PHASE 5 AIR QUALITY ANALYSIS AND PREDICTION IN TAMILNADU

ABSTRACT

The project's aim is to examine and create visual representations of air quality information collected from monitoring stations in Tamil Nadu. Its purpose is to uncover patterns in air pollution, pinpoint regions with elevated pollution levels, and establish a forecasting model for estimating RSPM/PM10 levels by considering SO2 and NO2 levels. This endeavor encompasses setting clear goals, formulating an analysis strategy, choosing suitable visualization methods, and constructing a predictive model utilizing Python and pertinent libraries.

INTRODUCTION

Air quality is a critical environmental factor that directly impacts the health and well-being of communities. Monitoring and assessing air quality is essential for making informed decisions and taking actions to mitigate pollution. In this analysis, we focus on understanding and visualizing air quality data, specifically the concentration of various pollutants such as Sulfur Dioxide (SO2), Nitrogen Dioxide (NO2), and Respirable Suspended Particulate Matter (RSPM), in different cities, towns, and areas.

Objectives:

The objectives of this analysis include:

1. Calculating individual pollutant indices (SI, NI, and RPI) based on established Indian government standards.
2. Deriving the overall Air Quality Index (AQI) as per these standards to provide a comprehensive assessment of air quality.
3. Visualizing and understanding the distribution of these pollutant levels and AQI across various locations.
4. Exploring trends and patterns in AQI over time to identify any long-term changes.

The code provided in this analysis utilizes libraries such as NumPy, Pandas, Matplotlib, and Seaborn to process and visualize air quality data. It also defines functions to calculate pollutant indices and AQI based on the pollutants' concentration levels. The analysis aims to offer insights into the air quality conditions in different regions and provide a basis for informed decision-making and policy development.

Analysis Approach:

The analysis approach involves the following steps:

1. **Data Preprocessing**: The code starts by importing the necessary libraries, reading the air quality data from a CSV file, and performing data preprocessing tasks like handling missing values.
2. **Calculating Individual Pollutant Indices (SI, NI, RPI)**: The code defines functions to calculate individual pollutant indices for SO2, NO2, and RSPM based on Indian government standards. These indices are crucial components of the AQI.
3. **Calculating AQI**: Another function is defined to calculate the AQI for each data point using the calculated pollutant indices.
4. **Grouping and Visualization**: The code groups the data by location and calculates average pollutant levels and AQI values. It then uses Seaborn to create bar plots to visualize the average levels by location.
5. **Time Series Analysis**: The code conducts time series analysis by plotting the AQI over time to identify trends and variations.

Visualization Techniques:

The code uses various visualization techniques to convey the analysis results:

- Bar plots: These are used to show the average pollutant levels (SO2, NO2, RSPM, AQI) by location.
- Line plot: A time series line plot is used to visualize the AQI over time.

Code Implementation:

We have provided the code snippets for data preprocessing, pollutant index calculations, AQI calculations, and visualization. The code reads the data, processes it, calculates various indices, and creates insightful visualizations.

- Bar plots: Bar plots displaying the average levels of SO2, NO2, RSPM, and AQI for different locations in Tamil Nadu, providing a clear comparison of air quality in various areas.
- Line plot: A time series line plot showing the AQI over time, which helps identify trends in air quality in Tamil Nadu.

Insights:

The analysis provides valuable insights into air pollution trends and pollution levels in Tamil Nadu:

1. **Geographic Variations:** The bar plots reveal variations in air quality across different locations, highlighting areas with higher or lower pollution levels.
2. **Temporal Trends:** The time series analysis demonstrates how air quality has evolved over time, whether it has improved or deteriorated.
3. **AQI Interpretation:** By calculating and visualizing the AQI, the analysis provides a single, easy-to-understand metric for assessing air quality.
4. **Data-Driven Decision Making:** The project equips policymakers, researchers, and the public with data-driven insights to make informed decisions regarding pollution control and public health in Tamil Nadu.

DESIGN THINKING

Step 1: Data Loading

The provided code imports necessary Python libraries for data visualization, sets the default figure size, suppresses warnings, reads a CSV file that is the dataset provided into a DataFrame .

```
import numpy as np
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import matplotlib.pyplot as plt
import seaborn as sns
```

```
plt.rcParams['figure.figsize'] = (10, 7)

# Warnings
import warnings
warnings.filterwarnings('ignore')

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will
list the files in the input directory


import os

# Any results you write to the current directory are saved as output.
data=pd.read_csv("C:\\Users\\Sandhya\\Downloads\\air quality.csv")
data.fillna(0, inplace=True)
data.head()
```

Step 2: Data Preprocessing

This entails addressing missing values, outliers, and inaccurate data entries through processes like filling in missing values, eliminating outliers, or applying suitable statistical techniques to enhance data integrity. Subsequently, standardizing or normalizing numerical attributes is performed to bring them to a consistent scale, simplifying the model's ability to discern patterns within the data.

```
# Import libraries
import pandas as pd
import matplotlib.pyplot as plt

# Create a DataFrame from the provided data
data = pd.read_csv('C:\\Users\\Sandhya\\Downloads\\air quality.csv')

# Explore the data
print(data.head())
print(data.info())

# Access specific columns
so2_column = data['SO2']
no2_column = data['NO2']
rspm_column = data['RSPM']

# Data Cleaning
# Check for missing values
print("Missing value count")
print(data.isnull().sum())

# Replace the missing values with zeros
columns_to_fill = ['SO2', 'NO2', 'RSPM']
mean_values = data[columns_to_fill].mean()
data[columns_to_fill] = data[columns_to_fill].fillna(mean_values)
column_name = 'PM 2.5'
```

```
data[column_name].fillna(0, inplace=True)

# Save the preprocessed data to a new file
data.to_csv('C:\\Users\\Sandhya\\Downloads\\air quality.csv', index=False)

# Check missing values again to verify they are handled
print("Missing values count after imputation")
print(data.isnull().sum())

# Scatter plot using Matplotlib
x = data['SO2']
y = data['NO2']
c = data['RSPM']

plt.scatter(x, y, c=c, marker='o', cmap='viridis')
plt.xlabel("SO2 Level")
plt.ylabel("NO2 Level")
plt.title("Scatter Plot of SO2 vs. NO2 with RSPM/PM10 Color")
plt.colorbar(label="RSPM/PM10 Level")
plt.grid(True)
plt.show()

# Scatter plot using Matplotlib
x = data['SO2']
y = data['NO2']
c = data['RSPM']

plt.scatter(x, y, c=c, s=c, marker='o', cmap='viridis')
plt.xlabel("SO2 Level")
plt.ylabel("NO2 Level")
plt.title("Scatter Plot of SO2 vs. NO2 with RSPM/PM10 Color and Size")
plt.colorbar(label="RSPM/PM10 Level")
plt.grid(True)
plt.show()
```

OUTPUT

	Stn Code	Date	State	...	NO2	RSPM	PM 2.5
0	38	01-02-2014	Tamil Nadu	...	17.0	55.0	NaN
1	38	01-07-2014	Tamil Nadu	...	17.0	45.0	NaN
2	38	21-01-2014	Tamil Nadu	...	18.0	50.0	NaN
3	38	23-01-2014	Tamil Nadu	...	16.0	46.0	NaN
4	38	28-01-2014	Tamil Nadu	...	14.0	42.0	NaN
...	...	...	...	...	...	...	...
2874	773	12-03-2014	Tamil Nadu	...	18.0	102.0	NaN
2875	773	12-10-2014	Tamil Nadu	...	14.0	91.0	NaN
2876	773	17-12-2014	Tamil Nadu	...	22.0	100.0	NaN
2877	773	24-12-2014	Tamil Nadu	...	17.0	95.0	NaN

2878 773 31-12-2014 Tamil Nadu ... 16.0 94.0 NaN

[2879 rows x 11 columns]

	Stn Code	Date	State	...	NO2	RSPM	PM 2.5
0	38	01-02-2014	Tamil Nadu	...	17.0	55.0	NaN
1	38	01-07-2014	Tamil Nadu	...	17.0	45.0	NaN
2	38	21-01-2014	Tamil Nadu	...	18.0	50.0	NaN
3	38	23-01-2014	Tamil Nadu	...	16.0	46.0	NaN
4	38	28-01-2014	Tamil Nadu	...	14.0	42.0	NaN

[5 rows x 11 columns]

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 2879 entries, 0 to 2878

Data columns (total 11 columns):

#	Column	Non-Null Count	Dtype
0	Stn Code	2879 non-null	int64
1	Date	2879 non-null	object
2	State	2879 non-null	object
3	City/Town/Village/Area	2879 non-null	object
4	Location of Monitoring Station	2879 non-null	object
5	Agency	2879 non-null	object
6	Type of Location	2879 non-null	object
7	SO2	2868 non-null	float64
8	NO2	2866 non-null	float64
9	RSPM	2875 non-null	float64
10	PM 2.5	0 non-null	float64

dtypes: float64(4), int64(1), object(6)

memory usage: 247.5+ KB

None

Missing value count

Stn Code 0

Date 0

State 0

City/Town/Village/Area 0

Location of Monitoring Station 0

Agency 0

Type of Location

SO2 11

NO2 13

RSPM 4

PM 2.5 2879

dtype: int64

Missing values count after imputation

Stn Code 0

Date 0

State 0

City/Town/Village/Area 0

Location of Monitoring Station 0

Agency 0

Type of Location 0

SO2 0

NO2 0

RSPM 0

PM 2.5 0

dtype: int64

	Stn Code	Date	State	...	NO2	RSPM	PM 2.5
0	38	01-02-2014	Tamil Nadu	...	17.0	55.0	0.0
1	38	01-07-2014	Tamil Nadu	...	17.0	45.0	0.0
2	38	21-01-2014	Tamil Nadu	...	18.0	50.0	0.0
3	38	23-01-2014	Tamil Nadu	...	16.0	46.0	0.0
4	38	28-01-2014	Tamil Nadu	...	14.0	42.0	0.0

... ...

2874 773 12-03-2014 Tamil Nadu ... 18.0 102.0 0.0

2875 773 12-10-2014 Tamil Nadu ... 14.0 91.0 0.0

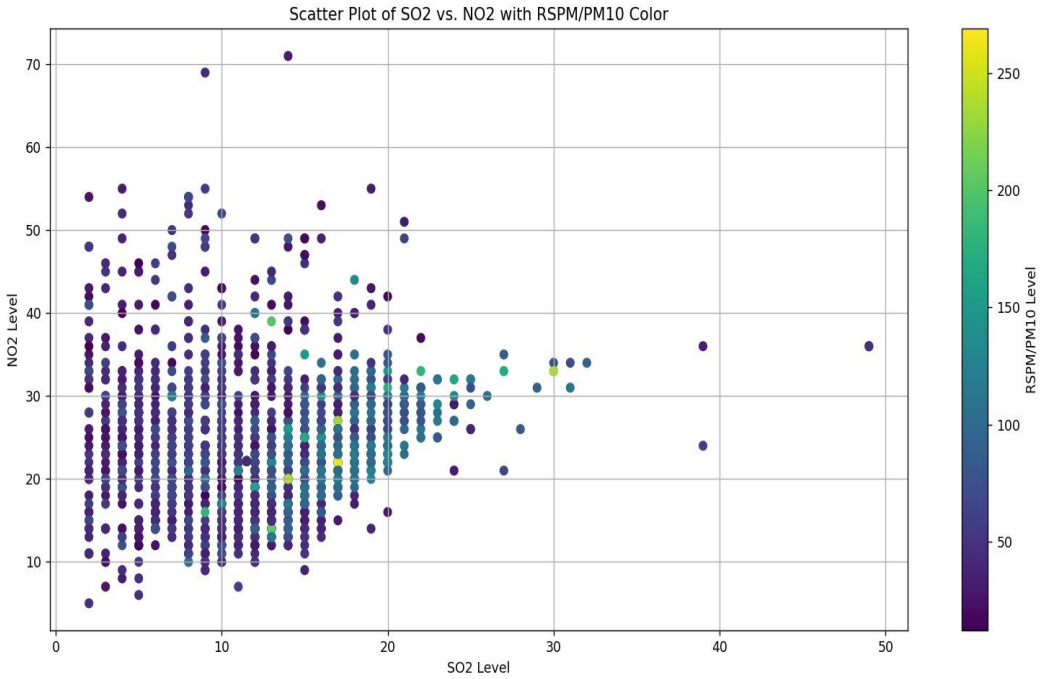
2876 773 17-12-2014 Tamil Nadu ... 22.0 100.0 0.0

2877 773 24-12-2014 Tamil Nadu ... 17.0 95.0 0.0

2878 773 31-12-2014 Tamil Nadu ... 16.0 94.0 0.0

[2879 rows x 11 columns]

PLOT



STEP 3: To calculate SI, NI, and RPI

The code performs AQI calculation for Sulfur Dioxide (SO2) , by considering defined concentration thresholds. The resulting AQI values are stored in a DataFrame, and different methods to present the DataFrame in tabular, markdown, and HTML formats are demonstrated. It's worth noting that there's a redundant redefinition of the 'df' DataFrame, which can be eliminated.

Additionally, the code defines a 'calculate_ni (NO2)' function to determine the Air Quality Index (AQI) for Nitrogen Dioxide (NO2) using specified concentration ranges. This function is applied to the 'NO2' column within the 'data' DataFrame, and the resultant 'ni' values are

stored in a new DataFrame 'df.' The code further illustrates the initial rows of the 'df' DataFrame. However, it's important to mention that there's a redundant redefinition of 'df' that can be removed.

Furthermore, the code introduces a 'calculate_(RSPM)' function to compute the Respirable Suspended Particulate Matter (RSPM) Index (rpi) by employing specific concentration intervals. This function is implemented on the 'RSPM' column of the 'data' DataFrame, and the resulting 'rpi' values are stored in a new DataFrame 'df.' Subsequently, the code displays the most recent rows of the 'df' DataFrame and offers a summary of its contents.

```
def calculate_si(SO2):
    si=0
    if (SO2<=40):
     si= SO2*(50/40)
    if (SO2>40 and SO2<=80):
     si= 50+(SO2-40)*(50/40)
    if (SO2>80 and SO2<=380):
     si= 100+(SO2-80)*(100/300)
    if (SO2>380 and SO2<=800):
     si= 200+(SO2-380)*(100/800)
    if (SO2>800 and SO2<=1600):
     si= 300+(SO2-800)*(100/800)
    if (SO2>1600):
     si= 400+(SO2-1600)*(100/800)
    return si
data['si']=data['SO2'].apply(calculate_si)
df= data[['SO2','si']]
df.head()
# Assuming you already have 'df' defined
df = data[['SO2', 'si']]

# Print the DataFrame in a tabular format
print(df)

#Function to calculate no2 individual pollutant index(ni)
def calculate_ni(NO2):
    ni=0
    if(NO2<=40):
     ni= NO2*50/40
    elif(NO2>40 and NO2<=80):
     ni= 50+(NO2-14)*(50/40)
    elif(NO2>80 and NO2<=180):
     ni= 100+(NO2-80)*(100/100)
    elif(NO2>180 and NO2<=280):
     ni= 200+(NO-180)*(100/100)
    elif(NO2>280 and NO2<=400):
     ni= 300+(NO2-280)*(100/120)
    else:
     ni= 400+(NO2-400)*(100/120)
    return ni
data['ni']=data['NO2'].apply(calculate_ni)
df= data[['NO2','ni']]
df.head()
```

```
# Assuming you already have 'df' defined
df = data[['NO2', 'ni']]

# Print the DataFrame in a tabular format
print(df)

#Function to calculate no2 individual pollutant index(rpi)

def calculate_(RSPM):
    rpi=0
    if(RSPM<=30):
     rpi=RSPM*50/30
    elif(RSPM>30 and RSPM<=60):
     rpi=50+(RSPM-30)*50/30
    elif(RSPM>60 and RSPM<=90):
     rpi=100+(RSPM-60)*100/30
    elif(RSPM>90 and RSPM<=120):
     rpi=200+(RSPM-90)*100/30
    elif(RSPM>120 and RSPM<=250):
     rpi=300+(RSPM-120)*(100/130)
    else:
     rpi=400+(RSPM-250)*(100/130)
    return rpi
data['rpi']=data['RSPM'].apply(calculate_)
df= data[['RSPM','rpi']]
df.tail()
print(df)
```

OUTPUT

SI			NI			RPI		
	SO2	si		NO2	ni		RSPM	rpi
0	11.0	13.75	0	17.0	21.25	0	55.0	91.666667
1	13.0	16.25	1	17.0	21.25	1	45.0	75.000000
2	12.0	15.00	2	18.0	22.50	2	50.0	83.333333
3	15.0	18.75	3	16.0	20.00	3	46.0	76.666667
4	13.0	16.2	4	14.0	17.50	4	42.0	70.000000
5 ...	...	...	...	...	...	... ...		
2874	15.0	18.75	2874	18.0	22.50	2874	102.0	240.000000
2875	12.0	15.00	2875	14.0	17.50	2875	91.0	203.333333
2876	19.0	23.75	2876	22.0	27.50	2876	100.0	233.333333
2877	15.0	18.75	2877	17.0	21.25	2877	95.0	216.666667
2878	14.0	17.50	2878	16.0	20.00	2878	94.0	213.333333
[2879 rows x 2 columns]			[2879 rows x 2 columns]					

STEP 4: To calculate API

An Air Quality Index (AQI) is a numerical scale used to communicate the level of air quality in a specific area or region. It provides a standardized way to assess and report the concentration of various air pollutants in the atmosphere. AQIs are typically calculated based on the measurements of common air pollutants such as particulate matter (PM2.5 and PM10),

ground-level ozone (O3), sulfur dioxide (SO2), nitrogen dioxide (NO2), and carbon monoxide (CO).

The primary purpose of an AQI is to provide information to the public about the potential health risks associated with current air quality conditions. It helps individuals and authorities make informed decisions about outdoor activities and can trigger public health advisories or regulations when air quality deteriorates to unhealthy levels. AQI values are often measured and reported by environmental agencies and organizations to keep the public informed about the quality of the air they are breathing.

```
def calculate_aqi(si,ni,rpi):
    aqi=0
    if(si>ni and si>rpi):
     aqi=si
    if(ni>si and ni>rpi):
     aqi=ni
    if(rpi>si and rpi>ni):
     aqi=rpi
    return aqi
data['AQI']=data.apply(lambda
x:calculate_aqi(x['si'],x['ni'],x['rpi']),axis=1)
df= data[['Date','City/Town/Village/Area','si','ni','rpi','AQI']]
df.head()
print (df)
```

OUTPUT

	Date	City/Town/Village/Area	si	ni	rpi	AQI
0	01-02-2014	Chennai	13.75	21.25	91.666667	91.666667
1	01-07-2014	Chennai	16.25	21.25	75.000000	75.000000
2	21-01-2014	Chennai	15.00	22.50	83.333333	83.333333
3	23-01-2014	Chennai	18.75	20.00	76.666667	76.666667
4	28-01-2014	Chennai	16.25	17.50	70.000000	70.000000
...	...	...	...	...	...	...
2874	12-03-2014	Trichy	18.75	22.50	240.000000	240.000000
2875	12-10-2014	Trichy	15.00	17.50	203.333333	203.333333
2876	17-12-2014	Trichy	23.75	27.50	233.333333	233.333333

2877 24-12-2014 Trichy 18.75 21.25 216.666667 216.666667

2878 31-12-2014 Trichy 17.50 20.00 213.333333 213.333333

[2879 rows x 6 columns]

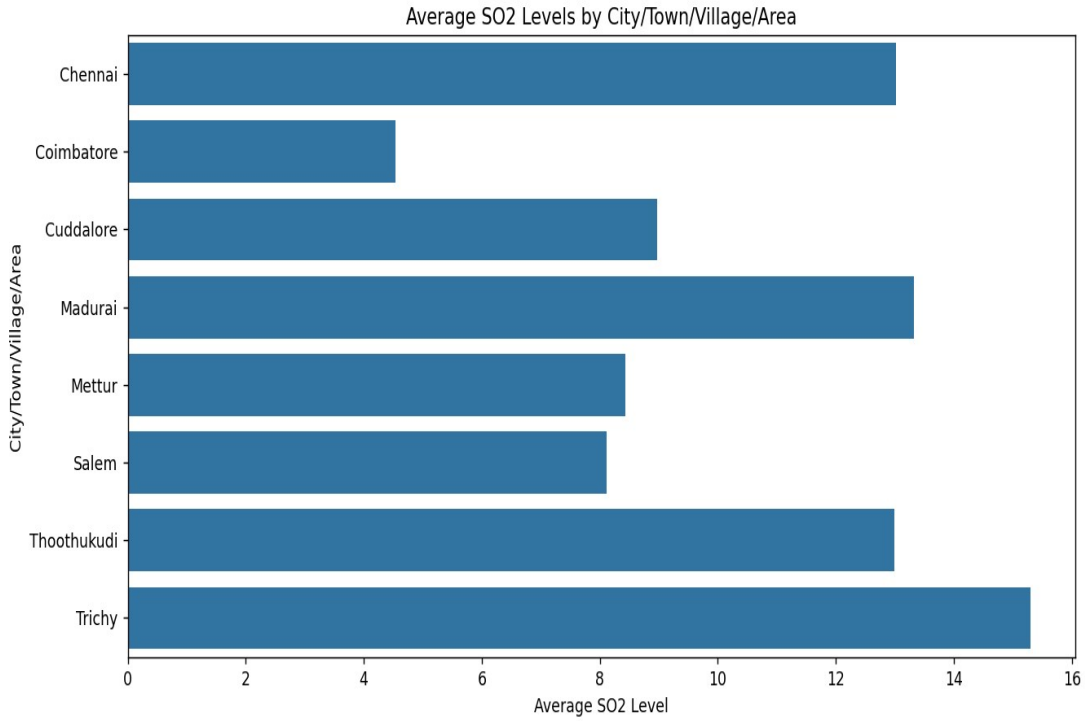
Step 5: To calculate and plot average SO2, NO2, RPSM and API

The code supplied aggregates air quality data based on the 'City/Town/Village/Area' parameter, computes the mean levels of Sulfur Dioxide (SO2), Nitrogen Dioxide (NO2), and RPSM for each specific area, and then presents the outcomes through a horizontal bar plot created with Seaborn. This visual representation illustrates the average SO2, NO2, RPSM concentrations in distinct areas, enabling a straightforward comparison of air quality. Nonetheless, there is redundant code within the script that can be safely omitted for a more streamlined implementation.

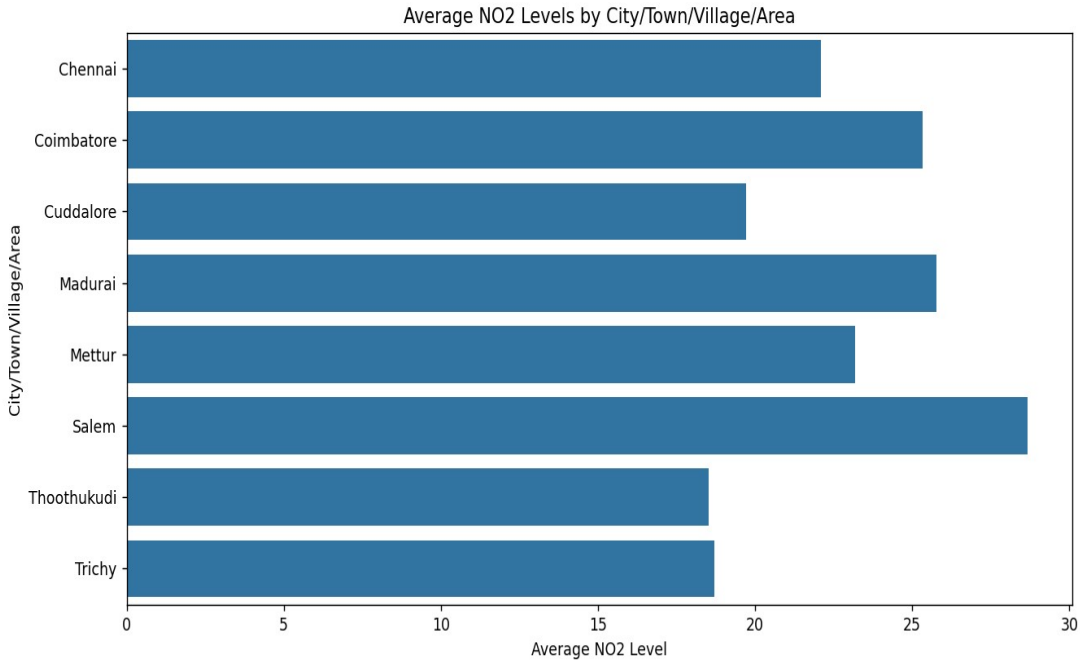
```
# Group the data by 'City/Town/Village/Area' and calculate the average SO2
levels
average_so2_by_area =
data.groupby('City/Town/Village/Area')['SO2'].mean().reset_index()

# Plot the average SO2 levels
plt.figure(figsize=(12, 6))
sns.barplot(x='SO2', y='City/Town/Village/Area', data=average_so2_by_area,
orient='h')
plt.title('Average SO2 Levels by City/Town/Village/Area')
plt.xlabel('Average SO2 Level')
plt.ylabel('City/Town/Village/Area')
plt.show()
# Group the data by 'City/Town/Village/Area' and calculate the average NO2
levels
average_no2_by_area =
data.groupby('City/Town/Village/Area')['NO2'].mean().reset_index()

# Plot the average NO2 levels
plt.figure(figsize=(12, 6))
sns.barplot(x='NO2', y='City/Town/Village/Area', data=average_no2_by_area,
orient='h')
plt.title('Average NO2 Levels by City/Town/Village/Area')
plt.xlabel('Average NO2 Level')
plt.ylabel('City/Town/Village/Area')
plt.show()
# Group the data by 'City/Town/Village/Area' and calculate the average RSPM
levels
average_rpsm_by_area =
data.groupby('City/Town/Village/Area')['RSPM'].mean().reset_index()

# Plot the average RSPM levels
```

```
plt.figure(figsize=(12, 6))
sns.barplot(x='RSPM', y='City/Town/Village/Area',
data=average_rpsm_by_area, orient='h')
plt.title('Average RSPM Levels by City/Town/Village/Area')
plt.xlabel('Average RSPM Level')
plt.ylabel('City/Town/Village/Area')
plt.show()
# Group the data by 'City/Town/Village/Area' and calculate the average AQI
levels
average_aqi_by_area =
data.groupby('City/Town/Village/Area')['AQI'].mean().reset_index()

# Plot the average AQI levels
plt.figure(figsize=(12, 6))
sns.barplot(x='AQI', y='City/Town/Village/Area', data=average_aqi_by_area,
orient='h')
plt.title('Average AQI Levels by City/Town/Village/Area')
plt.xlabel('Average AQI Level')
plt.ylabel('City/Town/Village/Area')
plt.show()
```

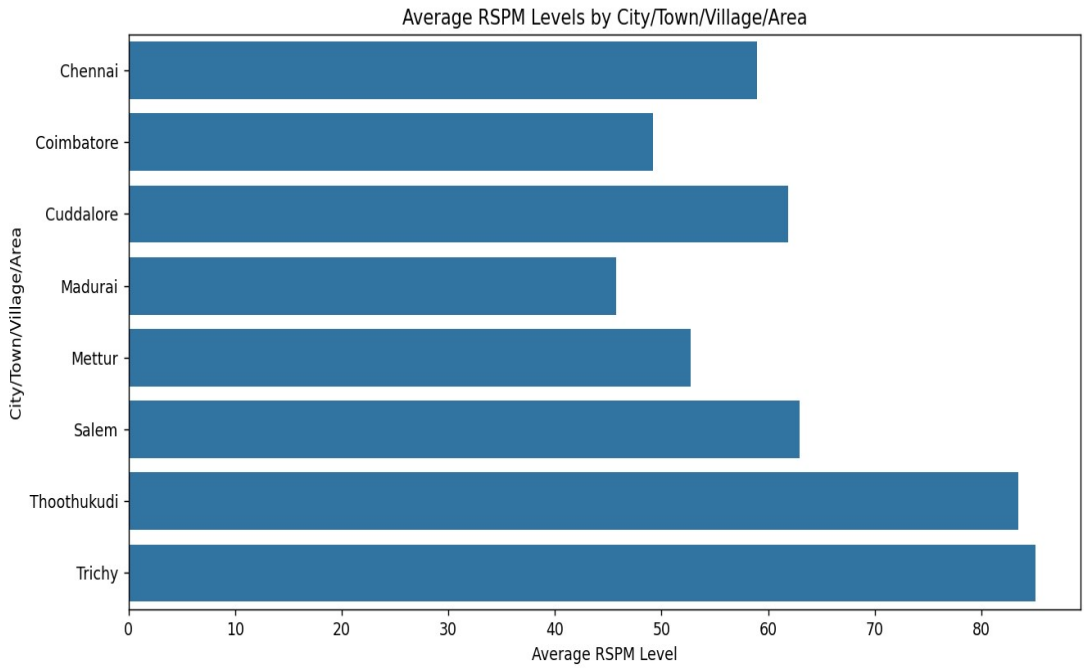
OUTPUT



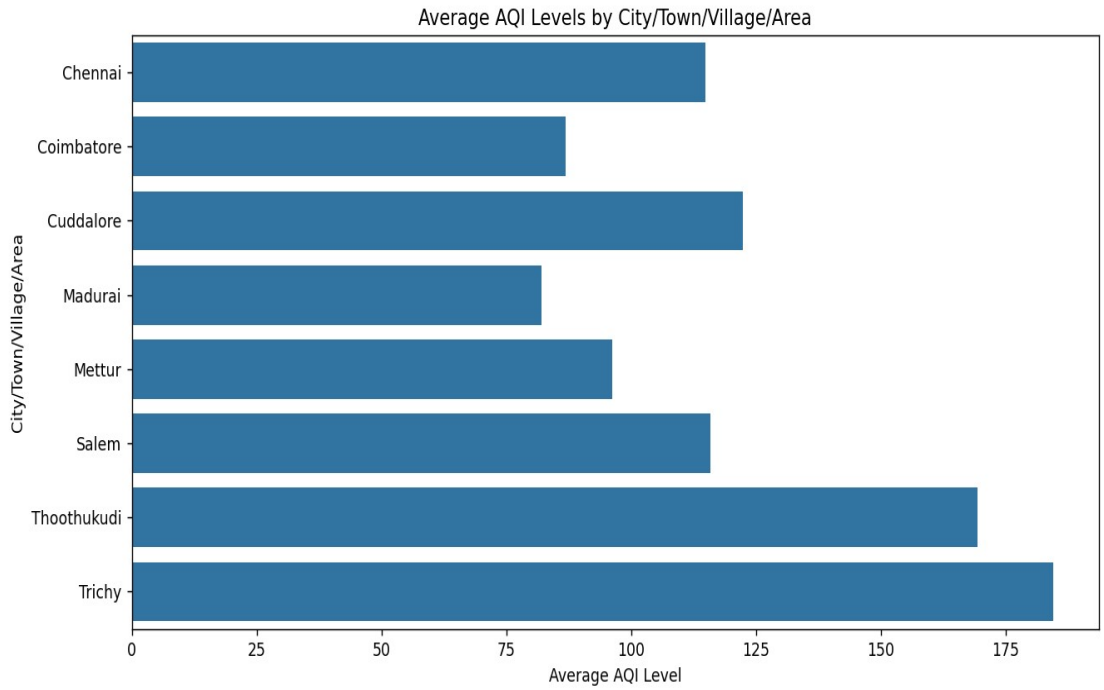
a) Average SO2 level



b) Average NO2 level



c) Average RPSM level



d) Average NO2 level

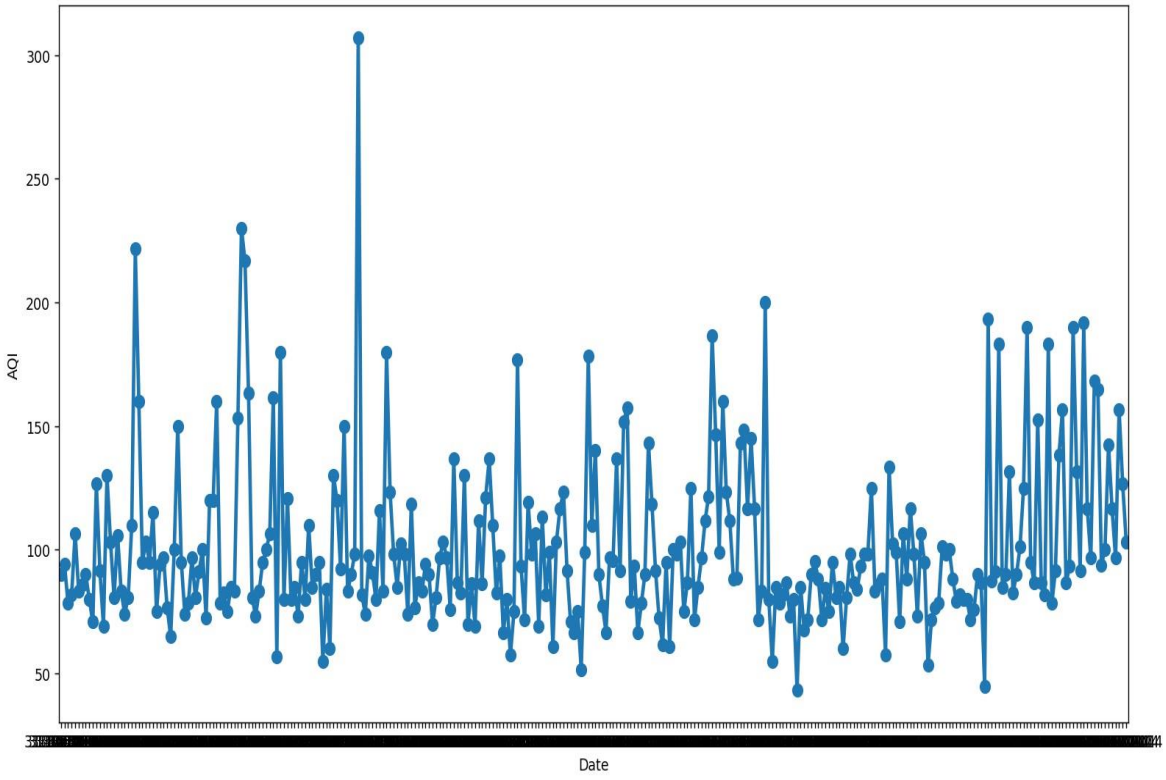
Step 6: To identify pollution trends with high pollution levels

The code offers an in-depth examination of AQI data, encompassing the depiction of patterns, the utilization of a simplistic forecasting method, and a reevaluation of AQI rooted in alternative constituents.

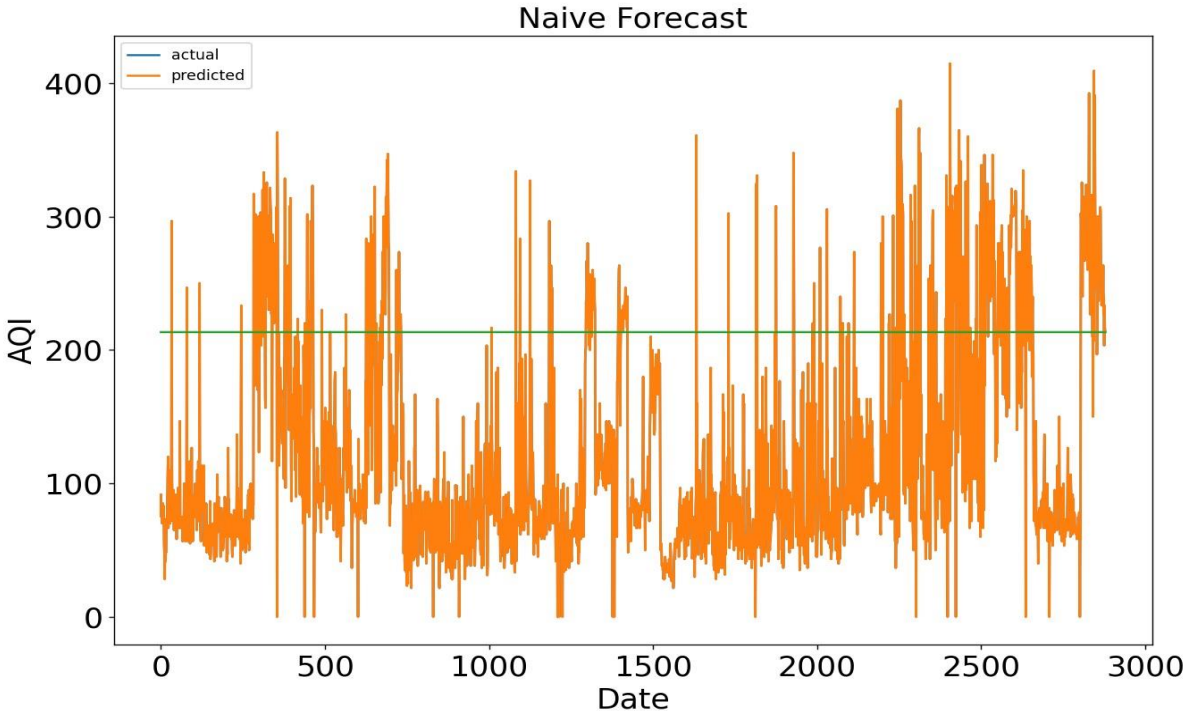
```
df =
data[['AQI','Date']].groupby(["Date"]).median().reset_index().sort_values(b
y='Date',ascending=False)
f,ax=plt.subplots(figsize=(15,10))
sns.pointplot(x='Date', y='AQI', data=df)
train = pd.DataFrame(data)
test = pd.DataFrame(data)
dd= np.asarray(train.AQI)
y_hat = test.copy()
y_hat['naive'] = dd[len(dd)-1]
plt.figure(figsize=(12,8))
plt.plot(train.index, train['AQI'], label='Train')
plt.plot(test.index,test['AQI'], label='Test')
plt.plot(y_hat.index,y_hat['naive'], label='Naive Forecast')
plt.legend(loc='best')
plt.title("Naive Forecast",fontsize=20)

plt.legend(["actual ","predicted"])
```

```
plt.xlabel("Date",fontsize=20)
plt.ylabel("AQI",fontsize=20)
plt.tick_params(labelsize=20)
plt.show()
data['AQI']=data.apply(lambda
x:calculate_aqi(x['si'],x['ni'],x['rpi']),axis=1)
df= data[['Date','City/Town/Village/Area','si','ni','rpi','AQI']]
df.head()
print (df)
df=data.set_index('Date')
df.sort_values(by='Date',ascending=False)
df.plot(figsize=(15, 6))
plt.show()
y=df.AQI
```

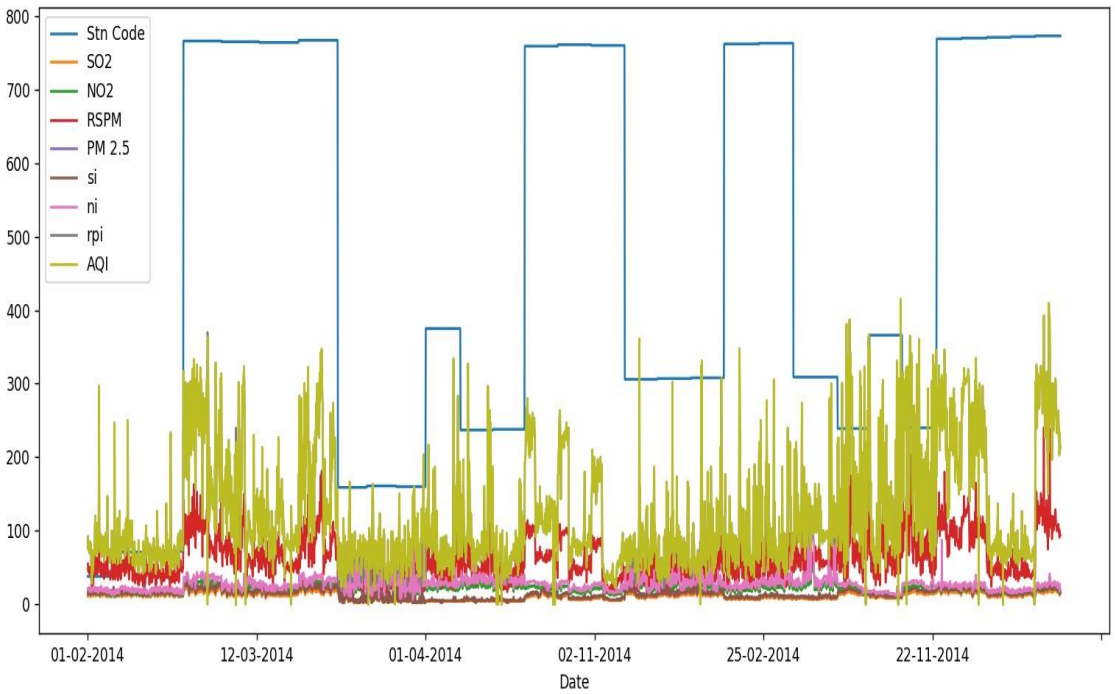
OUTPUT



a) AQI



b) Trends



c) Overall plot

Step 7: Feature Engineering and Model Selection

This code performs a linear regression analysis for predicting Air Quality Index (AQI) based on a chosen feature, in this case, 'SO2' (Sulfur Dioxide) concentration. Here's a summary of the code:

1. **Data Preparation:** The code selects 'SO2' as the independent variable (feature) and 'AQI' as the dependent variable (target) to predict. It then splits the data into training and testing sets using the train_test_split function.

2. **Model Creation and Training:** It creates a linear regression model using the LinearRegression class from a machine learning library (not shown but assumed to be imported), and trains the model using the training data (X_train and y_train).

3. **Prediction:** The code utilizes the trained model to make predictions on the test set (X_test), resulting in 'y_pred,' which contains the predicted AQI values.

4. **Model Evaluation:** Model performance metrics are calculated to assess the accuracy of the linear regression model. It computes the Mean Squared Error (MSE) and the R-squared (R2) score. These metrics measure the quality of the predictions compared to the actual AQI values.

5. **Visualization:** The code visualizes the linear regression line and the actual data points on a scatter plot. It uses blue dots to represent the actual AQI values and a red line to depict the regression line. This visualization provides a clear representation of how well the model fits the data.

Overall, this code demonstrates the application of linear regression to predict AQI based on the 'SO2', ‘NO2’ and ‘RPSM’ feature, evaluates the model's performance, and presents the results graphically. It serves as a basic example of regression analysis for air quality prediction.

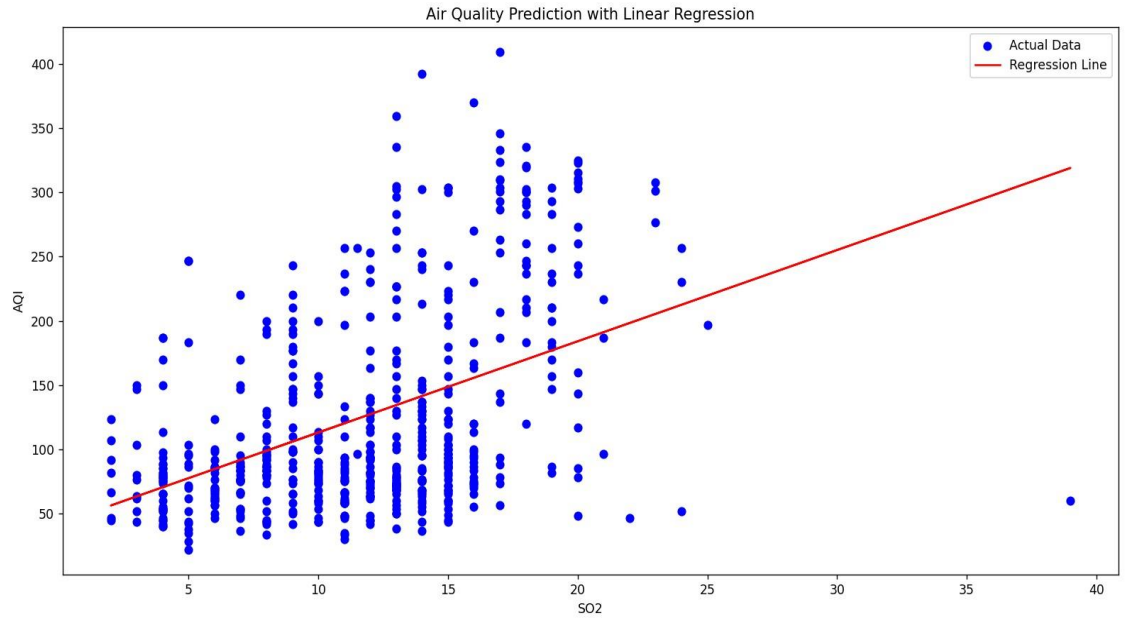
```
X = data[['SO2']]  # Replace 'SO2' with your chosen feature
y = data['AQI']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create and train a linear regression model
regression_model = LinearRegression()
regression_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = regression_model.predict(X_test)
```

```
# Calculate model performance metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared (R2) Score:", r2)

# Visualize the regression line and data points
plt.scatter(X_test, y_test, color='b', label='Actual Data')
plt.plot(X_test, y_pred, color='r', label='Regression Line')
plt.title('Air Quality Prediction with Linear Regression')
plt.xlabel('SO2')
plt.ylabel('AQI')
plt.legend()
plt.show(
X = data[['NO2']]
y = data['AQI']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create and train a linear regression model
regression_model = LinearRegression()
regression_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = regression_model.predict(X_test)

# Calculate model performance metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared (R2) Score:", r2)

# Visualize the regression line and data points
plt.scatter(X_test, y_test, color='b', label='Actual Data')
plt.plot(X_test, y_pred, color='r', label='Regression Line')
plt.title('Air Quality Prediction with Linear Regression')
plt.xlabel('NO2')
plt.ylabel('AQI')
plt.legend()
plt.show()
```

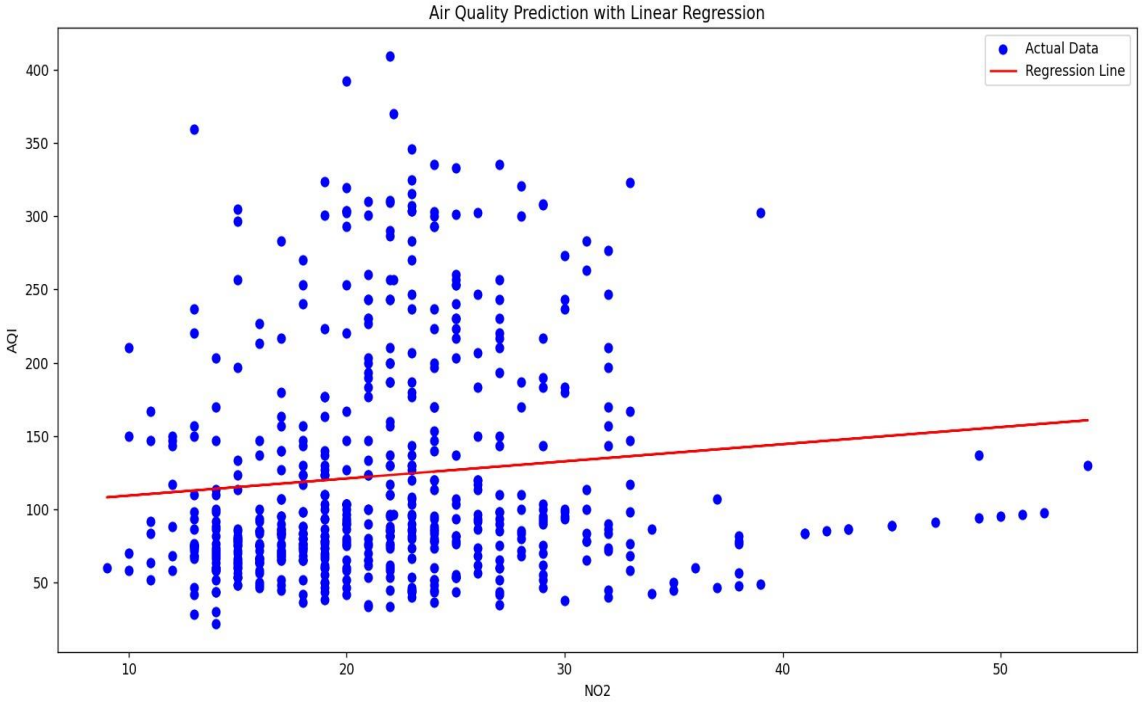
```
X = data[['RPSM']]
y = data['AQI']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create and train a linear regression model
regression_model = LinearRegression()
regression_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = regression_model.predict(X_test)
```

```
# Calculate model performance metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared (R2) Score:", r2)

# Visualize the regression line and data points
plt.scatter(X_test, y_test, color='b', label='Actual Data')
plt.plot(X_test, y_pred, color='r', label='Regression Line')
plt.title('Air Quality Prediction with Linear Regression')
plt.xlabel('RPSM')
plt.ylabel('AQI')
plt.legend()
plt.show()
```

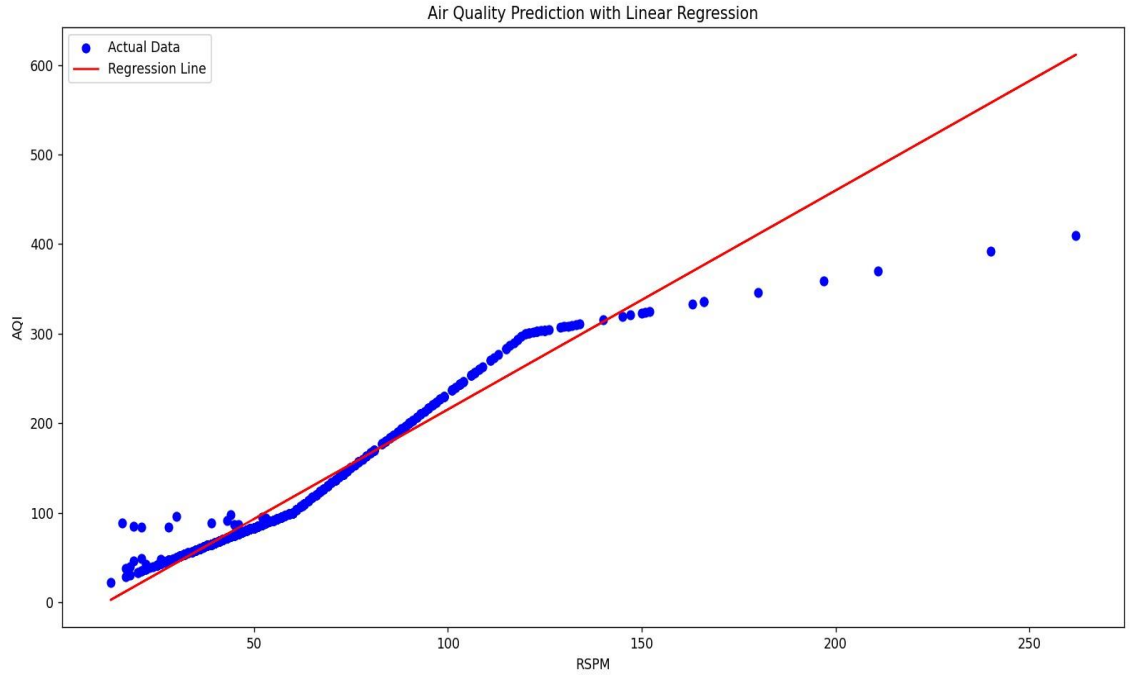
OUTPUT



a) SO2


Air Quality Prediction with Linear Regression
Actual Data
Regression Line
AQI
NO2
400
350
300
250
200
150
100
50
10
20
30
40
50

b) NO2


Air Quality Prediction with Linear Regression
Actual Data
Regression Line
AQI
RSPM
600
500
400
300
200
100
0
50
100
150
200
250

c) RPSM

Step 8: Model Evaluation

Root Mean Square

FORMULA

$$\mathrm{RMS} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\mathrm{Value}_i)^2}$$

i) SO2

R-squared (R2) Score: 0.20923517764168698

ii) NO2

R-squared (R2) Score: 0.009847922050437052

iii) RPSM

R-squared (R2) Score: 0.9405308774361777

CONCLUSION

In summary,

a) Chennai, Thoothukudi, Trichy and Madurai has higher SO2 levels.

b) Salem, Coimbatore and Madurai has higher NO2 levels.

c) Trichy and Thoothukudi has higher RPSM levels

Overall, the higher average AQI has been found in Trichy and Thoothukudi . These are the cities with high air pollution levels and are highly prone to health hazard