```
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
```

```
# loading the dataset to a Pandas DataFrame
credit_card_data = pd.read_csv('/content/creditcard.csv')
```

```
# first 5 rows of the dataset
credit_card_data.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	
0	0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.36
1	0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.25
2	1	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.5
3	1	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.38
4	2	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.8

5 rows × 31 columns

```
credit_card_data.tail()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8
27814	34711	1.443955	-1.052462	-0.141721	-1.564017	-0.966274	-0.333886	-0.777060	0.023616
27815	34711	-0.263364	0.931818	1.193111	-0.507924	0.862019	0.249381	0.815449	-0.090801
27816	34712	0.976345	-1.024867	0.978714	0.639442	-1.413711	0.311635	-0.909035	0.232423
27817	34712	1.464604	-0.437919	-0.018869	-1.057177	-0.154243	0.251215	-0.584866	-0.025483
27818	34	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 31 columns

```
# dataset informations
credit_card_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27819 entries, 0 to 27818
Data columns (total 31 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Time    27819 non-null  int64  
 1   V1      27818 non-null  float64
 2   V2      27818 non-null  float64
 3   V3      27818 non-null  float64
 4   V4      27818 non-null  float64
 5   V5      27818 non-null  float64
 6   V6      27818 non-null  float64
 7   V7      27818 non-null  float64
 8   V8      27818 non-null  float64
 9   V9      27818 non-null  float64
 10  V10     27818 non-null  float64
 11  V11     27818 non-null  float64
```

```
 12  V12     27818 non-null  float64
 13  V13     27818 non-null  float64
 14  V14     27818 non-null  float64
 15  V15     27818 non-null  float64
 16  V16     27818 non-null  float64
 17  V17     27818 non-null  float64
 18  V18     27818 non-null  float64
 19  V19     27818 non-null  float64
 20  V20     27818 non-null  float64
 21  V21     27818 non-null  float64
 22  V22     27818 non-null  float64
 23  V23     27818 non-null  float64
 24  V24     27818 non-null  float64
 25  V25     27818 non-null  float64
 26  V26     27818 non-null  float64
 27  V27     27818 non-null  float64
 28  V28     27818 non-null  float64
 29  Amount  27818 non-null  float64
 30  Class   27818 non-null  float64
dtypes: float64(30), int64(1)
memory usage: 6.6 MB
```

```
# checking the number of missing values in each column
credit_card_data.isnull().sum()
```

	0
Time	0
V1	1
V2	1
V3	1
V4	1
V5	1
V6	1
V7	1
V8	1
V9	1
V10	1
V11	1
V12	1
V13	1
V14	1
V15	1
V16	1
V17	1
V18	1
V19	1
V20	1
V21	1
V22	1
V23	1
V24	1
V25	1
V26	1
V27	1
V28	1
Amount	1
Class	1

dtype: int64

```
# distribution of legit transactions & fraudulent transactions
credit_card_data['Class'].value_counts()
```

Class	count
0.0	27725
1.0	93

dtype: int64

```
# distribution of legit transactions & fraudulent transactions
credit_card_data['Class'].value_counts()
```

Class	count
0.0	27725
1.0	93

dtype: int64

This Dataset is highly unblanced

0 --> Normal Transaction

1 --> fraudulent transaction

```
# separating the data for analysis
legit = credit_card_data[credit_card_data.Class == 0]
fraud = credit_card_data[credit_card_data.Class == 1]
```

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

```
(27725, 31)
(93, 31)
```

```
# statistical measures of the data
legit.Amount.describe()
```

	Amount
count	27725.000000
mean	77.232517
std	219.509762
min	0.000000
25%	6.490000
50%	19.950000
75%	69.320000
max	7879.420000

dtype: float64

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

	Amount
count	93.000000
mean	96.609677
std	259.128010
min	0.000000
25%	1.000000
50%	1.100000
75%	99.990000
max	1809.680000

dtype: float64

```
# compare the values for both transactions
credit_card_data.groupby('Class').mean()
```

	Time	V1	V2	V3	V4	V5	V6	V7	
Class									
0.0	20440.754518	-0.190595	0.129284	0.765200	0.201632	-0.180681	0.092904	-0.099318	0.
1.0	18829.451613	-8.165086	6.134379	-11.690379	6.070066	-5.753486	-2.388962	-7.986805	4.

2 rows × 30 columns

Under-Sampling

Build a sample dataset containing similar distribution of normal transactions and Fraudulent Transactions

Number of Fraudulent Transactions --> 93

```
legit_sample = legit.sample(n=93)
```

Concatenating two DataFrames

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

```
new_dataset.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8
20538	31109	-1.225281	1.394099	1.055144	-0.045803	-0.251273	-0.296868	-0.007193	-0.404626
2341	1882	1.317470	-0.040713	0.032014	-0.374985	-0.181072	-0.370094	-0.158119	-0.025468
22256	32133	1.264251	-0.050536	0.498255	-0.257808	-0.537794	-0.509947	-0.287907	-0.023872
26595	34157	1.176356	-1.116138	0.992765	-0.195482	-1.455736	0.451710	-1.034240	0.235439
20989	31397	-0.734116	1.400137	1.149041	0.230342	-0.535152	-1.317643	0.426698	0.293469

5 rows × 31 columns

```
new_dataset.tail()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8
26802	34256	0.539276	1.554890	-2.066180	3.241617	0.184736	0.028330	-1.515521	0.537035
27362	34521	1.081234	0.416414	0.862919	2.520863	-0.005021	0.563341	-0.123372	0.223122
27627	34634	0.333499	1.699873	-2.596561	3.643945	-0.585068	-0.654659	-2.275789	0.675229
27738	34684	-2.439237	2.591458	-2.840126	1.286244	-1.777016	-1.436139	-2.206056	-2.282725
27749	34687	-0.860827	3.131790	-5.052968	5.420941	-2.494141	-1.811287	-5.479117	1.189472

5 rows × 31 columns

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

	count
Class	
0.0	93
1.0	93

dtype: int64

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

ıe	V1	V2	V3	V4	V5	V6	V7	
6	-0.031798	0.072214	0.995712	-0.040946	-0.275561	-0.027665	-0.132501	-0
3	-8.165086	6.134379	-11.690379	6.070066	-5.753486	-2.388962	-7.986805	4

Splitting the data into Features & Targets

```
X = new_dataset.drop(columns='Class', axis=1)
Y = new_dataset['Class']
```

```
print(X)
```

```
        Time        V1        V2        V3        V4        V5        V6  \
20538  31109 -1.225281  1.394099  1.055144 -0.045803 -0.251273 -0.296868
2341    1882  1.317470 -0.040713  0.032014 -0.374985 -0.181072 -0.370094
22256  32133  1.264251 -0.050536  0.498255 -0.257808 -0.537794 -0.509947
26595  34157  1.176356 -1.116138  0.992765 -0.195482 -1.455736  0.451710
20989  31397 -0.734116  1.400137  1.149041  0.230342 -0.535152 -1.317643
...      ...       ...       ...       ...       ...       ...       ...
26802  34256  0.539276  1.554890 -2.066180  3.241617  0.184736  0.028330
27362  34521  1.081234  0.416414  0.862919  2.520863 -0.005021  0.563341
27627  34634  0.333499  1.699873 -2.596561  3.643945 -0.585068 -0.654659
27738  34684 -2.439237  2.591458 -2.840126  1.286244 -1.777016 -1.436139
27749  34687 -0.860827  3.131790 -5.052968  5.420941 -2.494141 -1.811287

             V7        V8        V9  ...       V20       V21       V22  \
20538 -0.007193 -0.404626 -0.468087  ... -0.416510  0.675851 -0.732189
2341  -0.158119 -0.025468  0.080709  ... -0.049134 -0.116096 -0.354175
22256 -0.287907 -0.023872  0.092298  ... -0.017027 -0.098897 -0.283946
26595 -1.034240  0.235439  0.147806  ... -0.386195 -0.797385 -1.547431
20989  0.426698  0.293469 -0.666237  ...  0.036198 -0.175618 -0.568957
...         ...       ...       ...  ...       ...       ...       ...
26802 -1.515521  0.537035 -1.999846  ...  0.302735  0.371773  0.111955
27362 -0.123372  0.223122 -0.673598  ... -0.165249 -0.159387 -0.305154
27627 -2.275789  0.675229 -2.042416  ...  0.329342  0.469212 -0.144363
27738 -2.206056 -2.282725 -0.292885  ...  0.513530  1.774460 -0.771390
27749 -5.479117  1.189472 -3.908206  ...  1.085760  1.192694  0.090356

            V23       V24       V25       V26       V27       V28  Amount
20538  0.239171  0.038782 -0.297973  0.069550 -0.313003 -0.001099    8.07
2341  -0.064966 -0.442757  0.314108  0.995446 -0.089684 -0.015753    0.77
22256  0.038827  0.065078  0.145656  0.906405 -0.071982 -0.003296    1.84
26595  0.099651  0.028148  0.092036  0.811710 -0.015834  0.011214   62.00
20989  0.125305  0.875873 -0.189152  0.039623  0.120212  0.042372   13.99
...         ...       ...       ...       ...       ...       ...     ...
26802 -0.305225 -1.053835  0.771175  0.240878  0.418435  0.232170   19.02
27362  0.053620  0.011761  0.375146 -0.106299  0.021008  0.010559    1.52
27627 -0.317981 -0.769644  0.807855  0.228164  0.551002  0.305473   18.96
27738  0.065727  0.103916 -0.057578  0.242652 -0.268649 -0.743713  125.30
27749 -0.341881 -0.215924  1.053032  0.271139  1.373300  0.691195   19.02

[186 rows x 30 columns]
```

```
print(Y)
```

```
20538    0.0
2341     0.0
22256    0.0
26595    0.0
20989    0.0
        ...
26802    1.0
27362    1.0
27627    1.0
27738    1.0
27749    1.0
Name: Class, Length: 186, dtype: float64
```

Split the data into Training data & Testing Data

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_
```

```
print(X.shape, X_train.shape, X_test.shape)
```

```
(186, 30) (148, 30) (38, 30)
```

Model Training

Logistic Regression

```
model = LogisticRegression()
```

```
# training the Logistic Regression Model with Training Data
```