credit_card_Fraud_Detection

April 19, 2025

[2]:
```
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.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

[3]:
```
credit_data=pd.read_csv('creditcard.csv')
```

[4]:
```
credit_data.head()
```

[4]:
```
   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]
```

[5]:
```
credit_data.tail()
```

[5]:
```
            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]
```

[6]:
```
credit_data.shape
```

[6]:
```
(284807, 31)
```

[7]:
```
credit_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
```

[8]:
```
credit_data.isnull().sum()
```

[8]:
```
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
```

[9]:
```
credit_data['Class'].value_counts()
```

[9]:
```
Class
0    284315
1       492
Name: count, dtype: int64
```

This Dataset is highly Unbalanced 0——>normal transaction 1——>fraudulent transaction

Seperating the data for analysis

[13]:
```
legit=credit_data[credit_data.Class==0]    #these are pandas series datatypes
```

[12]:
```
fraud=credit_data[credit_data.Class==1]
```

[14]:
```
print(legit.shape,fraud.shape)
```

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

Stastical measure of the data

[15]:
```
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]:
```
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
```

[19]:
```
credit_data.groupby('Class').mean()
```

[19]:

Class	Time	V1	V2	V3	V4	V5	…	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount
0	94838.202258	0.008258	-0.006271	0.012171	-0.007860	0.005453	…	-0.000644	-0.001235	-0.000024	0.000070	0.000182	-0.000072	-0.000089	-0.000295	-0.000131	88.291022
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	…	0.372319	0.713588	0.014049	-0.040308	-0.105130	0.041449	0.051648	0.170575	0.075667	122.211321

[2 rows x 30 columns]

Dealing with the Unbalanced Data

Under Sampling

Build a sample dataset containing similar distribution of Normal and Fraduluent Transactions

Number of Fraduluent Transactions is 492

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

Concatinating the two dataframes

[21]:
```
new_dataset=pd.concat([legit_Sample,fraud],axis=0)
```

[22]:
```
new_dataset.head()
```

[22]:
```
            Time        V1        V2        V3        V4        V5        V6  \
147157   88196.0  1.840186 -0.071982 -1.794965  1.264588  0.601569 -0.593841
186117  126962.0  2.169915 -1.157829 -0.708791 -1.058263 -0.843017  0.170768
83666    59970.0 -0.908381  0.985515 -0.002330 -0.712693  2.398197  3.547628
209244  137474.0 -0.336665  0.538751 -0.817160  0.020986  1.849731 -1.456884
103018   68461.0  1.188148  0.090220  0.585083  0.571258 -0.642525 -0.812744

                V7        V8        V9  …       V21       V22       V23  \
147157  0.683593 -0.301936 -0.119492  …  0.138677  0.419994 -0.137862
186117 -1.267143  0.119161  0.169604  …  0.393079  1.221556 -0.056083
83666  -0.054255  0.708240 -0.482761  … -0.140802 -0.819044 -0.083311
209244  2.218136 -0.639788 -0.673987  …  0.482810  1.349564 -0.129602
103018 -0.113174 -0.001407 -0.037428  … -0.202528 -0.702001  0.158177

                V24       V25       V26       V27       V28  Amount  Class
147157 -0.372130  0.456155 -0.497529 -0.032253 -0.059247   96.11      0
186117 -0.989350 -0.005786  0.054305  0.012392 -0.067498   17.09      0
83666   1.000594 -0.236621  0.165849 -0.618289  0.155527    1.29      0
209244  1.084858  0.076848 -0.527480 -0.045435 -0.000034   85.43      0
103018  0.495991  0.114510  0.067949 -0.042260  0.009442    8.99      0

[5 rows x 31 columns]
```

[23]:
```
new_dataset.tail()
```

[23]:
```
            Time        V1        V2        V3        V4        V5        V6  \
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  …       V21       V22       V23  \
279863 -0.882850  0.697211 -2.064945  …  0.778584 -0.319189  0.639419
280143 -1.413170  0.248525 -1.127396  …  0.370612  0.028234 -0.145640
280149 -2.234739  1.210158 -0.652250  …  0.751826  0.834108  0.190944
281144 -2.208002  1.058733 -1.632333  …  0.583276 -0.269209 -0.456108
281674  0.223050 -0.068384  0.577829  … -0.164350 -0.295135 -0.072173

                V24       V25       V26       V27       V28  Amount  Class
279863 -0.294885  0.537503  0.788395  0.292680  0.147968  390.00      1
280143 -0.081049  0.521875  0.739467  0.389152  0.186637    0.76      1
280149  0.032070 -0.739695  0.471111  0.385107  0.194361   77.89      1
281144 -0.183659 -0.328168  0.606116  0.884876 -0.253700  245.00      1
281674 -0.450261  0.313267 -0.289617  0.002988 -0.015309   42.53      1

[5 rows x 31 columns]
```

[24]:
```
new_dataset['Class'].value_counts()
```

[24]:
```
Class
0    492
1    492
Name: count, dtype: int64
```

[25]:
```
new_dataset.groupby('Class').mean()
```

[25]:
```
                Time        V1        V2        V3        V4        V5  \
Class
0      96753.717480 -0.095333  0.011770  0.052283 -0.070923 -0.073665
1      80746.806911 -4.771948  3.623778 -7.033281  4.542029 -3.151225

              V6        V7        V8        V9  …       V20       V21  \
Class                                           …
0      0.000134  0.066418  0.037255 -0.028800  …  0.047252  0.040457
1     -1.397737 -5.568731  0.570636 -2.581123  …  0.372319  0.713588

             V22       V23       V24       V25       V26       V27       V28  \
Class
0      0.067179 -0.021314  0.005568  0.012540  0.016787 -0.000930 -0.007599
1      0.014049 -0.040308 -0.105130  0.041449  0.051648  0.170575  0.075667

            Amount
Class
0      102.640793
1      122.211321

[2 rows x 30 columns]
```

[26]:
```
X=new_dataset.drop(['Class'],axis=1)
```

[27]:
```
Y=new_dataset['Class']
```

[28]:
```
print(X)
```

```
            Time        V1        V2        V3        V4        V5        V6  \
147157   88196.0  1.840186 -0.071982 -1.794965  1.264588  0.601569 -0.593841
186117  126962.0  2.169915 -1.157829 -0.708791 -1.058263 -0.843017  0.170768
83666    59970.0 -0.908381  0.985515 -0.002330 -0.712693  2.398197  3.547628
209244  137474.0 -0.336665  0.538751 -0.817160  0.020986  1.849731 -1.456884
103018   68461.0  1.188148  0.090220  0.585083  0.571258 -0.642525 -0.812744
…            …         …         …         …         …         …         …
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  \
147157  0.683593 -0.301936 -0.119492  … -0.070651  0.138677  0.419994
186117 -1.267143  0.119161  0.169604  … -0.000305  0.393079  1.221556
83666  -0.054255  0.708240 -0.482761  …  0.126970 -0.140802 -0.819044
209244  2.218136 -0.639788 -0.673987  … -0.067531  0.482810  1.349564
103018 -0.113174 -0.001407 -0.037428  … -0.155426 -0.202528 -0.702001
…           …         …        …  …         …         …         …
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
147157 -0.137862 -0.372130  0.456155 -0.497529 -0.032253 -0.059247   96.11
186117 -0.056083 -0.989350 -0.005786  0.054305  0.012392 -0.067498   17.09
83666  -0.083311  1.000594 -0.236621  0.165849 -0.618289  0.155527    1.29
209244 -0.129602  1.084858  0.076848 -0.527480 -0.045435 -0.000034   85.43
103018  0.158177  0.495991  0.114510  0.067949 -0.042260  0.009442    8.99
…           …         …         …         …         …         …     …
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]
```

[29]:
```
print(Y)
```

```
147157    0
186117    0
83666     0
209244    0
103018    0
         ..
279863    1
280143    1
280149    1
281144    1
281674    1
Name: Class, Length: 984, dtype: int64
```

[30]:
```
train_x,test_x,train_y,test_y=train_test_split(X,Y,test_size=0.2,random_state=2)
```

[31]:
```
Logistic=LogisticRegression()
```

[32]:
```
Logistic.fit(train_x,train_y)
```

```
C:\Users\indhu\anaconda3\Lib\site-
packages\sklearn\linear_model\_logistic.py:469: 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(
```

[32]:
```
LogisticRegression()
```

[33]:
```
train_x_prediction=Logistic.predict(train_x)
```

[34]:
```
train_x_accuracy=accuracy_score(train_x_prediction,train_y)
```

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

```
0.9428208386277002
```

[36]:
```
test_x_prediction=Logistic.predict(test_x)
```

[37]:
```
accuracy=accuracy_score(test_x_prediction,test_y)
```

[38]:
```
print(accuracy)
```

```
0.9289340101522843
```

[39]:
```
X_new=test_x.iloc[2]
```

[40]:
```
nparray=np.asarray(X_new)
```

[41]:
```
reshaped=nparray.reshape(1,-1)
```

[42]:
```
X_new_df=pd.DataFrame(reshaped,columns=train_x.columns)
```

[43]:
```
predict=Logistic.predict(X_new_df)
```

[44]:
```
print(predict)
```

```
[1]
```

[45]:
```
print(test_y.iloc[2])
```

```
1
```

[ ]: