ads-phase4

November 1, 2023

0.1 Date: 17-10-2023

0.2 Project Title: Credit Card Fraudlent Detection

TEAM MEMBERS:

VIGNESH S (2021506086)

SANJAY M (2021506123)

VIGNESH G (2021506121)

SISHAATH RA KRISHNA (20215060101)

SHREESH SHIVALINGAM (2021506097)

0.3 Importing required libraries

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
```

0.3.1 Set the jupyter notebook to show maximum number of columns

[2]:
```
pd.options.display.max_columns = None
```

0.3.2 Displaying top 5 rows

0.3.3 Loading the datasets

[3]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[14]:
```
ccfd = pd.read_csv('drive/MyDrive/ColabNotebooks/creditcard.csv')
```

[15]:
```
ccfd.head()
```

[15]:
```
   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       V10       V11       V12       V13       V14  \
0  0.098698  0.363787  0.090794 -0.551600 -0.617801 -0.991390 -0.311169
1  0.085102 -0.255425 -0.166974  1.612727  1.065235  0.489095 -0.143772
2  0.247676 -1.514654  0.207643  0.624501  0.066084  0.717293 -0.165946
3  0.377436 -1.387024 -0.054952 -0.226487  0.178228  0.507757 -0.287924
4 -0.270533  0.817739  0.753074 -0.822843  0.538196  1.345852 -1.119670

        V15       V16       V17       V18       V19       V20       V21  \
0  1.468177 -0.470401  0.207971  0.025791  0.403993  0.251412 -0.018307
1  0.635558  0.463917 -0.114805 -0.183361 -0.145783 -0.069083 -0.225775
2  2.345865 -2.890083  1.109969 -0.121359 -2.261857  0.524980  0.247998
3 -0.631418 -1.059647 -0.684093  1.965775 -1.232622 -0.208038 -0.108300
4  0.175121 -0.451449 -0.237033 -0.038195  0.803487  0.408542 -0.009431

        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

   Amount  Class
0  149.62      0
1    2.69      0
2  378.66      0
3  123.50      0
4   69.99      0
```

0.3.4 Displaying bottom 5 rows

[16]:
```
ccfd.tail()
```

[16]:
```
            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       V10       V11       V12  \
```

```
284802 -2.606837 -4.918215  7.305334  1.914428  4.356170 -1.593105  2.711941
284803  1.058415  0.024330  0.294869  0.584800 -0.975926 -0.150189  0.915802
284804  3.031260 -0.296827  0.708417  0.432454 -0.484782  0.411614  0.063119
284805  0.623708 -0.686180  0.679145  0.392087 -0.399126 -1.933849 -0.962886
284806 -0.649617  1.577006 -0.414650  0.486180 -0.915427 -1.040458 -0.031513

             V13       V14       V15       V16       V17       V18       V19  \
284802 -0.689256  4.626942 -0.924459  1.107641  1.991691  0.510632 -0.682920
284803  1.214756 -0.675143  1.164931 -0.711757 -0.025693 -1.221179 -1.545556
284804 -0.183699 -0.510602  1.329284  0.140716  0.313502  0.395652 -0.577252
284805 -1.042082  0.449624  1.962563 -0.608577  0.509928  1.113981  2.897849
284806 -0.188093 -0.084316  0.041333 -0.302620 -0.660377  0.167430 -0.256117

             V20       V21       V22       V23       V24       V25       V26  \
284802  1.475829  0.213454  0.111864  1.014480 -0.509348  1.436807  0.250034
284803  0.059616  0.214205  0.924384  0.012463 -1.016226 -0.606624 -0.395255
284804  0.001396  0.232045  0.578229 -0.037501  0.640134  0.265745 -0.087371
284805  0.127434  0.265245  0.800049 -0.163298  0.123205 -0.569159  0.546668
284806  0.382948  0.261057  0.643078  0.376777  0.008797 -0.473649 -0.818267

             V27       V28  Amount  Class
284802  0.943651  0.823731    0.77      0
284803  0.068472 -0.053527   24.79      0
284804  0.004455 -0.026561   67.88      0
284805  0.108821  0.104533   10.00      0
284806 -0.002415  0.013649  217.00      0
```

0.3.5 Shows number of rows and columns

[17]:
```python
print("Number of rows in given dataset ",ccfd.shape[0])
print("Number of columns in the given dataset ",ccfd.shape[1])
```

```
Number of rows in given dataset  284807
Number of columns in the given dataset  31
```

0.3.6 Getting basis information

[18]:
```python
ccfd.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
```

0.3.7 Checking null values in the given data

[19]:
```
ccfd.isnull().sum()
```

[19]:
```
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
```

0.3.8 Scaling the Amount features, removing the independent columns

[20]:
```
#removing the column name Time, it is unnecessary to our training purposes
ccfd.head(2)
```

[20]:
```
   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

         V8        V9       V10       V11       V12       V13       V14  \
0  0.098698  0.363787  0.090794 -0.551600 -0.617801 -0.991390 -0.311169
1  0.085102 -0.255425 -0.166974  1.612727  1.065235  0.489095 -0.143772

        V15       V16       V17       V18       V19       V20       V21  \
0  1.468177 -0.470401  0.207971  0.025791  0.403993  0.251412 -0.018307
1  0.635558  0.463917 -0.114805 -0.183361 -0.145783 -0.069083 -0.225775

        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

   Amount  Class
0  149.62      0
1    2.69      0
```

[21]:
```
#time features is unnecessary here
ccfd.drop('Time',axis = 1,inplace=True)
```

[22]:
```
ccfd.head(2)
```

[22]:
```
         V1        V2        V3        V4        V5        V6        V7  \
0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599
1  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803

         V8        V9       V10       V11       V12       V13       V14  \
0  0.098698  0.363787  0.090794 -0.551600 -0.617801 -0.991390 -0.311169
1  0.085102 -0.255425 -0.166974  1.612727  1.065235  0.489095 -0.143772

        V15       V16       V17       V18       V19       V20       V21  \
0  1.468177 -0.470401  0.207971  0.025791  0.403993  0.251412 -0.018307
1  0.635558  0.463917 -0.114805 -0.183361 -0.145783 -0.069083 -0.225775

        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

   Amount  Class
0  149.62      0
1    2.69      0
```

0.3.9 Scaling the Amount column data

[23]:
```
from sklearn.preprocessing import StandardScaler
```

[24]:
```
ss = StandardScaler()
```

[25]:
```
ccfd['Amounts'] = ss.fit_transform(pd.DataFrame(ccfd['Amount']))
```

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

[26]:
```
         V1        V2        V3        V4        V5        V6        V7  \
0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599
1  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803
2 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461
3 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609
4 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941

         V8        V9       V10       V11       V12       V13       V14  \
0  0.098698  0.363787  0.090794 -0.551600 -0.617801 -0.991390 -0.311169
1  0.085102 -0.255425 -0.166974  1.612727  1.065235  0.489095 -0.143772
2  0.247676 -1.514654  0.207643  0.624501  0.066084  0.717293 -0.165946
3  0.377436 -1.387024 -0.054952 -0.226487  0.178228  0.507757 -0.287924
```

```
4 -0.270533  0.817739  0.753074 -0.822843  0.538196  1.345852 -1.119670

        V15       V16       V17       V18       V19       V20       V21  \
0  1.468177 -0.470401  0.207971  0.025791  0.403993  0.251412 -0.018307
1  0.635558  0.463917 -0.114805 -0.183361 -0.145783 -0.069083 -0.225775
2  2.345865 -2.890083  1.109969 -0.121359 -2.261857  0.524980  0.247998
3 -0.631418 -1.059647 -0.684093  1.965775 -1.232622 -0.208038 -0.108300
4  0.175121 -0.451449 -0.237033 -0.038195  0.803487  0.408542 -0.009431

        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

   Amount  Class   Amounts
0  149.62      0  0.244964
1    2.69      0 -0.342475
2  378.66      0  1.160686
3  123.50      0  0.140534
4   69.99      0 -0.073403
```

[27]:
```
ccfd.shape
```

[27]: (284807, 31)

[28]:
```
ccfd.drop('Amount',axis=1,inplace=True)
```

[29]:
```
ccfd.shape
```

[29]: (284807, 30)

0.3.10 Dropping the duplicate records

[30]:
```
ccfd.duplicated().any()
```

[30]: True

[31]:
```
ccfd.drop_duplicates(inplace=True)
```

[32]:
```
ccfd.shape
```

[32]: (275663, 30)

[33]:
```
284807 - 275663
```

[33]: 9144

0.3.11 Exploring Class columns

[34]:
```
ccfd['Class'].unique()
```

[34]: array([0, 1])

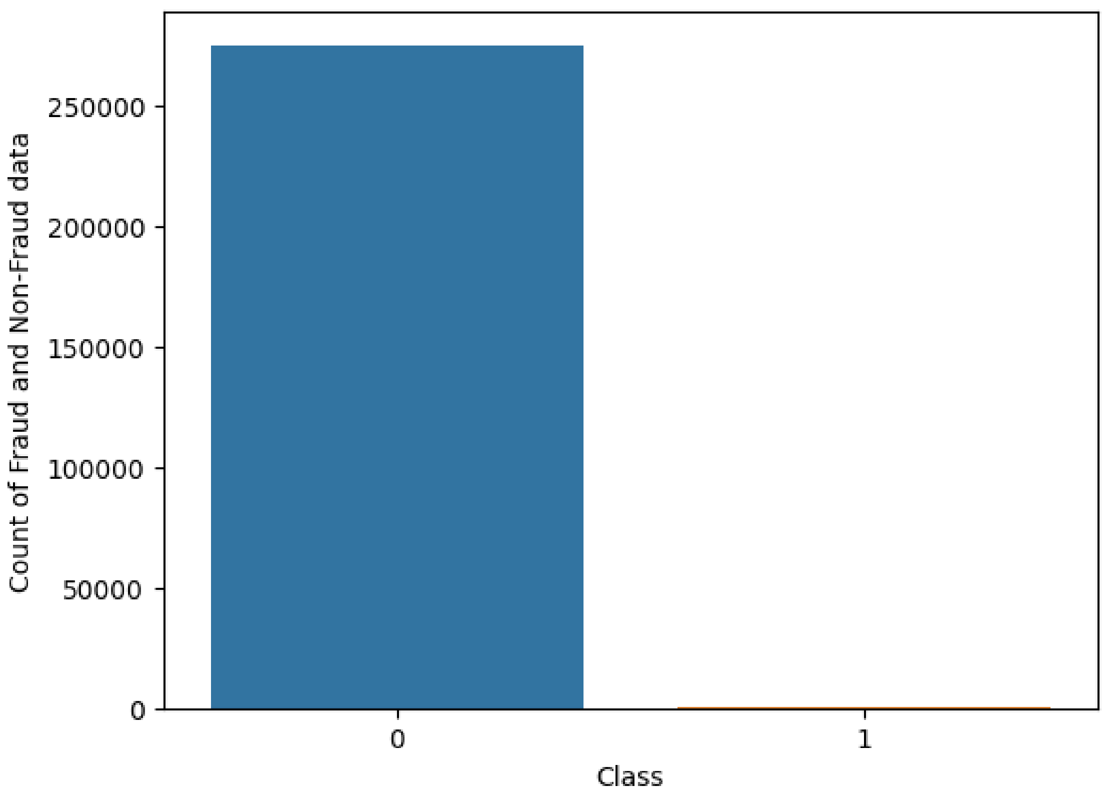
[35]:
```
ccfd['Class'].nunique()
```

[35]: 2

[36]:
```
ccfd['Class'].value_counts()
```

[36]:
```
0    275190
1       473
Name: Class, dtype: int64
```

[37]:
```
#visualizing the distribution of 0 and 1 using seaborn countplot
sns.countplot(ccfd,x = ccfd['Class'])
plt.xlabel('Class')
plt.ylabel('Count of Fraud and Non-Fraud data')
plt.show()
```



From the above information, We can say that our data is high imbalanced, so need to apply oversampling and undersampling technique to train our model

0.3.12 Storing feature matrix in X and response (Target) in vector y

[38]:
```
ccfd.head()
```

[38]:
```
         V1        V2        V3        V4        V5        V6        V7  \
0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599
1  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803
2 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461
3 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609
4 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941

         V8        V9       V10       V11       V12       V13       V14  \
0  0.098698  0.363787  0.090794 -0.551600 -0.617801 -0.991390 -0.311169
1  0.085102 -0.255425 -0.166974  1.612727  1.065235  0.489095 -0.143772
2  0.247676 -1.514654  0.207643  0.624501  0.066084  0.717293 -0.165946
3  0.377436 -1.387024 -0.054952 -0.226487  0.178228  0.507757 -0.287924
4 -0.270533  0.817739  0.753074 -0.822843  0.538196  1.345852 -1.119670

        V15       V16       V17       V18       V19       V20       V21  \
0  1.468177 -0.470401  0.207971  0.025791  0.403993  0.251412 -0.018307
1  0.635558  0.463917 -0.114805 -0.183361 -0.145783 -0.069083 -0.225775
2  2.345865 -2.890083  1.109969 -0.121359 -2.261857  0.524980  0.247998
3 -0.631418 -1.059647 -0.684093  1.965775 -1.232622 -0.208038 -0.108300
4  0.175121 -0.451449 -0.237033 -0.038195  0.803487  0.408542 -0.009431

        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

   Class   Amounts
0      0  0.244964
1      0 -0.342475
2      0  1.160686
3      0  0.140534
4      0 -0.073403
```

[39]:
```
X = ccfd.drop('Class',axis = 1)
```

[40]:
```
X
```

```
[40]:              V1         V2        V3        V4        V5        V6  \
0       -1.359807  -0.072781  2.536347  1.378155 -0.338321  0.462388
1        1.191857   0.266151  0.166480  0.448154  0.060018 -0.082361
2       -1.358354  -1.340163  1.773209  0.379780 -0.503198  1.800499
3       -0.966272  -0.185226  1.792993 -0.863291 -0.010309  1.247203
4       -1.158233   0.877737  1.548718  0.403034 -0.407193  0.095921
…             …          …         …         …         …         …
284802 -11.881118  10.071785 -9.834783 -2.066656 -5.364473 -2.606837
284803  -0.732789  -0.055080  2.035030 -0.738589  0.868229  1.058415
284804   1.919565  -0.301254 -3.249640 -0.557828  2.630515  3.031260
284805  -0.240440   0.530483  0.702510  0.689799 -0.377961  0.623708
284806  -0.533413  -0.189733  0.703337 -0.506271 -0.012546 -0.649617

              V7        V8        V9       V10       V11       V12       V13  \
0       0.239599  0.098698  0.363787  0.090794 -0.551600 -0.617801 -0.991390
1      -0.078803  0.085102 -0.255425 -0.166974  1.612727  1.065235  0.489095
2       0.791461  0.247676 -1.514654  0.207643  0.624501  0.066084  0.717293
3       0.237609  0.377436 -1.387024 -0.054952 -0.226487  0.178228  0.507757
4       0.592941 -0.270533  0.817739  0.753074 -0.822843  0.538196  1.345852
…            …         …         …         …         …         …         …
284802 -4.918215  7.305334  1.914428  4.356170 -1.593105  2.711941 -0.689256
284803  0.024330  0.294869  0.584800 -0.975926 -0.150189  0.915802  1.214756
284804 -0.296827  0.708417  0.432454 -0.484782  0.411614  0.063119 -0.183699
284805 -0.686180  0.679145  0.392087 -0.399126 -1.933849 -0.962886 -1.042082
284806  1.577006 -0.414650  0.486180 -0.915427 -1.040458 -0.031513 -0.188093

             V14       V15       V16       V17       V18       V19       V20  \
0      -0.311169  1.468177 -0.470401  0.207971  0.025791  0.403993  0.251412
1      -0.143772  0.635558  0.463917 -0.114805 -0.183361 -0.145783 -0.069083
2      -0.165946  2.345865 -2.890083  1.109969 -0.121359 -2.261857  0.524980
3      -0.287924 -0.631418 -1.059647 -0.684093  1.965775 -1.232622 -0.208038
4      -1.119670  0.175121 -0.451449 -0.237033 -0.038195  0.803487  0.408542
…            …         …         …         …         …         …         …
284802  4.626942 -0.924459  1.107641  1.991691  0.510632 -0.682920  1.475829
284803 -0.675143  1.164931 -0.711757 -0.025693 -1.221179 -1.545556  0.059616
284804 -0.510602  1.329284  0.140716  0.313502  0.395652 -0.577252  0.001396
284805  0.449624  1.962563 -0.608577  0.509928  1.113981  2.897849  0.127434
284806 -0.084316  0.041333 -0.302620 -0.660377  0.167430 -0.256117  0.382948

             V21       V22       V23       V24       V25       V26       V27  \
0      -0.018307  0.277838 -0.110474  0.066928  0.128539 -0.189115  0.133558
1      -0.225775 -0.638672  0.101288 -0.339846  0.167170  0.125895 -0.008983
2       0.247998  0.771679  0.909412 -0.689281 -0.327642 -0.139097 -0.055353
3      -0.108300  0.005274 -0.190321 -1.175575  0.647376 -0.221929  0.062723
4      -0.009431  0.798278 -0.137458  0.141267 -0.206010  0.502292  0.219422
…            …         …         …         …         …         …         …
284802  0.213454  0.111864  1.014480 -0.509348  1.436807  0.250034  0.943651
```

```
284803  0.214205  0.924384  0.012463 -1.016226 -0.606624 -0.395255  0.068472
284804  0.232045  0.578229 -0.037501  0.640134  0.265745 -0.087371  0.004455
284805  0.265245  0.800049 -0.163298  0.123205 -0.569159  0.546668  0.108821
284806  0.261057  0.643078  0.376777  0.008797 -0.473649 -0.818267 -0.002415

             V28   Amounts
0      -0.021053  0.244964
1       0.014724 -0.342475
2      -0.059752  1.160686
3       0.061458  0.140534
4       0.215153 -0.073403
…            …         …
284802  0.823731 -0.350151
284803 -0.053527 -0.254117
284804 -0.026561 -0.081839
284805  0.104533 -0.313249
284806  0.013649  0.514355

[275663 rows x 29 columns]
```

[41]:
```
y = ccfd.Class
```

[42]:
```
y
```

[42]:
```
0         0
1         0
2         0
3         0
4         0
         ..
284802    0
284803    0
284804    0
284805    0
284806    0
Name: Class, Length: 275663, dtype: int64
```

0.3.13 Splitting the dataset into the training set and test set

[43]:
```
from sklearn.model_selection import train_test_split
```

[44]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.
 ↪2,random_state = 42)
```

[45]:
```
X_train.shape
```

[45]:
```
(220530, 29)
```

0.3.14 Training into the Model

[46]:
```
from sklearn.linear_model import LogisticRegression
```

[47]:
```
LR = LogisticRegression()
```

[48]:
```
LR.fit(X_train,y_train)
```

[48]:
```
LogisticRegression()
```

0.3.15 Evaluating the accuracy_score, precision_score

[49]:
```
from sklearn.metrics import precision_score,recall_score,f1_score,accuracy_score
```

[50]:
```
y_pred = LR.predict(X_test)
```

[51]:
```
accuracy_score(y_test,y_pred)
```

[51]:
```
0.9992200678359603
```

[52]:
```
precision_score(y_test,y_pred)
```

[52]:
```
0.8870967741935484
```

[53]:
```
recall_score(y_test,y_pred)
```

[53]:
```
0.6043956043956044
```

Here, precision_score is very low so we have to perform the oversampling and undersampling technique

0.3.16 Handling Imbalanced dataset

[54]:
```
#undersampling
#oversampling
```

0.3.17 Undersampling

[55]:
```
fraud = ccfd[ccfd['Class'] == 1]
normal = ccfd[ccfd['Class'] == 0]
```

[56]:
```
fraud.shape
```

[56]:
```
(473, 30)
```

[57]:
```
normal.shape
```

[57]: (275190, 30)

[58]:
```
#selecting the 473 necessary samples to balance the class feature
equal_sample = normal.sample(n=473)
```

[59]:
```
equal_sample.shape
```

[59]: (473, 30)

[60]:
```
new_ccfd = pd.concat([equal_sample,fraud],ignore_index = True)
```

[61]:
```
new_ccfd['Class'].value_counts()
```

[61]:
```
0    473
1    473
Name: Class, dtype: int64
```

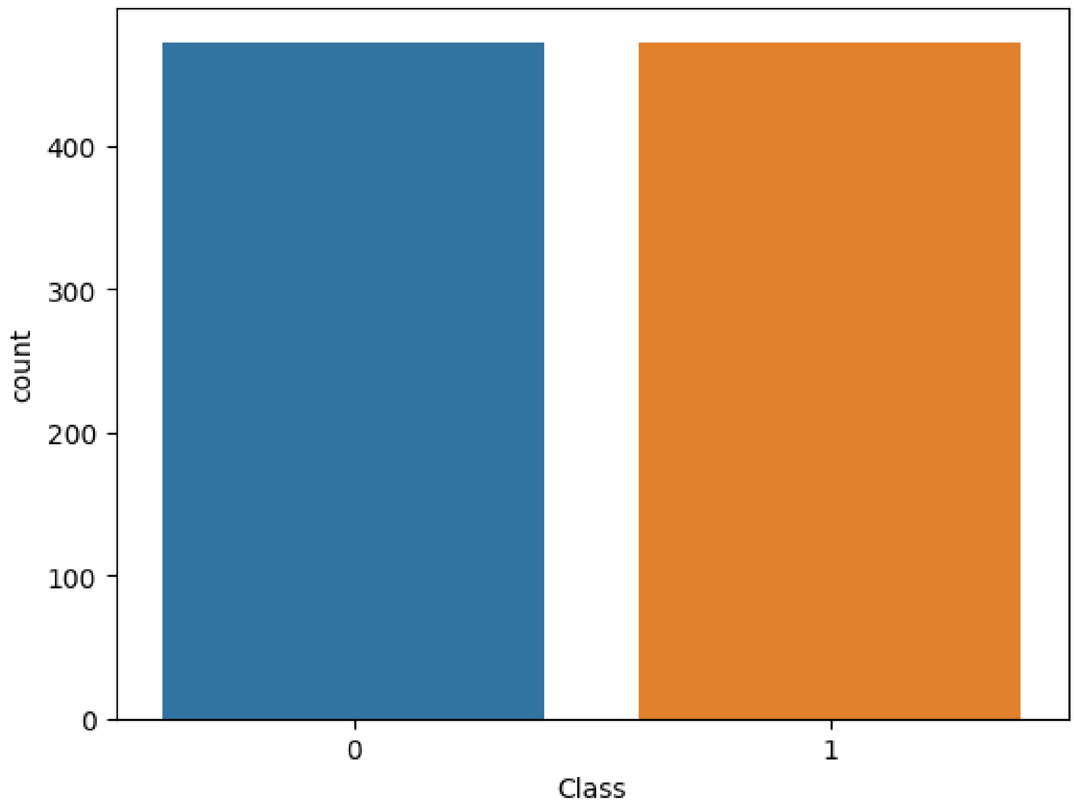
[62]:
```
new_ccfd.head()
```

[62]:
```
         V1        V2        V3        V4        V5        V6        V7  \
0  1.231088  0.817430  0.112470  2.463199  0.318203 -0.784330  0.632496
1 -1.630976 -2.884838  1.778903 -0.678444 -2.293574  0.474592  0.737225
2 -0.817516  1.126725 -0.873509 -1.952757  1.258358 -1.680604  2.106750
3 -6.378440  3.293830  1.730572 -1.586493  0.203319  0.213103  2.941403
4 -0.341223  1.180968  1.308514  0.056025  0.088179 -0.967221  0.760442

         V8        V9        V10       V11       V12       V13       V14  \
0 -0.333888 -1.197490   0.708084 -0.418512  0.373974  0.938349  0.324371
1 -0.123892 -2.113604   0.926001  0.677623 -0.930449 -0.679558 -0.686387
2 -0.938366  0.585487   0.751698  0.737783 -0.248623 -1.923379  0.463119
3 -3.415646  7.938980  11.519106  3.022639 -0.778760 -1.140137 -5.798989
4 -0.112704 -0.499812  -0.532183 -0.093211  0.287980  0.807955 -0.566582

        V15       V16       V17       V18       V19       V20       V21  \
0  0.104770  0.300458 -0.552556 -0.600918 -0.872732 -0.098796 -0.006789
1 -1.151933 -1.348177  1.211932  0.890796  1.434053  1.276694  0.281256
2 -1.335370 -0.224977 -1.250221  0.005941 -1.049938 -0.207060  0.240044
3  0.405939 -1.725896 -2.374682 -0.528987 -0.100571  2.954298 -1.610576
4  0.818074  0.350021 -0.030778 -0.223409 -0.096060  0.158729 -0.255591

        V22       V23       V24       V25       V26       V27       V28  \
0  0.004467 -0.123952  0.366885  0.769766  0.068267 -0.033828  0.011927
1  0.384174  1.064908  0.480130  0.543734  0.122207 -0.265463 -0.103553
2  1.007489 -0.146149  0.072756 -0.626661 -0.285701 -0.496262 -0.081747
3  0.385227 -0.366595  0.479819  0.153885 -0.536129 -4.795220 -3.693525
4 -0.632583 -0.022239  0.353115 -0.137185  0.069528  0.250039  0.099803
```

```
   Class   Amounts
0      0 -0.334838
1      0  1.877583
2      0 -0.289260
3      0 -0.329401
4      0 -0.338876
```

[63]:
```
sns.countplot(x = new_ccfd['Class'],data=new_ccfd)
```

[63]: <Axes: xlabel='Class', ylabel='count'>



Now we equalized the Class feature

[64]:
```
X = new_ccfd.drop('Class',axis = 1)
```

[65]:
```
X
```

[65]:
```
         V1        V2        V3        V4        V5        V6        V7  \
0  1.231088  0.817430  0.112470  2.463199  0.318203 -0.784330  0.632496
1 -1.630976 -2.884838  1.778903 -0.678444 -2.293574  0.474592  0.737225
2 -0.817516  1.126725 -0.873509 -1.952757  1.258358 -1.680604  2.106750
```

```
3   -6.378440  3.293830  1.730572 -1.586493  0.203319  0.213103  2.941403
4   -0.341223  1.180968  1.308514  0.056025  0.088179 -0.967221  0.760442
..        …         …         …         …         …         …         …
941 -1.927883  1.125653 -4.518331  1.749293 -1.566487 -2.010494 -0.882850
942  1.378559  1.289381 -5.004247  1.411850  0.442581 -1.326536 -1.413170
943 -0.676143  1.126366 -2.213700  0.468308 -1.120541 -0.003346 -2.234739
944 -3.113832  0.585864 -5.399730  1.817092 -0.840618 -2.943548 -2.208002
945  1.991976  0.158476 -2.583441  0.408670  1.151147 -0.096695  0.223050

           V8        V9        V10       V11       V12       V13       V14  \
0   -0.333888 -1.197490   0.708084 -0.418512  0.373974  0.938349  0.324371
1   -0.123892 -2.113604   0.926001  0.677623 -0.930449 -0.679558 -0.686387
2   -0.938366  0.585487   0.751698  0.737783 -0.248623 -1.923379  0.463119
3   -3.415646  7.938980  11.519106  3.022639 -0.778760 -1.140137 -5.798989
4   -0.112704 -0.499812  -0.532183 -0.093211  0.287980  0.807955 -0.566582
..        …         …          …         …         …         …         …
941  0.697211 -2.064945  -5.587794  2.115795 -5.417424 -1.235123 -6.665177
942  0.248525 -1.127396  -3.232153  2.858466 -3.096915 -0.792532 -5.210141
943  1.210158 -0.652250  -3.463891  1.794969 -2.775022 -0.418950 -4.057162
944  1.058733 -1.632333  -5.245984  1.933520 -5.030465 -1.127455 -6.416628
945 -0.068384  0.577829  -0.888722  0.491140  0.728903  0.380428 -1.948883

          V15       V16       V17       V18       V19       V20       V21  \
0    0.104770  0.300458 -0.552556 -0.600918 -0.872732 -0.098796 -0.006789
1   -1.151933 -1.348177  1.211932  0.890796  1.434053  1.276694  0.281256
2   -1.335370 -0.224977 -1.250221  0.005941 -1.049938 -0.207060  0.240044
3    0.405939 -1.725896 -2.374682 -0.528987 -0.100571  2.954298 -1.610576
4    0.818074  0.350021 -0.030778 -0.223409 -0.096060  0.158729 -0.255591
..        …         …         …         …         …         …         …
941  0.401701 -2.897825 -4.570529 -1.315147  0.391167  1.252967  0.778584
942 -0.613803 -2.155297 -3.267116 -0.688505  0.737657  0.226138  0.370612
943 -0.712616 -1.603015 -5.035326 -0.507000  0.266272  0.247968  0.751826
944  0.141237 -2.549498 -4.614717 -1.478138 -0.035480  0.306271  0.583276
945 -0.832498  0.519436  0.903562  1.197315  0.593509 -0.017652 -0.164350

          V22       V23       V24       V25       V26       V27       V28  \
0    0.004467 -0.123952  0.366885  0.769766  0.068267 -0.033828  0.011927
1    0.384174  1.064908  0.480130  0.543734  0.122207 -0.265463 -0.103553
2    1.007489 -0.146149  0.072756 -0.626661 -0.285701 -0.496262 -0.081747
3    0.385227 -0.366595  0.479819  0.153885 -0.536129 -4.795220 -3.693525
4   -0.632583 -0.022239  0.353115 -0.137185  0.069528  0.250039  0.099803
..        …         …         …         …         …         …         …
941 -0.319189  0.639419 -0.294885  0.537503  0.788395  0.292680  0.147968
942  0.028234 -0.145640 -0.081049  0.521875  0.739467  0.389152  0.186637
943  0.834108  0.190944  0.032070 -0.739695  0.471111  0.385107  0.194361
944 -0.269209 -0.456108 -0.183659 -0.328168  0.606116  0.884876 -0.253700
945 -0.295135 -0.072173 -0.450261  0.313267 -0.289617  0.002988 -0.015309
```

```
      Amounts
0   -0.334838
1    1.877583
2   -0.289260
3   -0.329401
4   -0.338876
..        …
941  1.206024
942 -0.350191
943 -0.041818
944  0.626302
945 -0.183191

[946 rows x 29 columns]
```

[66]:
```
y = new_ccfd.Class
```

[67]:
```
y
```

[67]:
```
0      0
1      0
2      0
3      0
4      0
      ..
941    1
942    1
943    1
944    1
945    1
Name: Class, Length: 946, dtype: int64
```

0.3.18 Again Splitting the data for training and testing

[68]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.
 ↪2,random_state = 42)
```

[69]:
```
X_train.shape
```

[69]:
```
(756, 29)
```

0.3.19 Logistis Regression

[70]:
```
LR.fit(X_train,y_train)
```

[70]: LogisticRegression()

[71]:
```
y_pred1 = LR.predict(X_test)
```

[72]:
```
accuracy_score(y_test,y_pred1)
```

[72]: 0.9526315789473684

[73]:
```
precision_score(y_test,y_pred1)
```

[73]: 0.9894736842105263

[74]:
```
f1_score(y_test,y_pred1)
```

[74]: 0.9543147208121827

0.3.20 Decision Tree Classification

[75]:
```
from sklearn.tree import DecisionTreeClassifier
```

[76]:
```
DTC = DecisionTreeClassifier()
```

[77]:
```
DTC.fit(X_train,y_train)
```

[77]: DecisionTreeClassifier()

[78]:
```
y_pred2 = DTC.predict(X_test)
```

0.3.21 Evaluating the precision_score, accuracy_score, f1_score

[79]:
```
accuracy_score(y_test,y_pred2)
```

[79]: 0.9157894736842105

[80]:
```
precision_score(y_test,y_pred2)
```

[80]: 0.9215686274509803

[81]:
```
f1_score(y_test,y_pred2)
```

[81]: 0.9215686274509803

0.3.22 RandomForest Classifier

[82]:
```
from sklearn.ensemble import RandomForestClassifier
```

[83]:
```
RFC = RandomForestClassifier()
```

[84]:
```
RFC.fit(X_train,y_train)
```

[84]: RandomForestClassifier()

[85]:
```
y_pred3 = RFC.predict(X_test)
```

0.3.23 Evaluating the precision_Score, accuracy_score,f1_score

[86]:
```
accuracy_score(y_test,y_pred3)
```

[86]: 0.9421052631578948

[87]:
```
precision_score(y_test,y_pred3)
```

[87]: 0.989247311827957

[88]:
```
f1_score(y_test,y_pred3)
```

[88]: 0.9435897435897437

0.3.24 LightBGM

[89]:
```
pip install lightgbm
```

```
Requirement already satisfied: lightgbm in /usr/local/lib/python3.10/dist-
packages (4.1.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from lightgbm) (1.23.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages
(from lightgbm) (1.11.3)
```

[90]:
```
from lightgbm import LGBMClassifier
```

[91]:
```
LGBM = LGBMClassifier()
```

[92]:
```
LGBM.fit(X_train,y_train)
```

```
[LightGBM] [Info] Number of positive: 371, number of negative: 385
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of
testing was 0.005193 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 7317
[LightGBM] [Info] Number of data points in the train set: 756, number of used
features: 29
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.490741 -> initscore=-0.037041
```

```
[LightGBM] [Info] Start training from score -0.037041
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

[92]:
```
LGBMClassifier()
```

[93]:
```
y_pred4 = LGBM.predict(X_test)
```

0.3.25 Evaluating the precision_Score, accuracy_score,f1_score

[94]:
```
accuracy_score(y_test,y_pred4)
```

[94]:
```
0.9421052631578948
```

[95]:
```
precision_score(y_test,y_pred4)
```

[95]:
```
0.989247311827957
```

[96]:
```
f1_score(y_test,y_pred4)
```

[96]:
```
0.9435897435897437
```

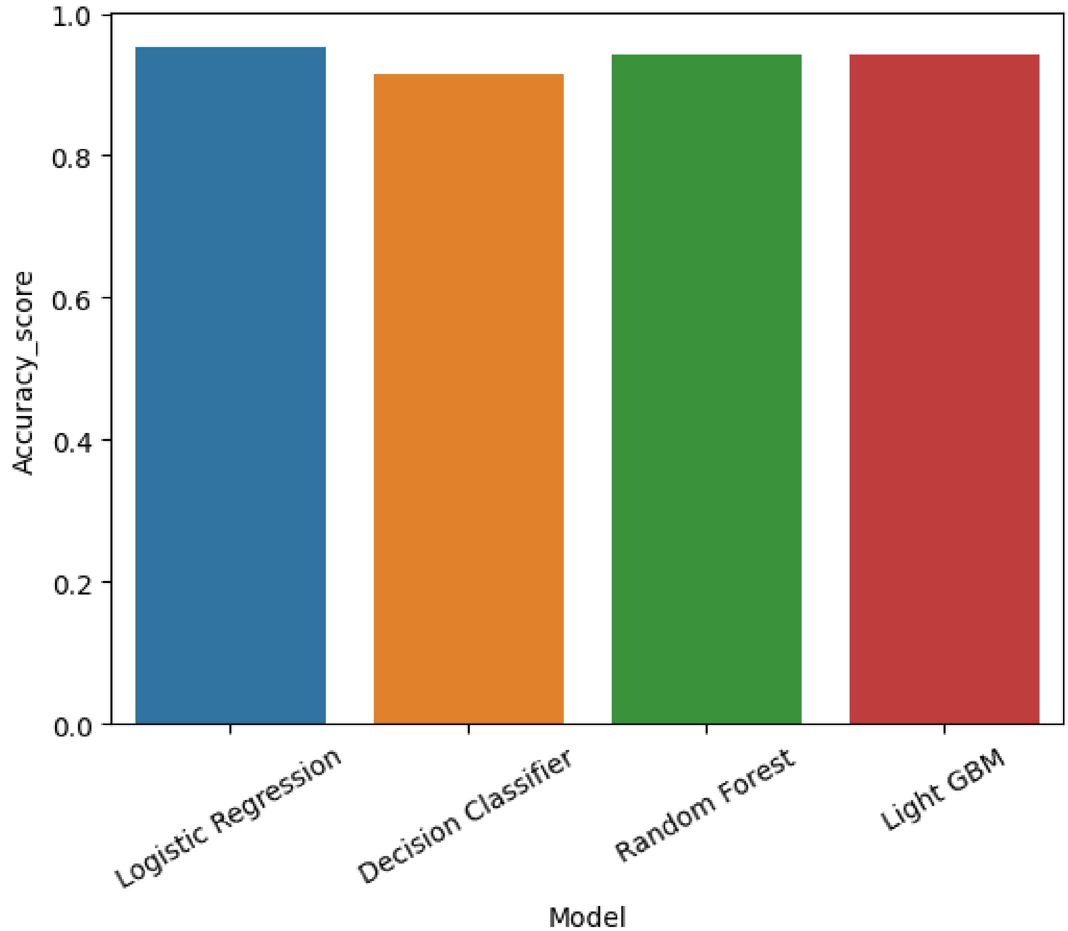
0.3.26 Checking which model is performing better accuracy_score

[97]:
```
stats = pd.DataFrame({'Model':['Logistic Regression','Decision␣
↪Classifier','Random Forest','Light GBM'],
                    'Accuracy_score':
↪[accuracy_score(y_test,y_pred1),accuracy_score(y_test,y_pred2),accuracy_score(y_test,y_pred
```

[98]:
```
stats
```

[98]:
```
                 Model  Accuracy_score
0  Logistic Regression        0.952632
1  Decision Classifier        0.915789
2        Random Forest        0.942105
3            Light GBM        0.942105
```

[99]:
```
ax = sns.barplot(x = 'Model',y = 'Accuracy_score',data = stats)
plt.xticks(rotation=30)
plt.show()
```



As we are losting so much of feature information in undersampling, so move head to oversampling

[99]:

```

```

0.3.27 Oversampling

[100]:

```
pip install imbalanced-learn==0.10.1
```

```
Requirement already satisfied: imbalanced-learn==0.10.1 in
/usr/local/lib/python3.10/dist-packages (0.10.1)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-
packages (from imbalanced-learn==0.10.1) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-
packages (from imbalanced-learn==0.10.1) (1.11.3)
Requirement already satisfied: scikit-learn>=1.0.2 in
/usr/local/lib/python3.10/dist-packages (from imbalanced-learn==0.10.1) (1.2.2)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from imbalanced-learn==0.10.1) (1.3.2)
```

```
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from imbalanced-learn==0.10.1) (3.2.0)
```

[101]:
```
pip install -U imbalanced-learn
```

```
Requirement already satisfied: imbalanced-learn in
/usr/local/lib/python3.10/dist-packages (0.10.1)
Collecting imbalanced-learn
  Downloading imbalanced_learn-0.11.0-py3-none-any.whl (235 kB)
                           235.6/235.6

kB 5.1 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.17.3 in
/usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.23.5)
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-
packages (from imbalanced-learn) (1.11.3)
Requirement already satisfied: scikit-learn>=1.0.2 in
/usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.2.2)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from imbalanced-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (3.2.0)
Installing collected packages: imbalanced-learn
  Attempting uninstall: imbalanced-learn
    Found existing installation: imbalanced-learn 0.10.1
    Uninstalling imbalanced-learn-0.10.1:
      Successfully uninstalled imbalanced-learn-0.10.1
Successfully installed imbalanced-learn-0.11.0
```

[102]:
```
from imblearn.over_sampling import SMOTE
```

[103]:
```
x2 = ccfd.drop('Class',axis=1)
```

[104]:
```
x2.head()
```

[104]:
```
         V1        V2        V3        V4        V5        V6        V7  \
0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599
1  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803
2 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461
3 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609
4 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941

         V8        V9       V10       V11       V12       V13       V14  \
0  0.098698  0.363787  0.090794 -0.551600 -0.617801 -0.991390 -0.311169
1  0.085102 -0.255425 -0.166974  1.612727  1.065235  0.489095 -0.143772
2  0.247676 -1.514654  0.207643  0.624501  0.066084  0.717293 -0.165946
3  0.377436 -1.387024 -0.054952 -0.226487  0.178228  0.507757 -0.287924
4 -0.270533  0.817739  0.753074 -0.822843  0.538196  1.345852 -1.119670
```

```
        V15       V16       V17       V18       V19       V20       V21  \
0  1.468177 -0.470401  0.207971  0.025791  0.403993  0.251412 -0.018307
1  0.635558  0.463917 -0.114805 -0.183361 -0.145783 -0.069083 -0.225775
2  2.345865 -2.890083  1.109969 -0.121359 -2.261857  0.524980  0.247998
3 -0.631418 -1.059647 -0.684093  1.965775 -1.232622 -0.208038 -0.108300
4  0.175121 -0.451449 -0.237033 -0.038195  0.803487  0.408542 -0.009431

        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

    Amounts
0  0.244964
1 -0.342475
2  1.160686
3  0.140534
4 -0.073403
```

[105]:
```
y2 = ccfd.Class
```

[106]:
```
y2
```

[106]:
```
0         0
1         0
2         0
3         0
4         0
         ..
284802    0
284803    0
284804    0
284805    0
284806    0
Name: Class, Length: 275663, dtype: int64
```

[107]:
```
X_res,y_res = SMOTE().fit_resample(x2,y2)
```

[108]:
```
y_res.value_counts()
```

[108]:
```
0    275190
1    275190
Name: Class, dtype: int64
```

0.3.28 Again split the training and testing data

[109]:
```
X_train,X_test,y_train,y_test = train_test_split(X_res,y_res,test_size = 0.
 ↪2,random_state=42)
```

0.3.29 Train the Model

0.3.30 Logistic Regression

[110]:
```
#already imported
LR.fit(X_train,y_train)
```

[110]: LogisticRegression()

0.3.31 Evaluating accuracy_score,precision_score,f1_score

[111]:
```
accuracy_score(y_test,LR.predict(X_test))
```

[111]: 0.9447563501580726

[112]:
```
precision_score(y_test,LR.predict(X_test))
```

[112]: 0.9733165634674923

[113]:
```
f1_score(y_test,LR.predict(X_test))
```

[113]: 0.9429993532240376

0.3.32 Decision Tree Classifier

[114]:
```
DTC.fit(X_train,y_train)
```

[114]: DecisionTreeClassifier()

0.3.33 Evaluating accuracy_Score,precision_Score,f1_score

[115]:
```
accuracy_score(y_test,DTC.predict(X_test))
```

[115]: 0.9981649042479741

[116]:
```
precision_score(y_test,DTC.predict(X_test))
```

[116]: 0.9972596773315427

[117]:
```
f1_score(y_test,DTC.predict(X_test))
```

[117]: 0.9981654042468168

0.3.34 Random Forest Classifier

[118]:
```
RFC.fit(X_train,y_train)
```

[118]:
```
RandomForestClassifier()
```

0.3.35 Evaluating accuracy_Score,precision_Score,f1_score

[119]:
```
accuracy_score(y_test,RFC.predict(X_test))
```

[119]:
```
0.999918238308078
```

[120]:
```
precision_score(y_test,RFC.predict(X_test))
```

[120]:
```
0.9998363993310551
```

[121]:
```
f1_score(y_test,RFC.predict(X_test))
```

[121]:
```
0.9999181929736854
```

0.3.36 LightGBM

[122]:
```
LGBM.fit(X_train,y_train)
```

```
[LightGBM] [Info] Number of positive: 220187, number of negative: 220117
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of
testing was 0.132663 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 7395
[LightGBM] [Info] Number of data points in the train set: 440304, number of used
features: 29
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.500079 -> initscore=0.000318
[LightGBM] [Info] Start training from score 0.000318
```

[122]:
```
LGBMClassifier()
```

0.3.37 Evaluating accuracy_Score,precision_Score,f1_score

[123]:
```
accuracy_score(y_test,LGBM.predict(X_test))
```

[123]:
```
0.9992823140375741
```

[124]:
```
precision_score(y_test,LGBM.predict(X_test))
```

[124]:
```
0.9985838779956427
```

[125]:
```
f1_score(y_test,LGBM.predict(X_test))
```

[125]: 0.9992823596740641

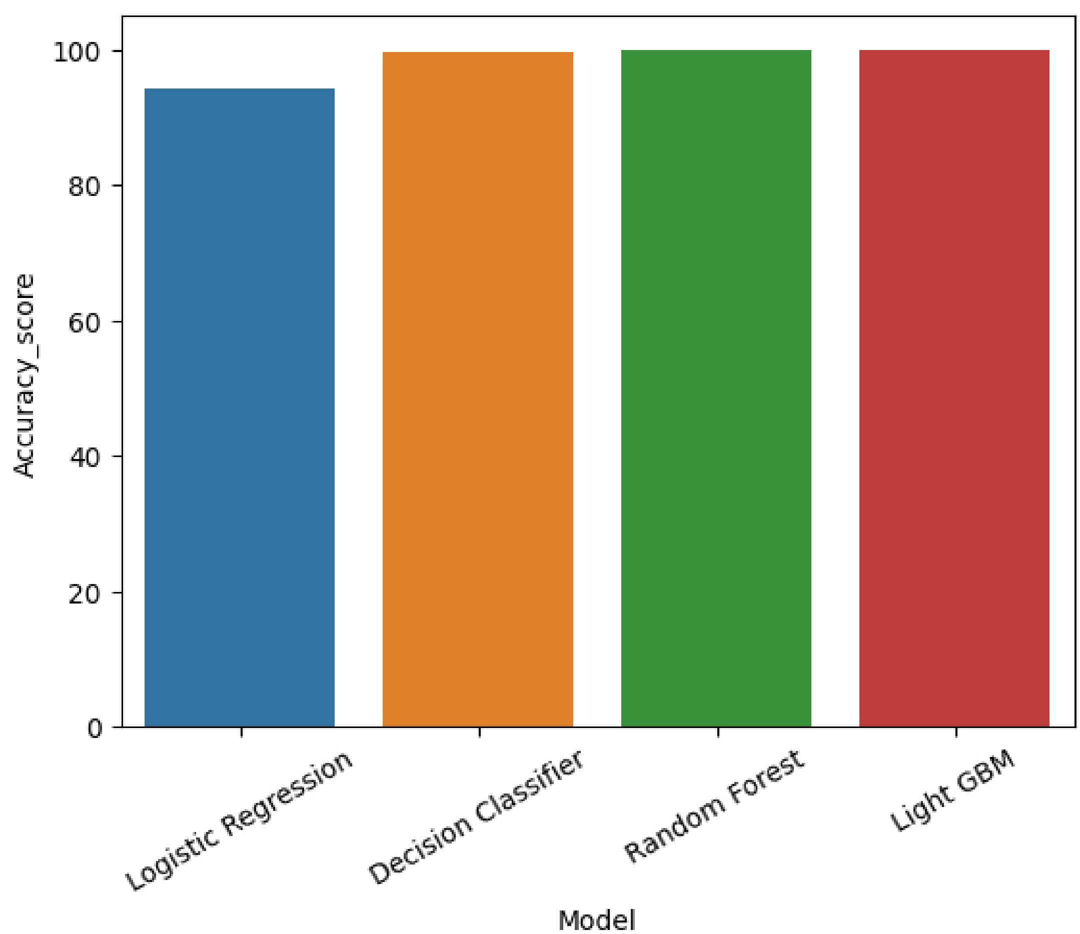
[126]:
```
stats_oversampling = pd.DataFrame({'Model':['Logistic Regression','Decision␣
↪Classifier','Random Forest','Light GBM'],
                      'Accuracy_score':[accuracy_score(y_test,LR.
↪predict(X_test))*100,accuracy_score(y_test,DTC.
↪predict(X_test))*100,accuracy_score(y_test,RFC.
↪predict(X_test))*100,accuracy_score(y_test,LGBM.predict(X_test))*100]})
```

[127]:
```
stats_oversampling
```

[127]:

	Model	Accuracy_score
0	Logistic Regression	94.475635
1	Decision Classifier	99.816490
2	Random Forest	99.991824
3	Light GBM	99.928231

[128]:
```
sns.barplot(x = 'Model',y = 'Accuracy_score',data = stats_oversampling)
plt.xticks(rotation=30)
plt.show()
```



0.3.38 Since Random Forest and Light Gradient Boosting Machine is performing better

[129]:
```
import joblib
```

[130]:
```
joblib.dump(RFC,"drive/MyDrive/ColabNotebooks/CCFD MODEL.txt")
```

[130]: ['drive/MyDrive/ColabNotebooks/CCFD MODEL.txt']

[131]:
```
model = joblib.load("drive/MyDrive/ColabNotebooks/CCFD MODEL.txt")
```

[132]:
```
predicted = model.
 ↪predict([[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1]])
if(predicted == 0):
    print("Normal Transaction")
else:
    print("Fraudlent Transaction")
```

```
Normal Transaction
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but RandomForestClassifier was fitted with feature
names
  warnings.warn(
```

[133]:
```
import pickle
```

[134]:
```
pickle.dump(RFC,open("drive/MyDrive/ColabNotebooks/ccfd.txt","wb"))
```