In [2]:

```
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 [3]:

```
card_data=pd.read_csv("C:\\Users\\racha\\OneDrive\\Desktop\\Datascience\\credit card fra
```

In [4]:

```
card_data.head(10)
```

Out[4]:

	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	2.0	-0.425966	0.960523	1.141109	-0.168252	0.420987	-0.029728	0.476201	0.260314
6	4.0	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.005159	0.081213
7	7.0	-0.644269	1.417964	1.074380	-0.492199	0.948934	0.428118	1.120631	-3.807864
8	7.0	-0.894286	0.286157	-0.113192	-0.271526	2.669599	3.721818	0.370145	0.851084
9	9.0	-0.338262	1.119593	1.044367	-0.222187	0.499361	-0.246761	0.651583	0.069539

10 rows × 31 columns

In [5]:

```
card_data.tail()
```

Out[5]:

	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 [6]:

```
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 [7]:

```
card_data.isnull().sum()
```

Out[7]:

```
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 [8]:

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

Out[8]:

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

In [9]:

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

In [10]:

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

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

In [11]:

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

Out[11]:

```
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 [12]:

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

Out[12]:

```
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 [13]:

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

Out[13]:

Class	Time	V1	V2	V3	V4	V5	V6	V
0	94838.202258	0.008258	-0.006271	0.012171	-0.007860	0.005453	0.002419	0.0096
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.5687

2 rows × 30 columns

In [14]:

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

In [19]:

```
new_set=pd.concat([l_sample,fraud],axis=0)
```

In [20]:

```
new_set.head()
```

Out[20]:

	Time	V1	V2	V3	V4	V5	V6	V7
25091	33514.0	-0.144788	0.163396	1.093703	-1.725958	0.179990	0.485831	-0.026689
26188	33945.0	-1.890667	2.034542	0.715240	-0.172394	-0.557209	-1.323174	0.535451
209864	137746.0	0.524951	0.707794	-0.095986	0.988461	0.064195	-0.797652	0.824637
106503	69991.0	-0.973682	-0.446373	2.163724	0.733064	0.096672	-0.108364	-0.557794
162	103.0	-0.940893	1.074155	1.759398	-0.601446	0.101693	-0.188520	0.455756

5 rows × 31 columns

In [21]:

```
new_set.tail()
```

Out[21]:

	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
281144	169966.0	-3.113832	0.585864	-5.399730	1.817092	-0.840618	-2.943548	-2.208002
281674	170348.0	1.991976	0.158476	-2.583441	0.408670	1.151147	-0.096695	0.223050

5 rows × 31 columns

In [22]:

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

Out[22]:

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

In [23]:

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

Out[23]:

Class	Time	V1	V2	V3	V4	V5	V6	V7
0	92199.058943	-0.088916	0.088548	0.015316	0.052784	0.034482	-0.090112	-0.05108
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.56873

2 rows × 30 columns

In [24]:

```
x=new_set.drop(columns='Class',axis=1)
y=new_set['Class']
```

In [25]:

```
print(x)
```

```
            Time        V1        V2        V3        V4        V5
V6  \
25091    33514.0 -0.144788  0.163396  1.093703 -1.725958  0.179990  0.4858
31
26188    33945.0 -1.890667  2.034542  0.715240 -0.172394 -0.557209 -1.3231
74
209864  137746.0  0.524951  0.707794 -0.095986  0.988461  0.064195 -0.7976
52
106503   69991.0 -0.973682 -0.446373  2.163724  0.733064  0.096672 -0.1083
64
162        103.0 -0.940893  1.074155  1.759398 -0.601446  0.101693 -0.1885
20
...          ...       ...       ...       ...       ...       ...
...
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  \
25091  -0.026689  0.227845 -1.571314  ...  0.125537  0.031578 -0.065322
26188   0.535451  0.026582  1.068794  ...  0.701742 -0.441593 -0.900118
209864  0.824637 -0.379256 -0.636096  ...  0.291999  0.427001  1.401982
106503 -0.557794  0.269943 -1.108954  ... -0.229461 -0.302300 -0.403205
162     0.455756 -3.460682  0.441525  ... -0.209018  2.270069 -0.143518
...          ...       ...       ...  ...       ...       ...       ...
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
25091  -0.185209 -1.177798 -0.107632 -0.374312  0.066538  0.033463    7.95
26188   0.157209  0.588234  0.048159  0.074433  0.724481  0.507292    5.36
209864  0.075510  0.159693 -1.345137  1.019752  0.253024  0.275106   49.50
106503 -0.096337  0.040405  0.314492 -0.099353  0.116410  0.094594   11.50
162     0.153908  0.700927 -0.413235  1.374031 -0.996161 -0.836301    9.99
...          ...       ...       ...       ...       ...       ...     ...
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 [26]:

```
print(y)
```

```
25091     0
26188     0
209864    0
106503    0
162       0
         ..
279863    1
280143    1
280149    1
281144    1
281674    1
Name: Class, Length: 984, dtype: int64
```

In [30]:

```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,stratify=y,random_state=2)
```

In [31]:

```
print(x.shape,xtrain.shape,xtest.shape)
```

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

In [32]:

```
model=LogisticRegression()
```

In [33]:

```
model.fit(xtrain,ytrain)
```

Out[33]:

```
LogisticRegression()
```

In [34]:

```
x_train_prediction=model.predict(xtrain)
training_data_accuracy=accuracy_score(x_train_prediction,ytrain)
```

In [35]:

```
print("Accuracy on training data=",training_data_accuracy)
```

```
Accuracy on training data= 0.9059720457433291
```

In [36]:

```
x_test_prediction=model.predict(xtest)
testing_data_accuracy=accuracy_score(x_test_prediction,ytest)
```

In [37]:

```
print("Accuracy on testing data=",testing_data_accuracy)
```

```
Accuracy on testing data= 0.9187817258883249
```