In [7]:
```
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 [11]:
```
credit_card_data = pd.read_csv("creditcard.csv")
```

In [12]:
```
credit_card_data.head()
```

Out[12]:

	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 [14]:
```
credit_card_data.tail()
```

Out[14]:

	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 [15]:
```
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 [19]:
```
credit_card_data.isnull().values.any()
```

Out[19]:
```
False
```

In [30]:
```
#credit_card_data.isnull().sum()
```

In [28]:
```
credit_card_data["Class"].value_counts()
```

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

In [34]:
```
legit = credit_card_data[credit_card_data.Class == 0]
fraud = credit_card_data[credit_card_data.Class == 1]
```

In [37]:
```
print(legit)
print(fraud)
```

```
            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]
            Time        V1        V2        V3        V4        V5        V6  \
541        406.0 -2.312227  1.951992 -1.609851  3.997906 -0.522188 -1.426545
623        472.0 -3.043541 -3.157307  1.088463  2.288644  1.359805 -1.064823
4920      4462.0 -2.303350  1.759247 -0.359745  2.330243 -0.821628 -0.075788
6108      6986.0 -4.397974  1.358367 -2.592844  2.679787 -1.128131 -1.706536
6329      7519.0  1.234235  3.019740 -4.304597  4.732795  3.624201 -1.357746
...          ...       ...       ...       ...       ...       ...       ...
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  ...       V21       V22       V23  \
541    -2.537387  1.391657 -2.770089  ...  0.517232 -0.035049 -0.465211
623     0.325574 -0.067794 -0.270953  ...  0.661696  0.435477  1.375966
4920    0.562320 -0.399147 -0.238253  ... -0.294166 -0.932391  0.172726
6108   -3.496197 -0.248778 -0.247768  ...  0.573574  0.176968 -0.436207
6329    1.713445 -0.496358 -1.282858  ... -0.379068 -0.704181 -0.656805
...          ...       ...       ...  ...       ...       ...       ...
279863 -0.882850  0.697211 -2.064945  ...  0.778584 -0.319189  0.639419
280143 -1.413170  0.248525 -1.127396  ...  0.370612  0.028234 -0.145640
280149 -2.234739  1.210158 -0.652250  ...  0.751826  0.834108  0.190944
281144 -2.208002  1.058733 -1.632333  ...  0.583276 -0.269209 -0.456108
281674  0.223050 -0.068384  0.577829  ... -0.164350 -0.295135 -0.072173

             V24       V25       V26       V27       V28  Amount  Class
541     0.320198  0.044519  0.177840  0.261145 -0.143276    0.00      1
623    -0.293803  0.279798 -0.145362 -0.252773  0.035764  529.00      1
4920   -0.087330 -0.156114 -0.542628  0.039566 -0.153029  239.93      1
6108   -0.053502  0.252405 -0.657488 -0.827136  0.849573   59.00      1
6329   -1.632653  1.488901  0.566797 -0.010016  0.146793    1.00      1
...          ...       ...       ...       ...       ...     ...    ...
279863 -0.294885  0.537503  0.788395  0.292680  0.147968  390.00      1
280143 -0.081049  0.521875  0.739467  0.389152  0.186637    0.76      1
280149  0.032070 -0.739695  0.471111  0.385107  0.194361   77.89      1
281144 -0.183659 -0.328168  0.606116  0.884876 -0.253700  245.00      1
281674 -0.450261  0.313267 -0.289617  0.002988 -0.015309   42.53      1

[492 rows x 31 columns]
```

In [38]:
```
legit.Amount.describe()
```

Out[38]:
```
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 [39]:
```
fraud.Amount.describe()
```

Out[39]:
```
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 [40]:
```
credit_card_data.groupby('Class').mean()
```

Out[40]:

Class	Time	V1	V2	V3	V4	V5	V6	V7	
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

```
In [41]: legit_sample = legit.sample(n=492)
```

```
In [43]: new_dataset = pd.concat([legit_sample,fraud], axis = 0)
```

```
In [44]: new_dataset.head()
```

Out[44]:

	Time	V1	V2	V3	V4	V5	V6	V7	
135062	81091.0	-0.887097	1.164267	0.942039	-0.034583	0.342566	0.799504	-0.006237	0.95
138051	82461.0	-0.797883	0.967002	1.268366	0.588408	-0.118438	-0.759572	0.538977	0.05
236438	148812.0	-0.333260	-4.238924	-1.279286	1.439904	-2.160105	-0.070001	0.784924	-0.28
22236	32121.0	1.193147	0.547676	-0.410321	1.318916	0.111914	-0.816554	0.113077	-0.00
24669	33323.0	-0.074199	0.642199	1.226669	0.338897	-0.929419	0.454092	-1.337717	-2.51

5 rows × 31 columns

```
In [45]: new_dataset.tail()
```

Out[45]:

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

```
Out[48]: Class
0    492
1    492
Name: count, dtype: int64
```

```
In [50]: new_dataset.groupby('Class').mean()
```

Out[50]:

	Time	V1	V2	V3	V4	V5	V6	V7	
Class									
0	91548.146341	-0.027461	0.032748	0.132098	-0.007365	-0.004814	-0.011415	0.112772	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 [53]:

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

In [54]:

```
print(x)
print(y)
```

```
              Time        V1        V2        V3        V4        V5        V6  \
135062   81091.0 -0.887097  1.164267  0.942039 -0.034583  0.342566  0.799504
138051   82461.0 -0.797883  0.967002  1.268366  0.588408 -0.118438 -0.759572
236438  148812.0 -0.333260 -4.238924 -1.279286  1.439904 -2.160105 -0.070001
22236    32121.0  1.193147  0.547676 -0.410321  1.318916  0.111914 -0.816554
24669    33323.0 -0.074199  0.642199  1.226669  0.338897 -0.929419  0.454092
...          ...       ...       ...       ...       ...       ...       ...
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  \
135062 -0.006237  0.952666 -0.551464  ... -0.168187 -0.085230 -0.171640
138051  0.538977  0.055627 -0.389907  ... -0.152601  0.230804  0.768534
236438  0.784924 -0.289524  1.419227  ...  2.204352  0.764133 -0.312505
22236   0.113077 -0.005366  0.175901  ... -0.244266 -0.060651 -0.153963
24669  -1.337717 -2.511914 -0.692689  ...  0.905438 -1.387993 -0.078156
...          ...       ...       ...  ...       ...       ...       ...
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
135062  0.208093 -0.697335 -0.506693  0.183887  0.190643  0.023043     1.98
138051 -0.098666  0.406806  0.324934 -0.280054 -0.336502 -0.285134    24.32
236438 -0.775900  0.020156 -0.822810 -0.557458 -0.169477  0.180189  1197.65
22236  -0.072616 -0.108231  0.557839 -0.291703  0.040050  0.050252     1.00
24669  -0.030818  0.053334  0.783078  1.289471 -0.047243  0.163684    28.75
...          ...       ...       ...       ...       ...       ...      ...
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]
135062    0
138051    0
236438    0
22236     0
24669     0
         ..
279863    1
280143    1
280149    1
281144    1
281674    1
Name: Class, Length: 984, dtype: int64
```

In [67]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, stratify
```

In [68]:
```
print(x.shape, x_train.shape, x_test.shape)
```

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

In [69]:
```
model = LogisticRegression()
```

In [70]:
```
model.fit(x_train, y_train)
```

Out[70]:
```
▾ LogisticRegression
LogisticRegression()
```

In [71]:
```
x_train_prediction = model.predict(x_train)
```

In [72]:
```
training_data_accuracy = accuracy_score(x_train_prediction, y_train)
```

In [73]:
```
print('Accuracy on training data:', training_data_accuracy)
```

```
Accuracy on training data: 0.9453621346886912
```

In [74]:
```
x_test_prediction = model.predict(x_test)
```

In [75]:
```
test_data_accuracy = accuracy_score(x_test_prediction, y_test)
```

In [77]:
```
print('Accuracy on test data;', test_data_accuracy)
```

```
Accuracy on test data; 0.9187817258883249
```

In [ ]:
```
# This is how i use Logistic Regression model to predict credit card fraud.
```