CREDIT CARD FRAUD DETECTION

Import Dependencies

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
```

The Dataset

In [3]:
```
credit_card_data=pd.read_csv(r'C:\Users\HP\Downloads\archive (2)\creditcard.csv')
```

In [4]:
```
credit_card_data.head()
```

Out[4]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23
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
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
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
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
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

5 rows × 31 columns

In [5]:
```
credit_card_data.tail()
```

Out[5]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...	0.213454	0.111864	1
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.924384	0
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.578229	-0
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.800049	-0
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057	0.643078	0

5 rows × 31 columns

Dataset Informations

In [6]:
```
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]:
```
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
```

Hence it can be concluded that this given dataset has no null values.

In [8]:
```
credit_card_data.shape
```

Out[8]:
```
(284807, 31)
```

The given dataset has 284807 rows and 31 columns.

Distribution of legit transactions and fraudulent transactions

In [9]:
```
credit_card_data.describe()
```

Out[9]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8
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
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
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
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
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
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
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
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

8 rows × 31 columns

In [10]:
```
credit_card_data['Class'].value_counts()
```

Out[10]:
```
0    284315
1       492
Name: Class, dtype: int64
```

0 ---> Legit transactions 1 ---> Fraudulent transactions

It can be observed that 492 fraud transactions were done using the particular credit card.

This dataset is highly unbalanced.

Seperating data of legit transactions and data of fraudulent transactions for data analysis

In [11]:
```
legit=credit_card_data[credit_card_data.Class==0]
fraud=credit_card_data[credit_card_data.Class==1]
```

In [12]:
```
print(legit.shape)
print(fraud.shape)
```

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

Statistical measures of the data

In [14]:
```
legit.Amount.describe()
```

Out[14]:
```
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 [15]:
```
fraud.Amount.describe()
```

Out[15]:
```
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
```

Compare the values for both transactions

In [16]:
```
credit_card_data.groupby('Class').mean()
```

Out[16]:

Class	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	V21
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
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

2 rows × 30 columns

Under Sampling

Build a sample dataset containing similar distribution of legit transactions and the fraudulent transactions

Number of fraudulent transactions ---> 492

In [17]:
```
legit_sample=legit.sample(n=492)
```

Concatenating two data frames

In [18]:
```
new_dataset =pd.concat([legit_sample,fraud],axis=0)
```

In [19]:
```
new_dataset.head()
```

Out[19]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	
216409	140464.0	-0.700138	2.095846	-2.057056	1.648525	1.139541	-1.327866	1.802385	0.022209	-1.164496	...	0.014722	0.099030	-(
154448	101573.0	2.001199	-0.071216	-0.747014	0.204079	-0.009969	-0.442587	-0.269615	-0.098480	1.578474	...	-0.558587	-1.478569	(
249119	154259.0	2.010719	-1.943531	-1.572516	-1.673460	-0.896737	0.035297	-0.928835	-0.156121	-1.445864	...	0.308526	1.126986	-(
250603	154991.0	-0.514317	1.281833	-0.236068	-0.632398	0.126544	-1.116174	0.636817	0.238145	-0.014978	...	-0.258763	-0.630457	(
90427	63015.0	0.921564	-0.391010	-0.426926	0.256354	-0.332200	-1.330426	0.766585	-0.498255	-0.293433	...	0.087153	-0.156378	-(

5 rows × 31 columns

In [20]:
```
new_dataset.tail()
```

Out[20]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	
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.(
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.
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.
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	-0.4
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.(

5 rows × 31 columns

In [21]:
```
new_dataset['Class'].value_counts()
```

Out[21]:
```
0    492
1    492
Name: Class, dtype: int64
```

In [22]:
```
new_dataset.groupby('Class').mean()
```

Out[22]:

Class	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	V21
0	96836.014228	-0.020909	-0.040966	0.045129	-0.047551	-0.060705	0.017105	0.035887	-0.020976	-0.068252	...	0.010844	-0.029636
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

2 rows × 30 columns

Splitting the data into Features & Targets

In [23]:
```
X=new_dataset.drop(columns='Class',axis=1)
Y=new_dataset['Class']
```

In [25]:
```
print(X)
```

```
            Time        V1        V2        V3        V4        V5        V6  \
216409  140464.0 -0.700138  2.095846 -2.057056  1.648525  1.139541 -1.327866
154448  101573.0  2.001199 -0.071216 -0.747014  0.204079 -0.009969 -0.442587
249119  154259.0  2.010719 -1.943531 -1.572516 -1.673460 -0.896737  0.035297
250603  154991.0 -0.514317  1.281833 -0.236068 -0.632398  0.126544 -1.116174
90427    63015.0  0.921564 -0.391010 -0.426926  0.256354 -0.332200 -1.330426
...          ...       ...       ...       ...       ...       ...       ...
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  \
216409  1.802385  0.022209 -1.164496  ...  0.397503  0.014722  0.099030
154448 -0.269615 -0.098480  1.578474  ... -0.198748 -0.558587 -1.478569
249119 -0.928835 -0.156121 -1.445864  ... -0.036273  0.308526  1.126986
250603  0.636817  0.238145 -0.014978  ... -0.017241 -0.258763 -0.630457
90427   0.766585 -0.498255 -0.293433  ...  0.397107  0.087153 -0.156378
...          ...       ...       ...  ...       ...       ...       ...
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
216409 -0.010420 -0.344222  0.132028 -0.377454  0.325363  0.281977  137.98
154448  0.588439  0.645595 -0.770757 -0.074031 -0.105877 -0.056268   11.99
249119 -0.320766 -1.083708  0.240181  0.285725 -0.021176 -0.050808  181.00
250603  0.108532 -0.120517 -0.394992  0.154329  0.231417  0.088870    3.87
90427  -0.278427  0.500507  0.501105  1.045782 -0.147562  0.028808  215.00
...          ...       ...       ...       ...       ...       ...     ...
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]
```

In [26]:
```
print(Y)
```

```
216409    0
154448    0
249119    0
250603    0
90427     0
         ..
279863    1
280143    1
280149    1
281144    1
281674    1
Name: Class, Length: 984, dtype: int64
```

Splitting the data into Training Data & Testing Data

In [31]:
```
X_train,X_test,Y_train, Y_test = train_test_split(X,Y,test_size=0.2,stratify=Y,random_state=2)
```

In [32]:
```
print(X.shape,X_train.shape,X_test.shape)
```

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

Model Training

Logistic Regression

In [33]:
```
model=LogisticRegression()
```

Training the Logistic Regression Model with Training Data

In [34]:
```
model.fit(X_train,Y_train)
```

Out[34]:
```
▾ LogisticRegression
LogisticRegression()
```

Model Evaluation

Accuracy Score

Accuracy on training data

In [35]:
```
X_train_prediction = model.predict(X_train)
```

In [36]:
```
training_data_accuracy = accuracy_score(X_train_prediction,Y_train)
```

In [37]:
```
print('Accuracy on Training Data:', training_data_accuracy)
```

```
Accuracy on Training Data: 0.9542566709021602
```

Accuracy on test data

In [39]:
```
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: 0.9238578680203046
```

In [ ]:
```

```