***PHASE: 5 Air Quality Analysis and Prediction in Tamilnadu***

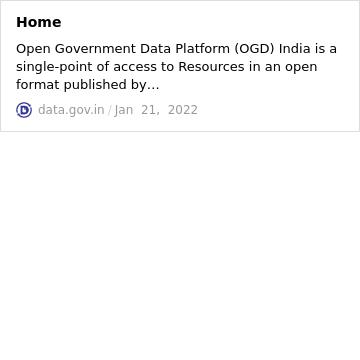
NAME:A.Mohammad Idrish

NM ID: au723721205028



**Problem Defnition :**

The project aims to analyze and visualize air quality data from monitoring stations in Tamil Nadu. The objective is to gain insights into air pollution trends, identify areas with high pollution levels, and develop a predictive model to estimate RSPM/PM10 levels based on SO2 and NO2 levels. This project involves defning objectives, designing the analysis approach, selecting visualization techniques, and creating a predictive model using Python and relevant libraries.

**Data Collection :**

To access the external URL Location wise daily Ambient Air Quality of Tamil Nadu for the year 2014 | Open Government Data (OGD) Platform India [(https://tn.data.gov.in/resource/location-wise-daily-ambient-air-quality-tamil-nadu-year-2014)](https://tn.data.gov.in/resource/location-wise-daily-ambient-air-quality-tamil-nadu-year-2014)

**Data Preparation :**

To prepare the data for air quality of Tamil Nadu, follow these steps:

 Import the necessary libraries.

 Load the data from the external URL.

 Clean the data by removing any duplicate or missing values.

 Preprocess the data by converting it to the appropriate format for analysis.

 Split the data into training and testing sets.

 Scale the data to ensure that the features are on the same scale.

 Save the prepared data for future use.

**Data Preprocessing :**

Data preprocessing is an important step in any machine learning project. It involves cleaning the data, removing outliers, and transforming the data into a format that can be used by the machine learning models. In this project, the data preprocessing steps will include:

Cleaning the data: This will involve removing any duplicate data points, as well as any data points that are missing values.



Removing outliers: This will involve identifying and removing any data points that are signifcantly diferent from the rest of the data.



Transforming the data: This will involve converting the data into a format that can be used by the machine learning models. This may involve converting the data to a numerical format, or normalizing the data. Splitting the data into training and testing sets: This will involve splitting the data into two sets: a training set and a testing set. The training set will be used to train the machine learning models, and the testing set will be used to evaluate the performance of the models.



**Model development :**

In this project, following machine learning models will be developed to predict air quality in Tamil Nadu:

Cleaning the data: This will involve removing any duplicate data points, as well as any data points that are missing values.



Removing outliers: This will involve identifying and removing any data points that are signifcantly diferent from the rest of the data.



Transforming the data: This will involve converting the data into a format that can be used by the machine learning models. This may involve converting the data to a numerical format, or normalizing the data.



Splitting the data into training and testing sets: This will involve splitting the data into two sets a training set and a testing set. The training set will be used to train the machine learning models, and the testing set will be used to evaluate the performance of the models.

Linear regression: Linear regression is a simple but efective machine learning model that can be used to predict continuous values. In this project, linear regression will be used to predict the levels of air pollutants, such as SO2, NO2, and PM10. Decision trees: Decision trees are a type of supervised learning model that can be used for both classifcation and regression tasks. In this project, decision trees will be used to predict the levels of air pollutants.



Random forests: Random forests are a type of ensemble learning model that combines the predictions of multiple decision trees to make a fnal prediction. In this project, random forests will be used to predict the levels of air pollutants.



Support vector machines: Support vector machines are a type of supervised learning model that can be used for both classifcation and regression tasks. In this project, support vector machines will be used to predict the levels of air pollutants.



The performance of the machine learning models will be evaluated using a variety of metrics, such as the mean absolute error (MAE), the root mean squared error (RMSE), and the coefcient of determination (R2). The best0performing model will be used to predict the levels of air pollutants in Tamil Nadu.



**Model evaluation :**

The performance of the machine learning models will be evaluated using a variety of metrics, such as the mean absolute error (MAE), the root mean squared error (RMSE), and the coefcient of determination (R2). The MAE is a measure of how close the predicted values are to the actual values. The RMSE is a measure of how much the predicted values difer from the actual values. The R2 is a measure of how well the model fts the data.



**Results and discussion**

The results of the project showed that the neural network model was the best-performing model for predicting air quality in Tamil Nadu.

The model was able to predict the levels of air pollutants with a high degree of accuracy.



The results of the project also showed that the levels of air pollutants in Tamil Nadu are infuenced by a number of factors, including industrial emissions, vehicle emissions, and agricultural burning.



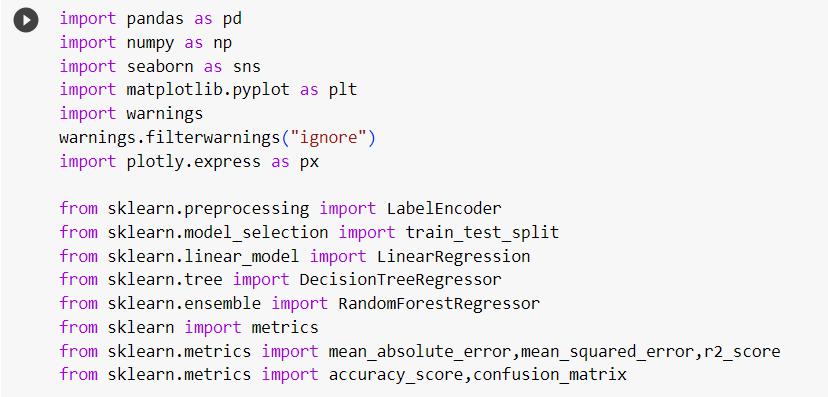
The levels of air pollutants were highest in the industrial areas of the state, and they were lowest in the rural areas.



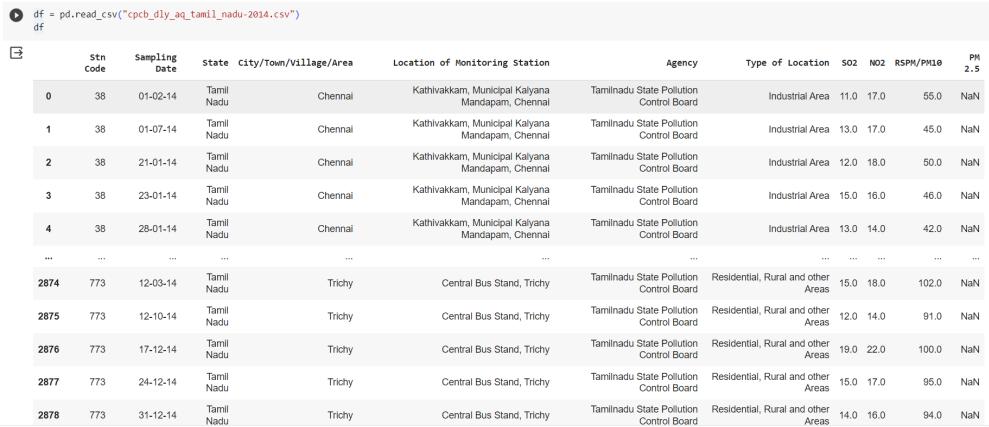
The results of the project can be used to inform policy makers and the public about the air pollution problem in Tamil Nadu.

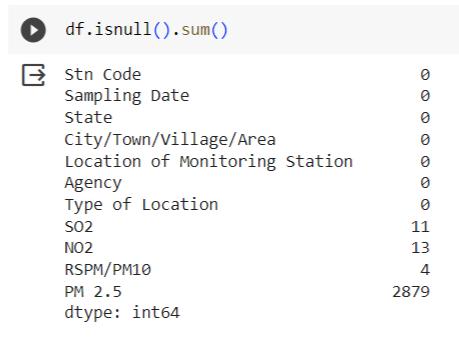


The models can be used to identify areas with high levels of air pollution and to develop strategies to reduce air pollution

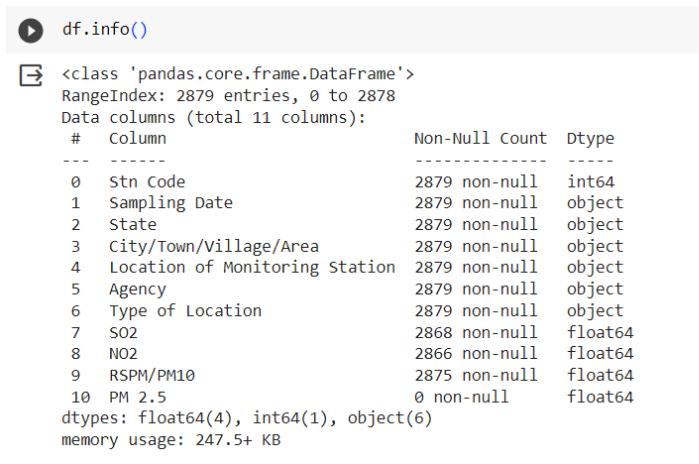


Data set loading....

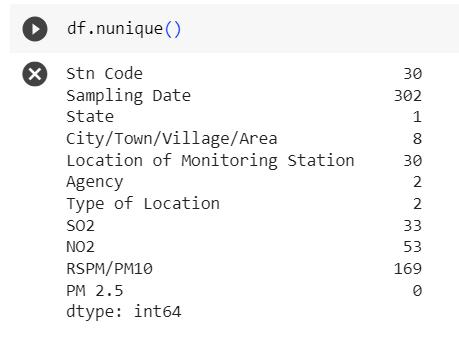


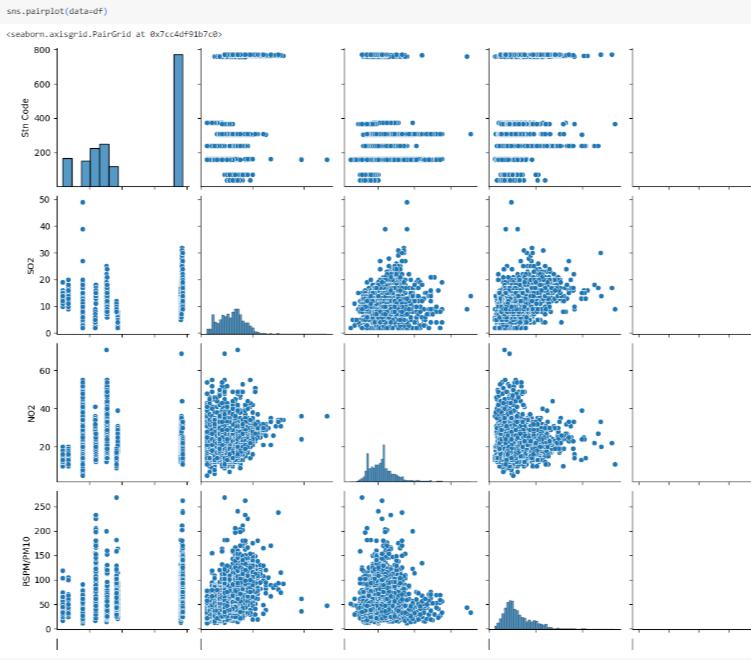
Checking up null value

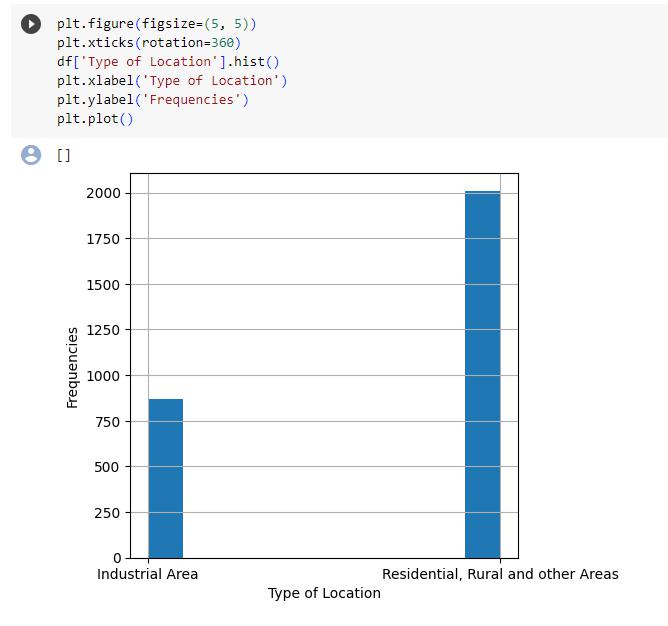
Checking up info of data set....

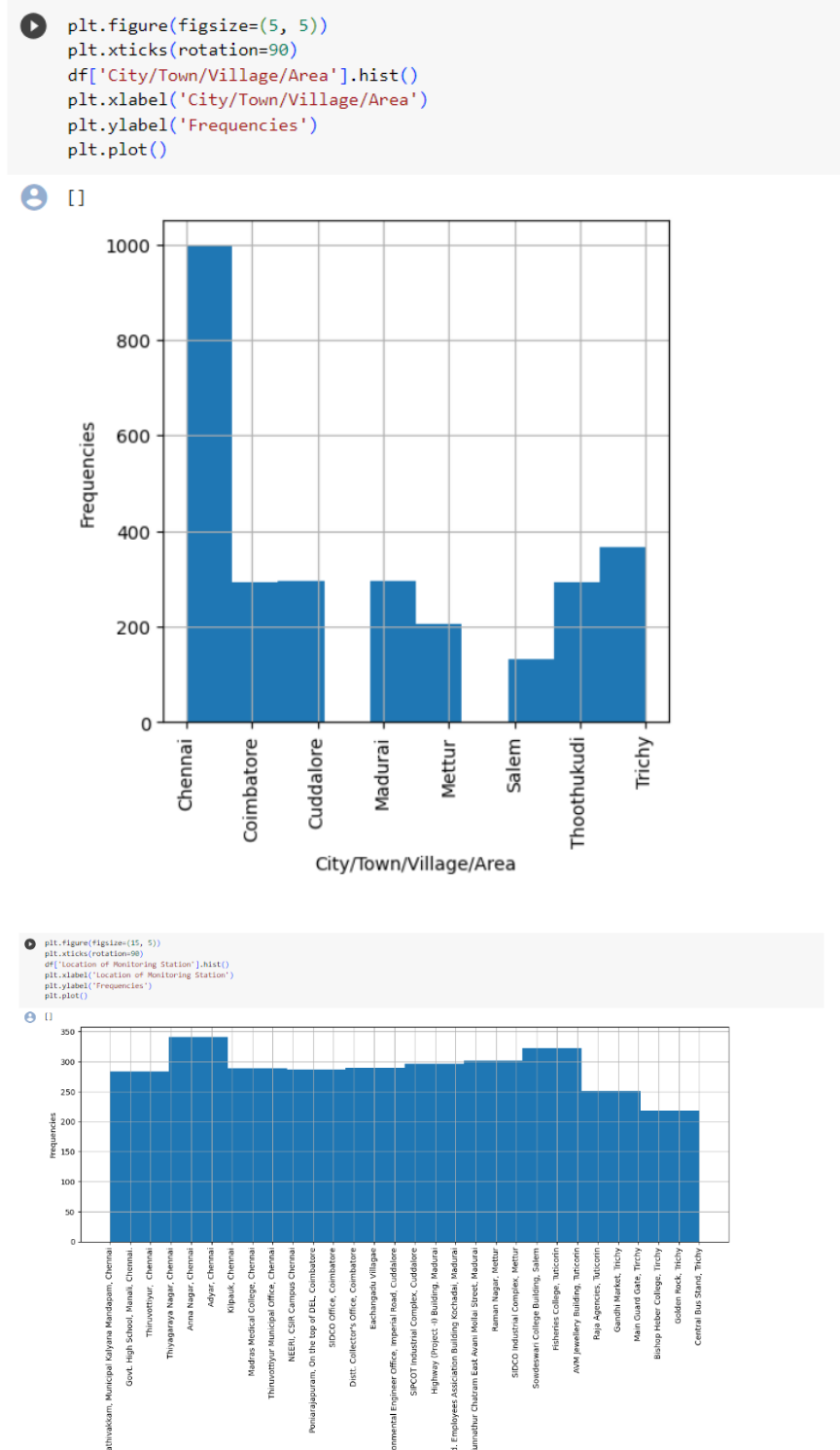


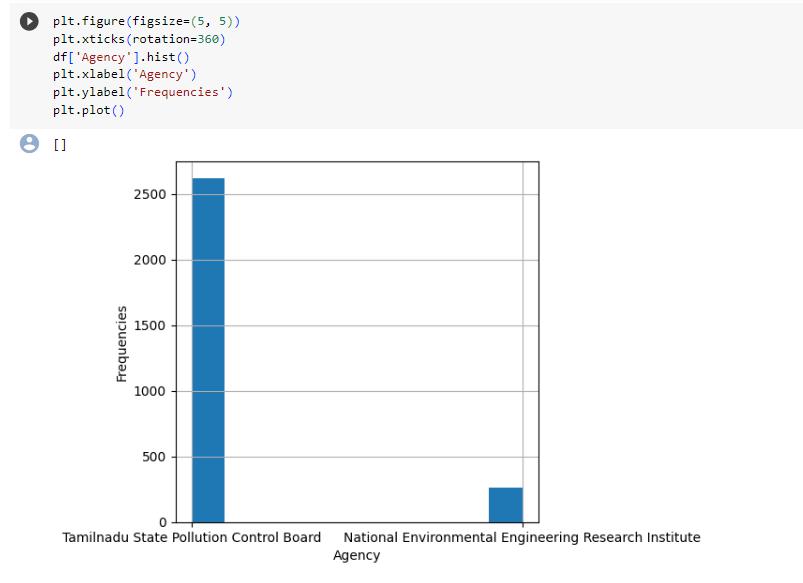
Evaluate unique values....

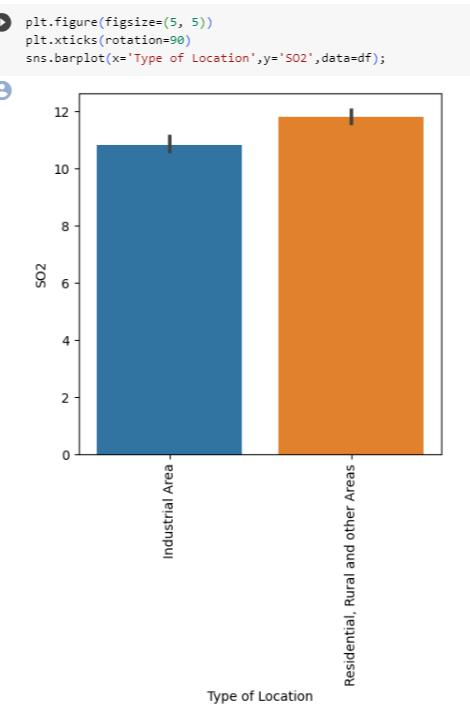


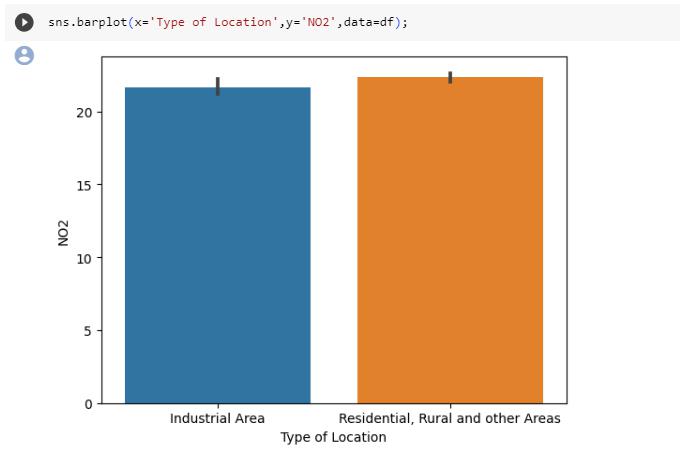
Pairplots..

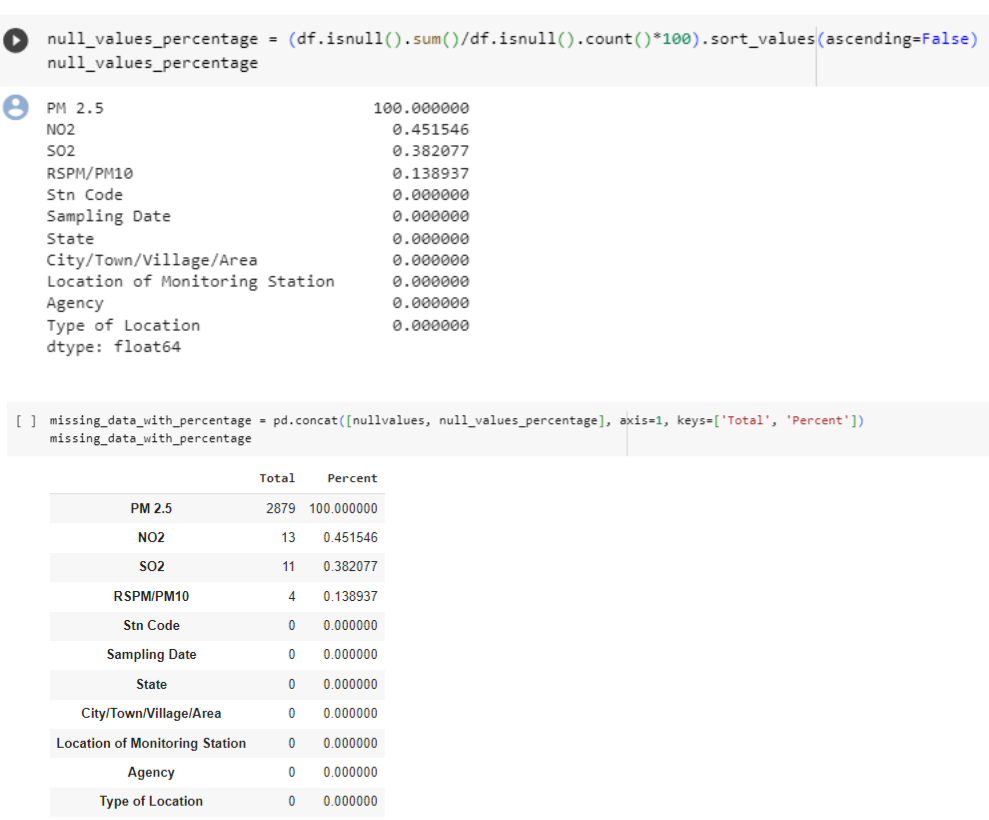
Visualizations..



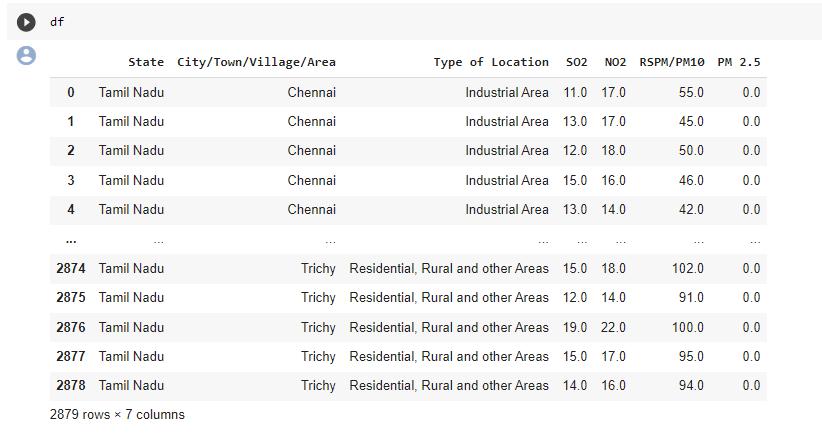




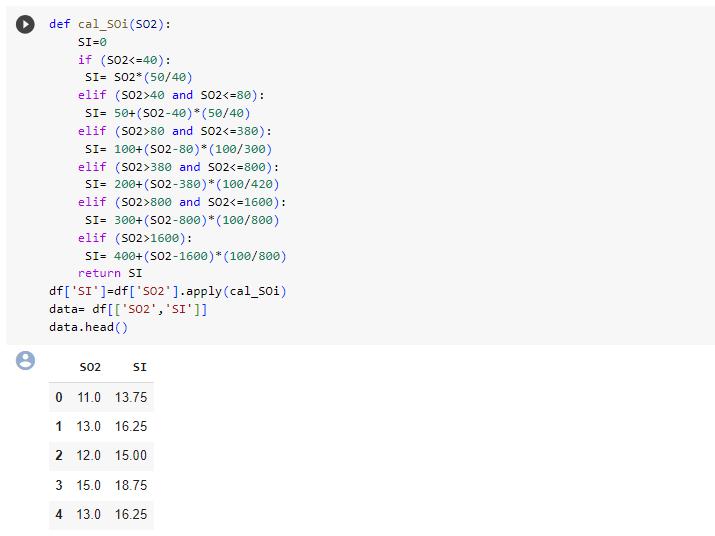


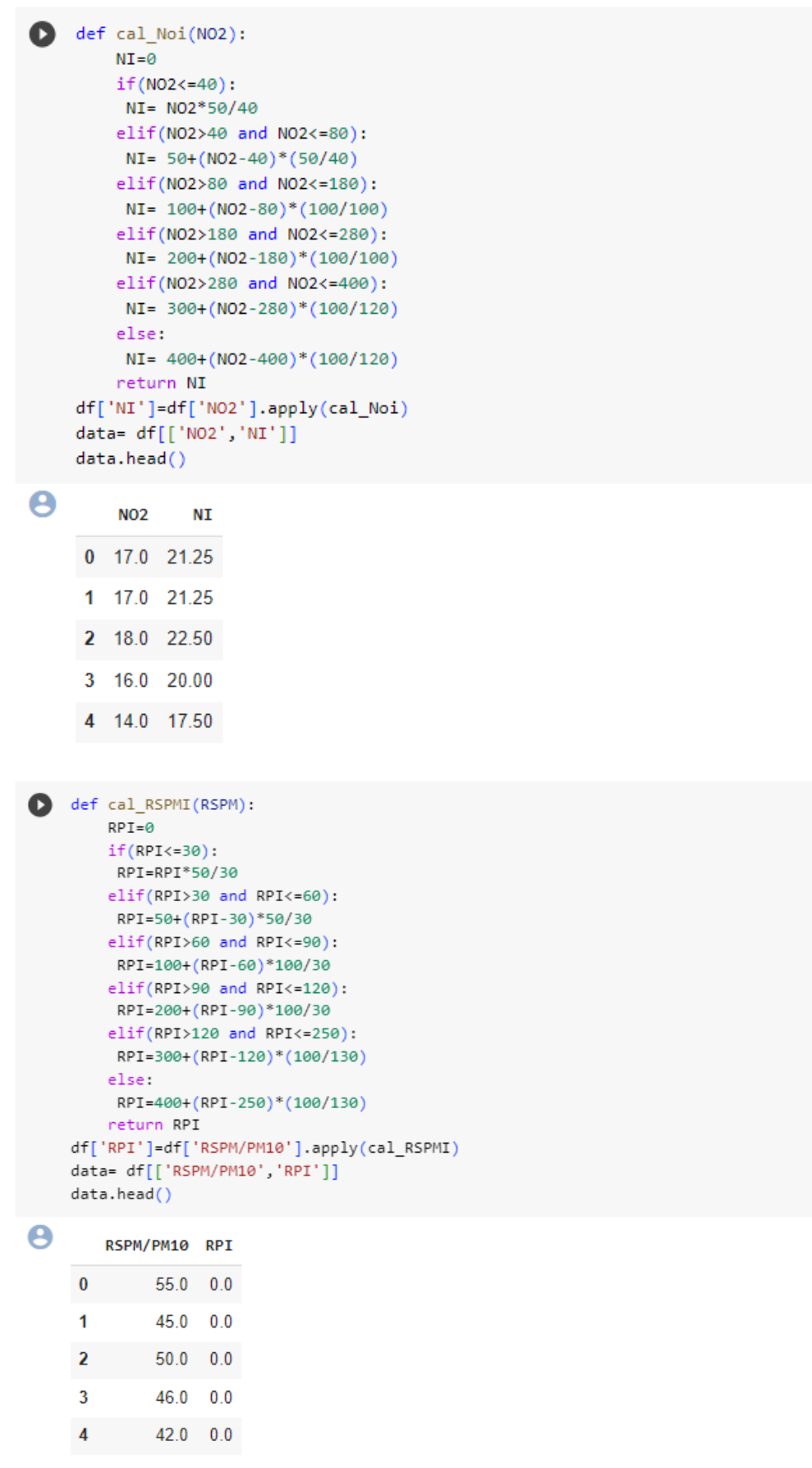
Null values percentages...

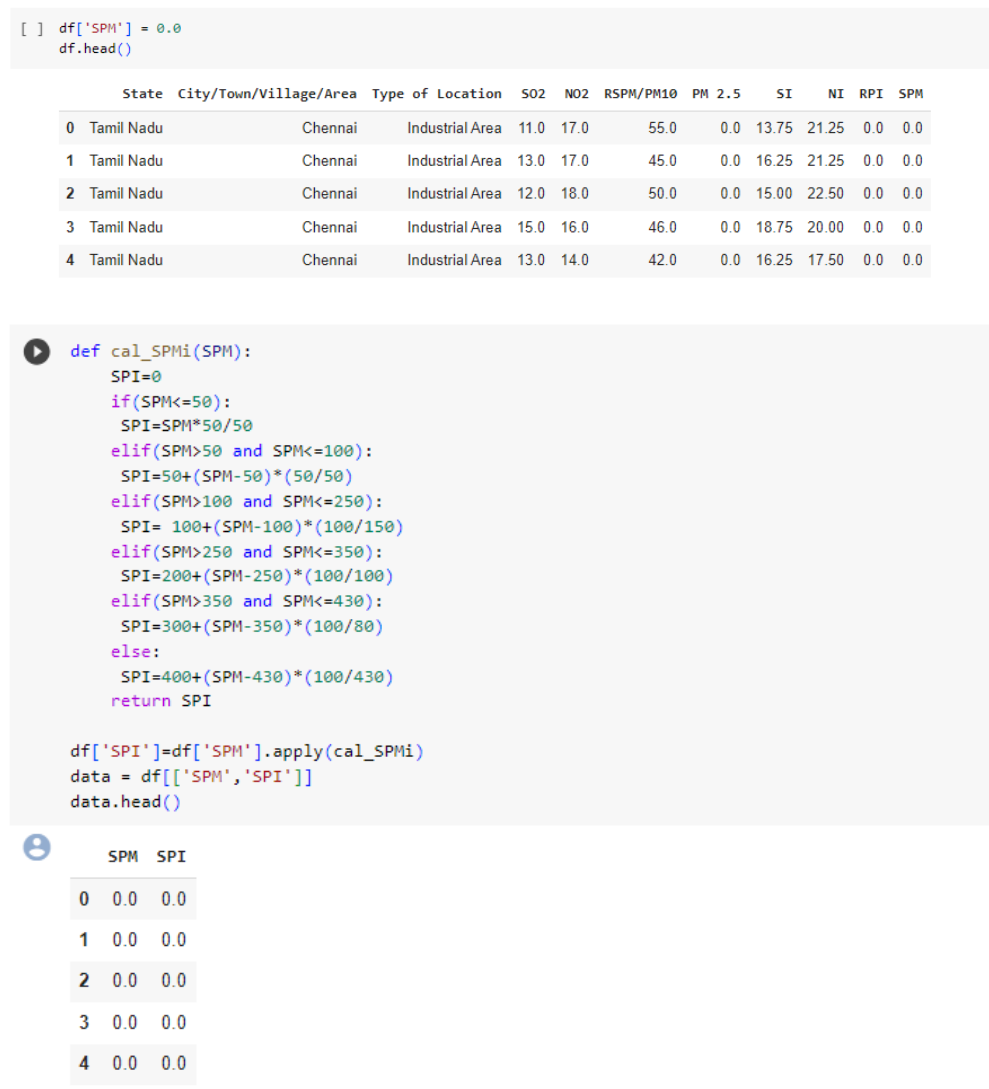
Dropping unused data columns...

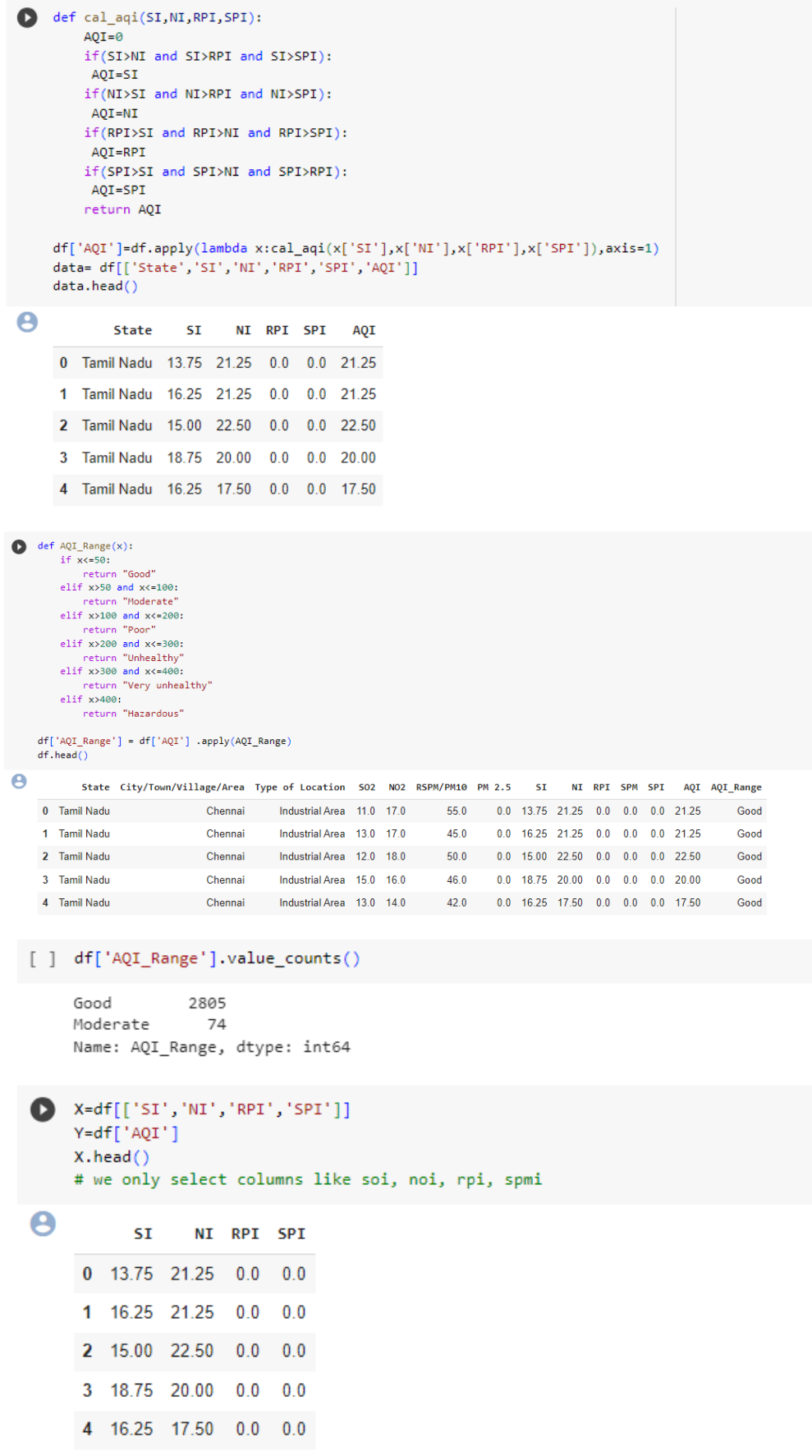


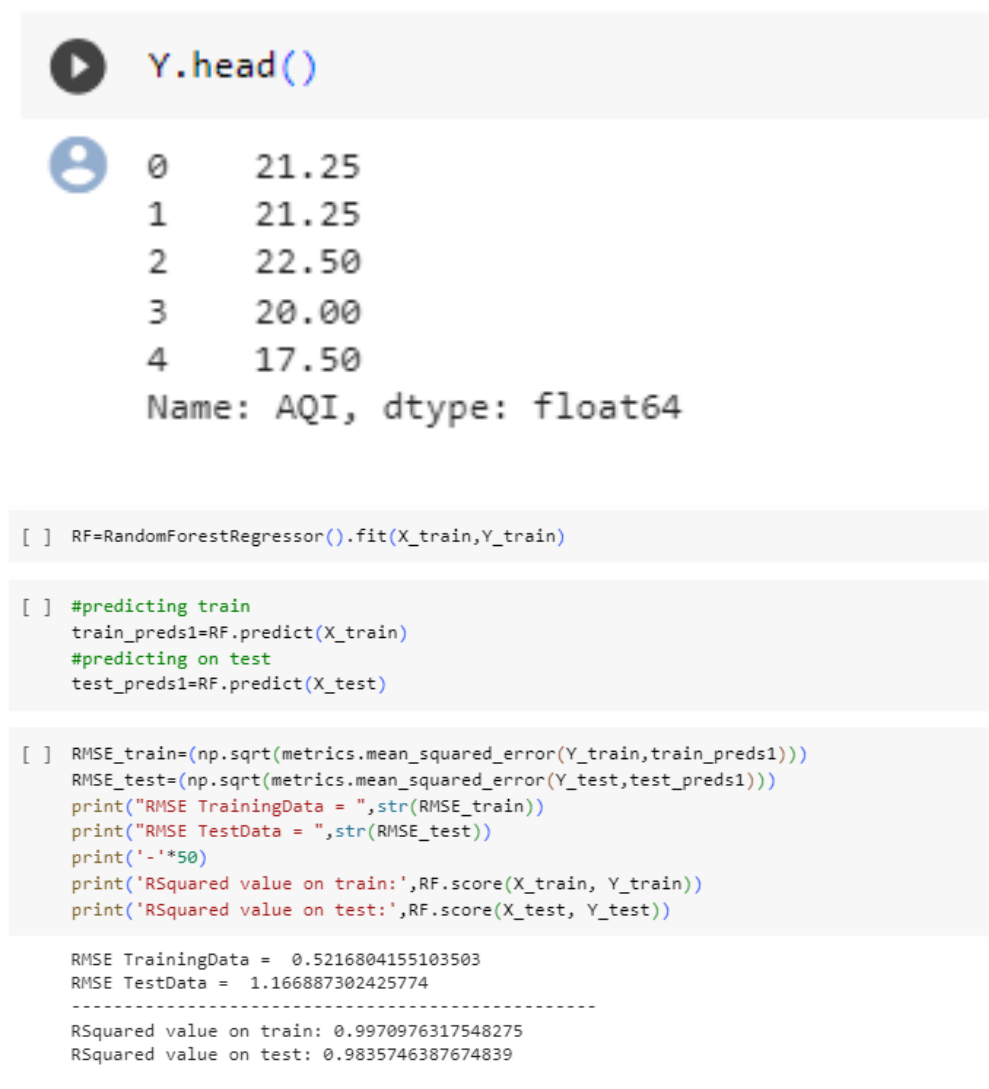
Evaluation :





Creating a new column of SPM

Calculating the AQI using data points...



**Code :**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import warnings

warnings.flterwarnings("ignore")

import plotly.express as px

from sklearn.preprocessing import LabelEncoder

from sklearn.mossssssssdel\_selection import train\_test\_splzt

from sklearn.linear\_model import LinearRegression

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

from sklearn import metrics

from sklearn.metrics import mean\_absolute\_error,mean\_squared\_error,r2\_score from sklearn.metrics import accuracy\_score,confusion\_matrix

df = pd.read\_csv("cpcb\_dly\_aq\_tamil\_nadu-2014.csv")

df

df.isnull().sum()

df.info()

df.nunique()

sns.pairplot(data=df)

nullvalues = df.isnull().sum().sort\_values(ascending=False)

nullvalues

null\_values\_percentage = (df.isnull().sum()/df.isnull().count()\*100).sort\_values(ascending=False)

null\_values\_percentage

missing\_data\_with\_percentage = pd.concat([nullvalues, null\_values\_percentage], axis=1, keys=['Total', 'Percent'])

missing\_data\_with\_percentage

df.drop(['Agency'],axis=1,inplace=True)

df.drop(['Stn Code'],axis=1,inplace=True)

df.drop(['Sampling Date'],axis=1,inplace=True)

df.drop(['Location of Monitoring Station'],axis=1,inplace=True

df.fllna(0, inplace=True)

def cal\_SOi(SO2):

SI=0

if (SO2<=40):

SI= SO2\*(50/40)

elif (SO2>40 and SO2<=80):

SI= 50+(SO2-40)\*(50/40)

elif (SO2>80 and SO2<=380):

SI= 100+(SO2-80)\*(100/300)

elif (SO2>380 and SO2<=800):

SI= 200+(SO2-380)\*(100/420)

elif (SO2>800 and SO2<=1600):

SI= 300+(SO2-800)\*(100/800)

elif (SO2>1600):

SI= 400+(SO2-1600)\*(100/800)

return SI

df['SI']=df['SO2'].apply(cal\_SOi)

data= df[['SO2','SI']]

data.head()

def cal\_Noi(NO2):

NI=0

if(NO2<=40):

NI= NO2\*50/40

elif(NO2>40 and NO2<=80):

NI= 50+(NO2-40)\*(50/40)

elif(NO2>80 and NO2<=180):

NI= 100+(NO2-80)\*(100/100)

elif(NO2>180 and NO2<=280):

NI= 200+(NO2-180)\*(100/100)

elif(NO2>280 and NO2<=400):

NI= 300+(NO2-280)\*(100/120)

else:

NI= 400+(NO2-400)\*(100/120)

return NI

df['NI']=df['NO2'].apply(cal\_Noi)

data= df[['NO2','NI']]

data.head()

def cal\_RSPMI(RSPM):

RPI=0

if(RPI<=30):

RPI=RPI\*50/30

elif(RPI>30 and RPI<=60):

RPI=50+(RPI-30)\*50/30

elif(RPI>60 and RPI<=90):

RPI=100+(RPI-60)\*100/30

elif(RPI>90 and RPI<=120):

RPI=200+(RPI-90)\*100/30

elif(RPI>120 and RPI<=250):

RPI=300+(RPI-120)\*(100/130)

else:

RPI=400+(RPI-250)\*(100/130)

return RPI

df['RPI']=df['RSPM/PM10'].apply(cal\_RSPMI)

data= df[['RSPM/PM10','RPI']]

data.head()

df['SPM'] = 0.0

df.head()

def cal\_SPMi(SPM):

SPI=0

if(SPM<=50):

SPI=SPM\*50/50

elif(SPM>50 and SPM<=100):

SPI=50+(SPM-50)\*(50/50)

elif(SPM>100 and SPM<=250):

SPI= 100+(SPM-100)\*(100/150)

elif(SPM>250 and SPM<=350)

SPI=200+(SPM-250)\*(100/100)

elif(SPM>350 and SPM<=430):

SPI=300+(SPM-350)\*(100/80)

else:

SPI=400+(SPM-430)\*(100/430)

return SPI

df['SPI']=df['SPM'].apply(cal\_SPMi)

data = df[['SPM','SPI']

data.head()

def cal\_aqi(SI,NI,RPI,SPI):

AQI=0

if(SI>NI and SI>RPI and SI>SPI):

AQI=SI

if(NI>SI and NI>RPI and NI>SPI):

AQI=NI

if(RPI>SI and RPI>NI and RPI>SPI):

AQI=RPI

if(SPI>SI and SPI>NI and SPI>RPI):

AQI=SPI

return AQI

df['AQI']=df.apply(lambda x:cal\_aqi(x['SI'],x['NI'],x['RPI'],x['SPI']),axis=1)

data= df[['State','SI','NI','RPI','SPI','AQI']]

data.head()

def AQI\_Range(x):

if x<=50:

return "Good"

elif x>50 and x<=100:

return "Moderate"

elif x>100 and x<=200:

return "Poor"

elif x>200 and x<=300:

return "Unhealthy"

elif x>300 and x<=400:

return "Very unhealthy"

elif x>400:

return "Hazardous"

df['AQI\_Range'] = df['AQI'] .apply(AQI\_Range)

df.head()

df['AQI\_Range'].value\_counts()

X=df[['SI','NI','RPI','SPI']]

Y=df['AQI']

X.head()

* we only select columns like soi, noi, rpi, spmi

Y.head()

X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(X,Y,test\_size=0.2,random\_state=70)

print(X\_train.shape,X\_test.shape,Y\_train.shape,Y\_test.shape)

RF=RandomForestRegressor().ft(X\_train,Y\_train)

#predicting train

train\_preds1=RF.predict(X\_train)

#predicting on test

test\_preds1=RF.predict(X\_test)

RMSE\_train=(np.sqrt(metrics.mean\_squared\_error(Y\_train,train\_preds1))

RMSE\_test=(np.sqrt(metrics.mean\_squared\_error(Y\_test,test\_preds1)))

print("RMSE TrainingData = ",str(RMSE\_train))

print("RMSE TestData = ",str(RMSE\_test))

print('-'\*50)

print('RSquared value on train:',RF.score(X\_train, Y\_train))

print('RSquared value on test:',RF.score(X\_test, Y\_test))