**Phase 3: Development**

**Project : Air Quality Analysis and Prediction in Tamil Nadu**

1.Introduction:

The objective of the project is to employ data science techniques to analyze air quality in various regions of Tamil Nadu , India. Air quality analysis and prediction in data science using Python is a field that leverages computational techniques to understand, assess, and forecast the quality of the air we breathe. By collecting and analyzing data from various sources, such as monitoring stations and sensors, the objective is to gain insights into pollutant levels, weather conditions, and their impact on air quality.



2.Data Collection:

Gathered air quality data from source like Kaggle(<http://www.kaggle.com/data>).This data typically includes information on pollutants like PM2.5, PM10, ozone, nitrogen dioxide, etc., as well as meteorological variables like temperature, humidity, and wind speed.

3.Details about columns:

We typically work with various data columns that provide information about air quality. Here are some common columns you might encounter in air quality datasets:

Timestamp/Date and Time: This column records the date and time of each data point, allowing you to track air quality changes over time.

Location/Station: Indicates the monitoring station or geographical location where the air quality data was collected. Multiple stations may be used for a broader analysis.

Pollutants: Different columns for various pollutants like:

- PM2.5 and PM10: Particulate matter with diameters of 2.5 and 10 micrometers, respectively.

- NO2 (Nitrogen Dioxide): A common air pollutant.

- SO2 (Sulfur Dioxide): Another common air pollutant.

- CO (Carbon Monoxide): Measures carbon monoxide levels.

- O3 (Ozone): Monitors ozone concentrations.

Other Environmental Factors: Columns related to other environmental factors like traffic density, industrial activities, or proximity to pollution sources.

4. Data Preprocessing:

Cleaned and preprocessed the collected data, handling missing values, outliers, and data inconsistencies.

5. Exploratory Data Analysis (EDA):

Conducted EDA to gain insights into the data. Visualized the data to identify trends, patterns, and correlations between air quality and environmental factors.

6. Model Selection:

Chosen appropriate machine learning or statistical models for air quality prediction.

7. Model Training:

Spliced the data into training and testing sets. Trained the selected models on the training data and tuned their hyperparameters to optimize performance.

8. Visualization:

Visualized the model's predictions and compare them to observed air quality data to assess the model's accuracy and reliability.

9.Libraries installed:

Python libraries like Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn, and TensorFlow or Torch for deep learning can be used for various stages of this process. Additionally, platforms like Jupyter Notebooks are often employed for interactive data analysis and model development.

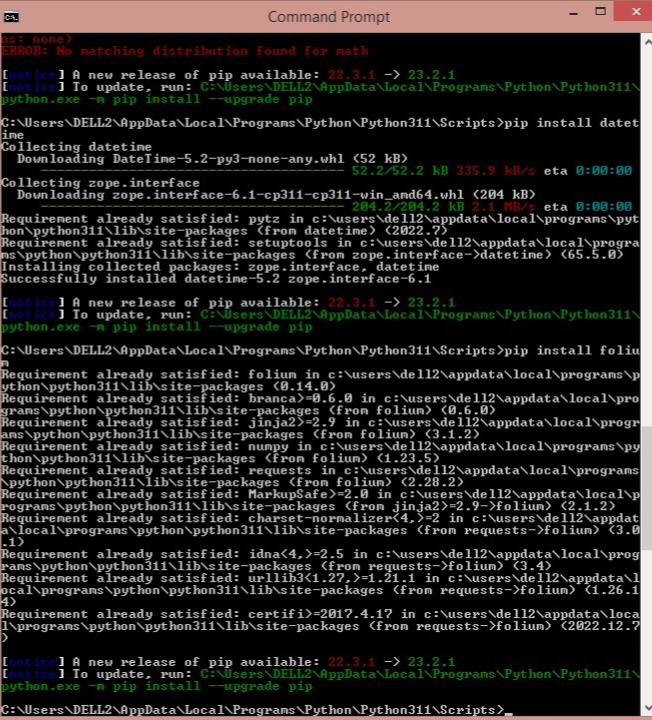
10.Accuracy and Metrics:

Evaluated model performance using metrics like Gaussian Naïve Bayes Classifier. GaussianNB to implement the Gaussian Naïve Bayes algorithm for classification.

CODINGS :

|  |
| --- |
| pip install numpy  Pip install pandas  pip install datetime  pip install geopandas  pip install folium  pip install seaborn |

1. Installing the required libraries .



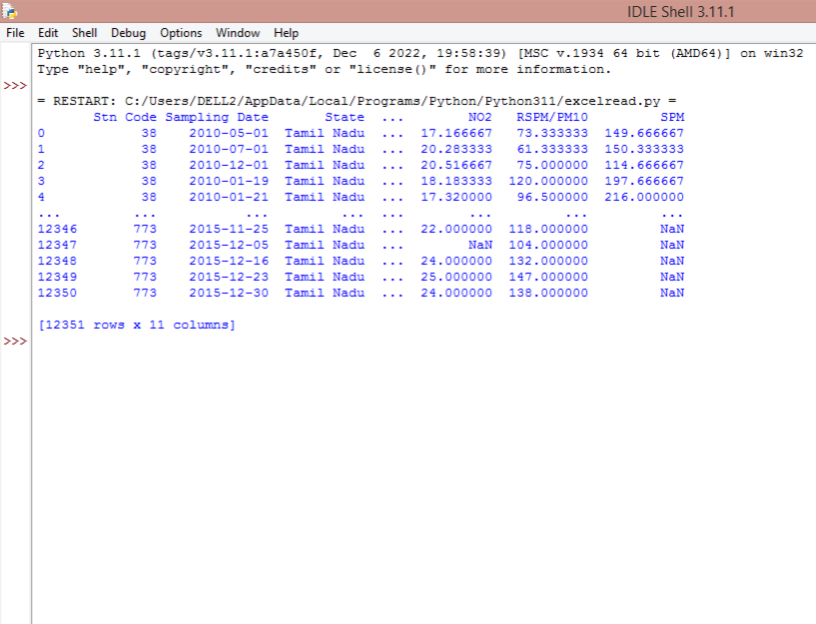
LOADING DATASET :

Dataset Reading :

* Pandas is a Python library used for working with data sets.
* Pandas module read\_excel() function to read the excel file data into a DataFrame object.

|  |
| --- |
| import pandas as pd  df=pd.read\_excel("C:\\Users\\DELL2\\Downloads\\Air\_Quality.xlsx")  print(df) |

OUTPUT :



Data Cleaning :

* Data cleansing, also referred to as data cleaning or data scrubbing, is the process of fixing incorrect, incomplete, duplicate or otherwise erroneous data in a data set.
* The drop() method removes the specified row or column.
* The ‘agency’ column is removed from the dataset as its contains duplicate columns.

|  |
| --- |
| df.drop(columns='Agency',inplace=True)  print(df) |

OUTPUT :

|  |
| --- |
| = RESTART: C:/Users/DELL2/AppData/Local/Programs/Python/Python311/excelread.py =  Stn Code Sampling Date State ... NO2 RSPM/PM10 SPM  0 38 2010-05-01 Tamil Nadu ... 17.166667 73.333333 149.666667  1 38 2010-07-01 Tamil Nadu ... 20.283333 61.333333 150.333333  2 38 2010-12-01 Tamil Nadu ... 20.516667 75.000000 114.666667  3 38 2010-01-19 Tamil Nadu ... 18.183333 120.000000 197.666667  4 38 2010-01-21 Tamil Nadu ... 17.320000 96.500000 216.000000  [5 rows x 11 columns]  Stn Code Sampling Date State ... NO2 RSPM/PM10 SPM  0 38 2010-05-01 Tamil Nadu ... 17.166667 73.333333 149.666667  1 38 2010-07-01 Tamil Nadu ... 20.283333 61.333333 150.333333  2 38 2010-12-01 Tamil Nadu ... 20.516667 75.000000 114.666667  3 38 2010-01-19 Tamil Nadu ... 18.183333 120.000000 197.666667  4 38 2010-01-21 Tamil Nadu ... 17.320000 96.500000 216.000000  ... ... ... ... ... ... ... ...  12346 773 2015-11-25 Tamil Nadu ... 22.000000 118.000000 NaN  12347 773 2015-12-05 Tamil Nadu ... NaN 104.000000 NaN  12348 773 2015-12-16 Tamil Nadu ... 24.000000 132.000000 NaN  12349 773 2015-12-23 Tamil Nadu ... 25.000000 147.000000 NaN  12350 773 2015-12-30 Tamil Nadu ... 24.000000 138.000000 NaN  [12351 rows x 10 columns] |

Train and testing the Data:

* The modelling data is divided into training and testing data.
* The simplest way to split the modelling dataset into training and testing sets is to assign 2/3 data points to the former and the remaining one-third to the latter.
* The dataframe gets divided into X\_train,X\_test , y\_train and y\_test.
* X\_train and y\_train sets are used for training and fitting the model.
* The X\_test and y\_test sets are used for testing the model if it's predicting the right outputs/labels.

|  |
| --- |
| #separate dependent and independent variable  X=df.drop(columns='State',axis=1)  y=df['State']  print(y)  #splitting dataset into training and test  X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.20,random\_state=0)  #shape  df.shape  #shape of train and test data  print(X\_train.shape)  print(X\_test.shape) |

OUTPUT :

|  |
| --- |
| 0 Tamil Nadu  1 Tamil Nadu  2 Tamil Nadu  3 Tamil Nadu  4 Tamil Nadu  ...  12347 Tamil Nadu  12348 Tamil Nadu  12349 Tamil Nadu  12350 Tamil Nadu  Name: State, Length: 12351, dtype: object  (12351, 11)  (9880, 10)  (2471, 10) |

Accuracy and Metrics :

* Accuracy can also be defined as the ratio of the number of correctly classified cases to the total of cases under evaluation.
* The best value of accuracy is 1 and the worst value is 0.
* Gaussian Naive Bayes (GNB) is a classification technique used in Machine Learning (ML) based on the probabilistic approach and Gaussian distribution.
* Gaussian Naïve Bayes is the extension of naïve Bayes.
* While other functions are used to estimate data distribution, Gaussian or normal distribution is the simplest to implement as you will need to calculate the mean and standard deviation for the training data.

|  |
| --- |
| import pandas as pd  from sklearn.model\_selection import train\_test\_split  from sklearn.naive\_bayes import GaussianNB  from sklearn import metrics  df=pd.read\_excel("C:\\Users\\DELL2\\Downloads\\Air\_Quality.xlsx")  X=df.drop(columns='State',axis=1)  y=df['State']  X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.20,random\_state=0)  print(X)  gnb=GaussianNB()  gnb.fit(X\_train,y\_train)  y\_pred=gnb.predict(X\_test)  print("accuracy is :",metrics.accuracy\_score(y\_test,y\_pred)\*100) |

OUTPUT:

