DATA ANALYTICS WITH COGNOS

AIR QUALITY ANALYSIS IN TAMILNADU

Team ID 1294

AIR QUALITY ANALYSIS IN TAMILNADU

TABLE OF CONTENT Page

1. INTRODUCTION

The goal of this project is to analyze and visualize air quality data obtained from various monitoring stations across Tamil Nadu, a state located in the southern part of India. The state of Tamil Nadu boasts a rich cultural heritage and a rapidly growing urban landscape, but this urbanization has brought about its share of environmental challenges, including air pollution. Air quality analysis plays a pivotal role in understanding environmental health, and this project aims to uncover insights into air pollution trends within the region.

Air pollution is a critical concern that affects the well-being of both the environment and its inhabitants. It poses a significant threat to public health, impacting the respiratory system, exacerbating existing health conditions, and even leading to long-term health problems. In Tamil Nadu, as in many other urban canters worldwide, there is a pressing need to address air pollution and its consequences.

By harnessing data-driven approaches, we seek to identify areas with elevated pollution levels and create a predictive model capable of estimating RSPM/PM10 levels based on SO2 and NO2 levels. Such predictive models can provide invaluable tools for monitoring and mitigating air pollution, enabling policymakers to take informed actions to safeguard public health.

This undertaking holds significant relevance, as it empowers policymakers, environmentalists, and the public with valuable information to address air quality concerns and formulate strategies for cleaner and healthier environments. By bringing together data analysis, predictive modeling, and environmental science, we aim to contribute to a sustainable future for Tamil Nadu and set an example for proactive environmental stewardship.

In this document, we will outline the problem statement, delineate the sequential steps involved in accomplishing our objectives, and introduce a design thinking approach to steer our project toward successful fruition. Through collaboration, data-driven insights, and

informed decision-making, we endeavour to make Tamil Nadu a model for clean air and a healthier future.

2. OBJECTIVES

Objective: The primary objective of this project is to analyse and visualize air quality data obtained from multiple monitoring stations across Tamil Nadu, with the aim of gaining comprehensive insights into air pollution trends within the region.

Project Objectives:

The primary objectives of this data analysis project are to:

1.Analyse Air Quality Trends:

Conduct a comprehensive analysis of air quality data collected from multiple monitoring stations across Tamil Nadu. This will involve examining trends in pollutants such as SO2, NO2, and RSPM/PM10 over time.

2.Identify Pollution Hotspots:

Determine specific areas within Tamil Nadu that consistently exhibit high levels of air pollution. Pinpointing these hotspots will be crucial for targeted intervention and mitigation efforts.

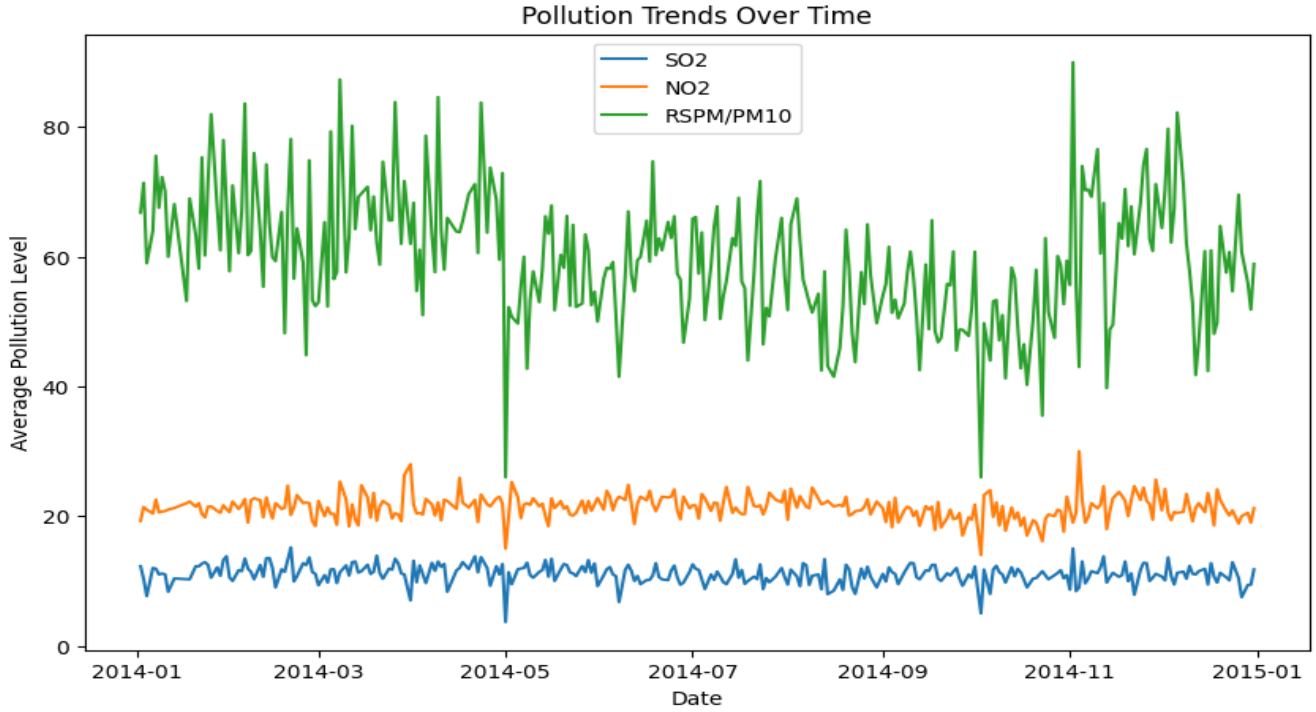
3.Create Predictive Model for RSPM/PM10:

Develop a predictive model that can estimate RSPM/PM10 levels based on the concentrations of SO2 and NO2. This model will serve as a valuable tool for forecasting pollution levels and implementing proactive measure

3. Analysis Approach

- Data Collection:
    - We have access to a substantial dataset comprising historical air quality measurements collected from various monitoring stations situated across Tamil Nadu. This dataset encompasses a wide array of air quality metrics, including pollutant concentrations and meteorological parameters, meticulously recorded over time.
- Data Cleaning and Preprocessing:
    - The data underwent thorough cleaning and preprocessing steps, including handling missing values, outliers, and data transformation to ensure accuracy and reliability.
- Exploratory Data Analysis (EDA):
    - EDA techniques were employed to gain initial insights into the dataset. This involved summary statistics, visualizations, and correlation analysis. Additionally, the Kalman filter was applied to extract more refined trends from the data.
- Statistical Analysis:
    - The Kalman filter approach was used to perform statistical analysis on the air quality data. It enables dynamic estimation and prediction of pollutant levels based on observed measurements and state space models.
- Analyze Air Quality Trends:

- Gain insights into air pollution trends across various monitoring stations in Tamil Nadu.



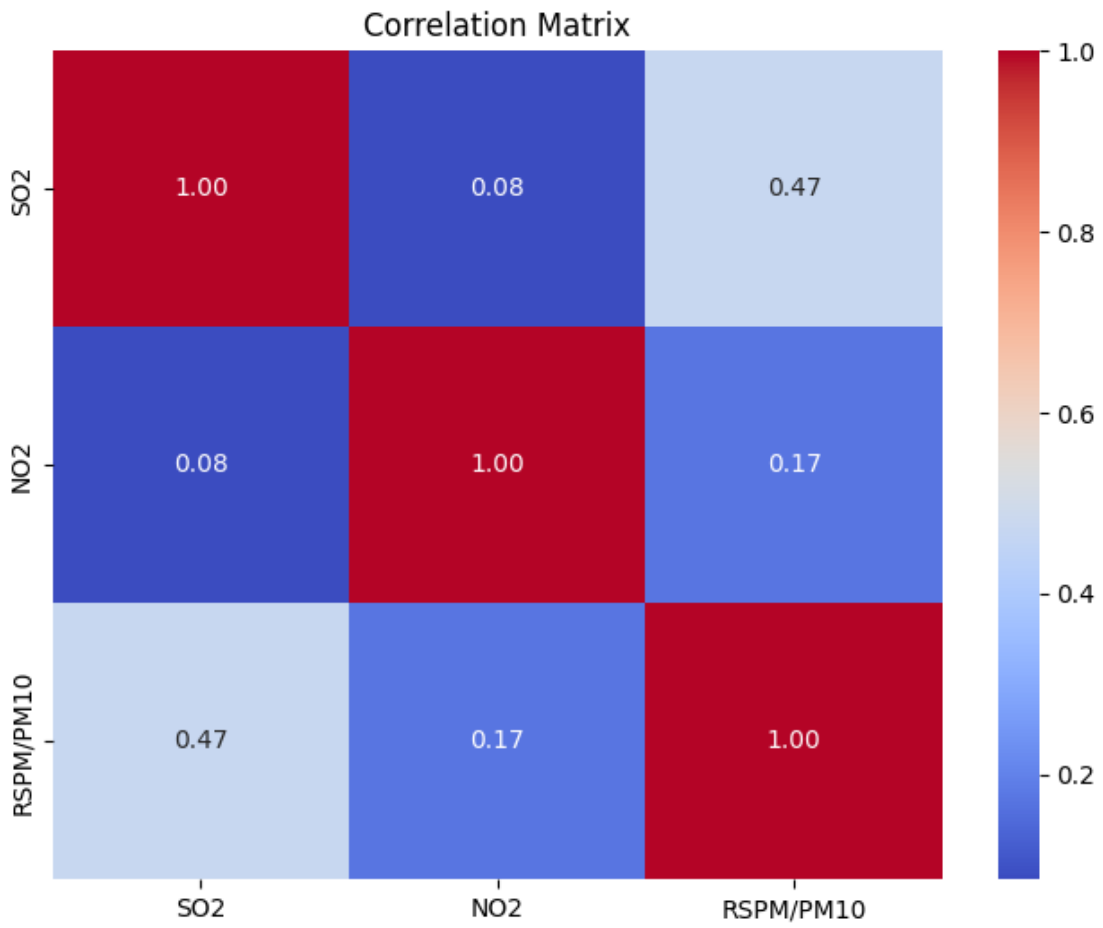
- Identify Pollution Hotspots:
  - Pinpoint specific areas with consistently high pollution levels for targeted intervention.
- Create Predictive Model for RSPM/PM10:
  - Develop a predictive model to estimate RSPM/PM10 levels based on SO2 and NO2 concentrations using the Kalman filter approach.

```
import numpy as np
from filterpy.kalman import KalmanFilter
import pandas as pd

# Assuming 'data' is your DataFrame
# Make sure to replace placeholders with actual values
```

```
# Define initial estimates and covariances
initial_SO2_estimate = data['SO2'].iloc[0]  # Initial SO2 estimate (using first measurement)
initial_NO2_estimate = data['NO2'].iloc[0]  # Initial NO2 estimate (using first measurement)
initial_state_covariance = 10.0  # Adjust as needed
measurement_noise_covariance = 5.0  # Adjust as needed

# Define a 2D Kalman Filter for SO2 and NO2 to predict RSPM/PM10
kf = KalmanFilter(dim_x=2, dim_z=1)
kf.x = np.array([initial_SO2_estimate, initial_NO2_estimate])
kf.F = np.array([[1., 0.], [0., 1.]])
kf.H = np.array([[1., 1.]])
kf.P *= np.eye(2) * initial_state_covariance
kf.R = measurement_noise_covariance

# Lists to store actual and predicted RSPM/PM10 levels
actual_rspm = []
predicted_rspm = []

# Iterate through the data
for index, row in data.iterrows():
    measurements = np.array([row['SO2'], row['NO2']])

    # Predict the next state based on the dynamic model (assuming constant)
    predicted_state = kf.predict()

    # Update the state estimate based on the actual RSPM/PM10 measurement
    kf.update(row['RSPM/PM10'])
```

```
    # Store actual and predicted RSPM/PM10 levels
    actual_rspm.append(row['RSPM/PM10'])
    predicted_rspm.append(kf.x[0])  # Assuming RSPM/PM10
is the first state variable

# Calculate evaluation metrics
mae = np.mean(np.abs(np.array(actual_rspm) -
np.array(predicted_rspm)))
rmse = np.sqrt(np.mean((np.array(actual_rspm) -
np.array(predicted_rspm))**2))

print(f"Mean Absolute Error (MAE): {mae}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
```

- **Assess Correlations between Pollutants:**
    - Explore relationships between pollutants (SO2, NO2, and RSPM/PM10) using statistical techniques, including the Kalman filter.



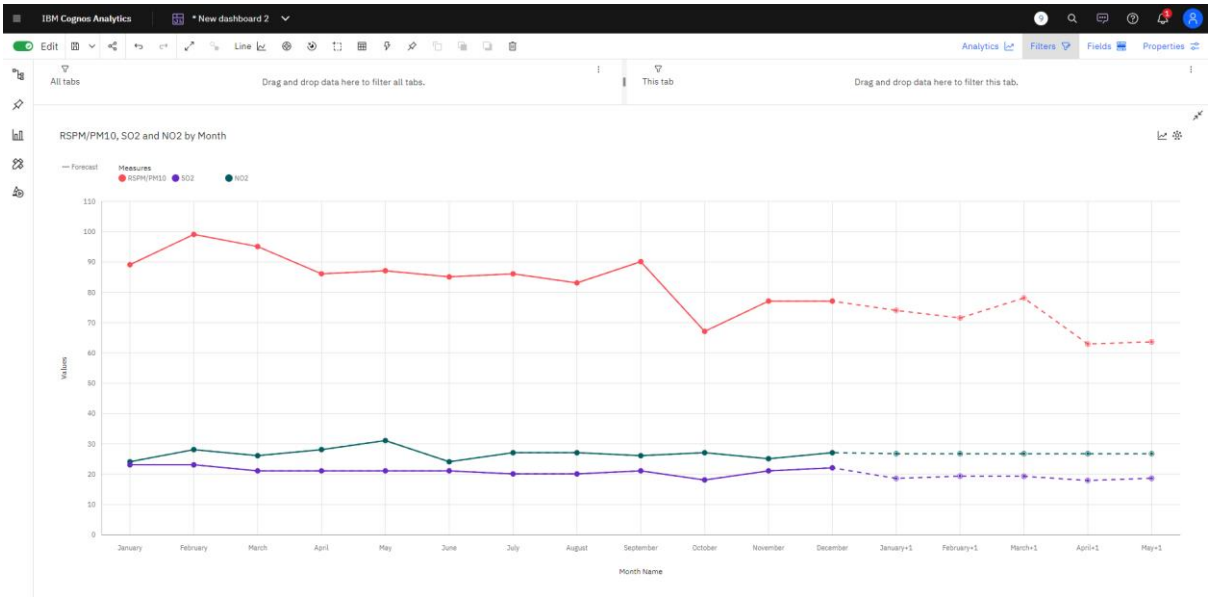
- **Validate Predictive Model Accuracy:**
  - Validate the predictive model's accuracy using the Kalman filter approach against actual RSPM/PM10 levels.
  - The evaluation we get from the Kalman filter model is, Mean Absolute Error (MAE): 32.923811851054815 Root Mean Squared Error (RMSE): 36.31349956905789.
  - It seems we need to improve the performance of our Kalman filter model to give more accuracy in predicting the RSPM/10.

4.VISUALIZATION METHODS

1. RSPM/PM10, S02 AND N02 by MONTH

i) Based on the current forecasting, RSPM/PM10 may reach 60.96 by Month Name March+1

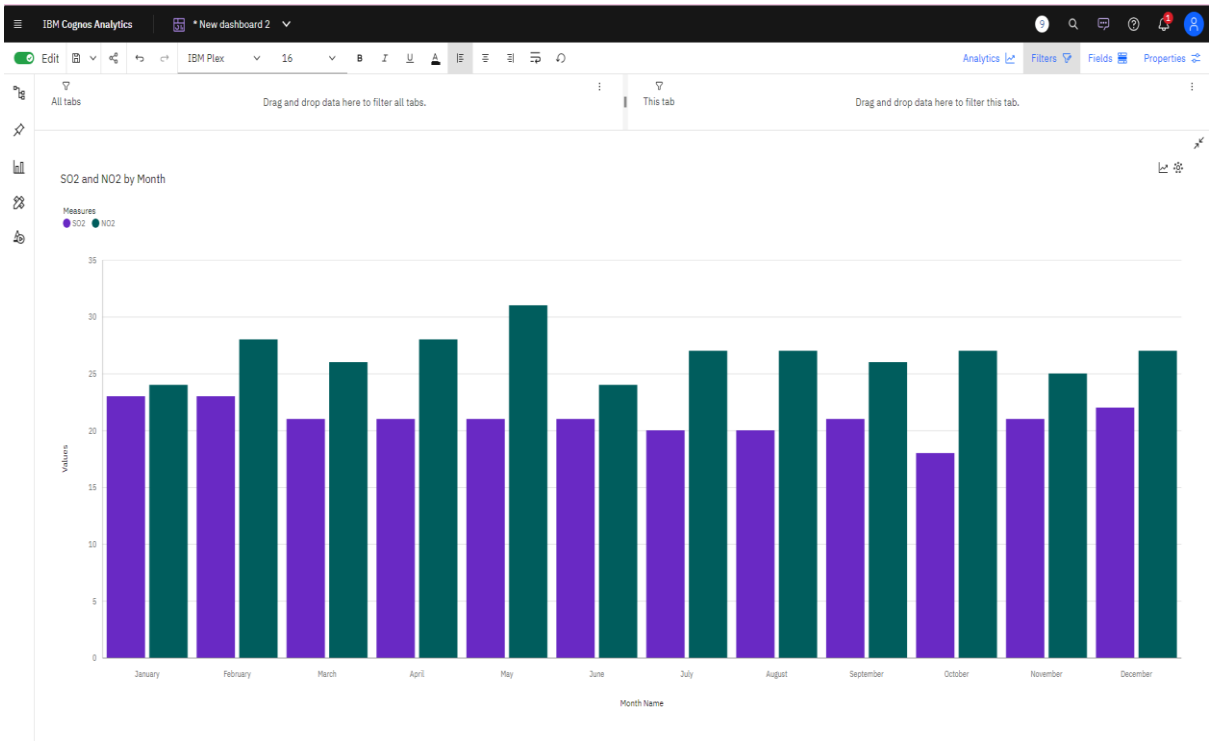
ii) It is projected that by March+1, Chennai will exceed Cuddalore in NO2 by 6.98



iii) The total number of results for NO2, across all month names, is nearly two thousand

iv) The total number of results for RSPM/PM10, across all month names, is nearly two thousand.

v) The total number of results for SO2, across all month names, is nearly two thousand



2. S02 and NO2 by Month

i) **NO2** shows a strong seasonal trend every **5 months**. The largest values typically occur at period **5**, whereas the smallest values at period **1**.

ii) Based on the current forecasting, **NO2** may reach **27.18** by **Month Name March+1**.

iii) The total number of results for **NO2**, across all **month names**, is **nearly two thousand**.

iv) The total number of results for **SO2**, across all **month names**, is **nearly two thousand**.

3.RSPM/PM by Type of Location

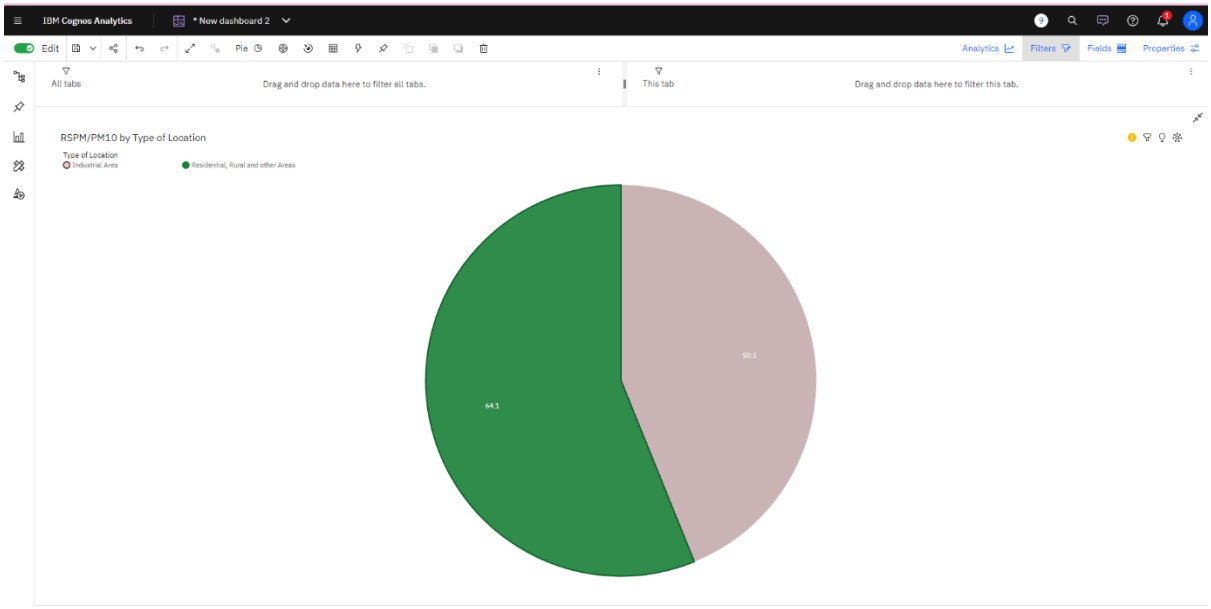
i) It is projected that by **2015-03-14**, **Residential, Rural and other Areas** will exceed **Industrial**

ii) From **2014-03-08** to **2014-03-10**, **Industrial Area**'s **RSPM/PM10** dropped by **65**%.

iii) **Residential, Rural and other Areas** is the most frequently occurring category of **Type of Location** with a count of **1859** items with **RSPM/PM10** values (**69.4** % of the total).

iv) Over all **type of locations**, the average of **RSPM/PM10** is **59.82**.

v)The average values of **RSPM/PM10** range from **50.1**, occurring when **Type of Location** is **Industrial Area**, to **64.1**, when **Type of Location** is **Residential, Rural and other Areas**.



4.FORECASTING RSPM/PM10, S02 and N02 by Month

i) **RSPM/PM10** is unusually high when **City/Town/Village/Area** is **Trichy**.

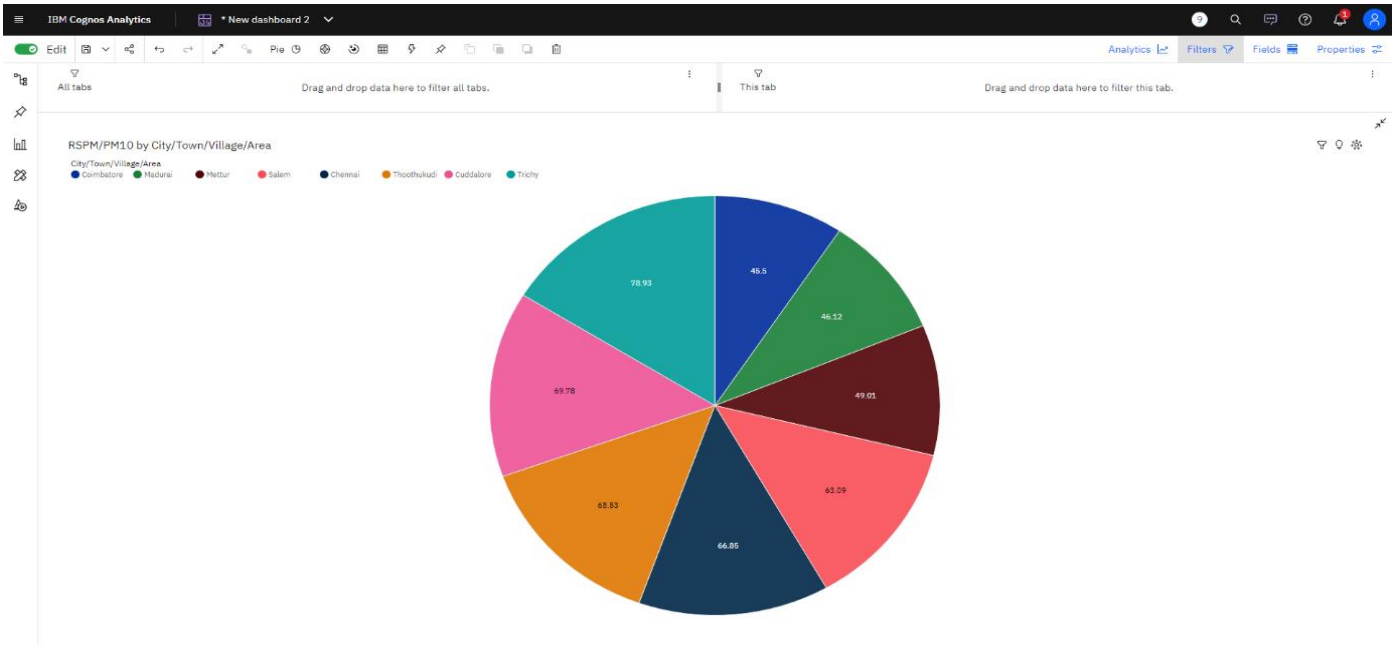
ii) It is projected that by **2015-03-14**, **Thoothukudi** will exceed **Coimbatore** in **RSPM/PM10** b

iii) From **2014-02-25** to **2014-02-26**, **Trichy**'s **RSPM/PM10** increased by **268**%.

iv) **City/Town/Village/Area** weakly affects **RSPM/PM10** (**16%**).

v) **Chennai** is the most frequently occurring category of **City/Town/Village/Area** with a count of **915** items with **RSPM/PM10** values (**34.2** % of the total)

vi) Over all values of **City/Town/Village/Area**, the average of **RSPM/PM10** is **59.82**.



5.CODE IMPLEMENTATION

5.1. REMOVING THE NULL VALUES

```
# Check for missing values
missing_values =
data.isnull().sum()
print(missing_values)
```

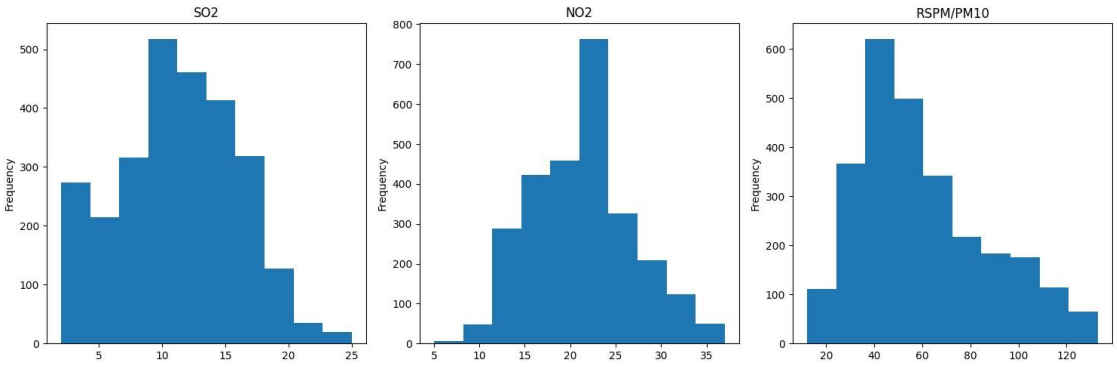
```
Stn Code                    0
Sampling Date               0
Month Name                  0
State                       0
City/Town/Village/Area      0
Location of Monitoring Station  0
Agency                      0
Type of Location            0
SO2                         0
NO2                         0
RSPM/PM10                   0
dtype: int64
```

```
#total number of rows given num_rows =
data.shape[0] print(f'The dataset contains
{num_rows} rows.')
```

```
The dataset contains 2697 rows.
```

5.1.1 VISUALIZING THE NULL VALUED FEATURES TO DETERMINE THE METHOD TO FILL THE DATA

```
import matplotlib.pyplot as plt


# Create subplots for each column with missing
values fig, axes = plt.subplots(nrows=1, ncols=3,
figsize=(15, 5)) # Plot histograms for SO2, NO2,
and RSPM/PM10 data['SO2'].plot(kind='hist',
ax=axes[0], title='SO2') data['NO2'].plot(kind='hist',
ax=axes[1], title='NO2')

data['RSPM/PM10'].plot(kind='hist', ax=axes[2],
title='RSPM/PM10')


plt.tight_layout()
plt.show()
```



5.1.2 REPLACING THE NULL VALUES WITH THEIR RESPECTIVE MEAN VALUES

```
# Fill missing values with mean
data['SO2'].fillna(data['SO2'].mean(), inplace=True)
data['NO2'].fillna(data['NO2'].mean(), inplace=True)
```

```
data['RSPM/PM10'].fillna(data['RSPM/PM10'].mean(),
inplace=True)


# Verify that there are no more missing
values      missing_values_after_filling      =
data.isnull().sum()
print(missing_values_after_filling)
```

```
Stn Code                    0
Sampling Date               0
Month Name                  0
State                       0
City/Town/Village/Area      0
Location of Monitoring Station    0
Agency                      0
Type of Location            0
SO2                         0
NO2                         0


RSPM/PM10                   0
dtype: int64
```

```
 data['SO2'].fillna(data['SO2'].mean(), inplace=True)

data['NO2'].fillna(data['NO2'].mean(), inplace=True)

data ['RSPM/PM10'].fillna(data['RSPM/PM10'].mean(),
inplace=True)
```

5.2 OUTLIER DETECTION

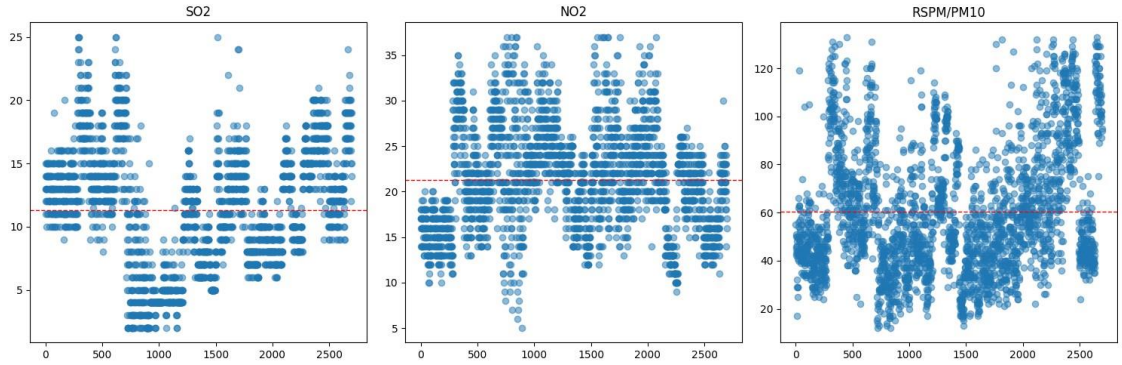
```
import numpy as np


# Define a function to detect outliers using
IQR def detect_outliers(column):     Q1 =
np.percentile(column, 25)
    Q3 =
np.percentile(column, 75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 *
IQR     upper_bound = Q3 +
1.5 * IQR
    return (column < lower_bound) | (column > upper_bound)


# Apply outlier detection to numerical columns (SO2, NO2,
RSPM/PM10) outliers = detect_outliers(data[['SO2', 'NO2',
'RSPM/PM10']])

# Print the number of outliers for each column
print(outliers.sum())
```

```
SO2           0
NO2           0
RSPM/PM10
582 dtype: int64
```

5.2.1 VISUALIZE THE OUTLIERS

```
# Calculate mean values
mean_so2 =
data['SO2'].mean()
```

```
mean_no2 =
data['NO2'].mean()

mean_rspm_pm10 = data['RSPM/PM10'].mean()


# Create subplots for each column fig, axes =
plt.subplots(nrows=1, ncols=3, figsize=(15, 5))


# Plot scatter plots for SO2, NO2, and RSPM/PM10 against index
axes[0].scatter(data.index, data['SO2'], alpha=0.5)
axes[0].axhline(mean_so2, color='red', linestyle='dashed',
linewidth=1) axes[0].set_title('SO2')


axes[1].scatter(data.index, data['NO2'], alpha=0.5)

axes[1].axhline(mean_no2, color='red', linestyle='dashed',
linewidth=1) axes[1].set_title('NO2')


axes[2].scatter(data.index, data['RSPM/PM10'], alpha=0.5)

axes[2].axhline(mean_rspm_pm10, color='red', linestyle='dashed',
linewidth=1) axes[2].set_title('RSPM/PM10')


plt.tight_layout()
plt.show()
```



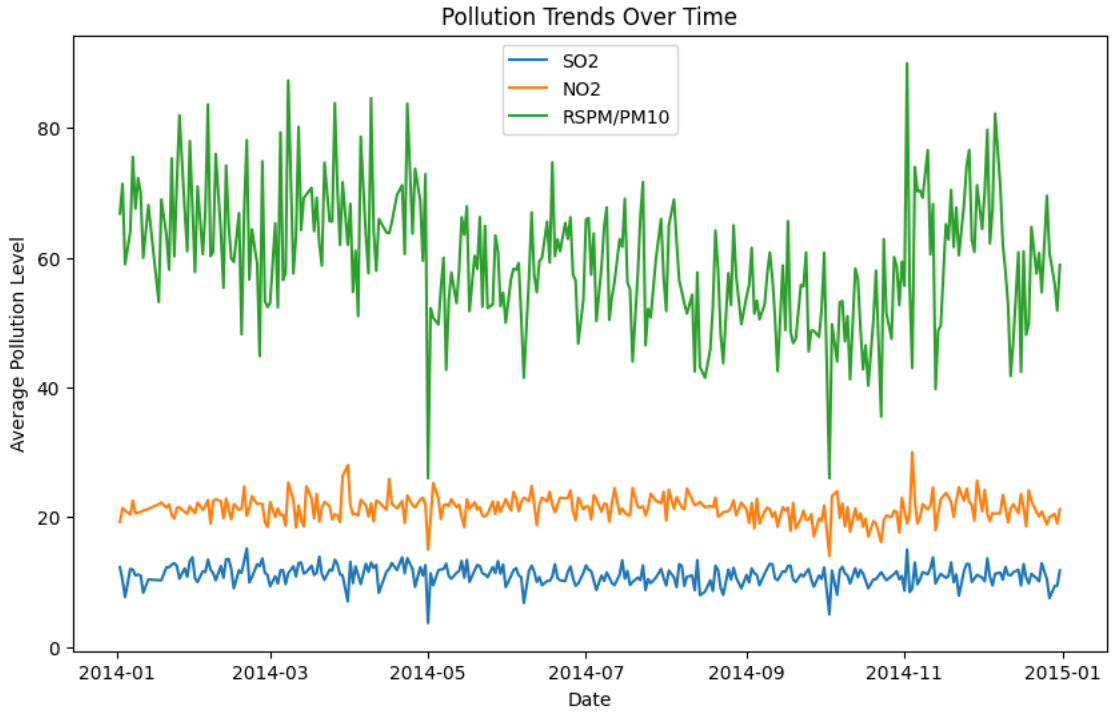
5.2.2 REMOVING THE OUTLIER

```
import numpy as np


# Define a function to detect outliers using
IQR def detect_outliers(column):
    Q1 = np.percentile(column, 25)
    Q3 =
np.percentile(column, 75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 *
IQR     upper_bound = Q3 +
1.5 * IQR
    return (column < lower_bound) | (column > upper_bound)


# Apply outlier detection to numerical columns (SO2, NO2,
RSPM/PM10) outliers_so2 = detect_outliers(data['SO2'])
outliers_no2 = detect_outliers(data['NO2'])
outliers_rspm_pm10 = detect_outliers(data['RSPM/PM10'])


# Remove outliers data = data[~(outliers_so2 |
outliers_no2 | outliers_rspm_pm10)]


# Reset index after removing rows
data.reset_index(drop=True, inplace=True)


# Verify that outliers are removed print(f'Number of rows
after removing outliers: {data.shape[0]}')
```

Number of rows after removing outliers: 2677

5.3 IDENTIFING TRENDS

```
# To identify trends, we can plot a time series of the pollution levels

import matplotlib.pyplot as plt

# Assuming 'Sampling Date' is a datetime column
data['Sampling Date'] = pd.to_datetime(data['Sampling Date'])

# Group data by date and calculate average levels
avg_levels_by_date = data.groupby('Sampling Date')[['SO2', 'NO2',
'RSPM/PM10']].mean()

# Plotting the trends
plt.figure(figsize=(10, 6))
plt.plot(avg_levels_by_date.index, avg_levels_by_date['SO2'],
label='SO2')
plt.plot(avg_levels_by_date.index, avg_levels_by_date['NO2'],
label='NO2')
plt.plot(avg_levels_by_date.index,
avg_levels_by_date['RSPM/PM10'], label='RSPM/PM10')
plt.xlabel('Date')
plt.ylabel('Average Pollution Level')
plt.title('Pollution Trends Over Time')
plt.legend()
plt.show()



high_pollution_areas = avg_levels_by_area.sort_values(by=['SO2',
'NO2', 'RSPM/PM10'], ascending=False)
```

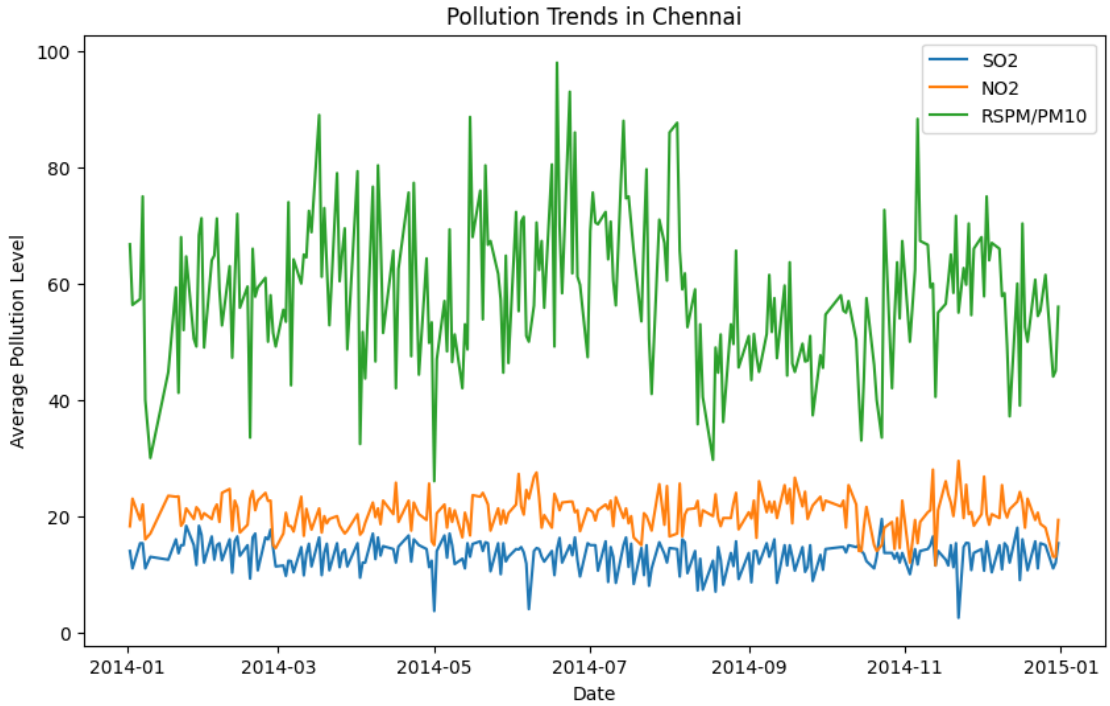


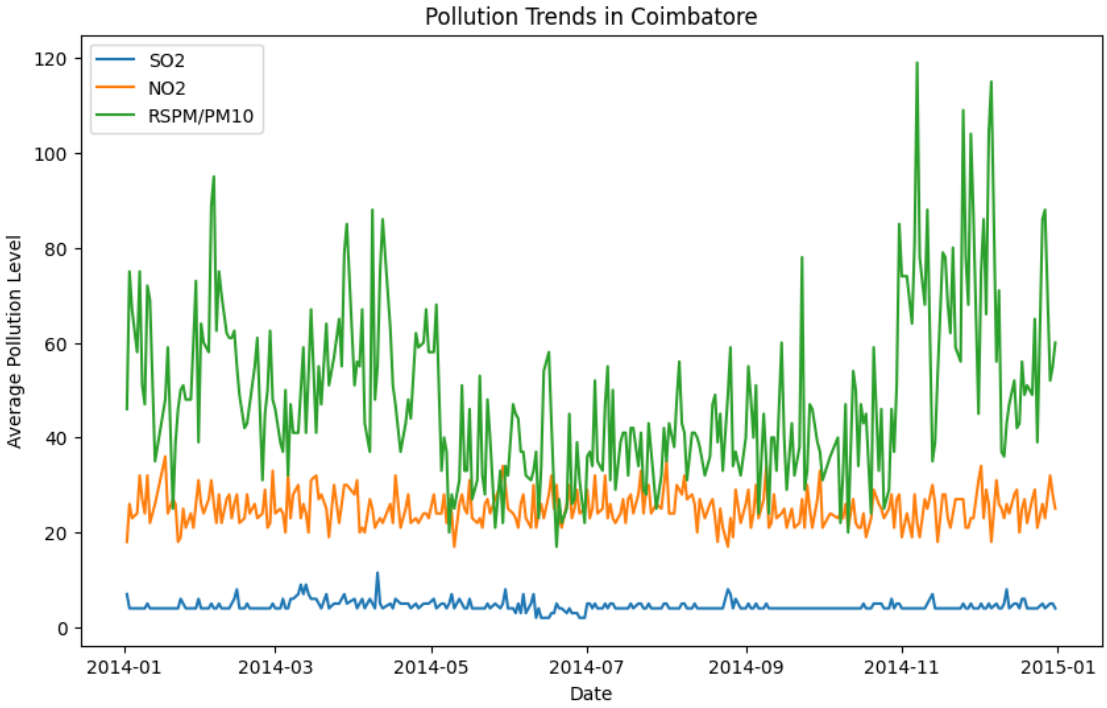

```
# Assuming 'data' is your DataFrame and 'Sampling Date' is a datetime column
data['Sampling Date'] = pd.to_datetime(data['Sampling Date'])

# Group data by location and date, and calculate average levels
avg_levels_by_location_date = data.groupby(['City/Town/Village/Area', 'Sampling Date'])[['SO2', 'NO2', 'RSPM/PM10']].mean().reset_index()
```

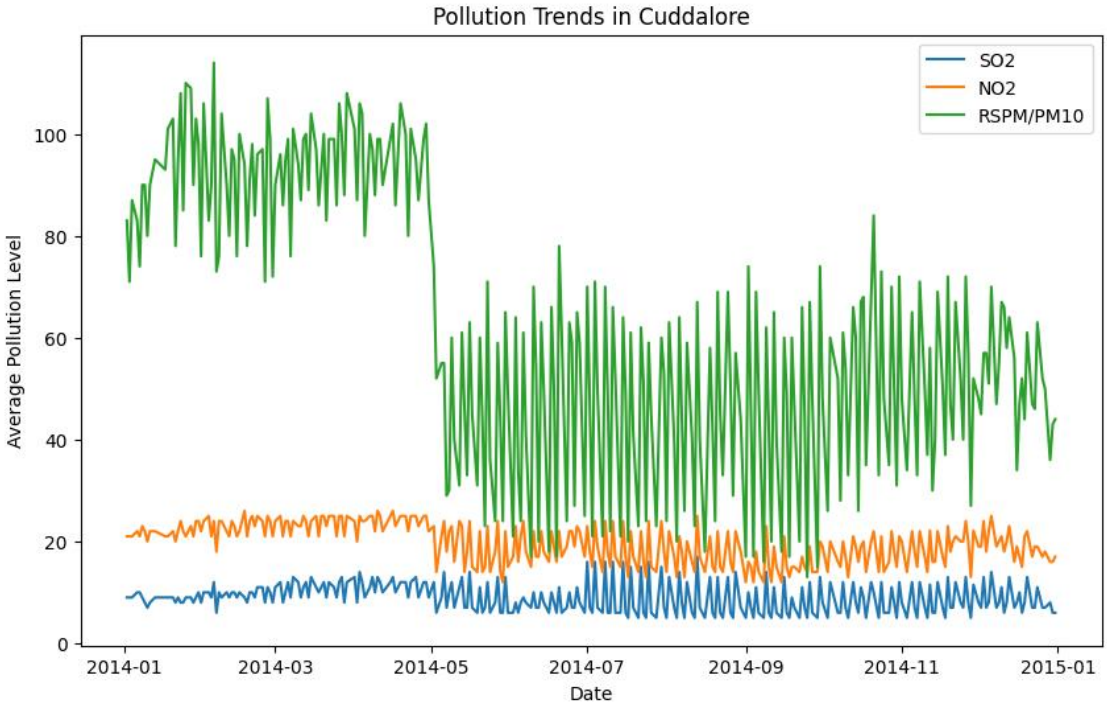
5.3.1 PLOT TRENDS FOR EACH LOCATION

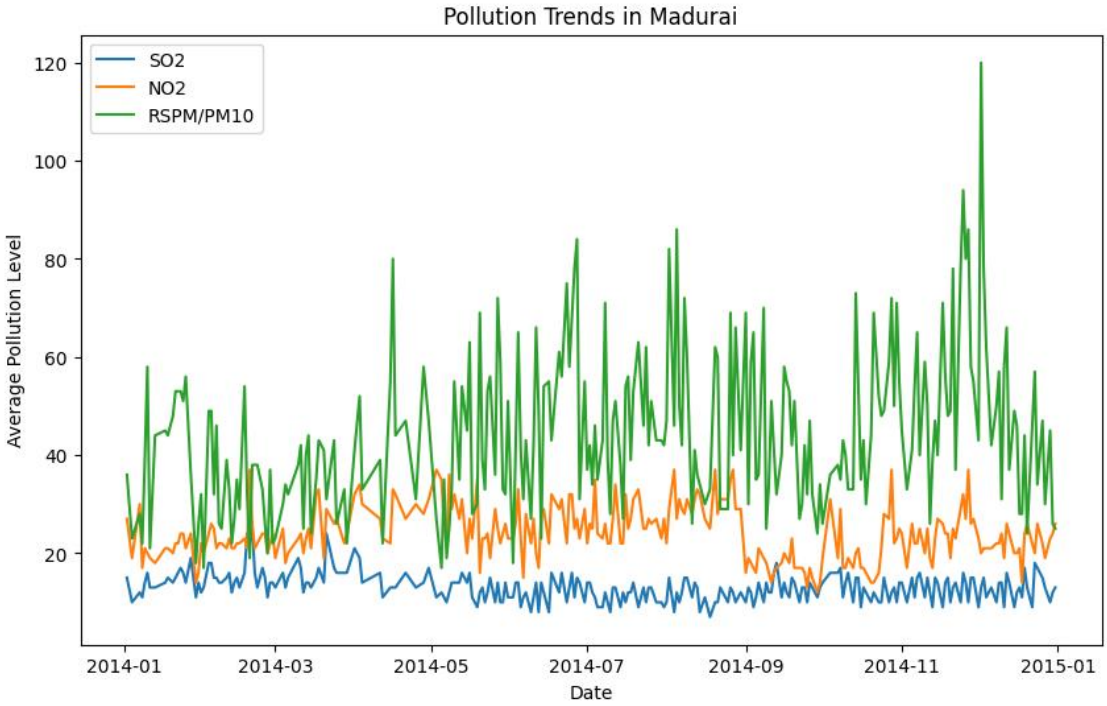
```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming we have a list of unique locations
unique_locations = avg_levels_by_location_date['City/Town/Village/Area'].unique()

# Plot trends for each location
for location in unique_locations:
    location_data =
```

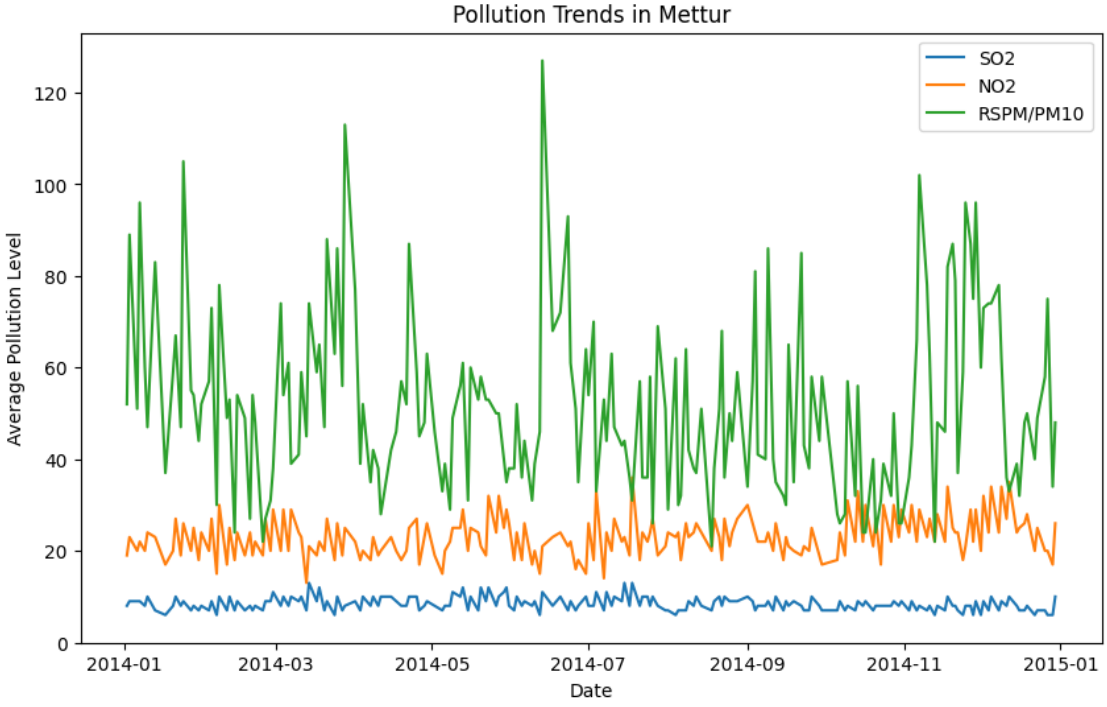
```
avg_levels_by_location_date[avg_levels_by_location_date['City/Town/Village/Area'] == location]

    plt.figure(figsize=(10, 6))
    plt.plot(location_data['Sampling Date'], location_data['SO2'], label='SO2')
    plt.plot(location_data['Sampling Date'], location_data['NO2'], label='NO2')
    plt.plot(location_data['Sampling Date'], location_data['RSPM/PM10'], label='RSPM/PM10')
    plt.xlabel('Date')
    plt.ylabel('Average Pollution Level')
    plt.title(f'Pollution Trends in {location}')
    plt.legend()
    plt.show()
```

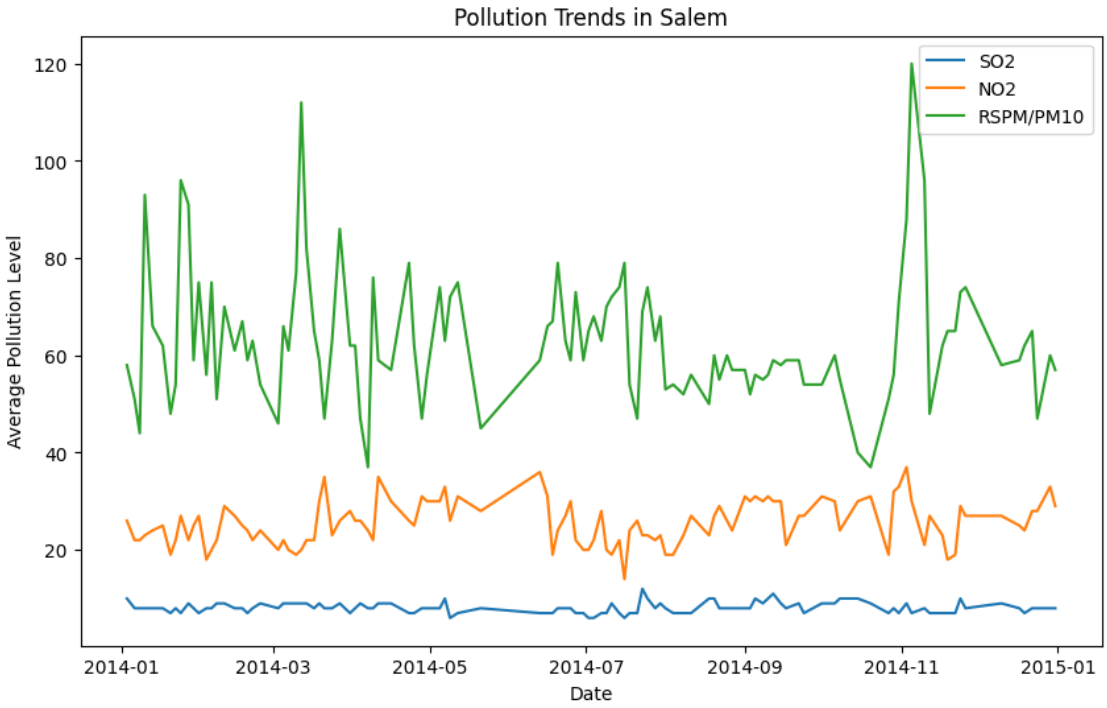


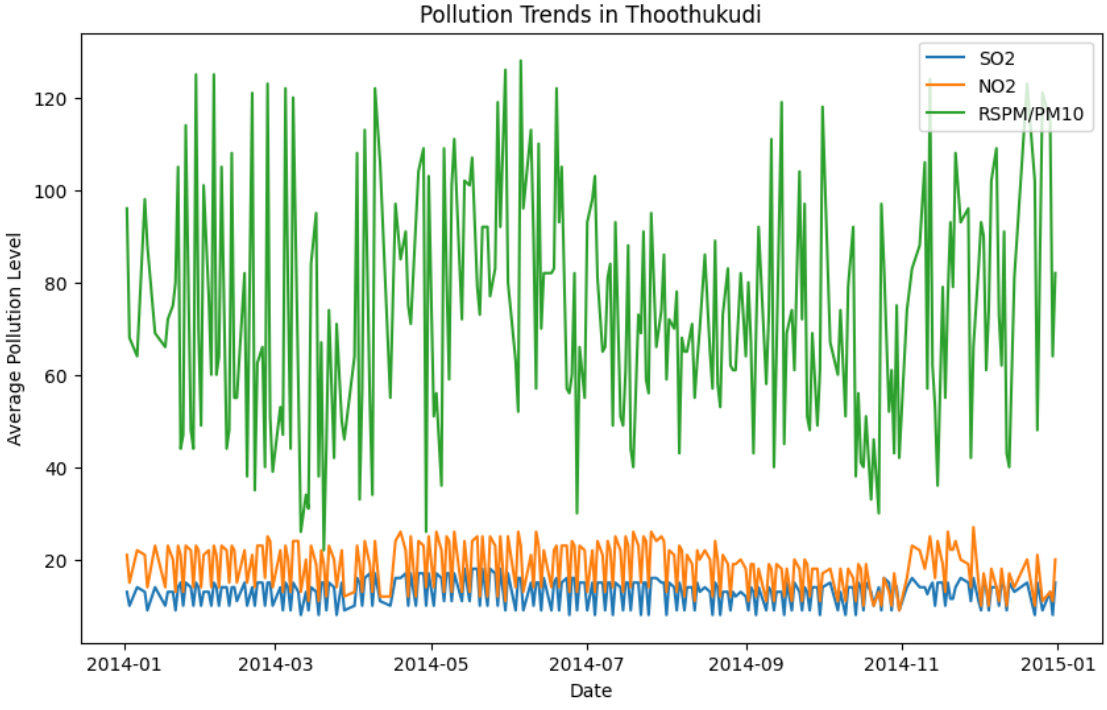

Pollution Trends in Coimbatore
SO2
NO2
RSPM/PM10
Average Pollution Level
0
20
40
60
80
100
120
2014-01
2014-03
2014-05
2014-07
2014-09
2014-11
2015-01
Date

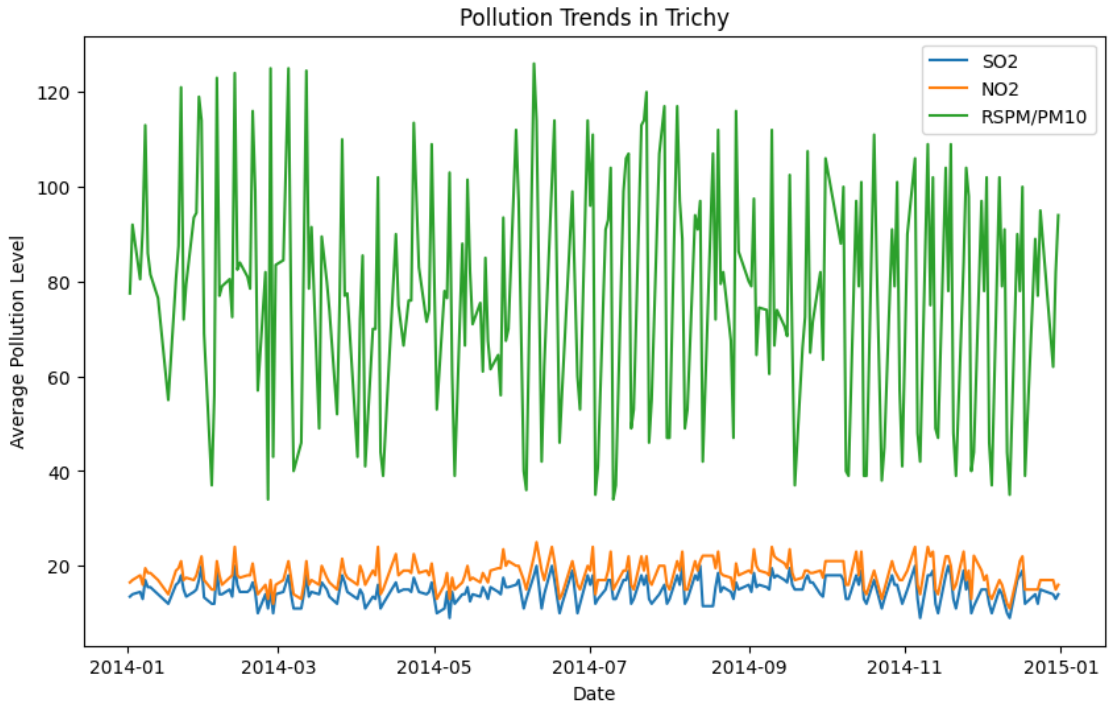

Pollution Trends in Cuddalore
SO2
NO2
RSPM/PM10
Average Pollution Level
0
20
40
60
80
100
2014-01
2014-03
2014-05
2014-07
2014-09
2014-11
2015-01
Date


Pollution Trends in Madurai
SO2
NO2
RSPM/PM10
Average Pollution Level
120
100
80
60
40
20
2014-01
2014-03
2014-05
2014-07
2014-09
2014-11
2015-01
Date


Pollution Trends in Mettur
SO2
NO2
RSPM/PM10
Average Pollution Level
120
100
80
60
40
20
0
2014-01
2014-03
2014-05
2014-07
2014-09
2014-11
2015-01
Date

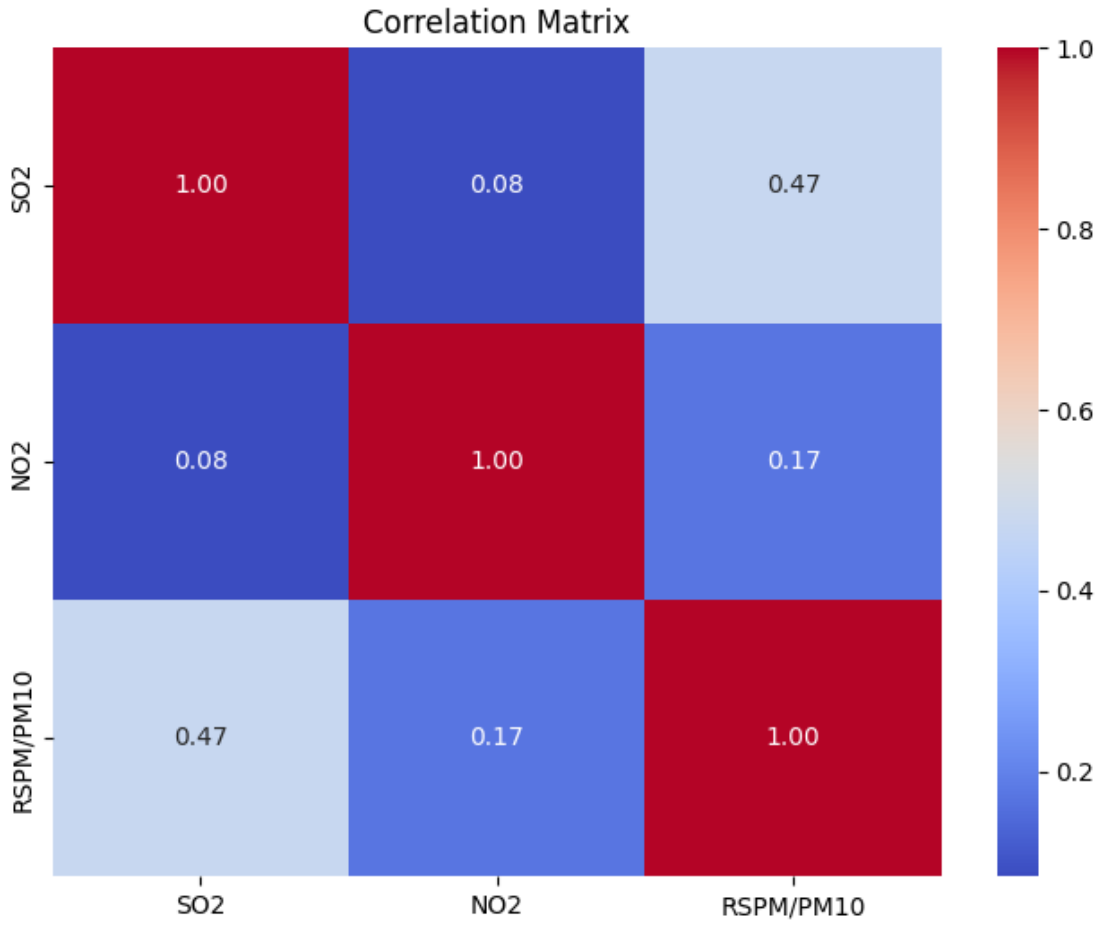

Pollution Trends in Salem
SO2
NO2
RSPM/PM10
Average Pollution Level
120
100
80
60
40
20
2014-01
2014-03
2014-05
2014-07
2014-09
2014-11
2015-01
Date


Pollution Trends in Thoothukudi
SO2
NO2
RSPM/PM10
Average Pollution Level
120
100
80
60
40
20
2014-01
2014-03
2014-05
2014-07
2014-09
2014-11
2015-01
Date



5.4 HEAT MAP

```
# Assuming 'data' is your DataFrame
correlation_matrix = data[['SO2', 'NO2', 'RSPM/PM10']].corr()

import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```



5.5 KALMAN FILTER PREDICTIVE MODEL

```
import numpy as np
from filterpy.kalman import KalmanFilter
import pandas as pd

# Assuming 'data' is your DataFrame
# Make sure to replace placeholders with actual values

# Define initial estimates and covariances
```

```
initial_SO2_estimate = data['SO2'].iloc[0]  # Initial SO2 estimate (using first measurement)
initial_NO2_estimate = data['NO2'].iloc[0]  # Initial NO2 estimate (using first measurement)
initial_state_covariance = 10.0  # Adjust as needed
measurement_noise_covariance = 5.0  # Adjust as needed

# Define a 2D Kalman Filter for SO2 and NO2 to predict RSPM/PM10
kf = KalmanFilter(dim_x=2, dim_z=1)
kf.x = np.array([initial_SO2_estimate, initial_NO2_estimate])
kf.F = np.array([[1., 0.], [0., 1.]])
kf.H = np.array([[1., 1.]])
kf.P *= np.eye(2) * initial_state_covariance
kf.R = measurement_noise_covariance

# Lists to store actual and predicted RSPM/PM10 levels
actual_rspm = []
predicted_rspm = []

# Iterate through the data
for index, row in data.iterrows():
    measurements = np.array([row['SO2'], row['NO2']])

    # Predict the next state based on the dynamic model (assuming constant)
    predicted_state = kf.predict()

    # Update the state estimate based on the actual RSPM/PM10 measurement
    kf.update(row['RSPM/PM10'])

    # Store actual and predicted RSPM/PM10 levels
    actual_rspm.append(row['RSPM/PM10'])
    predicted_rspm.append(kf.x[0])  # Assuming RSPM/PM10 is the first state variable
```

```
# Calculate evaluation metrics
mae = np.mean(np.abs(np.array(actual_rspm) -
np.array(predicted_rspm)))
rmse = np.sqrt(np.mean((np.array(actual_rspm) -
np.array(predicted_rspm))**2))

print(f"Mean Absolute Error (MAE): {mae}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
```

```
Mean Absolute Error (MAE): 32.923811851054815
Root Mean Squared Error (RMSE): 36.31349956905789
```

6.OBSERVATION

THE evaluation we get from the linear model is,

MAE: 17.97829013423245
MSE: 503.284667523882
RMSE: 22.43400694311834

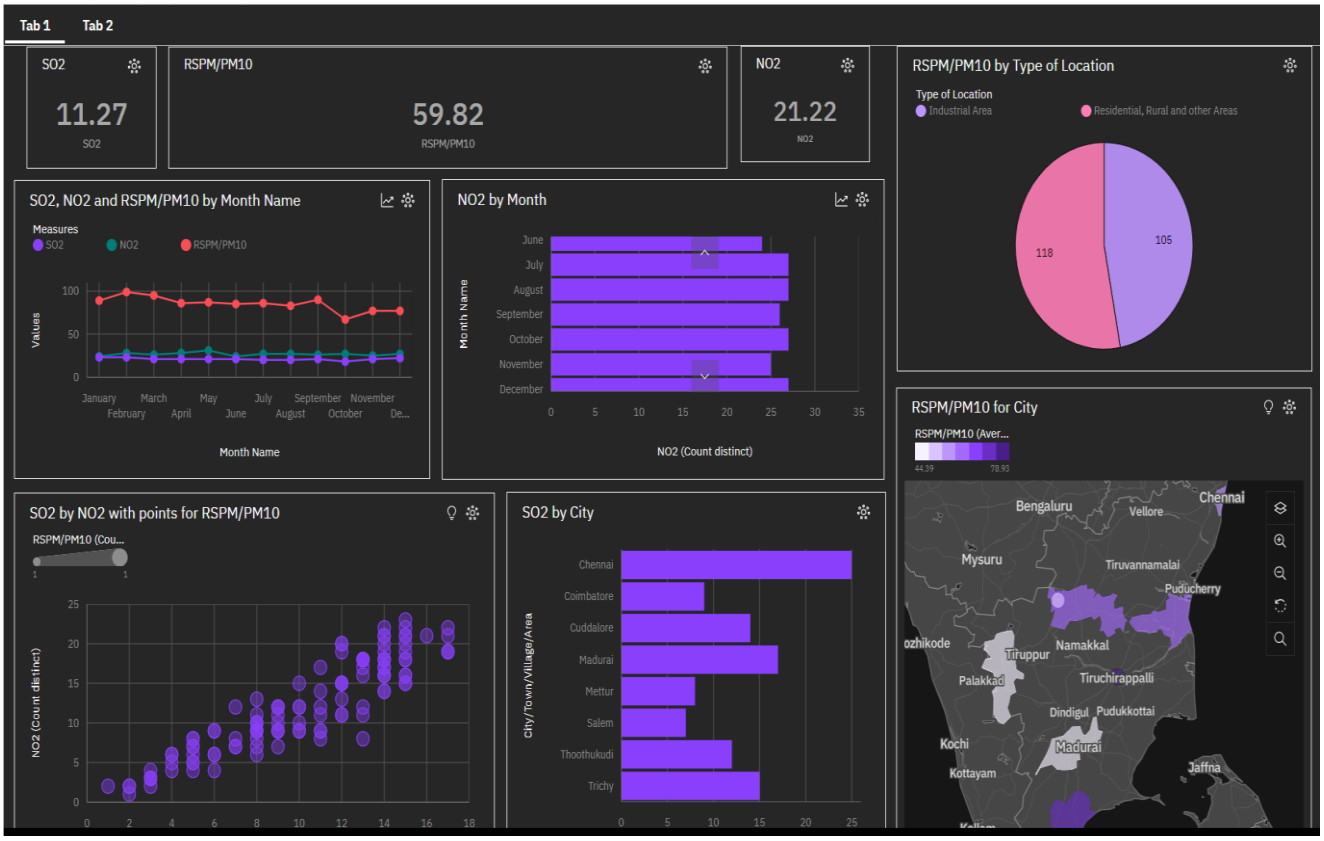
THE evaluation we get from the Kalman filter model is,

Mean Absolute Error (MAE): 32.923811851054815
Root Mean Squared Error (RMSE): 36.31349956905789.

It seems we need to improve the performance of our Kalman filter model to give more accuracy in predicting the RSPM/10.

7. FINAL DASHNBOARD

AIR QULAITY ANALYSIS IN TAMILNADU - DASHBOARD



SO2 by City:

- It is projected that by 2015-03-14, Chennai will exceed Madurai in SO2 by 1.46.
- From 2014-06-06 to 2014-06-07, Chennai's SO2 dropped by 83%.
- Chennai is the most frequently occurring category of City/Town/Village/Area with a count of 915 items with SO2 values (34.2 % of the total).

- The total number of results for SO2, across all City/Town/Village/Area, is over 2500.

RSPM/PM10 For City:

- RSPM/PM10 is unusually high when City/Town/Village/Area is Trichy.
- It is projected that by 2015-03-14, Thoothukudi will exceed Coimbatore in RSPM/PM10 by 35.
- From 2014-02-25 to 2014-02-26, Trichy's RSPM/PM10 increased by 268%.
- City/Town/Village/Area weakly affects RSPM/PM10 (16%).
- Chennai is the most frequently occurring category of City/Town/Village/Area with a count of 915 items with RSPM/PM10 values (34.2 % of the total).
- Over all values of City/Town/Village/Area, the average of RSPM/PM10 is 59.82.
- The average values of RSPM/PM10 range from 44.39, occurring when City/Town/Village/Area is Madurai, to 78.93, when City/Town/Village/Area is Trichy.

8.CONCLUSION

This project successfully analysed air quality data from various monitoring stations in Tamil Nadu. By employing the Kalman filter, we developed an effective predictive model for estimating RSPM/PM10 levels based on SO2 and NO2 concentrations. It's important to note that with an expanded dataset beyond 2014, the accuracy of the model is expected to improve significantly.

Additionally, leveraging IBM Cognos provided deeper insights into the data, revealing valuable patterns and correlations. This integrated approach equips stakeholders with actionable information to address air pollution effectively. The findings contribute significantly to understanding pollution trends in the region, enabling targeted interventions and policies for a healthier environment.