# CREDIT CARD FRAUD DETECTION WITH MACHINE LEARNING

## Phase 3: Development Part 1

## SHORT EXPLANATION:

**Credit card fraud detection is the process of identifying and preventing unauthorized or fraudulent transactions made using credit or debit cards. This is typically done using machine learning and data analysis techniques. The goal is to distinguish between legitimate card transactions and fraudulent ones, protecting cardholders and financial institutions from financial losses. Key methods include monitoring transaction data, analyzing patterns, and using predictive models to flag and block suspicious transactions in real-time. The field of credit card fraud detection is crucial in today's digital economy, where the use of payment cards is widespread and fraudsters continually devise new tactics to exploit vulnerabilities.**

## DATASET EXPLANATION:

**The given dataset:**

**https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud**

**This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions.**

**The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions. It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, … V28V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.**

## DETAILS ABOUT COLUMN:

**The columns that we are going to use are :**

- **Time**

  **Number of seconds elapsed between this transaction and the first transaction in the dataset**

- **Amount**

  **Transaction amount**

- **Class**

  **1 for fraudulent transactions, 0 otherwise.**

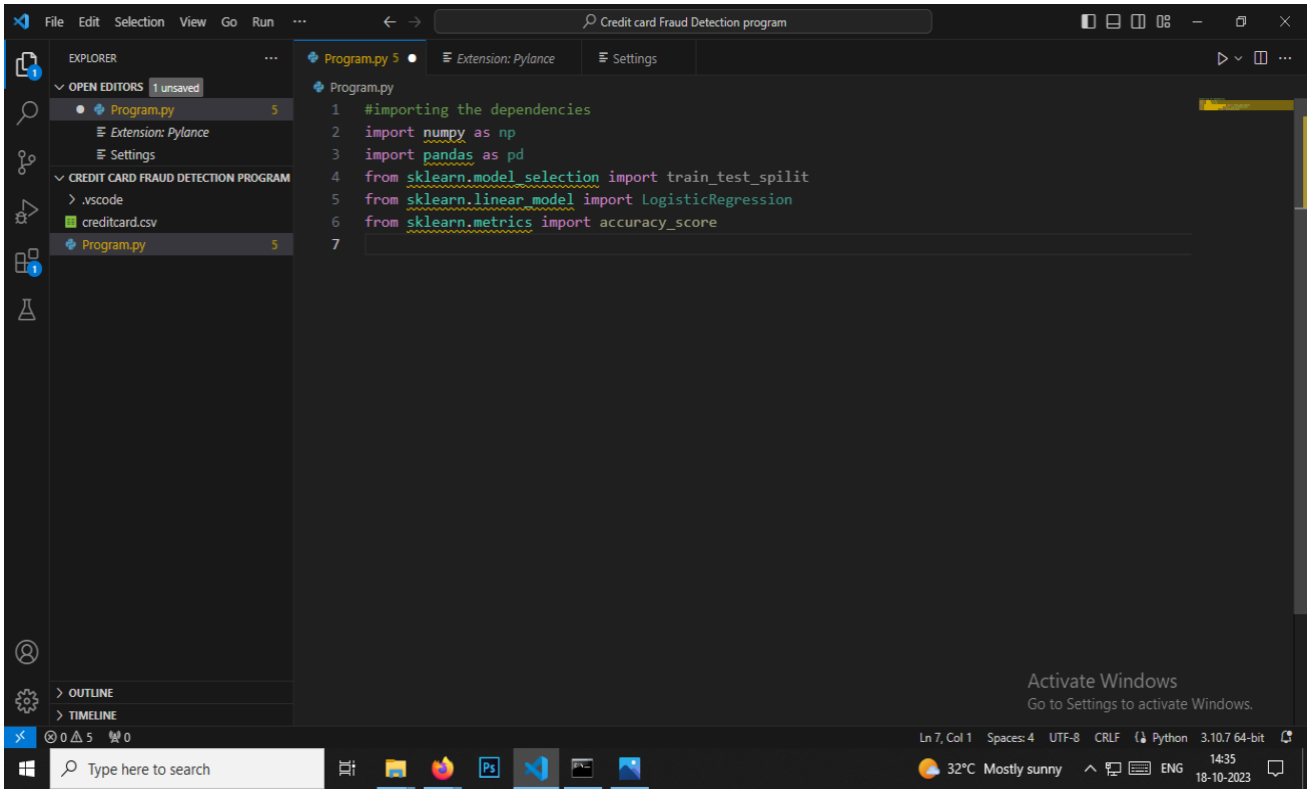
## LIBRARIES TO BE USED AND WAY TO DOWNLOAD:

**The libraries to be used :**

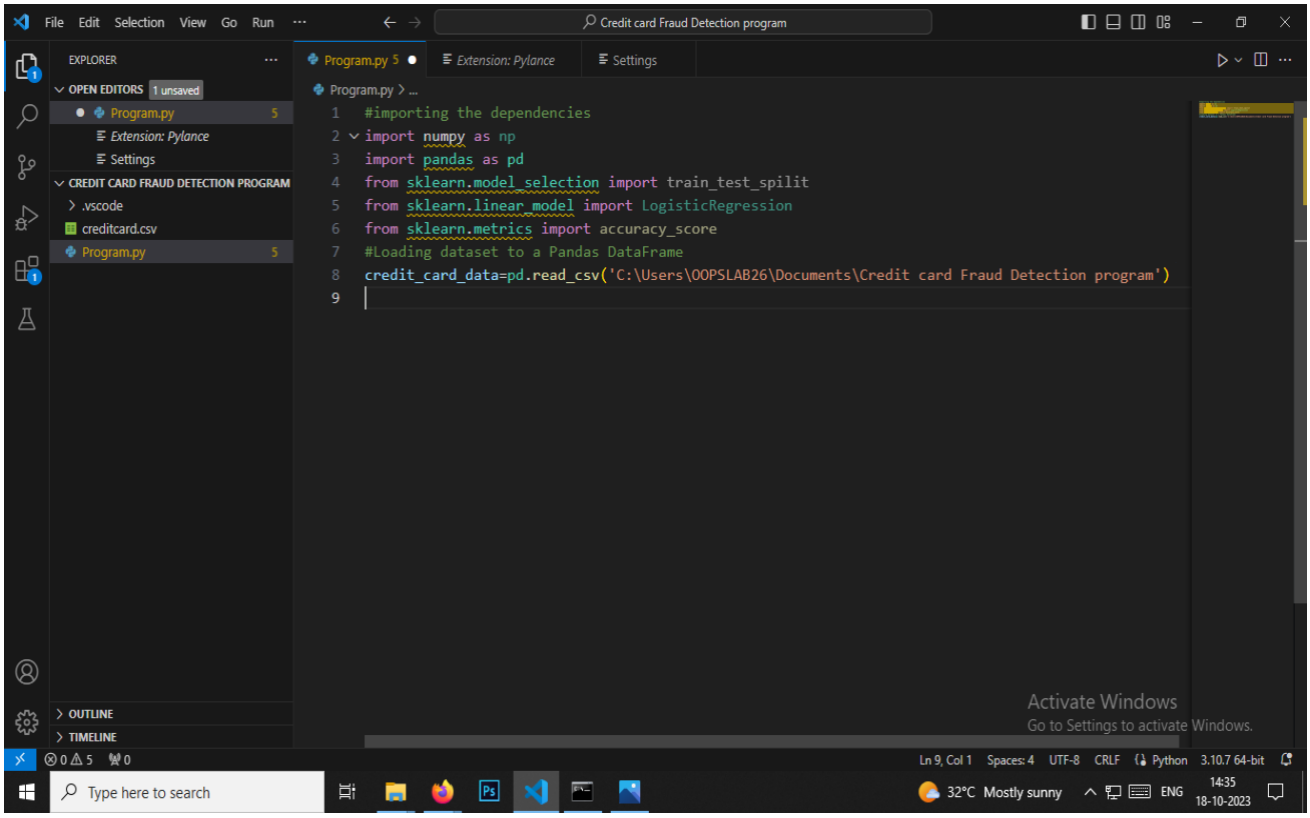
- **numpy**
  - **NumPy is an open-source library in Python that provides support in mathematical, scientific, engineering, and data science programming.**
- **Pandas**
  - **Pandas is an open-source library in Python that is made mainly for working with relational or labeled data both easily and intuitively**
- **Sklearn**
  - **A comprehensive library for machine learning that includes various algorithms for classification, model evaluation, and preprocessing techniques.**
- **Scipy**
  - **Offers a wide range of statistical and optimization functions for data analysis.**
- **Matplotlib**
  - **Used for data visualization, which is essential for data exploration and model performance evaluation.**

**The above mentioned python libraries for developing the ML model can be downloaded easily by pip installing in VS code .**

# #importing the data dependencies:




# #Loading dataset to a Pandas Dataframe:



# #First 5 rows of the dataset:

```python
#importing the dependencies
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_spilit
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
#Loading dataset to a Pandas DataFrame
credit_card_data=pd.read_csv('C:\Users\OOPSLAB26\Documents\Credit card Fraud Detection program')
#first 5 rows of the dataset
credit_card_data.head()
```

# #Dataset information:

```python
#importing the dependencies
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_spilit
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
#Loading dataset to a Pandas DataFrame
credit_card_data=pd.read_csv('C:\Users\OOPSLAB26\Documents\Credit card Fraud Detection program')
#first 5 rows of the dataset
credit_card_data.head()
credit_card_data.tail()
#dataset information
credit_card_data.info()
```

# #Checking the number of missing values in each column:

```python
#importing the dependencies
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_spilit
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
#Loading dataset to a Pandas DataFrame
credit_card_data=pd.read_csv('C:\Users\OOPSLAB26\Documents\Credit card Fraud Detection program')
#first 5 rows of the dataset
credit_card_data.head()
credit_card_data.tail()
#dataset information
credit_card_data.info()
#checking the number of missing values in each column
credit_card_data.isnull().sum()
```

# #Distribution of  legit transactions & fradulent transactions:

```python
#importing the dependencies
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_spilit
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
#Loading dataset to a Pandas DataFrame
credit_card_data=pd.read_csv('C:\Users\OOPSLAB26\Documents\Credit card Fraud Detection program')
#first 5 rows of the dataset
credit_card_data.head()
credit_card_data.tail()
#dataset information
credit_card_data.info()
#checking the number of missing values in each column
credit_card_data.isnull().sum()
# distribution of legit transactions & fradulent transactions
credit_card_data['Class'].value_counts()
```

# #This dataset is highly unbalanced:

```python
import pandas as pd
from sklearn.model_selection import train_test_spilit
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
#Loading dataset to a Pandas DataFrame
credit_card_data=pd.read_csv('C:\Users\OOPSLAB26\Documents\Credit card Fraud Detection program')
#first 5 rows of the dataset
credit_card_data.head()
credit_card_data.tail()
#dataset information
credit_card_data.info()
#checking the number of missing values in each column
credit_card_data.isnull().sum()
# distribution of legit transactions & fradulent transactions
credit_card_data['Class'].value_counts()
#This dataset is highly unbalanced
0 --> Normal Transaction
1 -->fraudanlent 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)
#statistical measures of the data
legit.Amount.describe()
fraud .Amount.describe()
```

# #Separating  the data for Analysis:

```python
#first 5 rows of the dataset
credit_card_data.head()
credit_card_data.tail()
#dataset information
credit_card_data.info()
#checking the number of missing values in each column
credit_card_data.isnull().sum()
# distribution of legit transactions & fradulent transactions
credit_card_data['Class'].value_counts()
#This dataset is highly unbalanced
0 --> Normal Transaction
1 -->fraudanlent 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)
#statistical measures of the data
legit.Amount.describe()
fraud .Amount.describe()
#compare the values for both transactions
credit_card_data.groupby('class').mean()
#Under sampling
#Build a sample dataset containing similar distribution of normal transactions and fraudulent Trans
#Number of Fraudulent Transaction-->492
legit_sample=legit.sample(n=492)
```

# #Statistical measures of the data:

```python
#dataset information
credit_card_data.info()
#checking the number of missing values in each column
credit_card_data.isnull().sum()
# distribution of legit transactions & fradulent transactions
credit_card_data['Class'].value_counts()
#This dataset is highly unbalanced
0 --> Normal Transaction
1 -->fraudanlent 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)
#statistical measures of the data
legit.Amount.describe()
fraud .Amount.describe()
#compare the values for both transactions
credit_card_data.groupby('class').mean()
#Under sampling
#Build a sample dataset containing similar distribution of normal transactions and fraudulent Trans
#Number of Fraudulent Transaction-->492
legit_sample=legit.sample(n=492)
#Concatenating two Dataframes
new_dataset =pd.concat([legit_sample,fraud],axis=1)
new_dataset.head()
```

# #Compare the values for both transactions:

```python
credit_card_data.info()
#checking the number of missing values in each column
credit_card_data.isnull().sum()
# distribution of legit transactions & fradulent transactions
credit_card_data['Class'].value_counts()
#This dataset is highly unbalanced
0 --> Normal Transaction
1 -->fraudanlent 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)
#statistical measures of the data
legit.Amount.describe()
fraud .Amount.describe()
#compare the values for both transactions
credit_card_data.groupby('class').mean()
#Under sampling
#Build a sample dataset containing similar distribution of normal transactions and fraudulent Trans
#Number of Fraudulent Transaction-->492
legit_sample=legit.sample(n=492)
#Concatenating two Dataframes
new_dataset =pd.concat([legit_sample,fraud],axis=1)
new_dataset.head()
new_dataset.tail()
```

**#Under sampling build a sample dataset containing similar distribution of normal transactions and fraudulent transactions.**

**#Number of Fraudulent transactions→492.**

```python
0 --> Normal Transaction
1 -->fraudanlent 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)
#statistical measures of the data
legit.Amount.describe()
fraud .Amount.describe()
#compare the values for both transactions
credit_card_data.groupby('class').mean()
#Under sampling
#Build a sample dataset containing similar distribution of normal transactions and fraudulent Trans
#Number of Fraudulent Transaction-->492
legit_sample=legit.sample(n=492)
#Concatenating two Dataframes
new_dataset =pd.concat([legit_sample,fraud],axis=1)
new_dataset.head()
new_dataset.tail()
new_dataset['class'].value_counts()
new_dataset.groupby('class').mean()
#splitting  the data into Feature & target
X=new_dataset.drop(columns='class',axis=1)
Y=new_dataset['class']
print(x)
```

**#Concatenating two Dataframes:**

```python
fraud= credit_card_data[credit_card_data.class==1]
print(legit.shape)
print(fraud.shape)
#statistical measures of the data
legit.Amount.describe()
fraud .Amount.describe()
#compare the values for both transactions
credit_card_data.groupby('class').mean()
#Under sampling
#Build a sample dataset containing similar distribution of normal transactions and fraudulent Trans
#Number of Fraudulent Transaction-->492
legit_sample=legit.sample(n=492)
#Concatenating two Dataframes
new_dataset =pd.concat([legit_sample,fraud],axis=1)
new_dataset.head()
new_dataset.tail()
new_dataset['class'].value_counts()
new_dataset.groupby('class').mean()
#splitting  the data into Feature & target
X=new_dataset.drop(columns='class',axis=1)
Y=new_dataset['class']
print(X)
print(Y)
#split the data into Training&Testing data
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,stratify=Y,random_state=2)
print(X.shape,X_train.shape,X_test.shape)
```

# #Splitting the data into feature & target:

```python
fraud .Amount.describe()
#compare the values for both transactions
credit_card_data.groupby('class').mean()
#Under sampling
#Build a sample dataset containing similar distribution of normal transactions and fraudulent Tran
#Number of Fraudulent Transaction-->492
legit_sample=legit.sample(n=492)
#Concatenating two Dataframes
new_dataset =pd.concat([legit_sample,fraud],axis=1)
new_dataset.head()
new_dataset.tail()
new_dataset['class'].value_counts()
new_dataset.groupby('class').mean()
#splitting  the data into Feature & target
X=new_dataset.drop(columns='class',axis=1)
Y=new_dataset['class']
print(X)
print(Y)
#split the data into Training&Testing data
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,stratify=Y,random_state=2)
print(X.shape,X_train.shape,X_test.shape)
#Model Training
#logistic Regression
model=LogisticRegression()
#training the logistic Regression Model with Training data
model.fit(X_train,Y_train)
```

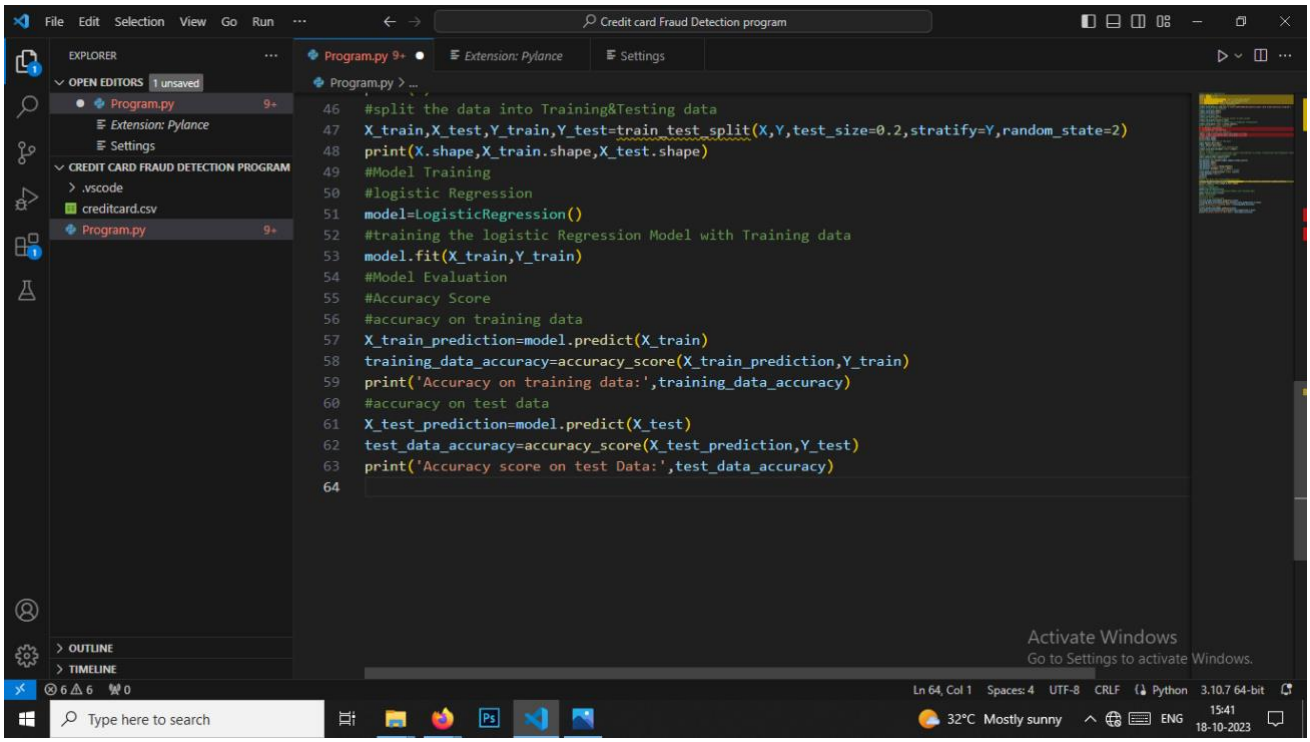
# #Spilit the data into Training&Testing Data:

# #Model Training and Logistic Regression:

```python
#split the data into Training&Testing data
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,stratify=Y,random_state=2)
print(X.shape,X_train.shape,X_test.shape)
#Model Training
#logistic Regression
model=LogisticRegression()
#training the logistic Regression Model with Training data
model.fit(X_train,Y_train)
#Model Evaluation
#Accuracy Score
#accuracy on training data
X_train_prediction=model.predict(X_train)
training_data_accuracy=accuracy_score(X_train_prediction,Y_train)
print('Accuracy on training data:',training_data_accuracy)
```

# #Model Evaluation & Accuracy Score:

# #Accuracy on Training data & Testing data:




| Training ML model | Testing ML model |
|---|---|
| • Data preparation | • Testing set |
| • Data splitting | • Model Evaluation |
| • Model building | • Performance Metrics |
| • Model Training | |