In [1]:
```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import accuracy_score
```

In [3]:
```python
df = pd.read_csv(r"C:\Users\ho\Desktop\creditcard.csv", encoding='latin1')
df.head()
```

Out[3]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
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	-0.021053	149.62	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	0.014724	2.69	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	-0.059752	378.66	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	0.061458	123.50	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	0.215153	69.99	0

5 rows × 31 columns

In [4]:
```python
df.describe()
```

Out[4]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amou
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+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	284807.0000
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.604066e-16	1.487313e-15	-5.556467e-16	1.213481e-16	-2.406331e-15	...	1.654067e-16	-3.568593e-16	2.578648e-16	4.473266e-15	5.340915e-16	1.683437e-15	-3.660091e-16	-1.227390e-16	88.3496
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	7.257016e-01	6.244603e-01	6.056471e-01	5.212781e-01	4.822270e-01	4.036325e-01	3.300833e-01	250.1201
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+01	-4.480774e+01	-2.836627e+00	-1.029540e+01	-2.604551e+00	-2.256568e+01	-1.543008e+01	0.0000
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	-5.423504e-01	-1.618463e-01	-3.545861e-01	-3.171451e-01	-3.269839e-01	-7.083953e-02	-5.295979e-02	5.6000
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	6.781943e-03	-1.119293e-02	4.097606e-02	1.659350e-02	-5.213911e-02	1.342146e-03	1.124383e-02	22.0000
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	5.285536e-01	1.476421e-01	4.395266e-01	3.507156e-01	2.409522e-01	9.104512e-02	7.827995e-02	77.1650
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+01	2.252841e+01	4.584549e+00	7.519589e+00	3.517346e+00	3.161220e+01	3.384781e+01	25691.1600

8 rows × 31 columns

In [5]:
```python
df.isnull().sum()
```

Out[5]:
```
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 [6]:
```python
df.duplicated().sum()
```

Out[6]:
```
1081
```

In [7]:
```python
df = df.drop_duplicates( keep='first')
df.duplicated().sum()
```

Out[7]:
```
0
```

In [8]:
```python
# disreibution of legit transection and fradulent transection
df['Class'].value_counts()
```

Out[8]:
```
Class
0    283253
1       473
Name: count, dtype: int64
```

In [9]:
```python
#this data set is higly unbalanced
#0 represents legit trans
#1 represents fraud trans
legit =df[df.Class == 0]
fraud =df[df.Class == 1]
print(legit.shape)
print(fraud.shape)
```

```
(283253, 31)
(473, 31)
```

In [10]:
```python
legit.Amount.describe()
```

Out[10]:
```
count    283253.000000
mean         88.413575
std         250.379023
min           0.000000
25%           5.670000
50%          22.000000
75%          77.460000
max       25691.160000
Name: Amount, dtype: float64
```

In [11]:
```python
fraud.Amount.describe()
```

Out[11]:
```
count     473.000000
mean      123.871860
std       260.211041
min         0.000000
25%         1.000000
50%         9.820000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

In [12]:
```python
#Compares the values for both trans
df.groupby('Class').mean()
```

Out[12]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount
Class																					
0	94835.058093	0.013439	-0.009829	0.012853	-0.010440	0.006769	0.001251	0.010447	-0.002448	0.002613	...	-0.000489	-0.00115	-0.000160	0.000360	0.000393	-0.000301	0.000065	0.001409	0.000418	88.413575
1	80450.513742	-4.498280	3.405965	-6.729599	4.472591	-2.957197	-1.432518	-5.175912	0.953255	-2.522124	...	0.405043	0.46655	0.086639	-0.096464	-0.106643	0.040615	0.050456	0.213774	0.078270	123.871860

2 rows × 30 columns

In [13]:
```python
#under sampling
legit_sample = legit.sample(n=492)
new_dataset = pd.concat([legit_sample, fraud], axis=0)
new_dataset.describe()
```

Out[13]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
count	965.000000	965.000000	965.000000	965.000000	965.000000	965.000000	965.000000	965.000000	965.000000	965.000000	...	965.000000	965.000000	965.000000	965.000000	965.000000	965.000000	965.000000	965.000000	965.000000	965.000000
mean	90597.838342	-2.130328	1.637450	-3.341179	2.146328	-1.449719	-0.683438	-2.561234	0.422773	-1.217347	...	0.262285	0.068234	-0.048649	-0.066775	0.010485	0.018022	0.104398	0.049485	104.535834	0.490155
std	48798.850609	5.333961	3.537586	5.952418	3.208984	4.071912	1.704932	5.504779	4.024009	2.310665	...	1.999010	0.965123	1.239507	0.562323	0.697709	0.474647	0.917176	0.440567	225.918085	0.500162
min	14.000000	-30.552380	-13.207144	-31.103685	-3.659864	-22.105532	-6.406267	-43.557242	-41.044261	-13.434066	...	-22.797604	-8.887017	-19.254328	-2.346415	-4.781606	-1.253686	-7.263482	-1.869290	0.000000	0.000000
25%	47545.000000	-2.589617	-0.255776	-4.850575	-0.204131	-1.627859	-1.555962	-2.935856	-0.218795	-2.219570	...	-0.156435	-0.512626	-0.247452	-0.395924	-0.317402	-0.298771	-0.069485	-0.057621	1.520000	0.000000
50%	85573.000000	-0.695222	0.913677	-1.367020	1.266109	-0.444313	-0.639268	-0.661110	0.150759	-0.735756	...	0.149896	0.089320	-0.031228	-0.009181	0.044146	-0.032941	0.045917	0.032557	19.590000	0.000000
75%	138489.000000	1.125336	2.660670	0.272952	4.161802	0.450145	0.095951	0.268049	0.831040	0.248835	...	0.641211	0.601514	0.190944	0.331862	0.392847	0.321985	0.428610	0.208830	99.990000	1.000000
max	172227.000000	2.368114	22.057729	3.246135	12.114672	11.095089	6.474115	7.585578	20.007208	7.088803	...	27.202839	8.361985	5.933244	1.188232	2.327538	2.745261	3.748396	5.050808	2125.870000	1.000000

8 rows × 31 columns

In [14]:
```python
new_dataset['Class'].value_counts()
```

Out[14]:
```
Class
0    492
1    473
Name: count, dtype: int64
```

In [15]:
```python
new_dataset.groupby('Class').mean()
```

Out[15]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount
Class																					
0	100353.294715	0.146179	-0.062769	-0.083612	-0.090101	-0.000457	0.036713	-0.047529	-0.087222	0.037042	...	-0.007203	0.065907	0.050540	-0.002680	-0.028448	-0.018481	-0.013159	-0.000753	0.021811	85.946524
1	80450.513742	-4.498280	3.405965	-6.729599	4.472591	-2.957197	-1.432518	-5.175912	0.953255	-2.522124	...	0.405043	0.466550	0.086639	-0.096464	-0.106643	0.040615	0.050456	0.213774	0.078270	123.871860

2 rows × 30 columns

In [16]:
```python
X1= new_dataset.drop(columns='Class', axis=1)
print(X1)
```

```
            Time        V1         V2        V3        V4        V5        V6  \
190924  129033.0 -8.950470 -13.207144 -1.913437  1.742184  7.956615 -3.972827
118178   74977.0  1.049017  -1.047323  0.456481 -0.797563 -0.605684  1.066689
148630   90089.0 -0.625582   0.829973  1.170429  0.048991  1.112722 -0.375808
53486    46007.0  1.170878   0.239949  0.561753  1.525445 -0.502140 -0.834816
231843  146949.0 -0.029280   0.690416  0.316517 -0.670210  0.340164 -0.799111
...          ...       ...        ...       ...       ...       ...       ...
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  \
190924 -5.192870  1.608447  1.196896  ...  3.870355  1.199890 -1.152893
118178 -0.976916  0.487551 -0.525511  ...  0.028638  0.146858  0.215007
148630  0.759440 -0.209708  0.583296  ...  0.158623 -0.439370 -1.041486
53486   0.160931 -0.175112  0.275127  ... -0.155781 -0.103878 -0.116620
231843  0.809005  0.035165 -0.111515  ... -0.110569 -0.209555 -0.528980
...          ...       ...       ...  ...       ...       ...       ...
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
190924  1.302578 -0.871068 -2.702872 -0.134537  0.373359 -1.550573  324.66
118178  0.168288 -1.012736 -0.144023 -0.286922  0.065266  0.021481   93.00
148630 -0.287726 -0.750465  0.427674  0.508354 -0.046429  0.045093    9.99
53486  -0.057312  0.732144  0.689046 -0.363414  0.022284  0.023974   15.68
231843  0.100562 -0.025596 -0.588442  0.125053  0.236376  0.085207    6.45
...          ...       ...       ...       ...       ...       ...     ...
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

[965 rows x 30 columns]
```

In [17]:
```python
y1 = new_dataset['Class']
print(y1)
```

```
190924    0
118178    0
148630    0
53486     0
231843    0
         ..
279863    1
280143    1
280149    1
281144    1
281674    1
Name: Class, Length: 965, dtype: int64
```

In [18]:
```python
# MODEL 1
X1_train,X1_test,y1_train,y1_test=train_test_split(X1,y1,test_size=0.3,random_state=123)
print(X1.shape,X1_train.shape , X1_test.shape)
```

```
(965, 30) (675, 30) (290, 30)
```

In [19]:
```python
lr=LogisticRegression()
model2=lr.fit(X1_train,y1_train)
prediction2=model2.predict(X1_test)
accuracy_score(y1_test,prediction2)
```

```
C:\Users\ho\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:460: 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[19]:
```
0.9620689655172414
```

In [22]:
```python
# MODEL 2
from sklearn.model_selection import train_test_split
X2= new_dataset.drop(columns='Class', axis=1)
y2 = new_dataset['Class']
X2_train,X2_test,y2_train,y2_test=train_test_split(X2,y2,test_size=0.30,random_state=123)
```

In [23]:
```python
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
model=rfc.fit(X2_train,y2_train)
prediction=model.predict(X2_test)
accuracy_score(y2_test,prediction)
```

Out[23]:
```
0.9482758620689655
```

Hence, Both of the models are performing with good accuracy

In [ ]: