**Phase 4 : Development Part 2**

**Project 9: Air Quality Analysis and Prediction in Tamil Nadu**

**DESCRIPTION:**

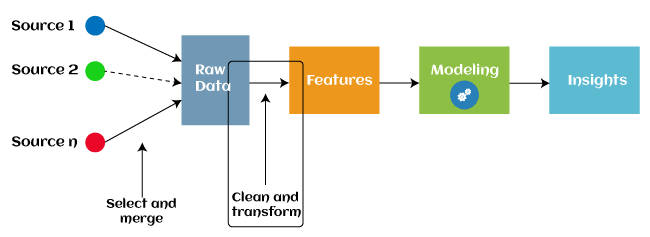
The objective of this project is to employ data science techniques to analyze air quality in various regions of Tamil Nadu, India. The project aims to analyze and visualize air quality data from monitoring stations in Tamil Nadu. The objective is to gain insights into air pollution trends, identify areas with high pollution levels.

**FEATURE ENGINEERING :**

* Feature engineering is the pre-processing step of machine learning, which extracts features from raw data**.**
* It helps to represent an underlying problem to predictive models in a better way, which as a result, improve the accuracy of the model for unseen data.

**STEPS INVOLVED IN FEATURE ENGINEERING ARE:**

* Selecting, extracting, and transforming the most relevant features from the available data
* Manipulating the data set to improve machine learning model training
* Creating new variables that aren't in the training set
* Remodelling raw data into a format that successfully represents the underlying patterns within the data
* Ensuring the model is flexible in the variety of data it can ingest
* Ensuring variables are on the same scale
* Making the model easier to understand
* Improving accuracy
* Avoiding computational errors

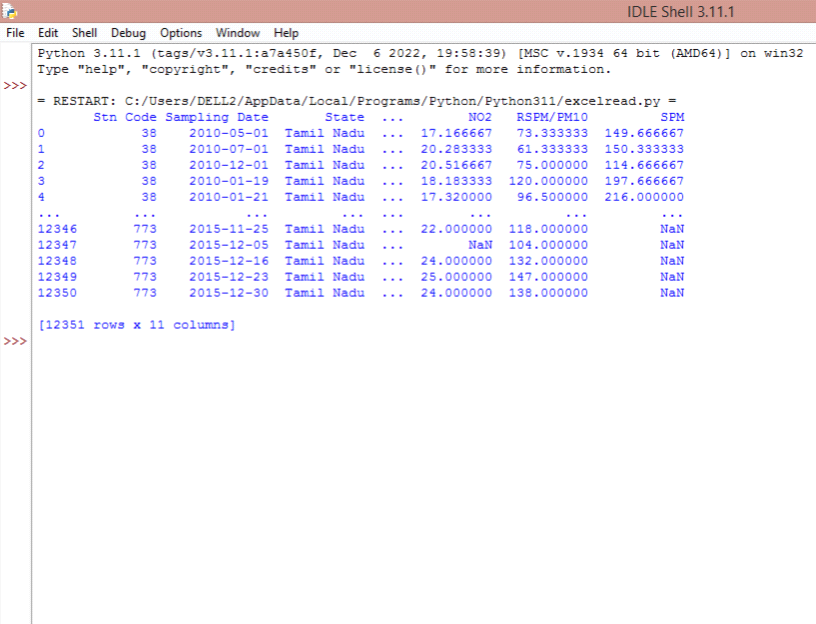


**EXPLORATORY DATA ANALYSIS:**

* EDA is a phenomenon under data analysis used for gaining a better understanding of data aspects like main features of data, variables and relationships that hold between them ,identifying which variables are important for our problem.
* It uses data manipulation techniques and several statistical tools to describe and understand the relationship between variables and how these can impact business.

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

**OUTPUT:**



**RENAMING THE COLUMNS :**

|  |
| --- |
| df.rename(columns={'City/Town/Village/Area':'City'})  #RENAMING COLUMN  d=df.rename(columns={'City/Town/Village/Area':'City'})  s=df.columns  print(s)  #TITLE OF COLUMNS  print(d.columns) |

**OUTPUT** :

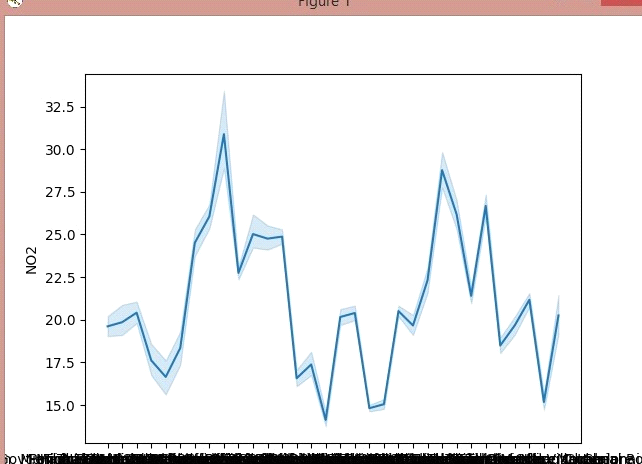
|  |
| --- |
| Index(['Stn Code', 'Sampling Date', 'State', 'City','Location of Monitoring Station', 'Agency', 'Type of Location', 'SO2', 'NO2', 'RSPM/PM10', 'SPM'],dtype='object') |

**DATA VISUALIZATION :**

**LINECHART:**

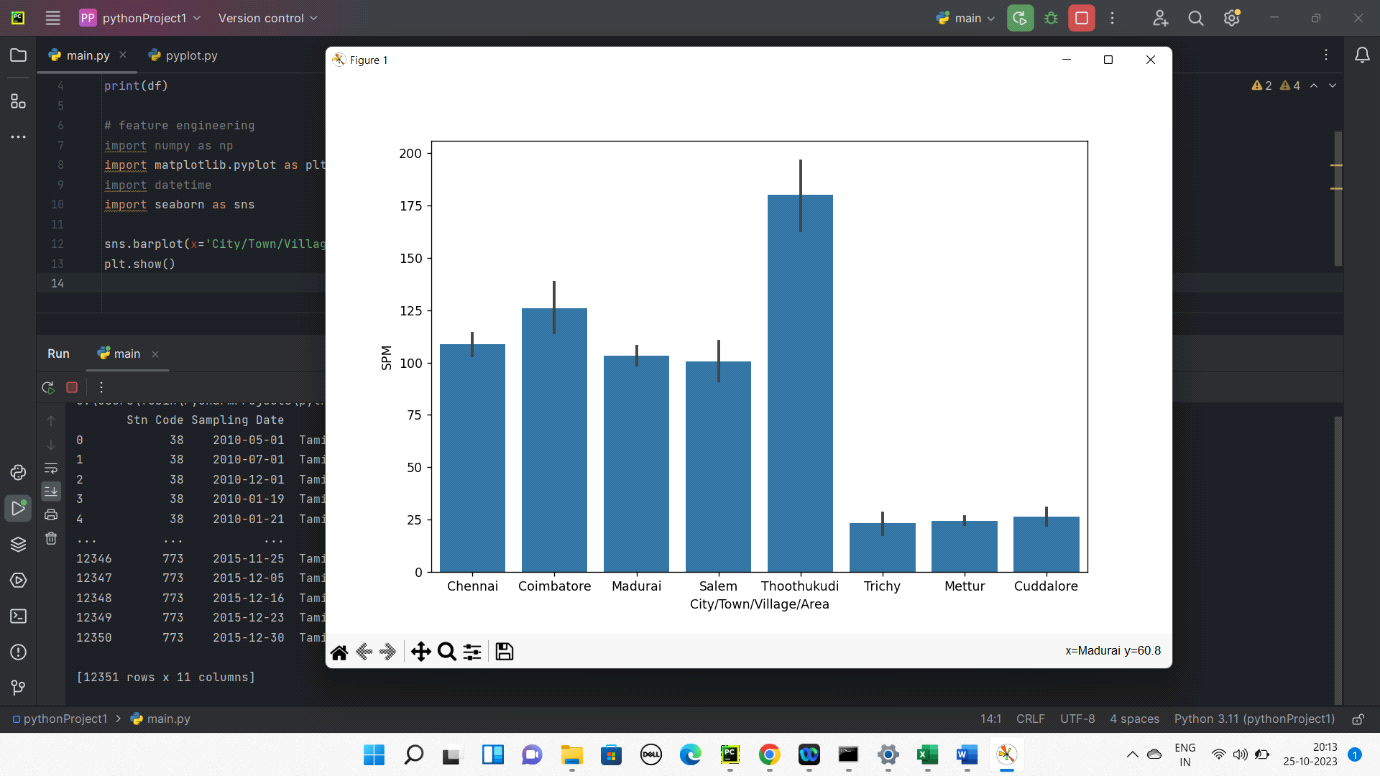
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.lineplot(x='Location of Monitoring Station', y='NO2', data=df)  
plt.show()

**OUTPUT:**



**BARCHART:**

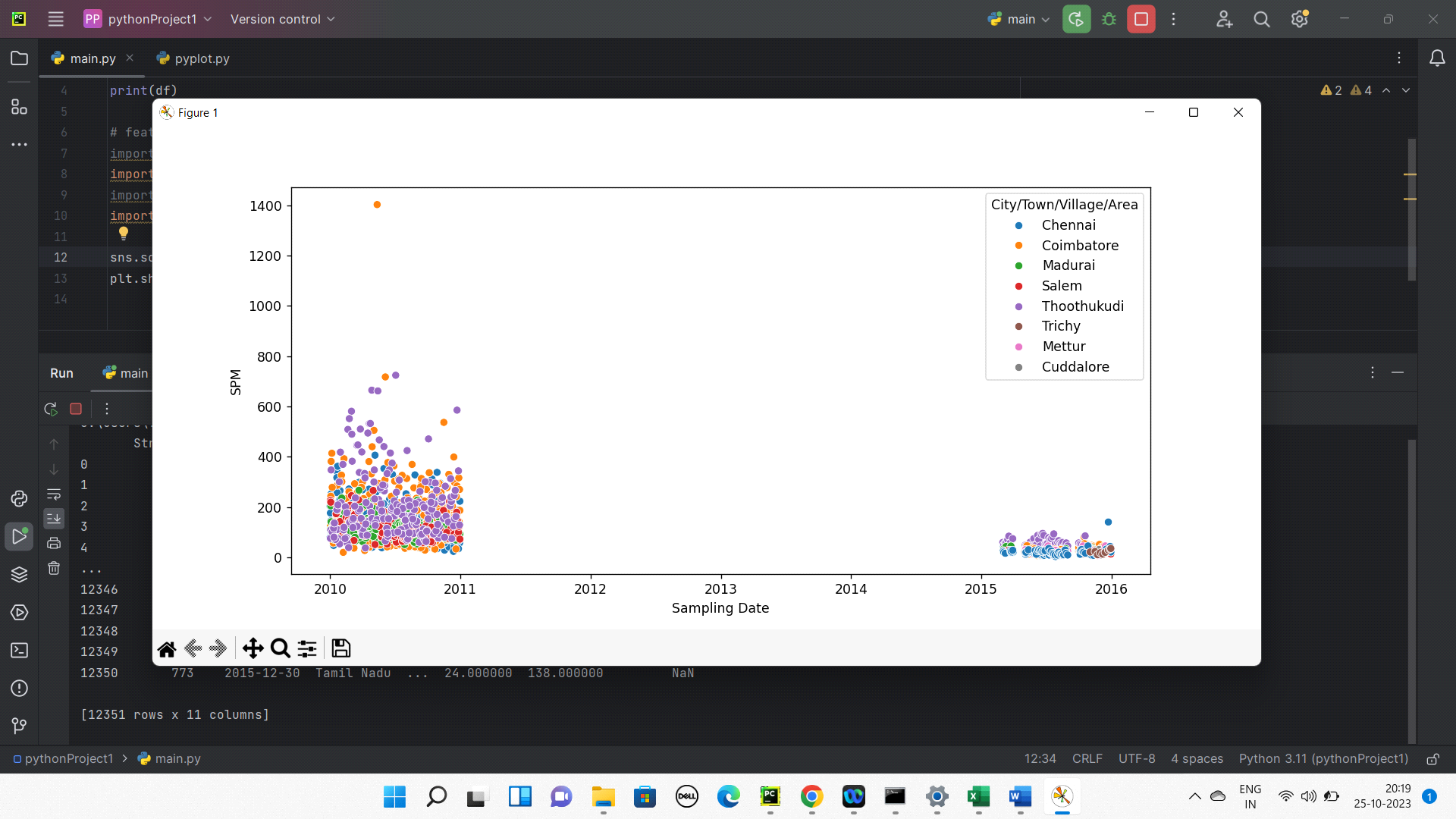
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.barplot(x='City/Town/Village/Area', y='SPM', data=df)  
plt.show()



**SCATTER CHART:**

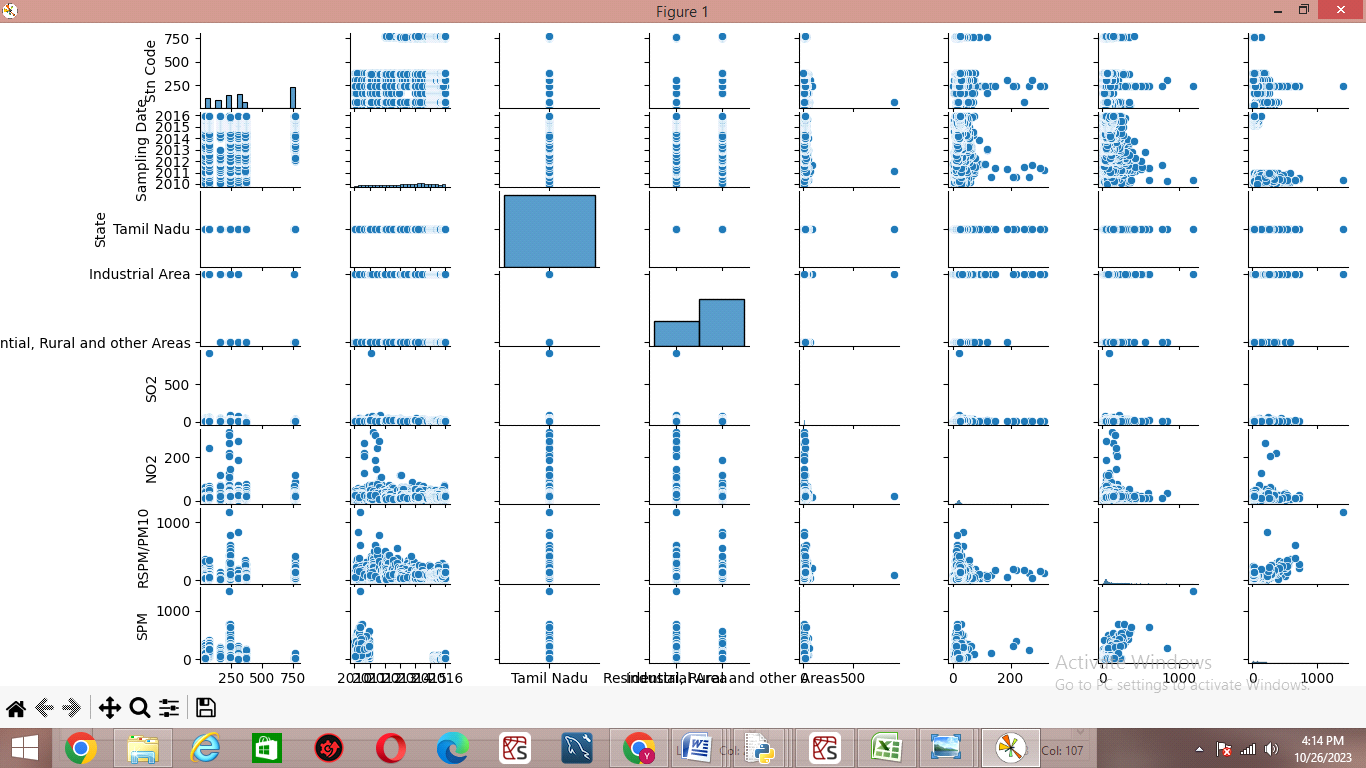
import matplotlib.pyplot as plt  
import seaborn as sns  
  
sns.scatterplot(x='Sampling Date', y='SPM', data=df, hue='City/Town/Village/Area')  
plt.show()

**OUTPUT:**



**MULTI VALUES CHART:**

|  |
| --- |
| import pandas as pd  import matplotlib.pyplot as plt  import seaborn as sns  sns.pairplot(df,vars=['Stn Code','Sampling Date','State','Type of Location','SO2','NO2','RSPM/PM10','SPM'])  plt.show() |



**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.

**CODING:**

|  |
| --- |
| #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.

|  |
| --- |
| 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:**

