       ...       ...       ...       ...       ...       ...
279863  169142.0 -1.927883  1.125653 -4.518331  1.749293 -1.566487 -2.010494
280143  169347.0  1.378559  1.289381 -5.004247  1.411850  0.442581 -1.326536
280149  169351.0 -0.676143  1.126366 -2.213700  0.468308 -1.120541 -0.003346
281144  169966.0 -3.113832  0.585864 -5.399730  1.817092 -0.840618 -2.943548
281674  170348.0  1.991976  0.158476 -2.583441  0.408670  1.151147 -0.096695

              V7        V8        V9  ...       V20       V21       V22  \
200774 -0.033031  0.558319  0.466986  ...  0.089371  0.147437  0.669019
117125  0.658266  0.050057 -0.407778  ...  0.138755 -0.195017 -0.530140
209001  0.520433  0.777343 -0.561185  ... -0.274478 -0.032542 -0.071493
```

```
256070  0.106794 -0.293809  0.795575  ... -0.168508 -0.040658  0.225180
62189   1.809276 -0.581791 -0.142721  ...  2.423336  0.751071 -0.614360
...          ...       ...       ...  ...       ...       ...       ...
279863 -0.882850  0.697211 -2.064945  ...  1.252967  0.778584 -0.319189
280143 -1.413170  0.248525 -1.127396  ...  0.226138  0.370612  0.028234
280149 -2.234739  1.210158 -0.652250  ...  0.247968  0.751826  0.834108
281144 -2.208002  1.058733 -1.632333  ...  0.306271  0.583276 -0.269209
281674  0.223050 -0.068384  0.577829  ... -0.017652 -0.164350 -0.295135

             V23       V24       V25       V26       V27       V28   Amount
200774 -0.269699 -0.939491  0.169019  0.705460  0.342493  0.167799    11.99
117125  0.128338  0.342867 -0.267032  0.080999  0.250565  0.101375    26.99
209001 -0.015945 -1.269778 -0.268364  0.394606 -0.277641  0.062518    19.91
256070  0.018667 -0.171552  0.120334  0.645225 -0.038508 -0.031551     1.69
62189  -1.159144  0.484825  0.076043  0.265392 -0.275332  0.225451  1244.42
...          ...       ...       ...       ...       ...       ...      ...
279863  0.639419 -0.294885  0.537503  0.788395  0.292680  0.147968   390.00
280143 -0.145640 -0.081049  0.521875  0.739467  0.389152  0.186637     0.76
280149  0.190944  0.032070 -0.739695  0.471111  0.385107  0.194361    77.89
281144 -0.456108 -0.183659 -0.328168  0.606116  0.884876 -0.253700   245.00
281674 -0.072173 -0.450261  0.313267 -0.289617  0.002988 -0.015309    42.53

[984 rows x 30 columns]
```

[36]:
```
print(Y)
```

```
200774    0
117125    0
209001    0
256070    0
62189     0
         ..
279863    1
280143    1
280149    1
281144    1
281674    1
Name: Class, Length: 984, dtype: int64
```

[37]:
```
#split the data to train and Test
Xtrain,Xtest,Ytrain,Ytest=train_test_split(X,Y,test_size=0.
↪2,stratify=Y,random_state=2)
```

[38]:
```
print(X.shape,Xtrain.shape,Xtest.shape)
```

```
(984, 30) (787, 30) (197, 30)
```

[39]:
```
#training the model using logestic Regression Model
model = LogisticRegression()
```

[40]:
```
#training the model using logestic Regression Model using training data
model.fit(Xtrain,Ytrain)
```

```
C:\Users\KIIT\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[40]:
```
LogisticRegression()
```

[41]:
```
#Finding the performance
#accuracy on traing data
Xtrain_prediction = model.predict(Xtrain)
training_data_accuracy = accuracy_score(Xtrain_prediction, Ytrain)
```

[42]:
```
print("Accuracy on training data : ",training_data_accuracy)
```

```
Accuracy on training data :  0.9466327827191868
```

[43]:
```
# Finding the performance on test data
Xtest_prediction = model.predict(Xtest)
test_data_accuracy = accuracy_score(Xtest_prediction, Ytest)
```

[44]:
```
print('Accuracy score on Test Data : ', test_data_accuracy)
```

```
Accuracy score on Test Data :  0.9137055837563451
```

[45]:
```
#Conclusion
#Here we can see the Accuracy Score of Trained Data = 94%
#and the Accuracy Score of Test Data = 91%
#which is very Smilar to Accuracy Score of Trained Data
```

[46]:
```
model.predict(Xtest)
```

[46]:
```
array([1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0,
       0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0,
       1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0,
       0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0,
       0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1,
       0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0],
```

```
dtype=int64)
```

[47]:
```
len(Y)
```

[47]:
```
984
```

[ ]: