In [3]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

In [5]:
```python
creditcard=pd.read_csv("C:\\Users\\Pranav\\Desktop\\DATA SCIENCE DATA\\CVC file\\creditcard.csv")
creditcard.head()
```

Out[5]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0

5 rows × 31 columns

In [6]:
```python
#shape of dataframe
creditcard.shape
```

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

In [8]:
```python
#information of dataframe
creditcard.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 [9]:
```python
##describe mathamatical data
creditcard.describe()
```

Out[9]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	...	2.848070e+05	2.848070e
mean	94813.859575	3.918649e-15	5.682686e-16	-8.761736e-15	2.811118e-15	-1.552103e-15	2.040130e-15	-1.698953e-15	-1.893285e-16	-3.147640e-15	...	1.473120e-16	8.042109
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.098632e+00	...	7.345240e-01	7.257016
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.343407e+01	...	-3.483038e+01	-1.093314e
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.430976e-01	...	-2.283949e-01	-5.423504
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5.142873e-02	...	-2.945017e-02	6.781943
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01	5.971390e-01	...	1.863772e-01	5.285536
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01	1.559499e+01	...	2.720284e+01	1.050309e

8 rows × 31 columns

In [11]:
```python
#null value in dataframe
creditcard.isnull().sum()
```

Out[11]:
```
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 [28]:
```python
#distribution of legit Transaction & fraudulent Transaction
creditcard['Class'].value_counts()
```

Out[28]:
```
0    284315
1       492
Name: Class, dtype: int64
```

In [29]:
```python
#This dataest is highly unblanced
#O--->legit Transaction    1---> fraudulent Transaction
legit=creditcard[creditcard.Class==0]
fraudulent=creditcard[creditcard.Class==1]
```

In [30]:
```python
print(legit)
```

```
            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  

[284315 rows x 31 columns]
```

In [35]:
```python
print(legit.shape)
print(fraudulent.shape)
```

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

In [36]:
```python
#statatical measure of the data
legit.Amount.describe()
```

Out[36]:
```
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 [37]:
```python
fraudulent.Amount.describe()
```

Out[37]:
```
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 [39]:
```python
#compare the values for both transaction
creditcard.groupby('Class').mean()
```

Out[39]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	V21	V22	V23	V24	V25
Class																	
0	94838.202258	0.008258	-0.006271	0.012171	-0.007860	0.005453	0.002419	0.009637	-0.000987	0.004467	...	-0.000644	-0.001235	-0.000024	0.000070	0.000182	-0.000072
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.568731	0.570636	-2.581123	...	0.372319	0.713588	0.014049	-0.040308	-0.105130	0.041449

2 rows × 30 columns

In [40]:
```python
#Under sampling
#build a sample dataset containg simlar distribution  of normal transcation and fraudulent transcation
#Number of fraudulent transcation------->498
legit_sample=legit.sample(n=498)
```

In [41]:
```python
#concatentaing the dataframe
new_dataset=pd.concat([legit_sample,fraudulent],axis=0)
```

In [42]:
```python
new_dataset.head()
```

Out[42]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	
11297	19645.0	1.090666	0.336109	-0.072621	1.384967	0.060575	-1.096974	0.623667	-0.489797	0.919411	...	-0.082523	-0.131621	-0.226813	0.358635	0.810954	-0.346882	-0.0
90804	63175.0	-0.462649	1.025205	1.677640	-0.104036	-0.099363	-0.685613	0.615145	0.058498	-0.693063	...	-0.162656	-0.430978	-0.016826	0.520999	-0.247077	0.040631	0.2
81618	59016.0	-0.754173	0.630477	1.501071	0.340628	0.283952	0.113798	1.030670	0.134642	-0.605788	...	0.132816	0.391467	0.019666	0.224513	-0.025233	-0.544298	0.1
119629	75524.0	1.356796	0.394406	-0.818778	0.586709	0.922007	0.115650	0.399982	-0.088114	-0.394345	...	-0.061278	-0.152709	-0.368772	-1.382826	1.047681	-0.206090	-0.0
223760	143508.0	-0.398991	0.627150	1.899792	1.143005	-0.958310	0.905860	-1.561209	-2.265146	-0.186177	...	-1.227650	0.379234	-0.257101	0.012453	1.175474	-0.380277	0.0

5 rows × 31 columns

In [43]:
```python
new_dataset.tail()
```

Out[43]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	
279863	169142.0	-1.927883	1.125653	-4.518331	1.749293	-1.566487	-2.010494	-0.882850	0.697211	-2.064945	...	0.778584	-0.319189	0.639419	-0.294885	0.537503	0.788395	0.29
280143	169347.0	1.378559	1.289381	-5.004247	1.411850	0.442581	-1.326536	-1.413170	0.248525	-1.127396	...	0.370612	0.028234	-0.145640	-0.081049	0.521875	0.739467	0.38
280149	169351.0	-0.676143	1.126366	-2.213700	0.468308	-1.120541	-0.003346	-2.234739	1.210158	-0.652250	...	0.751826	0.834108	0.190944	0.032070	-0.739695	0.471111	0.38
281144	169966.0	-3.113832	0.585864	-5.399730	1.817092	-0.840618	-2.943548	-2.208002	1.058733	-1.632333	...	0.583276	-0.269209	-0.456108	-0.183659	-0.328168	0.606116	0.88
281674	170348.0	1.991976	0.158476	-2.583441	0.408670	1.151147	-0.096695	0.223050	-0.068384	0.577829	...	-0.164350	-0.295135	-0.072173	-0.450261	0.313267	-0.289617	0.00

5 rows × 31 columns

In [44]:
```python
new_dataset.groupby('Class').mean()
```

Out[44]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	V21	V22	V23	V24	V25	
Class																		
0	94738.997992	0.016676	0.000781	0.053246	-0.133896	-0.102939	-0.078752	0.020556	-0.038796	0.007250	...	0.018323	-0.030734	-0.021125	0.008720	0.032289	0.038756	-0
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.568731	0.570636	-2.581123	...	0.372319	0.713588	0.014049	-0.040308	-0.105130	0.041449	0

2 rows × 30 columns

In [47]:
```python
#Splitting the data into Features & Targets
X=new_dataset.drop(columns='Class',axis=1)
y=new_dataset['Class']
```

In [48]:
```python
print(X)
```

```
            Time        V1        V2        V3        V4        V5        V6  \
11297    19645.0  1.090666  0.336109 -0.072621  1.384967  0.060575 -1.096974   
90804    63175.0 -0.462649  1.025205  1.677640 -0.104036 -0.099363 -0.685613   
81618    59016.0 -0.754173  0.630477  1.501071  0.340628  0.283952  0.113798   
119629   75524.0  1.356796  0.394406 -0.818778  0.586709  0.922007  0.115650   
223760  143508.0 -0.398991  0.627150  1.899792  1.143005 -0.958310  0.905860   
...          ...       ...       ...       ...       ...       ...       ...   
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  \
11297   0.623667 -0.489797  0.919411  ...  0.034026 -0.082523 -0.131621   
90804   0.615145  0.058498 -0.693063  ...  0.105801 -0.162656 -0.430978   
81618   1.030670  0.134642 -0.605788  ...  0.001318  0.132816  0.391467   
119629  0.399982 -0.088114 -0.394345  ... -0.100168 -0.061278 -0.152709   
223760 -1.561209 -2.265146 -0.186177  ...  0.683994 -1.227650  0.379234   
...          ...       ...       ...  ...       ...       ...       ...   
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  
11297  -0.226813  0.358635  0.810954 -0.346882 -0.050001  0.022698   99.99  
90804  -0.016826  0.520999 -0.247077  0.040631  0.263694  0.112968    0.89  
81618   0.019666  0.224513 -0.025233 -0.544298  0.119455  0.136135   85.00  
119629 -0.368772 -1.382826  1.047681 -0.206090 -0.015007 -0.017213    1.00  
223760 -0.257101  0.012453  1.175474 -0.380277  0.074112  0.190663   45.00  
...          ...       ...       ...       ...       ...       ...     ...  
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  

[990 rows x 30 columns]
```

In [49]:
```python
print(y)
```

```
11297     0
90804     0
81618     0
119629    0
223760    0
         ..
279863    1
280143    1
280149    1
281144    1
281674    1
Name: Class, Length: 990, dtype: int64
```

In [50]:
```python
#training and test data
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=2,stratify=y)
print("shape of X_train= ",X_train.shape)
print("shape of X_test= ",X_test.shape)
print("shape of y_train= ",y_train.shape)
print("shape of y_test= ",y_test.shape)
```

```
shape of X_train=  (792, 30)
shape of X_test=  (198, 30)
shape of y_train=  (792,)
shape of y_test=  (198,)
```

In [51]:
```python
#model trainig
model=LogisticRegression()
```

In [52]:
```python
model.fit(X_train,y_train)
```

Out[52]:
```
LogisticRegression()
```

In [53]:
```python
#model evalution
#checking  the accuracy score of traing and testing data
X_train_prediction=model.predict(X_train)
training_data_accuracy=accuracy_score(X_train_prediction,y_train)
print('Accuracy on training data:',training_data_accuracy*100)
```

```
Accuracy on training data: 92.67676767676768
```

In [54]:
```python
X_test_prediction=model.predict(X_test)
testing_data_accuracy=accuracy_score(X_test_prediction,y_test)
print('Accuracy on test data:',testing_data_accuracy*100)
```

```
Accuracy on test data: 90.40404040404042
```

In [64]:
```python
#Making a Predictive system
#changing the input _data to a numpy array
input_data1=(0,-1.3598071336738,-0.0727811733098497,2.53634673796914,1.37815522427443,-0.338320769942518,0.462387777762292,0.239598554061257,0.098697
input_data_numpy_array1=np.asarray(input_data1)
```

In [65]:
```python
#reshape the np array as we are predicting for one instance
input_data_reshaped1=input_data_numpy_array1.reshape(1,-1)
input_data_reshaped1
```

Out[65]:
```
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]])
```

In [66]:
```python
prediction=model.predict(input_data_reshaped1)
print(prediction)
```

```
[0]
C:\Users\Pranav\Searches\hjhkh\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but LogisticRegression was fi
tted with feature names
  warnings.warn(
```

In [67]:
```python
if(prediction[0]==0):
    print('The Transaction is  legit')
else:
    print('The The Transaction is r is fraudulent')
```

```
The Transaction is  legit
```

In [ ]: