In [106]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

In [107]:
```
credit_card_data=pd.read_csv(r'C:\Users\ranim\Downloads\creditcard.csv\credi
```

In [108]:
```
print(credit_card_data)
```

```
            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]
```

In [109]:
```
credit_card_data.head()
```

Out[109]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533

5 rows × 31 columns

In [110]:
```
credit_card_data.shape
```

Out[110]:
```
(284807, 31)
```

In [111]:
```
credit_card_data.tail()
```

Out[111]:

	Time	V1	V2	V3	V4	V5	V6	V7
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006

5 rows × 31 columns

In [112]:
```
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
```

In [113]:
```
#checking number of missing values of each columns
credit_card_data.isnull().sum()
```

Out[113]:
```
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
```

In [114]:
```
#distribution of legit  trasaction &fraudulent trasection
credit_card_data['Class'].value_counts()
```

Out[114]:
```
Class
0    284315
1       492
Name: count, dtype: int64
```

This Dataset is Highly Unbalanced

0->Normat Trasaction

1->Fraudult Trasaction

In [115]:
```
#separating the data for analysis
legit = credit_card_data[credit_card_data.Class==0]
fraud = credit_card_data[credit_card_data.Class==1]
```

In [116]:
```
print(legit.shape)
print(fraud.shape)
```

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

In [117]:
```
#Statistical message of the data

legit.Amount.describe()
```

Out[117]:
```
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
```

In [118]:
```
fraud.Amount.describe()
```

Out[118]:
```
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 [119]:
```
#compare values for both trasaction
credit_card_data.groupby('Class').mean()
```

Out[119]:

Class	Time	V1	V2	V3	V4	V5	V6	V
0	94838.202258	0.008258	-0.006271	0.012171	-0.007860	0.005453	0.002419	0.00963
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.56873

2 rows × 30 columns

Under-Sampling

Build a sample dataset containing similar distribution of normal trasactions and Fraudulent Transactions

Number of Fraudulent Transactions --> 492

In [120]:
```
legit_sample=legit.sample(n=492)
```

Concatening two DataFrames

In [121]:
```
new_dataset=pd.concat([legit_sample,fraud],axis=0)
```

In [122]:
```
new_dataset.head()
```

Out[122]:

	Time	V1	V2	V3	V4	V5	V6	V7
21856	31897.0	-1.188394	-1.602988	1.295811	-3.545147	0.181809	0.144712	-0.491763
155591	105862.0	2.139933	0.027373	-2.396735	-0.072527	0.944102	-0.874399	0.525409
121419	76184.0	-0.405210	1.039527	1.384470	0.001575	-0.069324	-0.781118	0.593321
209987	137793.0	2.252485	-0.532105	-2.510164	-1.138217	0.562797	-0.983605	0.351924
223690	143484.0	1.893135	-0.479994	-0.985241	0.060864	-0.291145	-0.511276	-0.222454

5 rows × 31 columns

In [123]:
```
new_dataset.tail()
```

Out[123]:

	Time	V1	V2	V3	V4	V5	V6	V7	
279863	169142.0	-1.927883	1.125653	-4.518331	1.749293	-1.566487	-2.010494	-0.882850	(
280143	169347.0	1.378559	1.289381	-5.004247	1.411850	0.442581	-1.326536	-1.413170	(
280149	169351.0	-0.676143	1.126366	-2.213700	0.468308	-1.120541	-0.003346	-2.234739	1
281144	169966.0	-3.113832	0.585864	-5.399730	1.817092	-0.840618	-2.943548	-2.208002	1
281674	170348.0	1.991976	0.158476	-2.583441	0.408670	1.151147	-0.096695	0.223050	-(

5 rows × 31 columns

In [124]:
```
new_dataset['Class'].value_counts()
```

Out[124]:
```
Class
0    492
1    492
Name: count, dtype: int64
```

In [125]:
```
new_dataset.groupby('Class').mean()
```

Out[125]:

Class	Time	V1	V2	V3	V4	V5	V6	V7
0	98323.113821	0.059713	0.014442	-0.015339	-0.022739	0.051276	-0.029666	0.072361
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.568731

2 rows × 30 columns

Splitting data into features & Targets

In [126]:
```
x=new_dataset.drop(columns='Class',axis=1)
y=new_dataset['Class']
```

In [127]:
```
print(x)
```

```
            Time         V1         V2        V3         V4         V5
V6
21856    31897.0 -1.188394 -1.602988  1.295811 -3.545147  0.181809  0.1447
12  \
155591  105862.0  2.139933  0.027373 -2.396735 -0.072527  0.944102 -0.8743
99
121419   76184.0 -0.405210  1.039527  1.384470  0.001575 -0.069324 -0.7811
18
209987  137793.0  2.252485 -0.532105 -2.510164 -1.138217  0.562797 -0.9836
05
223690  143484.0  1.893135 -0.479994 -0.985241  0.060864 -0.291145 -0.5112
76
...          ...       ...       ...       ...       ...       ...
...
279863  169142.0 -1.927883  1.125653 -4.518331  1.749293 -1.566487 -2.0104
94
280143  169347.0  1.378559  1.289381 -5.004247  1.411850  0.442581 -1.3265
36
280149  169351.0 -0.676143  1.126366 -2.213700  0.468308 -1.120541 -0.0033
46
281144  169966.0 -3.113832  0.585864 -5.399730  1.817092 -0.840618 -2.9435
48
281674  170348.0  1.991976  0.158476 -2.583441  0.408670  1.151147 -0.0966
95

               V7         V8         V9  ...       V20       V21       V22
\
21856  -0.491763  0.460799  0.661532  ... -0.290061  0.026230  0.526782
\
155591  0.525409 -0.413893  1.509907  ... -0.301723 -0.028861  0.185612
121419  0.593321  0.087993 -0.594281  ...  0.094685 -0.202138 -0.580632
209987  0.351924 -0.580009 -1.127677  ...  0.160610  0.335664  0.912548
223690 -0.222454  0.065980  0.547990  ... -0.156479 -0.395350 -1.461735
...          ...       ...       ...  ...       ...       ...       ...
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
21856   0.391664 -0.654446 -0.427227 -0.852091  0.237973  0.171892   95.89
155591 -0.172178 -1.136685  0.477765  0.252391 -0.129659 -0.106796   18.00
121419  0.013173  0.470099 -0.233264  0.043357  0.237441  0.089384    7.18
209987 -0.170458  0.183196  0.586231  0.078141 -0.085014 -0.070959   55.70
223690  0.550013  0.662219 -0.825634 -0.035387 -0.100550 -0.044203   65.69
...          ...       ...       ...       ...       ...       ...     ...
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]
```

In [128]:
```
print(y)
```

```
21856     0
155591    0
121419    0
209987    0
223690    0
         ..
279863    1
280143    1
280149    1
281144    1
281674    1
Name: Class, Length: 984, dtype: int64
```

Split Data into training data & Testing data

In [129]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=2,stratify=y,ra
```

In [130]:
```
print(x.shape,x_train.shape,x_test.shape)
```

```
(984, 30) (982, 30) (2, 30)
```

Model Training

Logistic Regression

In [131]:
```
model = LogisticRegression()
```

In [132]:
```
#training the logistic regression model with training data
model.fit(x_train,y_train)
```

```
C:\Users\ranim\AppData\Local\Programs\Python\Python310\lib\site-packages\s
klearn\linear_model\_logistic.py:458: 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 i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-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-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
```

Out[132]:
```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Model Evaluation

Accuracy

In [133]:
```
x_train_prediction = model.predict(x_train)
training_data_accuracy = accuracy_score(x_train_prediction,y_train)
```

In [134]:
```
print('Accuracy on Training Data : ',training_data_accuracy)
```

```
Accuracy on Training Data :  0.9490835030549898
```

In [135]:
```
x_test_prediction = model.predict(x_test)
test_data_accuracy = accuracy_score(x_test_prediction,y_test)
```

In [136]:
```
print('Accuracy on Training Data : ',test_data_accuracy)
```

```
Accuracy on Training Data :  1.0
```

In [ ]: