TASK-5 CREDIT CARD FRAUD DETECTION

Importing required libraries

First step to build any model is to import neccessary libaries. Below are the libaries that we would use in above task:

- Numpy
- Pandas
- sklearn

```
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 data and preprocessing

After importing the required libraries we load the dataset using **read_csv()** and view the first 5 element of dataset using **head()**. Along with this we will find out null values and remove them if any.

```
data=pd.read_csv('/content/drive/MyDrive/Dataset/creditcard.csv')
```

```
data.head()
```

	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	0.0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.2
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.3
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.2

5 rows × 31 columns

```
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
```

```
#checking null values
data.isnull().sum()
```

```
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
```

As it's a classification problem,so we would investigate the data about diffrent classes present in data and divide the data into diffrent sets for each class.

```
#diffrent classes
data['Class'].value_counts()
```

```
0    284315
1       492
Name: Class, dtype: int64
```

```
#segregting data foreach class
legal_class=data[data.Class==0]
fraud_class=data[data.Class==1]
```

```
print(legal_class.shape)
```

```
(284315, 31)
```

```
print(fraud_class.shape)
```

```
(492, 31)
```

```
legal_class['Amount'].describe()
```

```
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
```

```
fraud_class['Amount'].describe()
```

```
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
```

```
# grouping data by class andgetting mean value
data.groupby('Class').mean()
```

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

2 rows × 30 columns

```
#concating the data
legal_sample=legal_class.sample(n=492)
df=pd.concat([legal_sample,fraud_class],axis=0)
```

```
df.head()
```

	Time	V1	V2	V3	V4	V5	V6	
100785	67622.0	-1.157629	-0.265964	-0.690266	-2.452611	2.330012	2.841980	-0.84
248772	154088.0	-1.151891	1.065933	-0.738016	-0.420535	0.254825	-0.966131	-0.01
275453	166539.0	-0.231390	1.089148	-0.812853	-0.676097	0.728772	-0.234758	0.48
221870	142731.0	-0.130150	0.470468	-1.310311	-1.360620	3.384781	3.540088	0.60
218637	141397.0	2.033832	-0.361866	-1.324293	-0.033820	0.237224	0.101318	-0.28

5 rows × 31 columns

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

```
0    492
1    492
Name: Class, dtype: int64
```

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

Class	Time	V1	V2	V3	V4	V5	V6	
0	101062.644309	0.125889	0.050480	-0.069378	-0.132570	0.052947	0.105487	-0.(
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.5

2 rows × 30 columns

Feature and Target split and Training and Testing split

Before proceeding toward building the model we would seprate our feature variables/attribute and target variable from data. Then, this splited data would be further used for training an testing data split.

```
#seprating target and feature sets
x=df.drop(columns='Class',axis=1)
y=df['Class']
```

```
#spliting traing and testing datasets
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.15,stratify=Y,random_state=2)
print(x_train.shape)
print(x_test.shape)
print(x.shape)
print(y_train.shape)
print(y_test.shape)
print(y.shape)
```

```
(836, 30)
(148, 30)
(984, 30)
(836,)
(148,)
(984,)
```

Building Model

For building the model weload the model and fit our data into it. We then predict the values based on which model would be evaluated.

```
model=LogisticRegression(solver='lbfgs', max_iter=150)
# fitting the model
model.fit(x_train,y_train)
#predicting the values
x_train_prediction=model.predict(x_train)
```

Model Evaluation

To evaluate the model, accuracy ofmodel is calculated, which tells how accurate the model is trained and how accurate the model predict the result.

```
# calculating training accuracy
training_data_accuracy=accuracy_score(x_train_prediction,y_train)
print('Training accuracy:',training_data_accuracy)
```

```
Training accuracy: 0.9485645933014354
```

```
# calculating testing accuracy
x_test_prediction=model.predict(x_test)
testing_data_accuracy=accuracy_score(x_test_prediction,y_test)
print('Testing accuracy:',testing_data_accuracy)
```

```
Testing accuracy: 0.9391891891891891
```