

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, ... V28 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**

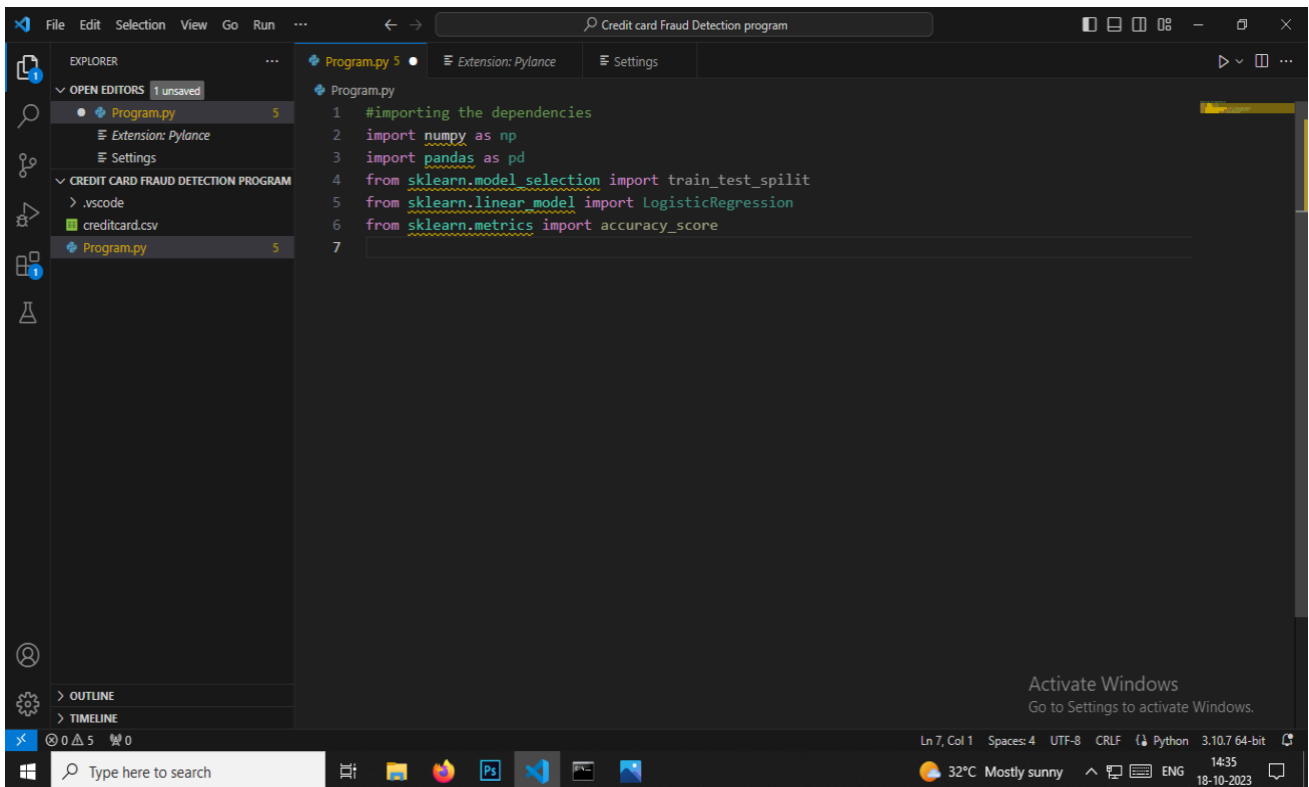
- Offers a wide range of statistical and optimization functions for data analysis.

- **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:

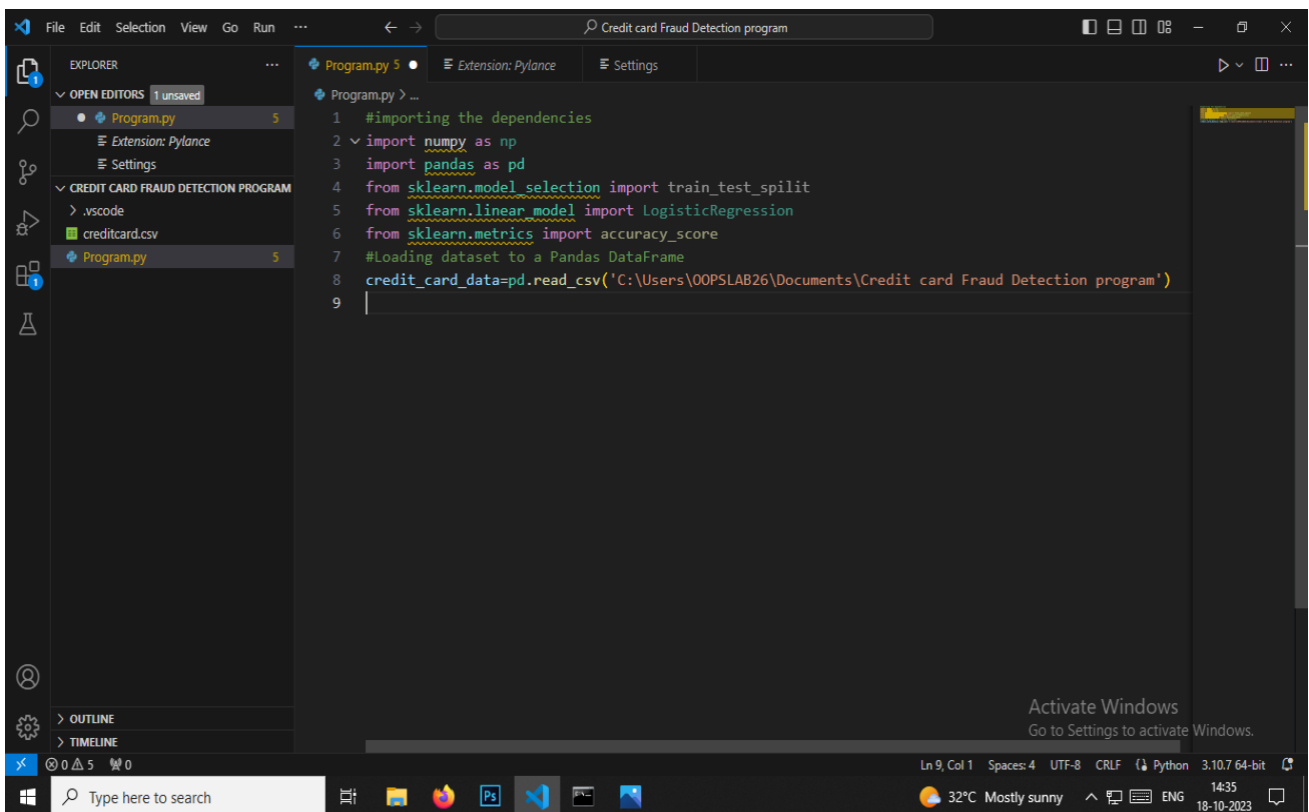


The screenshot shows the Visual Studio Code interface with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'CREDIT CARD FRAUD DETECTION PROGRAM' with files '.vscode', 'creditcard.csv', and 'Program.py'. The code editor shows the following Python code:

```
1 #importing the dependencies
2 import numpy as np
3 import pandas as pd
4 from sklearn.model_selection import train_test_split
5 from sklearn.linear_model import LogisticRegression
6 from sklearn.metrics import accuracy_score
7
```

The status bar at the bottom indicates the file is 'Program.py' with 5 lines, using the 'Pylance' extension. The system tray shows the date and time as 14:35 on 18-10-2023.

#Loading dataset to a Pandas Dataframe:

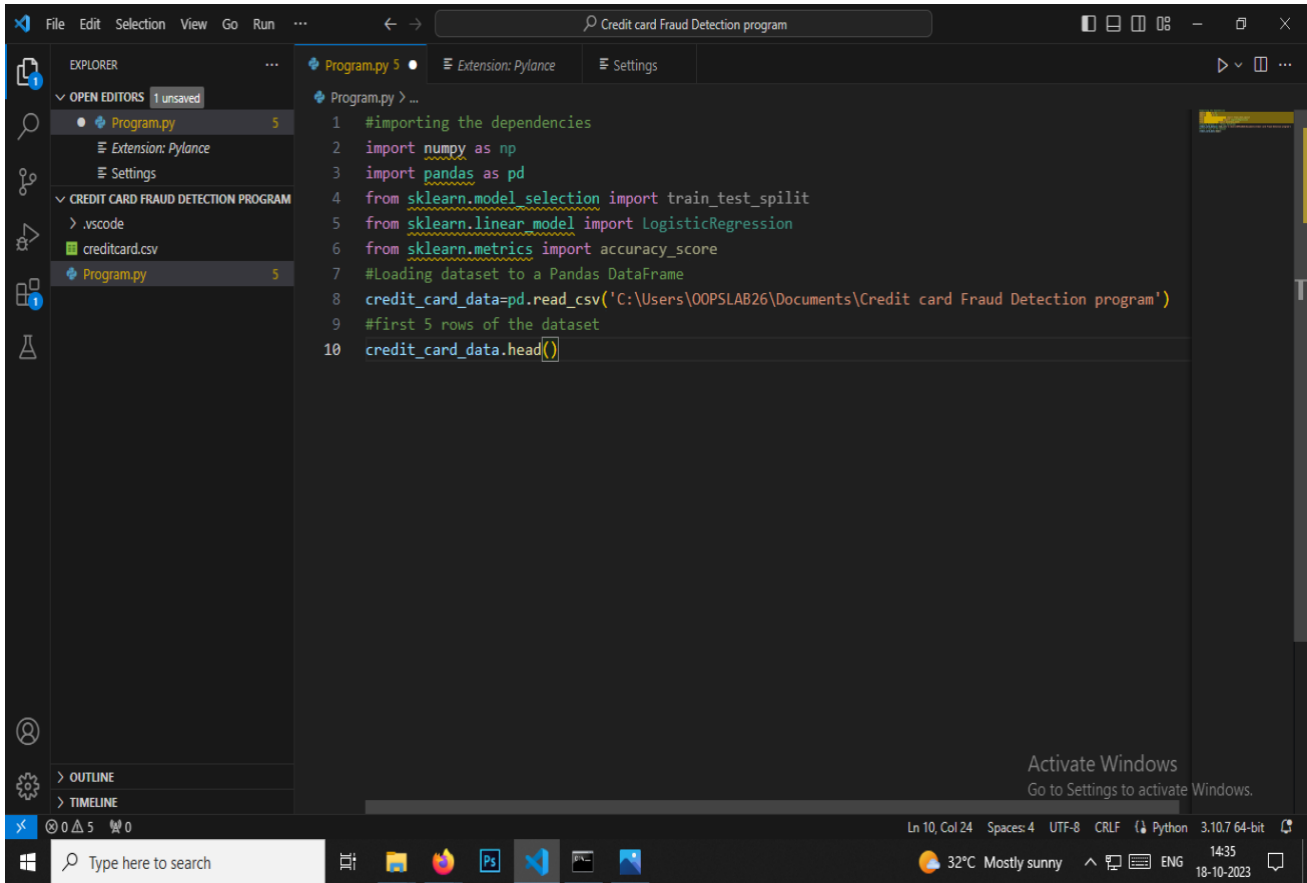


The screenshot shows the Visual Studio Code interface with the same file explorer and code editor. The code editor now shows the following Python code:

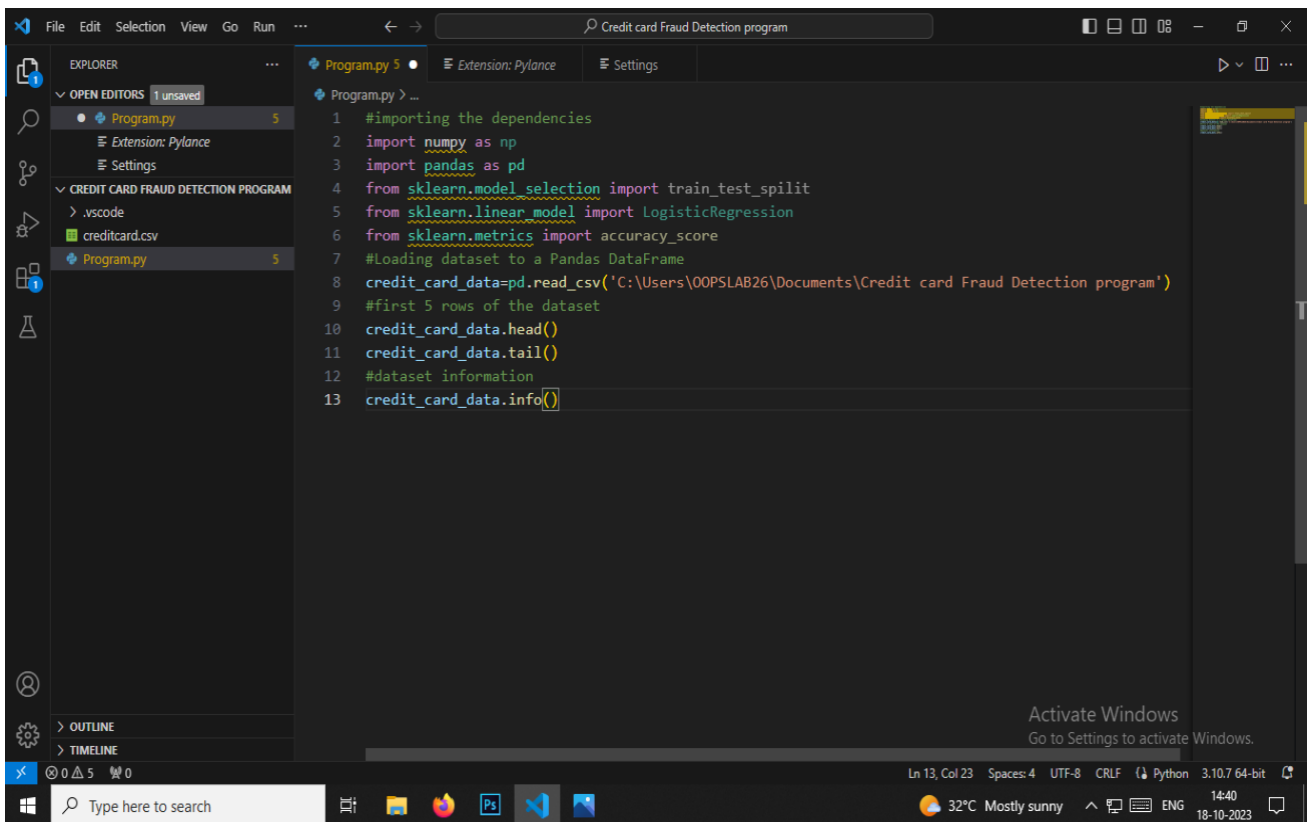
```
1 #importing the dependencies
2 import numpy as np
3 import pandas as pd
4 from sklearn.model_selection import train_test_split
5 from sklearn.linear_model import LogisticRegression
6 from sklearn.metrics import accuracy_score
7 #Loading dataset to a Pandas DataFrame
8 credit_card_data=pd.read_csv('C:\Users\OOPSLAB26\Documents\Credit card Fraud Detection program')
9
```

The status bar at the bottom indicates the file is 'Program.py' with 9 lines. The system tray shows the date and time as 14:35 on 18-10-2023.

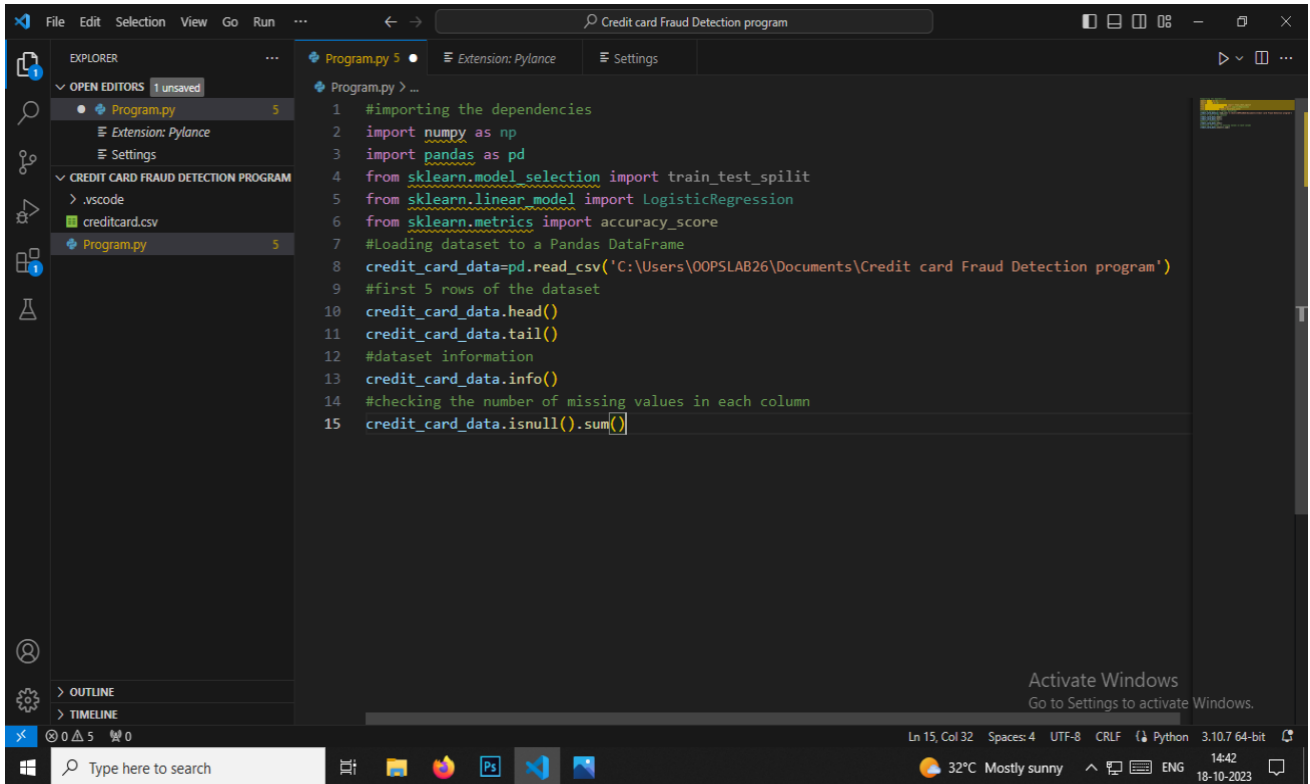
#First 5 rows of the dataset:



#Dataset information:

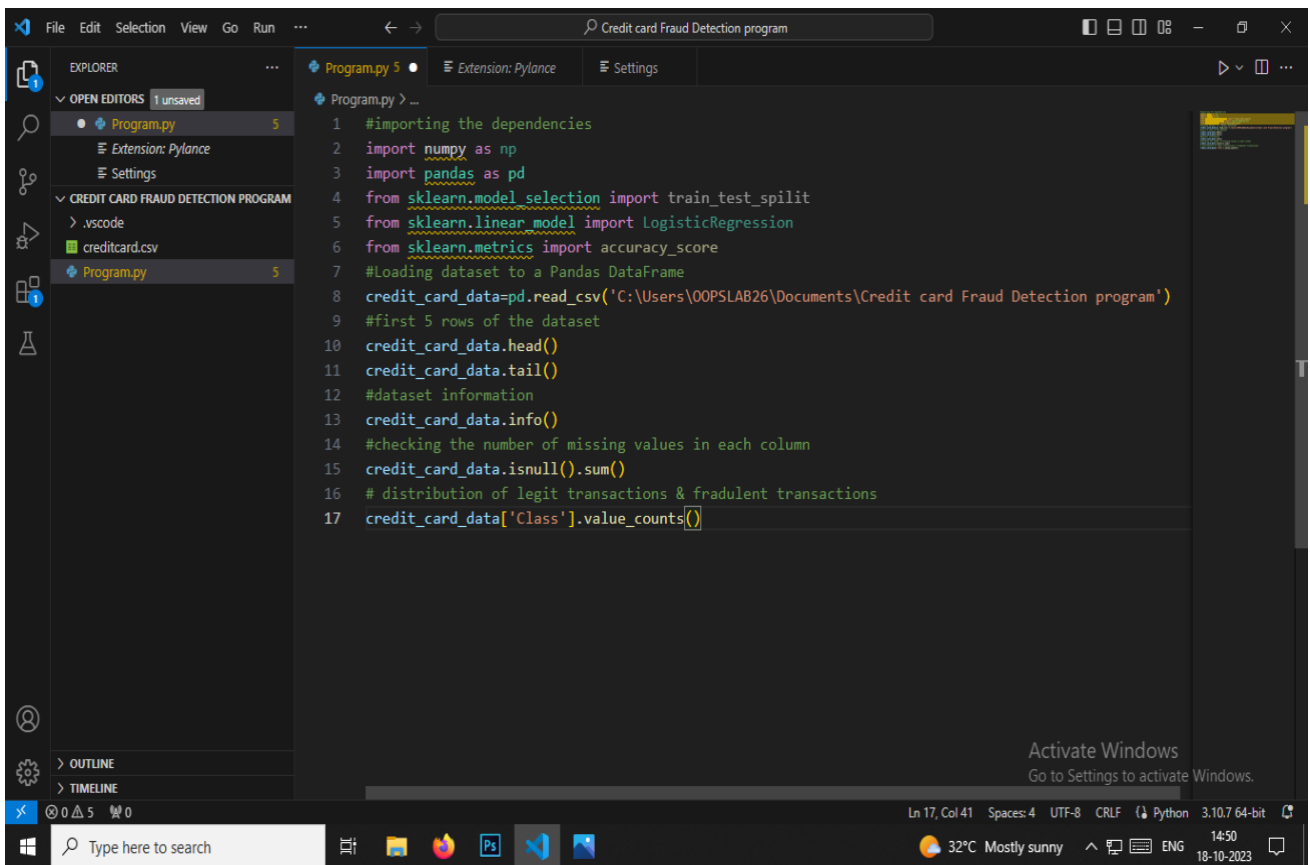


#Checking the number of missing values in each column:



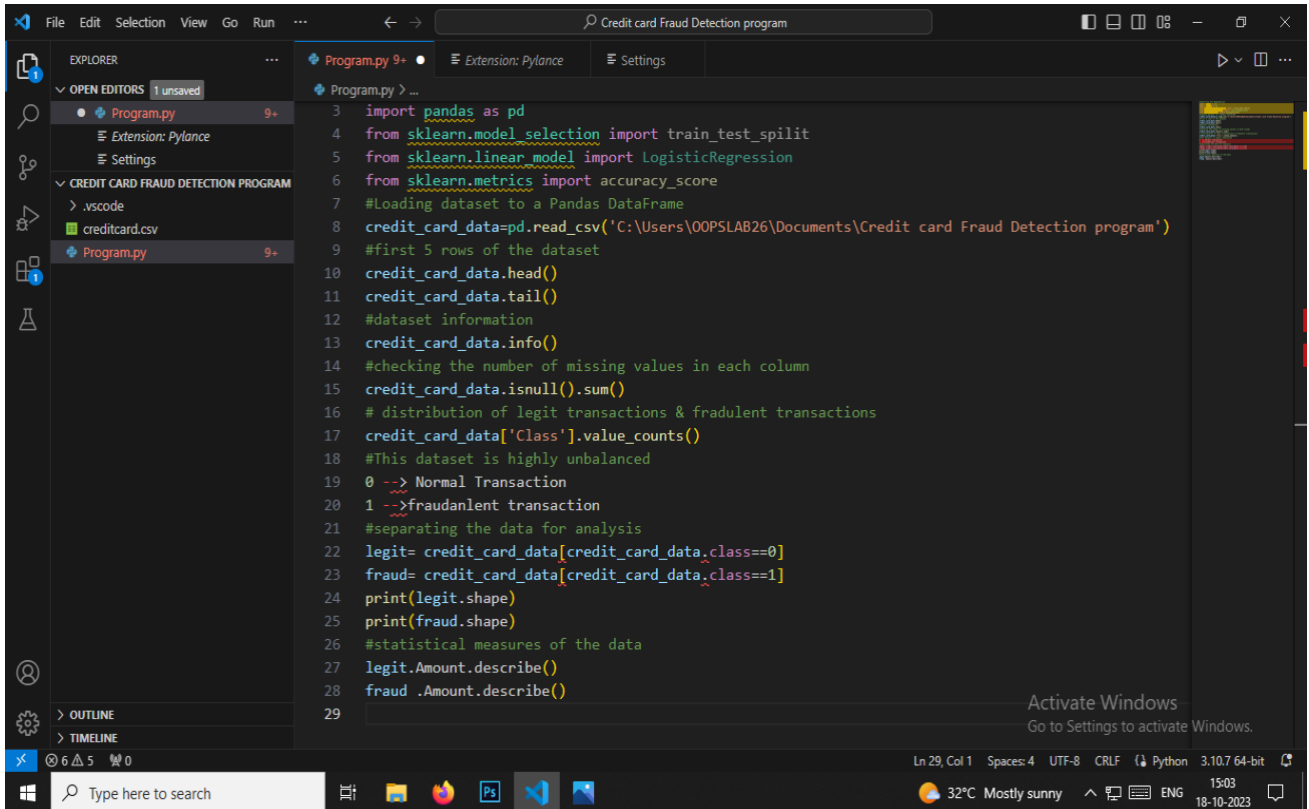
```
1 #importing the dependencies
2 import numpy as np
3 import pandas as pd
4 from sklearn.model_selection import train_test_split
5 from sklearn.linear_model import LogisticRegression
6 from sklearn.metrics import accuracy_score
7 #Loading dataset to a Pandas DataFrame
8 credit_card_data=pd.read_csv('C:\Users\OOPSLAB26\Documents\Credit card Fraud Detection program')
9 #first 5 rows of the dataset
10 credit_card_data.head()
11 credit_card_data.tail()
12 #dataset information
13 credit_card_data.info()
14 #checking the number of missing values in each column
15 credit_card_data.isnull().sum()
```

#Distribution of legit transactions & fradulent transactions:



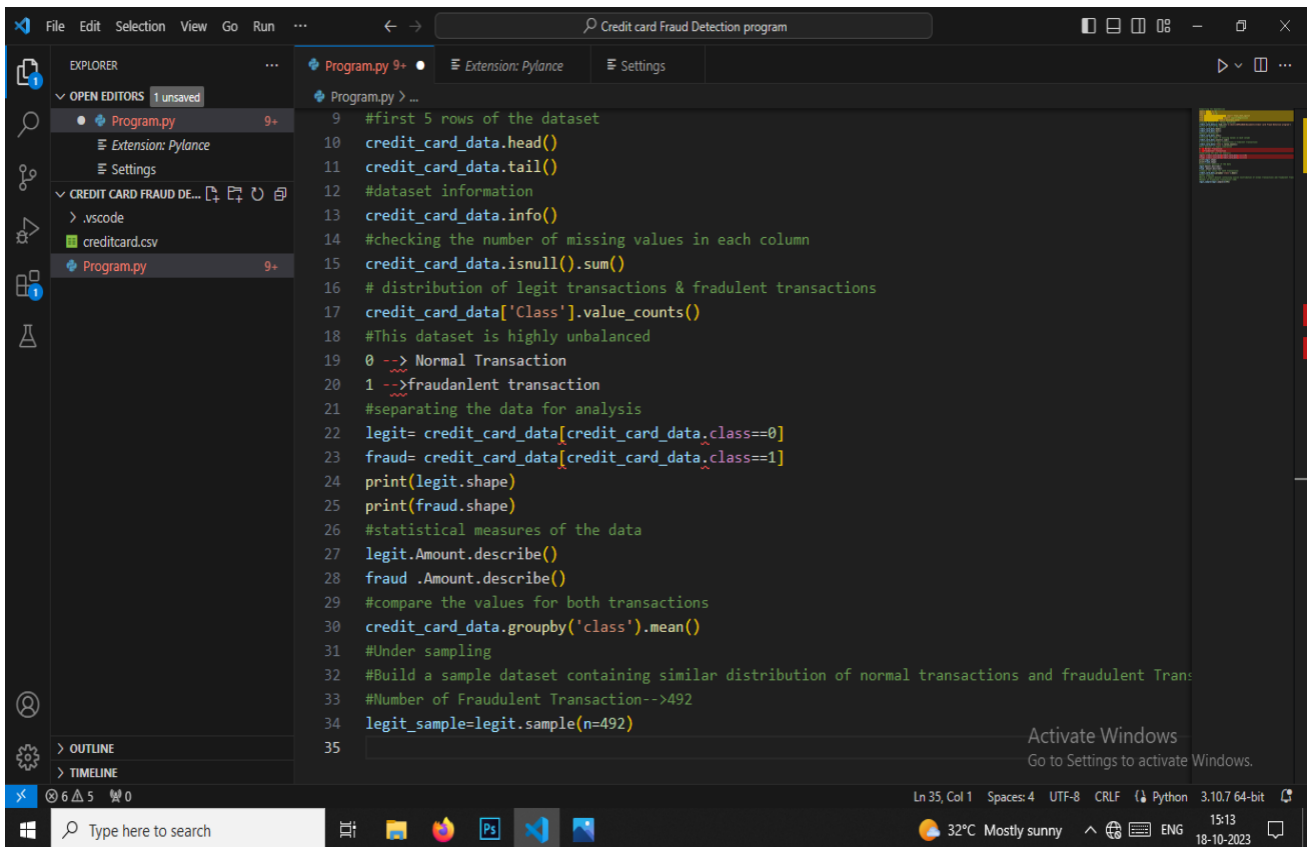
```
1 #importing the dependencies
2 import numpy as np
3 import pandas as pd
4 from sklearn.model_selection import train_test_split
5 from sklearn.linear_model import LogisticRegression
6 from sklearn.metrics import accuracy_score
7 #Loading dataset to a Pandas DataFrame
8 credit_card_data=pd.read_csv('C:\Users\OOPSLAB26\Documents\Credit card Fraud Detection program')
9 #first 5 rows of the dataset
10 credit_card_data.head()
11 credit_card_data.tail()
12 #dataset information
13 credit_card_data.info()
14 #checking the number of missing values in each column
15 credit_card_data.isnull().sum()
16 # distribution of legit transactions & fradulent transactions
17 credit_card_data['Class'].value_counts()
```

#This dataset is highly unbalanced:



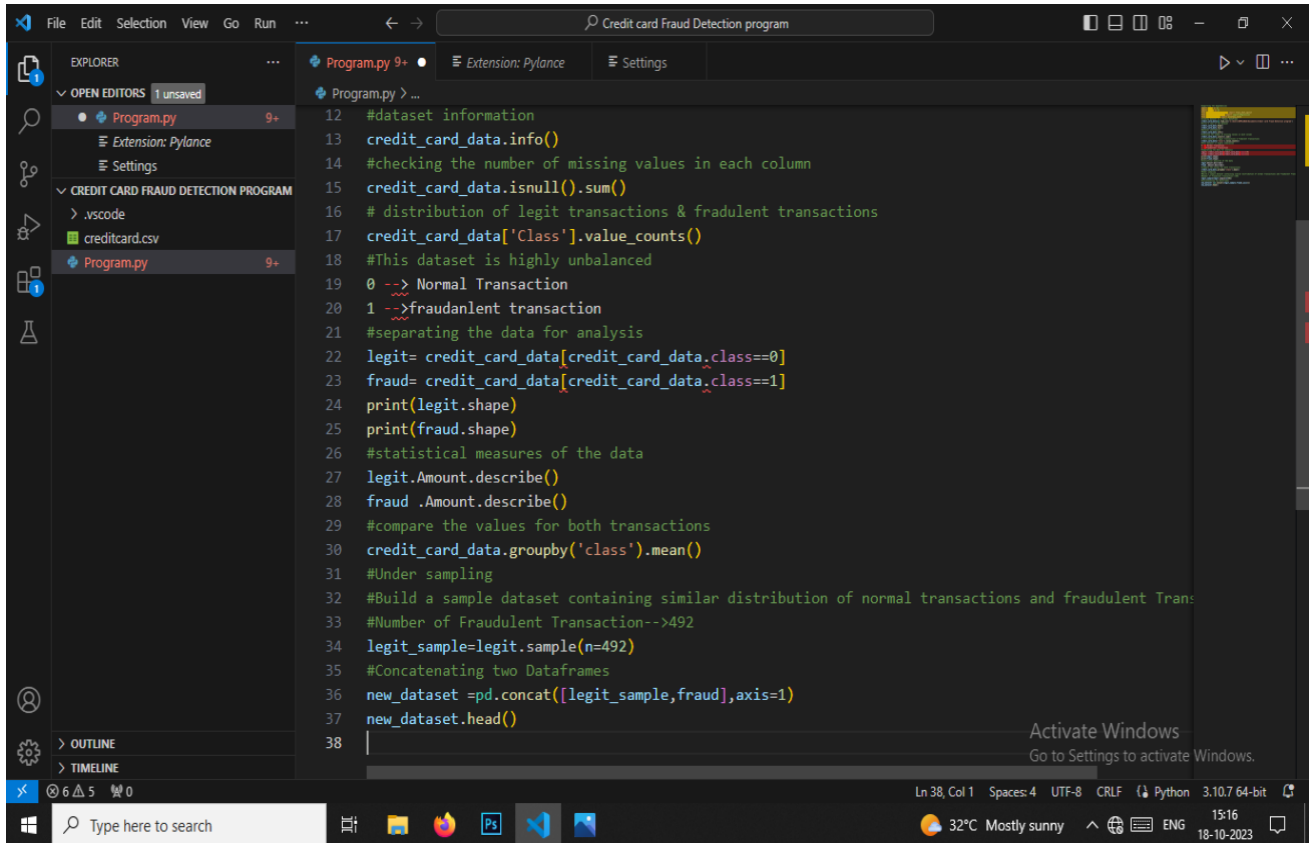
```
3 import pandas as pd
4 from sklearn.model_selection import train_test_split
5 from sklearn.linear_model import LogisticRegression
6 from sklearn.metrics import accuracy_score
7 #Loading dataset to a Pandas DataFrame
8 credit_card_data=pd.read_csv('C:\Users\OOPSLAB26\Documents\Credit card Fraud Detection program')
9 #first 5 rows of the dataset
10 credit_card_data.head()
11 credit_card_data.tail()
12 #dataset information
13 credit_card_data.info()
14 #checking the number of missing values in each column
15 credit_card_data.isnull().sum()
16 # distribution of legit transactions & fraudulent transactions
17 credit_card_data['Class'].value_counts()
18 #This dataset is highly unbalanced
19 0 --> Normal Transaction
20 1 --> fraudulent transaction
21 #separating the data for analysis
22 legit= credit_card_data[credit_card_data.class==0]
23 fraud= credit_card_data[credit_card_data.class==1]
24 print(legit.shape)
25 print(fraud.shape)
26 #statistical measures of the data
27 legit.Amount.describe()
28 fraud .Amount.describe()
29
```

#Separating the data for Analysis:



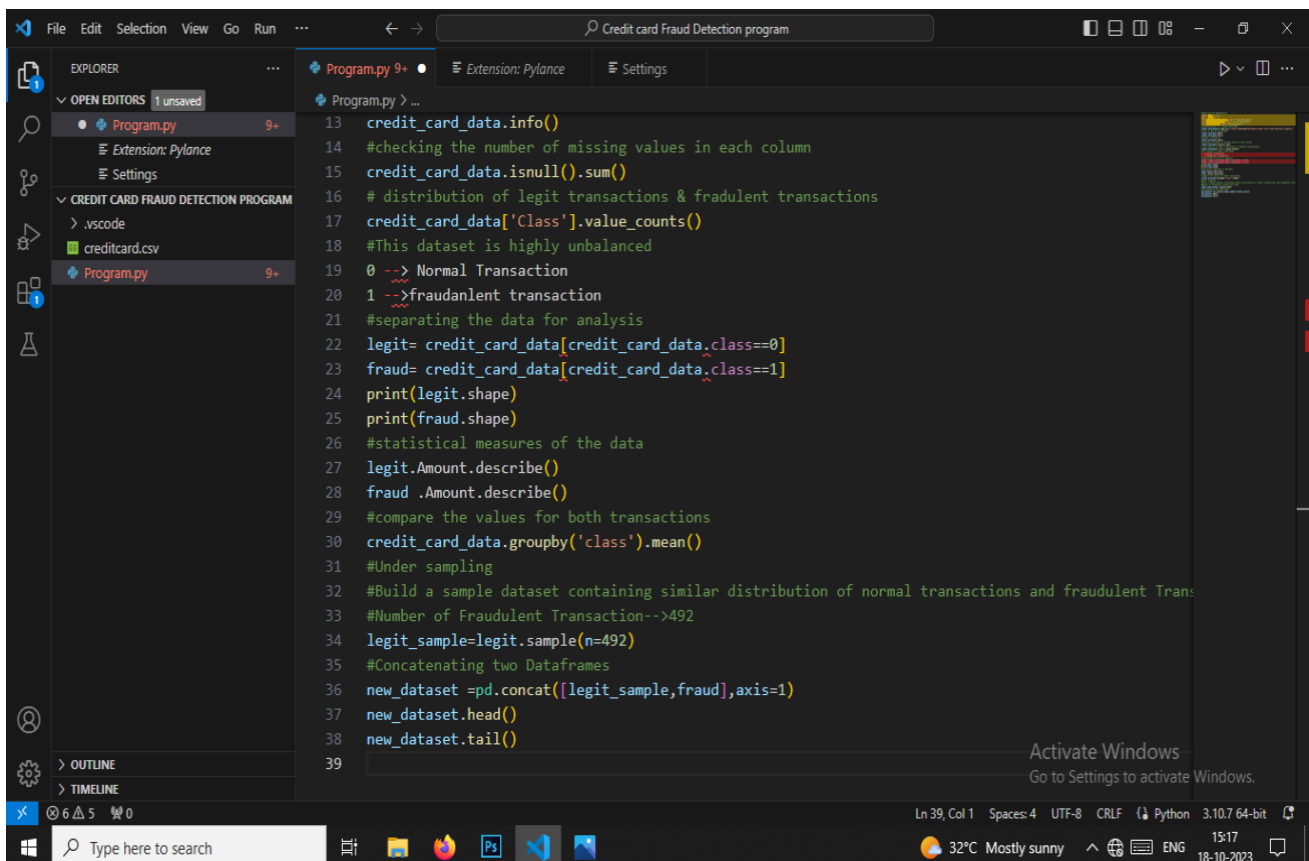
```
9 #first 5 rows of the dataset
10 credit_card_data.head()
11 credit_card_data.tail()
12 #dataset information
13 credit_card_data.info()
14 #checking the number of missing values in each column
15 credit_card_data.isnull().sum()
16 # distribution of legit transactions & fraudulent transactions
17 credit_card_data['Class'].value_counts()
18 #This dataset is highly unbalanced
19 0 --> Normal Transaction
20 1 --> fraudulent transaction
21 #separating the data for analysis
22 legit= credit_card_data[credit_card_data.class==0]
23 fraud= credit_card_data[credit_card_data.class==1]
24 print(legit.shape)
25 print(fraud.shape)
26 #statistical measures of the data
27 legit.Amount.describe()
28 fraud .Amount.describe()
29 #compare the values for both transactions
30 credit_card_data.groupby('class').mean()
31 #Under sampling
32 #Build a sample dataset containing similar distribution of normal transactions and fraudulent Trans
33 #Number of Fraudulent Transaction-->492
34 legit_sample=legit.sample(n=492)
35
```

#Statistical measures of the data:



```
12 #dataset information
13 credit_card_data.info()
14 #checking the number of missing values in each column
15 credit_card_data.isnull().sum()
16 # distribution of legit transactions & fraudulent transactions
17 credit_card_data['Class'].value_counts()
18 #This dataset is highly unbalanced
19 0 --> Normal Transaction
20 1 --> fraudulent transaction
21 #separating the data for analysis
22 legit= credit_card_data[credit_card_data.class==0]
23 fraud= credit_card_data[credit_card_data.class==1]
24 print(legit.shape)
25 print(fraud.shape)
26 #statistical measures of the data
27 legit.Amount.describe()
28 fraud .Amount.describe()
29 #compare the values for both transactions
30 credit_card_data.groupby('class').mean()
31 #Under sampling
32 #Build a sample dataset containing similar distribution of normal transactions and fraudulent Trans
33 #Number of Fraudulent Transaction-->492
34 legit_sample=legit.sample(n=492)
35 #Concatenating two Dataframes
36 new_dataset =pd.concat([legit_sample,fraud],axis=1)
37 new_dataset.head()
38
```

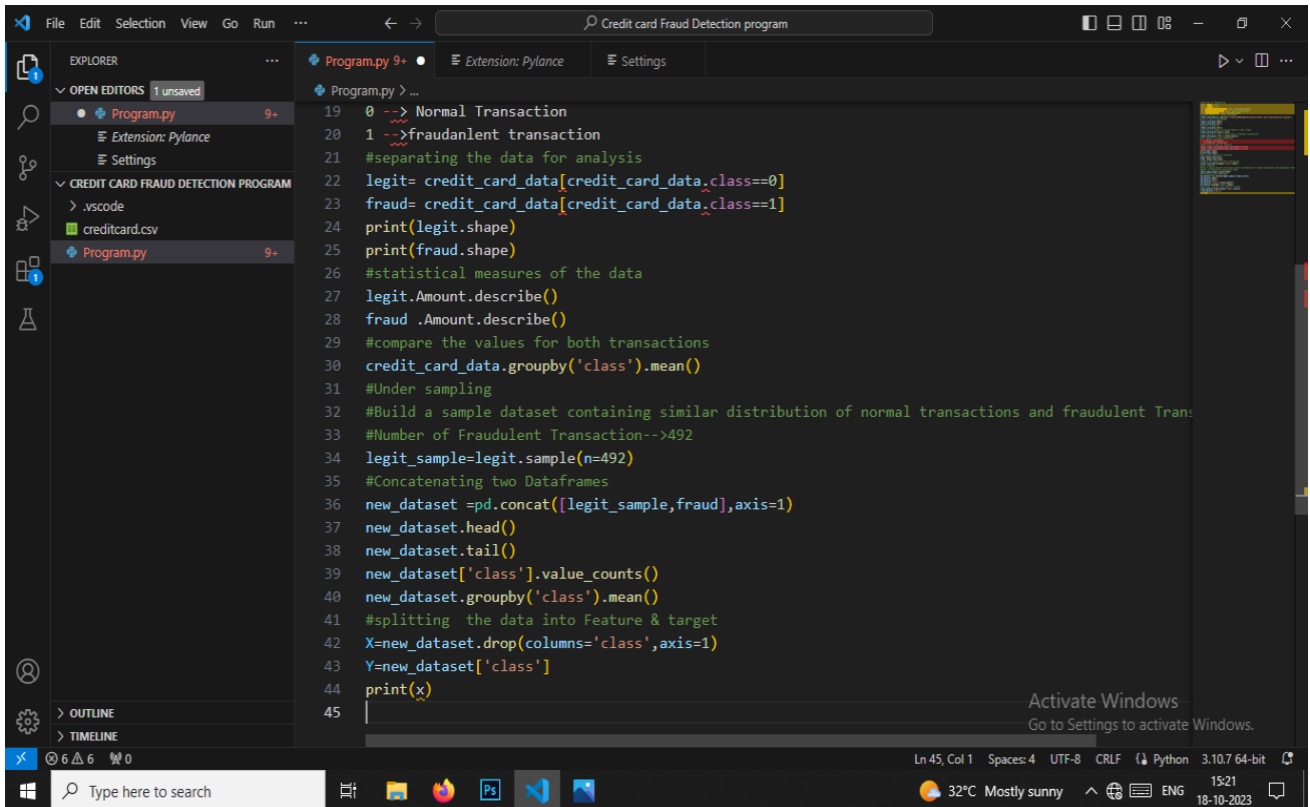
#Compare the values for both transactions:



```
13 credit_card_data.info()
14 #checking the number of missing values in each column
15 credit_card_data.isnull().sum()
16 # distribution of legit transactions & fraudulent transactions
17 credit_card_data['Class'].value_counts()
18 #This dataset is highly unbalanced
19 0 --> Normal Transaction
20 1 --> fraudulent transaction
21 #separating the data for analysis
22 legit= credit_card_data[credit_card_data.class==0]
23 fraud= credit_card_data[credit_card_data.class==1]
24 print(legit.shape)
25 print(fraud.shape)
26 #statistical measures of the data
27 legit.Amount.describe()
28 fraud .Amount.describe()
29 #compare the values for both transactions
30 credit_card_data.groupby('class').mean()
31 #Under sampling
32 #Build a sample dataset containing similar distribution of normal transactions and fraudulent Trans
33 #Number of Fraudulent Transaction-->492
34 legit_sample=legit.sample(n=492)
35 #Concatenating two Dataframes
36 new_dataset =pd.concat([legit_sample,fraud],axis=1)
37 new_dataset.head()
38 new_dataset.tail()
39
```

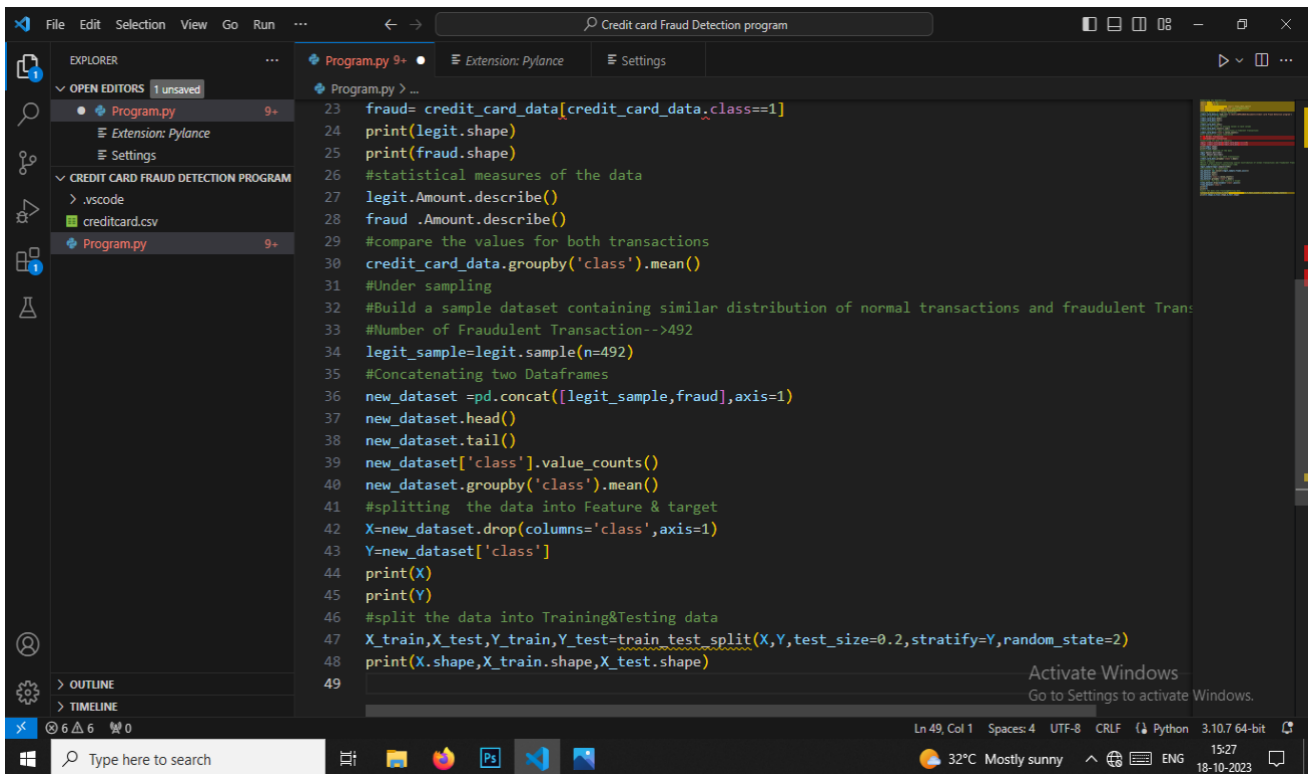
#Under sampling build a sample dataset containing similar distribution of normal transactions and fraudulent transactions.

#Number of Fraudulent transactions→492.



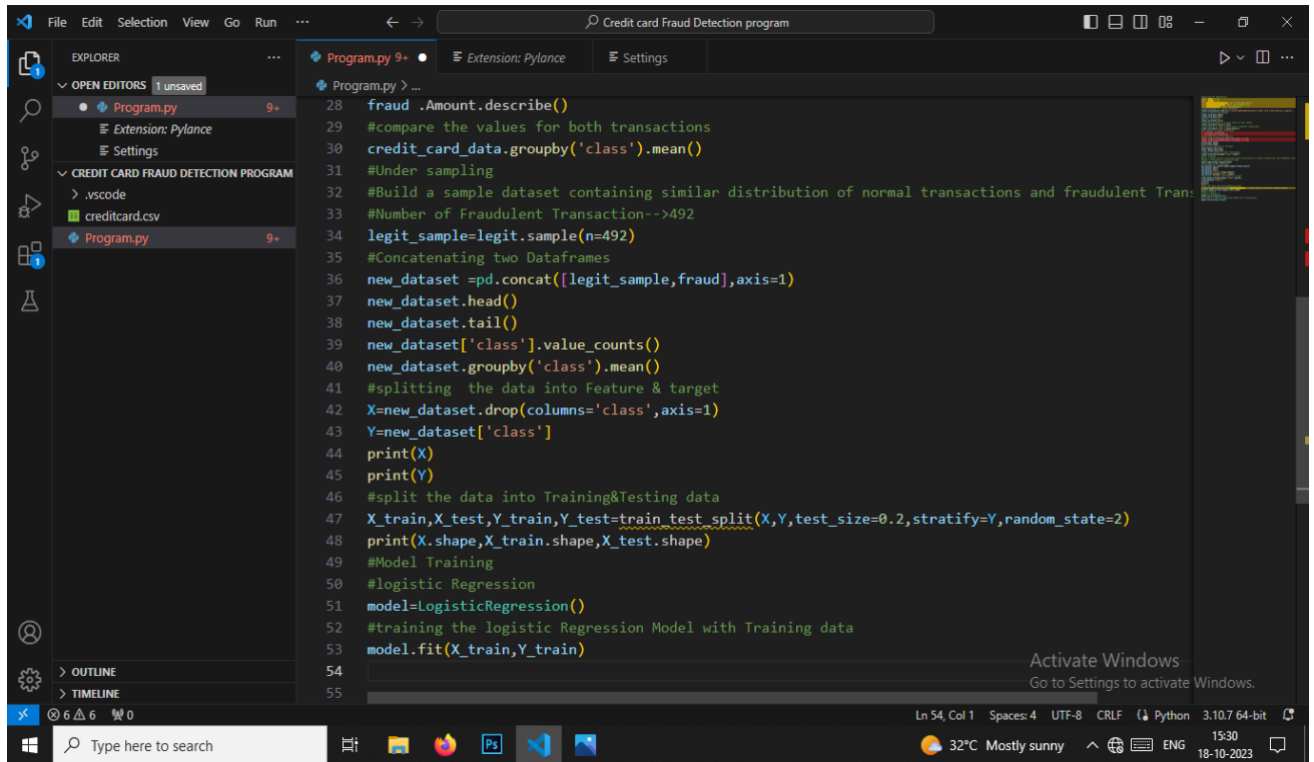
```
19 0 --> Normal Transaction
20 1 --> fraudulent transaction
21 #separating the data for analysis
22 legit= credit_card_data[credit_card_data.class==0]
23 fraud= credit_card_data[credit_card_data.class==1]
24 print(legit.shape)
25 print(fraud.shape)
26 #statistical measures of the data
27 legit.Amount.describe()
28 fraud .Amount.describe()
29 #compare the values for both transactions
30 credit_card_data.groupby('class').mean()
31 #Under sampling
32 #Build a sample dataset containing similar distribution of normal transactions and fraudulent Transactions
33 #Number of Fraudulent Transaction-->492
34 legit_sample=legit.sample(n=492)
35 #Concatenating two Dataframes
36 new_dataset =pd.concat([legit_sample,fraud],axis=1)
37 new_dataset.head()
38 new_dataset.tail()
39 new_dataset['class'].value_counts()
40 new_dataset.groupby('class').mean()
41 #splitting the data into Feature & target
42 X=new_dataset.drop(columns='class',axis=1)
43 Y=new_dataset['class']
44 print(X)
45
```

#Concatenating two Dataframes:



```
23 fraud= credit_card_data[credit_card_data.class==1]
24 print(legit.shape)
25 print(fraud.shape)
26 #statistical measures of the data
27 legit.Amount.describe()
28 fraud .Amount.describe()
29 #compare the values for both transactions
30 credit_card_data.groupby('class').mean()
31 #Under sampling
32 #Build a sample dataset containing similar distribution of normal transactions and fraudulent Transactions
33 #Number of Fraudulent Transaction-->492
34 legit_sample=legit.sample(n=492)
35 #Concatenating two Dataframes
36 new_dataset =pd.concat([legit_sample,fraud],axis=1)
37 new_dataset.head()
38 new_dataset.tail()
39 new_dataset['class'].value_counts()
40 new_dataset.groupby('class').mean()
41 #splitting the data into Feature & target
42 X=new_dataset.drop(columns='class',axis=1)
43 Y=new_dataset['class']
44 print(X)
45 print(Y)
46 #split the data into Training&Testing data
47 X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,stratify=Y,random_state=2)
48 print(X.shape,X_train.shape,X_test.shape)
49
```


#Splitting the data into feature & target:



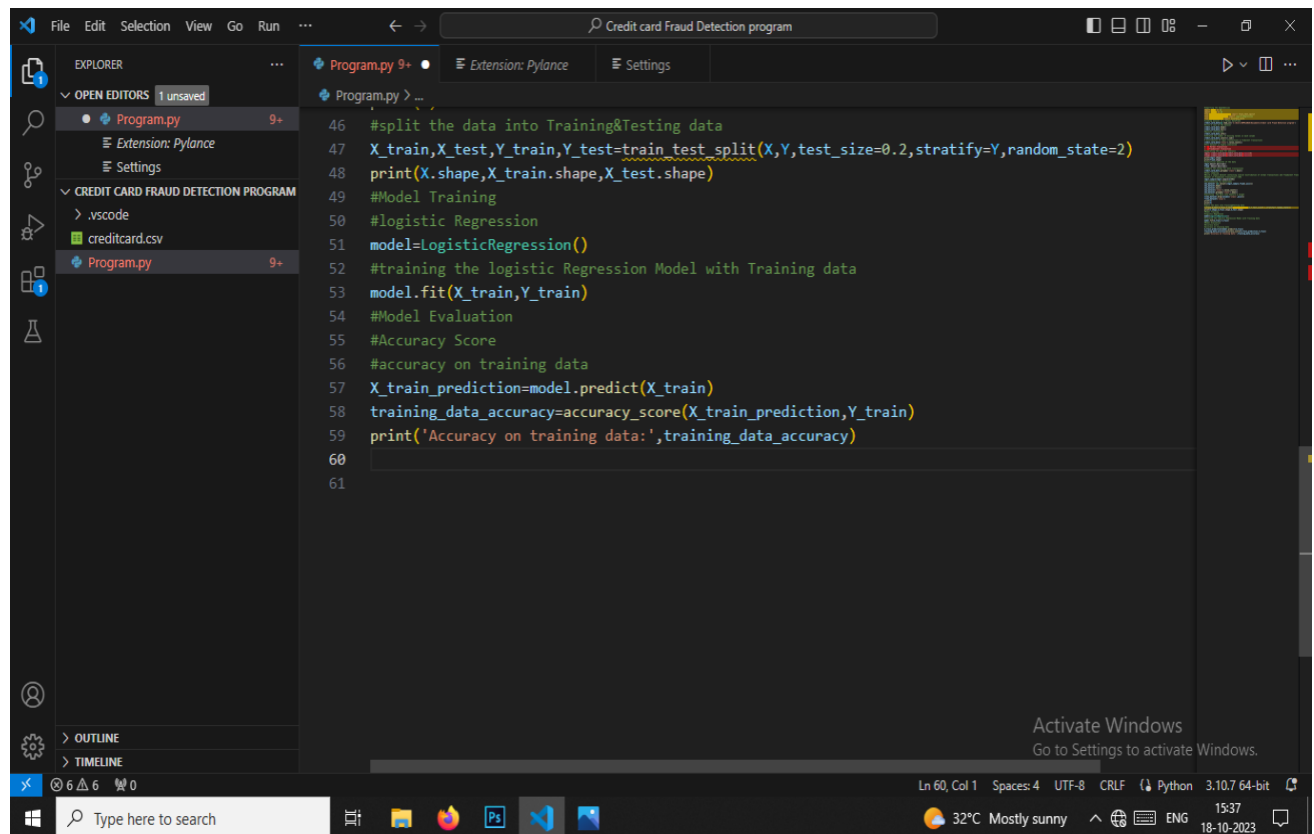
The screenshot shows a Visual Studio Code editor window titled "Credit card Fraud Detection program". The Explorer sidebar on the left shows a project named "CREDIT CARD FRAUD DETECTION PROGRAM" containing files ".vscode", "creditcard.csv", and "Program.py". The main editor displays the "Program.py" file with the following code:

```
28 fraud.Amount.describe()
29 #compare the values for both transactions
30 credit_card_data.groupby('class').mean()
31 #Under sampling
32 #Build a sample dataset containing similar distribution of normal transactions and fraudulent Transactions
33 #Number of Fraudulent Transaction-->492
34 legit_sample=legit.sample(n=492)
35 #Concatenating two Dataframes
36 new_dataset=pd.concat([legit_sample,fraud],axis=1)
37 new_dataset.head()
38 new_dataset.tail()
39 new_dataset['class'].value_counts()
40 new_dataset.groupby('class').mean()
41 #splitting the data into Feature & target
42 X=new_dataset.drop(columns='class',axis=1)
43 Y=new_dataset['class']
44 print(X)
45 print(Y)
46 #split the data into Training&Testing data
47 X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,stratify=Y,random_state=2)
48 print(X.shape,X_train.shape,X_test.shape)
49 #Model Training
50 #logistic Regression
51 model=LogisticRegression()
52 #training the logistic Regression Model with Training data
53 model.fit(X_train,Y_train)
54
```

The status bar at the bottom indicates "Ln 54, Col 1", "Spaces: 4", "UTF-8", "CRLF", "Python", "3.10.7 64-bit", "15:30", "18-10-2023", and "32°C Mostly sunny".

#Spilit the data into Training&Testing Data:

#Model Training and Logistic Regression:



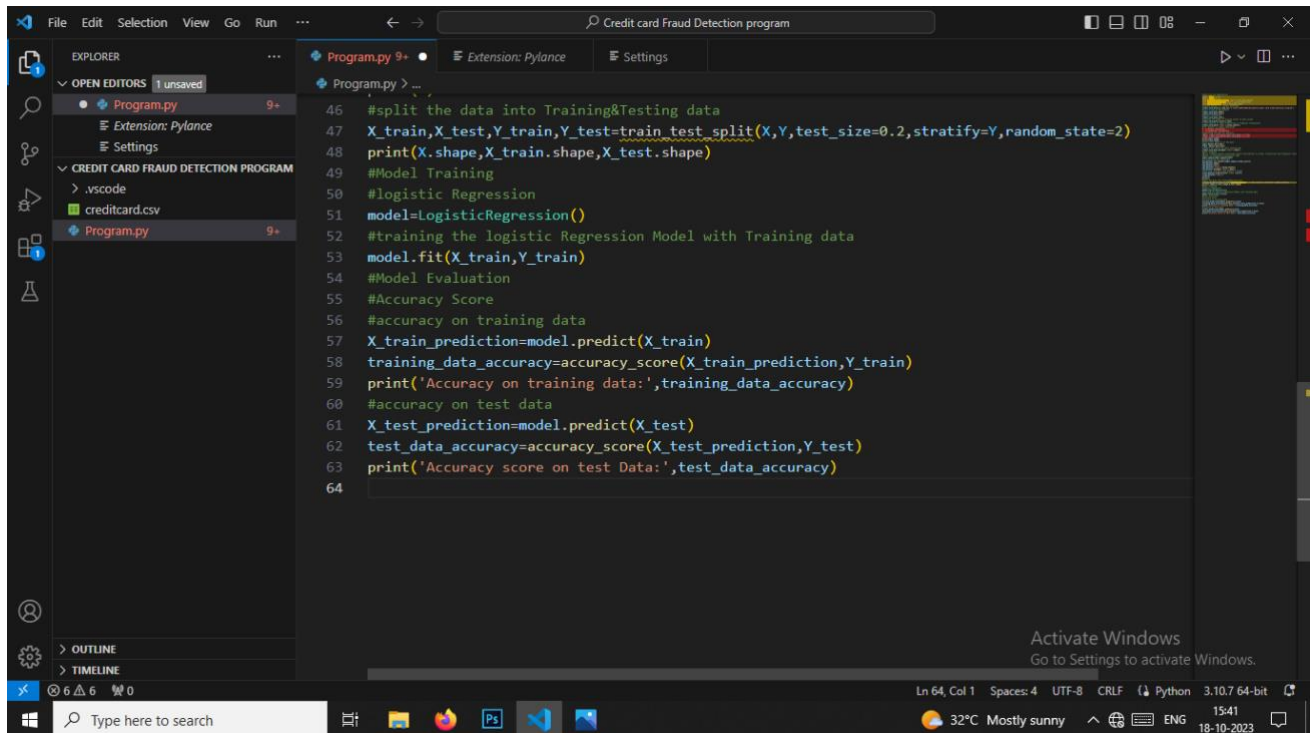
The screenshot shows the same Visual Studio Code editor window, but the code in "Program.py" has been updated to include model evaluation:

```
46 #split the data into Training&Testing data
47 X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,stratify=Y,random_state=2)
48 print(X.shape,X_train.shape,X_test.shape)
49 #Model Training
50 #logistic Regression
51 model=LogisticRegression()
52 #training the logistic Regression Model with Training data
53 model.fit(X_train,Y_train)
54 #Model Evaluation
55 #Accuracy Score
56 #accuracy on training data
57 X_train_prediction=model.predict(X_train)
58 training_data_accuracy=accuracy_score(X_train_prediction,Y_train)
59 print('Accuracy on training data:',training_data_accuracy)
60
61
```

The status bar at the bottom indicates "Ln 60, Col 1", "Spaces: 4", "UTF-8", "CRLF", "Python", "3.10.7 64-bit", "15:37", "18-10-2023", and "32°C Mostly sunny".

#Model Evaluation & Accuracy Score:

#Accuracy on Training data & Testing data:



```
46 #split the data into Training&Testing data
47 X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,stratify=Y,random_state=2)
48 print(X.shape,X_train.shape,X_test.shape)
49 #Model Training
50 #logistic Regression
51 model=LogisticRegression()
52 #training the logistic Regression Model with Training data
53 model.fit(X_train,Y_train)
54 #Model Evaluation
55 #Accuracy Score
56 #accuracy on training data
57 X_train_prediction=model.predict(X_train)
58 training_data_accuracy=accuracy_score(X_train_prediction,Y_train)
59 print('Accuracy on training data:',training_data_accuracy)
60 #accuracy on test data
61 X_test_prediction=model.predict(X_test)
62 test_data_accuracy=accuracy_score(X_test_prediction,Y_test)
63 print('Accuracy score on test Data:',test_data_accuracy)
64
```

Training ML model	Testing ML model
<ul style="list-style-type: none">• Data preparation	<ul style="list-style-type: none">• Testing set
<ul style="list-style-type: none">• Data splitting	<ul style="list-style-type: none">• Model Evaluation
<ul style="list-style-type: none">• Model building	<ul style="list-style-type: none">• Performance Metrics
<ul style="list-style-type: none">• Model Training	

