Fraud Detection

February 18, 2025

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

[4]:
```
credit_card_data = pd.read_csv(r'C:\Users\User\Desktop\creditcard.csv')
```

[5]:
```
credit_card_data.head()
```

[5]:
```
   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]
```

[6]:
```
credit_card_data.tail()
```

[6]:
```
            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]
```

[7]:
```
credit_card_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]:
```
#checking missing values
credit_card_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
```

[11]:
```
#legit and fraudulent trastraction
credit_card_data['Class'].value_counts()
```

[11]:
```
Class
0    284315
1       492
Name: count, dtype: int64
```

0 - Normal transaction ,
1 - Fraudulent transaction

[18]:
```
#seperating data
legit = credit_card_data[credit_card_data.Class==0]
fraud = credit_card_data[credit_card_data.Class==1]
```

[19]:
```
print(legit.shape)
print(fraud.shape)
```

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

[20]:
```
#statistical measures
legit.Amount.describe()
```

[20]:
```
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
```

[21]:
```
fraud.Amount.describe()
```

[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
```

[23]:
```
# compare the values for both  trannsactions
credit_card_data.groupby('Class').mean()
```

[23]:
```
               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]
```

Under sampling

Build a sample dataset

[24]:
```
legit_sample = legit.sample(n=492)
```

[25]:
```
#concatenating two dataframes
new_dataset = pd.concat([legit_sample,fraud], axis=0)
```

[26]:
```
new_dataset.head()
```

[26]:
```
             Time        V1        V2        V3        V4        V5        V6  \
151729    96131.0 -0.372930  1.248888 -0.787021 -1.033541  1.440630 -1.252267
209705   137668.0  0.077335  0.756205 -0.123442 -0.936956  0.909080 -0.227595
82913     59606.0  1.181215 -0.034055  0.194496  0.551489 -0.047601  0.113685
52813     45656.0 -1.171559  1.745159  1.441715 -0.047893 -0.300120 -0.571349
96983     66030.0 -1.296644  0.463238  2.319948 -1.224178 -0.374589  0.781035
```

```
              V7        V8        V9  …       V21       V22       V23  \
151729  1.586265 -0.476941  0.689877  …  0.109581  0.851321 -0.244570
209705  0.852820  0.035400 -0.152126  … -0.282858 -0.690799 -0.064498
82913  -0.031618  0.000099  0.115126  … -0.202921 -0.465568 -0.143721
52813  -0.029771 -2.498441 -0.393213  …  2.098413 -1.220703  0.347408
96983  -0.253823 -0.385727  0.371812  …  0.735516 -0.235528 -0.195494

              V24       V25       V26       V27       V28  Amount  Class
151729 -0.318726 -0.221219  0.052125  0.475266  0.330501   20.00      0
209705 -1.058742 -0.402132  0.185889  0.238838  0.075587    2.67      0
82913  -0.437644  0.553061  0.296367 -0.027554  0.002303   41.33      0
52813   0.680678 -0.140620  0.065747  0.376654  0.150813    2.69      0
96983  -0.246506  0.051560  0.936465 -0.051510  0.021942   44.22      0

[5 rows x 31 columns]
```

[ ]:

[27]:
```
new_dataset.tail()
```

[27]:
```
            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]
```

[28]:
```
new_dataset['Class'].value_counts()
```

[28]:
```
Class
0    492
1    492
Name: count, dtype: int64
```

[29]:
```
new_dataset.groupby('Class').mean()
```

[29]:
```
               Time        V1        V2        V3        V4        V5  \
Class
0      97089.123984  0.027821  0.043579  0.014307 -0.068252  0.056295
1      80746.806911 -4.771948  3.623778 -7.033281  4.542029 -3.151225

              V6        V7        V8        V9  …       V20       V21  \
Class                                           …
0      0.028465  0.007776  0.032129 -0.104161  …  0.010657  0.005072
1     -1.397737 -5.568731  0.570636 -2.581123  …  0.372319  0.713588

             V22       V23      V24       V25       V26       V27       V28  \
Class
0      0.012520  0.030913 -0.00655  0.012172  0.014567 -0.022231 -0.001005
1      0.014049 -0.040308 -0.10513  0.041449  0.051648  0.170575  0.075667

            Amount
Class
0       81.372439
1      122.211321

[2 rows x 30 columns]
```

splitting data

[30]:
```
X = new_dataset.drop(columns='Class' , axis =1)
Y = new_dataset['Class']
```

[32]:
```
print(X)
```

```
            Time        V1        V2        V3        V4        V5        V6  \
151729   96131.0 -0.372930  1.248888 -0.787021 -1.033541  1.440630 -1.252267
209705  137668.0  0.077335  0.756205 -0.123442 -0.936956  0.909080 -0.227595
82913    59606.0  1.181215 -0.034055  0.194496  0.551489 -0.047601  0.113685
52813    45656.0 -1.171559  1.745159  1.441715 -0.047893 -0.300120 -0.571349
96983    66030.0 -1.296644  0.463238  2.319948 -1.224178 -0.374589  0.781035
…            …       …       …       …       …       …       …
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  \
151729  1.586265 -0.476941  0.689877  …  0.107605  0.109581  0.851321
209705  0.852820  0.035400 -0.152126  … -0.031941 -0.282858 -0.690799
82913  -0.031618  0.000099  0.115126  …  0.047802 -0.202921 -0.465568
52813  -0.029771 -2.498441 -0.393213  … -0.328226  2.098413 -1.220703
96983  -0.253823 -0.385727  0.371812  … -0.348094  0.735516 -0.235528
…            …         …         …  …         …         …         …
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
151729 -0.244570 -0.318726 -0.221219  0.052125  0.475266  0.330501   20.00
209705 -0.064498 -1.058742 -0.402132  0.185889  0.238838  0.075587    2.67
82913  -0.143721 -0.437644  0.553061  0.296367 -0.027554  0.002303   41.33
52813   0.347408  0.680678 -0.140620  0.065747  0.376654  0.150813    2.69
96983  -0.195494 -0.246506  0.051560  0.936465 -0.051510  0.021942   44.22
…            …         …         …         …         …         …       …
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]
```

[33]:
```
print(Y)
```

```
151729    0
209705    0
82913     0
52813     0
96983     0
         ..
279863    1
280143    1
280149    1
281144    1
281674    1
Name: Class, Length: 984, dtype: int64
```

Split data into training and testing

[35]:
```
X_train , X_test , Y_train , Y_test = train_test_split(X,Y, test_size = 0.2 ,␣
↪stratify = Y , random_state = 2)
```

[36]:
```
print(X.shape , X_train.shape , X_test.shape)
```

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

Model Training

[38]:
```
model = LogisticRegression()
```

[39]:
```
#Training data
model.fit(X_train,Y_train)
```

```
C:\Users\User\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\linear_model\_logistic.py:465: 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(
```

[39]:
```
LogisticRegression()
```

Model Evaluation

[41]:
```
#Accuracy score
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

[42]:
```
print('Training data accuracy : ' , training_data_accuracy)
```

```
Training data accuracy :  0.9479034307496823
```

[43]:
```
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction , Y_test)
```

[44]:
```
print('Test data accuracy : ' , test_data_accuracy)
```

```
Test data accuracy :  0.9187817258883249
```

Checking Predictions for New data

[55]:
```
new_data = np.array([ 0.00000000e+00, -1.35980713e+00, -7.27811733e-02,  2.
↪53634674e+00,
 1.37815522e+00, -3.38320770e-01,  4.62387778e-01,  2.39598554e-01,
 9.86979013e-02,  3.63786970e-01,  9.07941720e-02, -5.51599533e-01,
-6.17800856e-01, -9.91389847e-01, -3.11169354e-01,  1.46817697e+00,
-4.70400525e-01,  2.07971242e-01,  2.57905802e-02,  4.03992960e-01,
 2.51412098e-01, -1.83067779e-02,  2.77837576e-01, -1.10473910e-01,
```

```
  6.69280749e-02,  1.28539358e-01, -1.89114844e-01,  1.33558377e-01,
 -2.10530535e-02,  1.49620000e+02])

new_data = new_data.reshape(1, -1)
```

[57]:
```
new_data_prediction = model.predict(new_data)
print(new_data_prediction)
```

```
[0]

C:\Users\User\AppData\Local\Programs\Python\Python310\lib\site-
packages\sklearn\utils\validation.py:2739: UserWarning: X does not have valid
feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
```

[ ]: