In [6]:

```
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
```

In [7]:

```
credit_card_data =pd.read_csv('creditcard.csv')
```

In [8]:

```
credit_card_data.head()
```

Out[8]:

| | 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 | 0.3 |
| **1** | 0.0 | 1.191857 | 0.266151 | 0.166480 | 0.448154 | 0.060018 | -0.082361 | -0.078803 | 0.085102 | -0.2 |
| **2** | 1.0 | -1.358354 | -1.340163 | 1.773209 | 0.379780 | -0.503198 | 1.800499 | 0.791461 | 0.247676 | -1.5 |
| **3** | 1.0 | -0.966272 | -0.185226 | 1.792993 | -0.863291 | -0.010309 | 1.247203 | 0.237609 | 0.377436 | -1.3 |
| **4** | 2.0 | -1.158233 | 0.877737 | 1.548718 | 0.403034 | -0.407193 | 0.095921 | 0.592941 | -0.270533 | 0.8 |

5 rows × 31 columns

In [9]:

```
credit_card_data.tail()
```

Out[9]:

| | 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 | 7.3 |
| **284803** | 172787.0 | -0.732789 | -0.055080 | 2.035030 | -0.738589 | 0.868229 | 1.058415 | 0.024330 | 0.2 |
| **284804** | 172788.0 | 1.919565 | -0.301254 | -3.249640 | -0.557828 | 2.630515 | 3.031260 | -0.296827 | 0.7 |
| **284805** | 172788.0 | -0.240440 | 0.530483 | 0.702510 | 0.689799 | -0.377961 | 0.623708 | -0.686180 | 0.6 |
| **284806** | 172792.0 | -0.533413 | -0.189733 | 0.703337 | -0.506271 | -0.012546 | -0.649617 | 1.577006 | -0.4 |

5 rows × 31 columns

In [6]:

```
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 [10]:

```
credit_card_data.isnull().sum()
```

Out[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
```

In [11]:

```
credit_card_data['Class'].value_counts()
```

Out[11]:

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

# The data is highly unbalanced

# 0---> Normal Transaction

# 1---> Fraudulent Transaction

In [12]:

```
legit=credit_card_data[credit_card_data.Class==0]
fraud=credit_card_data[credit_card_data.Class==1]
```

In [13]:

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

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

In [14]:

```
legit.Amount.describe()
```

Out[14]:

```
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 [15]:

```
fraud.Amount.describe()
```

Out[15]:

```
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 [16]:

```
credit_card_data.groupby('Class').mean()
```

Out[16]:

| | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | |
|---|---|---|---|---|---|---|---|---|---|
| **Class** | | | | | | | | | |
| **0** | 94838.202258 | 0.008258 | -0.006271 | 0.012171 | -0.007860 | 0.005453 | 0.002419 | 0.009637 | -0. |
| **1** | 80746.806911 | -4.771948 | 3.623778 | -7.033281 | 4.542029 | -3.151225 | -1.397737 | -5.568731 | 0. |

2 rows × 30 columns

# Under- Sampling

In [17]:

```
legit_sample = legit.sample(n=492)
```

In [18]:

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

In [19]:

```
new_dataset.head()
```

Out[19]:

| | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | |
|---|---|---|---|---|---|---|---|---|---|
| **1825** | 1418.0 | -0.814336 | 1.139683 | 1.067270 | 1.231870 | -0.192121 | 0.319095 | 0.106998 | 0.74 |
| **195983** | 131307.0 | 1.818970 | -0.820930 | -0.477969 | 0.091696 | -0.726897 | 0.046201 | -0.751738 | 0.19 |
| **19681** | 30466.0 | 1.452901 | -0.475756 | 0.155140 | -0.575338 | -0.718861 | -0.603680 | -0.435483 | -0.20 |
| **173524** | 121522.0 | 1.878979 | -1.217169 | -1.639267 | -0.887792 | -0.366040 | -0.709471 | -0.050754 | -0.36 |
| **66669** | 52163.0 | 1.489330 | -0.281605 | -0.548476 | -0.916856 | 0.013046 | -0.451147 | -0.107197 | -0.24 |

5 rows × 31 columns

In [20]:

```
new_dataset.tail()
```

Out[20]:

| | 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 | 0.697 |
| **280143** | 169347.0 | 1.378559 | 1.289381 | -5.004247 | 1.411850 | 0.442581 | -1.326536 | -1.413170 | 0.248 |
| **280149** | 169351.0 | -0.676143 | 1.126366 | -2.213700 | 0.468308 | -1.120541 | -0.003346 | -2.234739 | 1.210 |
| **281144** | 169966.0 | -3.113832 | 0.585864 | -5.399730 | 1.817092 | -0.840618 | -2.943548 | -2.208002 | 1.058 |
| **281674** | 170348.0 | 1.991976 | 0.158476 | -2.583441 | 0.408670 | 1.151147 | -0.096695 | 0.223050 | -0.068 |

5 rows × 31 columns

In [21]:

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

Out[21]:

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

In [22]:

```
new_dataset.groupby('Class').mean()
```

Out[22]:

| | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | |
|---|---|---|---|---|---|---|---|---|---|
| **Class** | | | | | | | | | |
| **0** | 93245.941057 | 0.059877 | -0.024429 | 0.011189 | -0.033723 | -0.092501 | 0.069108 | -0.058183 | 0.0 |
| **1** | 80746.806911 | -4.771948 | 3.623778 | -7.033281 | 4.542029 | -3.151225 | -1.397737 | -5.568731 | 0.5 |

2 rows × 30 columns

In [23]:

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

In [24]:

```
print(X)
```

```
            Time        V1        V2        V3        V4        V5        V6  \
1825      1418.0 -0.814336  1.139683  1.067270  1.231870 -0.192121  0.319095
195983  131307.0  1.818970 -0.820930 -0.477969  0.091696 -0.726897  0.046201
19681    30466.0  1.452901 -0.475756  0.155140 -0.575338 -0.718861 -0.603680
173524  121522.0  1.878979 -1.217169 -1.639267 -0.887792 -0.366040 -0.709471
66669    52163.0  1.489330 -0.281605 -0.548476 -0.916856  0.013046 -0.451147
...          ...       ...       ...       ...       ...       ...       ...
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  \
1825    0.106998  0.743026 -0.557680  ... -0.013667  0.052109  0.278061
195983 -0.751738  0.192960  1.359420  ... -0.074017  0.261781  0.734425
19681  -0.435483 -0.207618 -0.554431  ...  0.075195  0.021318  0.213959
173524 -0.050754 -0.361780 -0.609896  ...  0.425656  0.037658 -0.351184
66669  -0.107197 -0.243042 -1.168847  ...  0.064548  0.160686  0.490414
...          ...       ...       ...  ...       ...       ...       ...
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
1825   -0.078368  0.010284 -0.130791 -0.235254  0.291107  0.125879   18.42
195983  0.110689  0.788187 -0.239800  0.064404 -0.011390 -0.033860   79.95
19681  -0.165682 -0.058139  0.772633 -0.115557  0.009892  0.003235    2.99
173524  0.120269  0.482956 -0.196914 -0.466271 -0.070382 -0.018032  201.96
66669  -0.233210 -0.711561  0.852530 -0.003296 -0.007937 -0.011206    1.00
...          ...       ...       ...       ...       ...       ...     ...
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 [25]:
```
print(Y)
```

```
1825      0
195983    0
19681     0
173524    0
66669     0
         ..
279863    1
280143    1
280149    1
281144    1
281674    1
Name: Class, Length: 984, dtype: int64
```

In [27]:
```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,stratify=Y,random_
```

In [28]:
```
print(X.shape,X_train.shape,X_test.shape)
```

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

In [29]:
```
model=LogisticRegression()
```

In [30]:

```
model.fit(X_train,Y_train)
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\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 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(
```

Out[30]:

```
▾ LogisticRegression
LogisticRegression()
```

In [31]:

```
X_train_prediction=model.predict(X_train)
training_data_accuracy=accuracy_score(X_train_prediction, Y_train)
```

In [33]:

```
print('Accuracy on Training data:', training_data_accuracy)
```

```
Accuracy on Training data: 0.9491740787801779
```

In [34]:

```
X_test_prediction=model.predict(X_test)
```

In [37]:

```
test_Data_accuracy=accuracy_score(X_test_prediction, Y_test)
```

In [40]:

```
print('Accuracy score on Test data:', test_Data_accuracy)
```

```
Accuracy score on Test data: 0.9137055837563451
```

In [ ]: