Importing all the Dependencies

In [48]:
```
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 [2]:
```
# loading the dataset to a Pandas DataFrame
credit_card_data = pd.read_csv('creditcard.csv')
```

In [3]:
```
# first 5 rows of the dataset
credit_card_data.head()
```

Out[3]:

	Time	V1	V2	V3	V4	V5	V6	V7	'
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.0986
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.0851
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.2476
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.3774
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.2705

5 rows × 31 columns

In [6]:
```
# dataset informations
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 [7]:

```
# checking the number of missing values in each column
credit_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]:
```
# distribution of legit transactions & fraudulent transactions
credit_card_data['Class'].value_counts()
```

Out[8]:
```
Class
0    284315
1       492
Name: count, dtype: int64
```

This Dataset is highly unblanced

0 --> Normal Transaction

1 --> Fraudulent Transaction

In [9]:
```
# separating the data for analysis
legit = credit_card_data[credit_card_data.Class == 0]
fraud = credit_card_data[credit_card_data.Class == 1]
```

In [10]:
```
print(legit.shape)
print(fraud.shape)
```

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

In [11]:
```
# statistical measures of the data
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]:
```
# compare the values for both transactions
credit_card_data.groupby('Class').mean()
```

Out[13]:

	Time	V1	V2	V3	V4	V5	V6	
Class								
0	94838.202258	0.008258	-0.006271	0.012171	-0.007860	0.005453	0.002419	0.009
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.568

2 rows × 30 columns

Under-Sampling:

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

In [34]:
```
# as the number of Fraudulent Transactions is 492 we need 492 samples from legit da
legit_sample = legit.sample(n=492)
```

Concatenating two DataFrames

In [15]:
```python
new_dataset = pd.concat([legit_sample, fraud], axis=0)
```

In [16]:
```python
new_dataset.head()
```

Out[16]:

	Time	V1	V2	V3	V4	V5	V6	V7
283569	171698.0	2.158602	0.750699	-3.464304	0.547413	1.664424	-1.236616	0.87473
114379	73457.0	1.217124	0.408112	0.560617	1.099548	-0.313754	-0.870446	0.18690
251234	155278.0	1.962023	-0.323978	-2.553668	0.359218	2.522439	3.705626	-0.44559
22129	32051.0	1.421877	-1.347204	0.403220	-1.160002	-1.677039	-0.530347	-1.07918
112646	72743.0	1.135110	0.623557	0.783507	2.604152	-0.403401	-1.010331	0.30605

5 rows × 31 columns

In [17]:
```python
new_dataset.tail()
```

Out[17]:

	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 [18]:
```python
new_dataset['Class'].value_counts()
```

Out[18]:
```
Class
0    492
1    492
Name: count, dtype: int64
```

In [19]:
```python
new_dataset.groupby('Class').mean()
```

Out[19]:

Class	Time	V1	V2	V3	V4	V5	V6	V
0	98767.806911	0.083077	0.107013	-0.099756	0.014187	0.070482	-0.003784	-0.00742
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.5687

2 rows × 30 columns

Splitting the data into Features & Targets

In [20]:
```
X = new_dataset.drop(columns='Class', axis=1)
Y = new_dataset['Class']
```

In [37]:
```
X.head()
```

Out[37]:

	Time	V1	V2	V3	V4	V5	V6	V7
283569	171698.0	2.158602	0.750699	-3.464304	0.547413	1.664424	-1.236616	0.874736
114379	73457.0	1.217124	0.408112	0.560617	1.099548	-0.313754	-0.870446	0.186907
251234	155278.0	1.962023	-0.323978	-2.553668	0.359218	2.522439	3.705626	-0.445594
22129	32051.0	1.421877	-1.347204	0.403220	-1.160002	-1.677039	-0.530347	-1.079182
112646	72743.0	1.135110	0.623557	0.783507	2.604152	-0.403401	-1.010331	0.306052

5 rows × 30 columns

In [22]:
```
print(Y)
```

```
283569    0
114379    0
251234    0
22129     0
112646    0
         ..
279863    1
280143    1
280149    1
281144    1
281674    1
Name: Class, Length: 984, dtype: int64
```

Split the data into Training data & Testing Data

In [23]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y
```

In [24]:
```
print(X.shape, X_train.shape, X_test.shape)
```

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

Model Training

Logistic Regression

In [56]:
```
model = LogisticRegression()
```

In [57]:
```
# training the Logistic Regression Model with Training Data
model.fit(X_train, Y_train)
```

Out[57]:
```
▾ LogisticRegression
LogisticRegression()
```

Model Evaluation

Accuracy Score

In [58]:
```
# accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

In [60]:
```
print('Accuracy on Training data : ', training_data_accuracy)
```

```
Accuracy on Training data :  0.9466327827191868
```

In [62]:
```
# accuracy on test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

In [63]:
```
print('Accuracy score on Test Data : ', test_data_accuracy)
```

```
Accuracy score on Test Data :  0.934010152284264
```