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 [4]:
```
credit_card_data = pd.read_csv('creditcard.csv')
credit_card_data.head()
```

Out[4]:

	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 [6]:
```
credit_card_data.sample()
```

Out[6]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
84047	60140.0	1.057529	-0.127356	0.900452	1.577774	-0.565966	0.298478	-0.332411	0.156855	0.846328	...	-0.064373	0.122969	-0.100207	0.115336	0.60709	-0.262013	0.067994	0.0285	40.0	0

1 rows × 31 columns

In [8]:
```
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 [12]:
```
credit_card_data['Class'].value_counts()
```

Out[12]:
```
Class
0    284315
1       492
Name: count, dtype: int64
```

In [14]:
```
legit = credit_card_data[credit_card_data.Class==0]
fraud = credit_card_data[credit_card_data['Class']==1]
```

In [16]:
```
fraud['Class']
```

Out[16]:
```
541       1
623       1
4920      1
6108      1
6329      1
         ..
279863    1
280143    1
280149    1
281144    1
281674    1
Name: Class, Length: 492, dtype: int64
```

In [18]:
```
legit.Amount.describe()
```

Out[18]:
```
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 [20]:
```
fraud.Amount.describe()
```

Out[20]:
```
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 [22]:
```
credit_card_data.groupby('Class').mean()
```

Out[22]:

Class	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount
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	-0.000089	-0.000295	-0.000131	88.291022
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	0.051648	0.170575	0.075667	122.211321

2 rows × 30 columns

In [24]:
```
legit_sample = legit.sample(n=492)
```

In [26]:
```
Df = pd.concat([legit_sample,fraud],axis=0)
```

In [28]:
```
Df
```

Out[28]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
119522	75481.0	-0.410341	1.118510	0.488614	-0.037651	0.176083	-0.349123	0.449458	0.411255	-1.116376	...	0.178869	0.434948	-0.014137	0.069851	-0.272335	0.270254	-0.105427	-0.045919	10.62	0
202192	134210.0	2.077038	-0.014961	-2.100799	0.119556	0.582078	-1.121689	0.593713	-0.369255	-0.131720	...	0.101643	0.297303	-0.042122	-0.446975	0.242132	0.625305	-0.127526	-0.093778	21.74	0
257602	158249.0	-0.794127	-1.151332	-2.608456	0.315455	-1.217514	-0.291114	2.212935	-0.128182	-2.527802	...	0.912803	1.402599	1.276140	-0.432939	-0.719215	0.301387	-0.044643	0.199628	610.98	0
199451	132974.0	0.356572	-0.134729	-1.499157	-4.488220	0.254756	-0.832070	0.357918	-2.087665	0.458446	...	1.263518	-0.644885	-0.191495	0.021998	1.009419	-0.897916	0.311652	0.277715	59.99	0
191456	129264.0	-0.128263	-0.065892	-0.873523	-0.722465	1.502633	-0.410419	0.670331	-0.043873	0.072201	...	0.005265	0.084429	0.634069	0.083833	-1.968686	-0.104937	0.176572	0.367159	21.66	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
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	0.292680	0.147968	390.00	1
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	0.389152	0.186637	0.76	1
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	0.385107	0.194361	77.89	1
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	0.884876	-0.253700	245.00	1
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	0.002988	-0.015309	42.53	1

984 rows × 31 columns

In [30]:
```
Df['Class'].value_counts()
```

Out[30]:
```
Class
0    492
1    492
Name: count, dtype: int64
```

In [32]:
```
Df.groupby('Class').mean()
```

Out[32]:

Class	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount
0	96664.479675	-0.032894	-0.059021	-0.027920	-0.030430	-0.076448	0.001536	0.104060	0.023887	0.038025	...	0.055565	-0.043044	-0.002856	-0.051717	-0.022704	-0.026194	0.000657	0.000499	0.005631	116.613394
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	0.051648	0.170575	0.075667	122.211321

2 rows × 30 columns

In [34]:
```
X = Df.drop(columns='Class', axis=1)
y = Df['Class']
```

In [36]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=2)
```

In [40]:
```
model=LogisticRegression()
model.fit(X_train, y_train)
```

```
C:\Users\Admin\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:469: 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[40]:
```
LogisticRegression()
```

In [42]:
```
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)
```

```
Accuracy on Training data :  0.9504447268106735
```

In [44]:
```
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, y_test)
print('Accuracy score on Test Data : ', test_data_accuracy)
```

```
Accuracy score on Test Data :  0.9441624365482234
```

In [46]:
```
Df.head()
```

Out[46]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
119522	75481.0	-0.410341	1.118510	0.488614	-0.037651	0.176083	-0.349123	0.449458	0.411255	-1.116376	...	0.178869	0.434948	-0.014137	0.069851	-0.272335	0.270254	-0.105427	-0.045919	10.62	0
202192	134210.0	2.077038	-0.014961	-2.100799	0.119556	0.582078	-1.121689	0.593713	-0.369255	-0.131720	...	0.101643	0.297303	-0.042122	-0.446975	0.242132	0.625305	-0.127526	-0.093778	21.74	0
257602	158249.0	-0.794127	-1.151332	-2.608456	0.315455	-1.217514	-0.291114	2.212935	-0.128182	-2.527802	...	0.912803	1.402599	1.276140	-0.432939	-0.719215	0.301387	-0.044643	0.199628	610.98	0
199451	132974.0	0.356572	-0.134729	-1.499157	-4.488220	0.254756	-0.832070	0.357918	-2.087665	0.458446	...	1.263518	-0.644885	-0.191495	0.021998	1.009419	-0.897916	0.311652	0.277715	59.99	0

191456	129264.0	-0.128263	-0.065892	-0.873523	-0.722465	1.502633	-0.410419	0.670331	-0.043873	0.072201	...	0.005265	0.084429	0.634069	0.083833	-1.968686	-0.104937	0.176572	0.367159	21.66	0

5 rows × 31 columns