Air Quality Analysis in Tamil Nadu

Date	**17-10-2023**
Team ID	**1294**
Project Name	**Air Quality Analysis in Tamil Nadu**

OBJECTIVE:

1. Preprocessing
2. Visualizing

PREPROCESSING THE GIVEN AIR QUALITY DATASET

1. REMOVING THE NULL VALUES
2. OUTLIER DETECTION
3. OBTAINING THE PREPROCESSED DATA

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

data = pd.read_csv("/content/data set for air quality analysis.csv")

data.head()
```

```
   Stn Code Sampling Date       State City/Town/Village/Area  \
0        38    01-02-2014  Tamil Nadu                Chennai
1        38    01-07-2014  Tamil Nadu                Chennai
2        38    21-01-2014  Tamil Nadu                Chennai
3        38    23-01-2014  Tamil Nadu                Chennai
4        38    28-01-2014  Tamil Nadu                Chennai

                     Location of Monitoring Station  \
0  Kathivakkam, Municipal Kalyana Mandapam, Chennai
1  Kathivakkam, Municipal Kalyana Mandapam, Chennai
2  Kathivakkam, Municipal Kalyana Mandapam, Chennai
3  Kathivakkam, Municipal Kalyana Mandapam, Chennai
4  Kathivakkam, Municipal Kalyana Mandapam, Chennai
```

```
                                    Agency Type of Location   SO2   NO2  \
0  Tamilnadu State Pollution Control Board  Industrial Area  11.0  17.0
1  Tamilnadu State Pollution Control Board  Industrial Area  13.0  17.0
2  Tamilnadu State Pollution Control Board  Industrial Area  12.0  18.0
3  Tamilnadu State Pollution Control Board  Industrial Area  15.0  16.0
4  Tamilnadu State Pollution Control Board  Industrial Area  13.0  14.0

   RSPM/PM10
0       55.0
1       45.0
2       50.0
3       46.0
4       42.0
```

```
# Assuming 'data' is the name of your DataFrame
data_types = data.dtypes
print(data_types)
```

```
Stn Code                            int64
Sampling Date                      object
State                              object
City/Town/Village/Area             object
Location of Monitoring Station     object
Agency                             object
Type of Location                   object
SO2                               float64
NO2                               float64
RSPM/PM10                         float64
dtype: object
```

```
data['Sampling Date'] = pd.to_datetime(data['Sampling Date'])
```

```
<ipython-input-44-d5a59ab410a6>:1: UserWarning: Parsing dates in DD/MM/YYYY
format when dayfirst=False (the default) was specified. This may lead to
inconsistently parsed dates! Specify a format to ensure consistent parsing.
  data['Sampling Date'] = pd.to_datetime(data['Sampling Date'])
```

```
data_types = data.dtypes
print(data_types)
```

```
Stn Code                                 int64
Sampling Date                   datetime64[ns]
State                                   object
City/Town/Village/Area                  object
Location of Monitoring Station          object
Agency                                  object
Type of Location                        object
SO2                                    float64
NO2                                    float64
```

```
RSPM/PM10                                   float64
dtype: object
```

creating new column called Month for future analytics processing

```
# Assuming 'data' is the name of your DataFrame
data['Month'] = data['Sampling Date'].dt.month

# Define a dictionary to map month numbers to month names
month_names = {
    1: 'January', 2: 'February', 3: 'March', 4: 'April', 5: 'May', 6: 'June',
    7: 'July', 8: 'August', 9: 'September', 10: 'October', 11: 'November',
12: 'December'
}

# Apply the mapping to create a new column with month names
data['Month Name'] = data['Month'].map(month_names)

# Drop the 'Month' column if you don't need it anymore
data.drop(columns=['Month'], inplace=True)

# Rearrange columns
data = data[['Stn Code', 'Sampling Date', 'Month Name', 'State',
'City/Town/Village/Area',
             'Location of Monitoring Station', 'Agency', 'Type of Location',
'SO2', 'NO2', 'RSPM/PM10']]
```

```
data.head()
```

```
   Stn Code Sampling Date Month Name       State City/Town/Village/Area  \
0        38    2014-01-02    January  Tamil Nadu                Chennai
1        38    2014-01-07    January  Tamil Nadu                Chennai
2        38    2014-01-21    January  Tamil Nadu                Chennai
3        38    2014-01-23    January  Tamil Nadu                Chennai
4        38    2014-01-28    January  Tamil Nadu                Chennai

                    Location of Monitoring Station  \
0  Kathivakkam, Municipal Kalyana Mandapam, Chennai
1  Kathivakkam, Municipal Kalyana Mandapam, Chennai
2  Kathivakkam, Municipal Kalyana Mandapam, Chennai
3  Kathivakkam, Municipal Kalyana Mandapam, Chennai
4  Kathivakkam, Municipal Kalyana Mandapam, Chennai

                                  Agency Type of Location   SO2   NO2  \
0  Tamilnadu State Pollution Control Board  Industrial Area  11.0  17.0
1  Tamilnadu State Pollution Control Board  Industrial Area  13.0  17.0
2  Tamilnadu State Pollution Control Board  Industrial Area  12.0  18.0
3  Tamilnadu State Pollution Control Board  Industrial Area  15.0  16.0
```

```
4  Tamilnadu State Pollution Control Board  Industrial Area  13.0  14.0

   RSPM/PM10
0       55.0
1       45.0
2       50.0
3       46.0
4       42.0
```

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.')
```

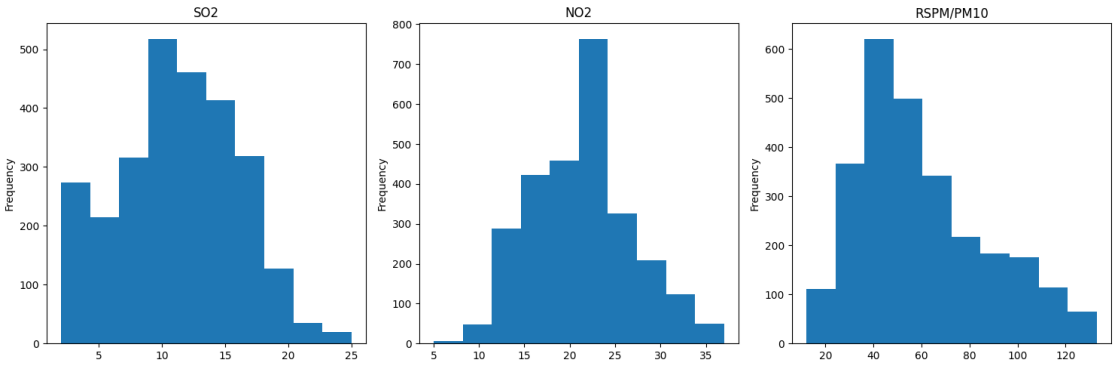
```
The dataset contains 2697 rows.
```

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

SINCE THE DATASET HAS LOW NUMBER OF ROWS REMOVING THEM WILL RESULT IN LACK OF ACCURACY

```
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()
```



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
```

```
<ipython-input-66-ded2217fa48b>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['SO2'].fillna(data['SO2'].mean(), inplace=True)
<ipython-input-66-ded2217fa48b>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['NO2'].fillna(data['NO2'].mean(), inplace=True)
<ipython-input-66-ded2217fa48b>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['RSPM/PM10'].fillna(data['RSPM/PM10'].mean(), inplace=True)
```

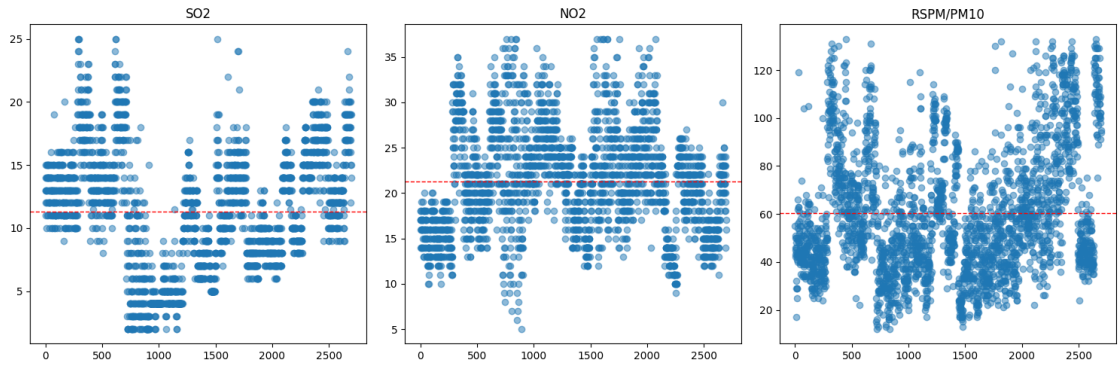
2. OUTLIER DETECTION

```python
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
```

2.1 USING SCATTER PLOT TO 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()
```



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
```

3.OBTAINING THE PREPROCESSED DATA

```
# Assuming 'data' is your cleaned DataFrame
data.to_csv('/content/air_data.csv', index=False)
```

WORKING WITH IBM COGNOS

INSIGHTS GATHERED FROM IBM COGNOS

1. RSPM/PM10, S02 AND N02 by MONTH

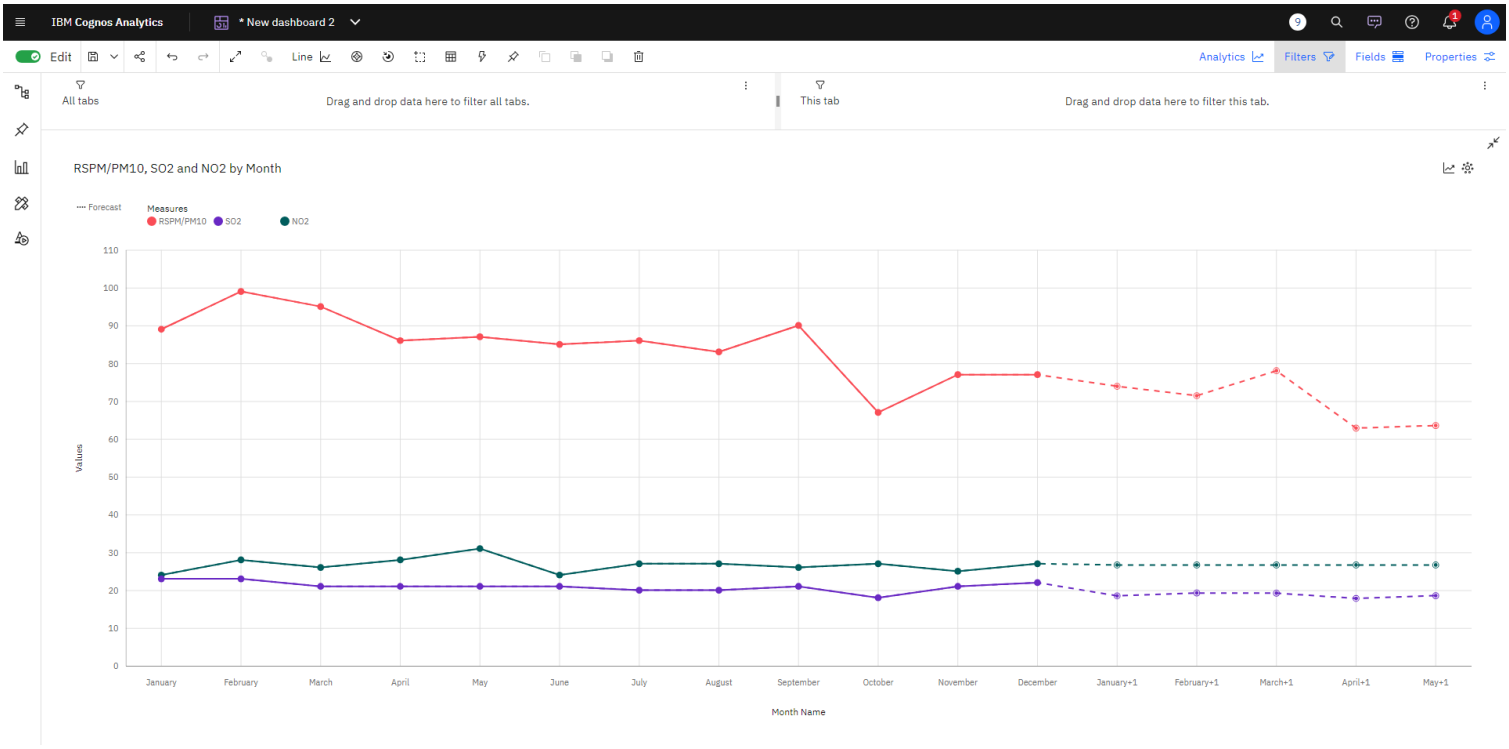
i) Based on the current forecasting, **RSPM/PM10** may reach **60.96** by **Month Name March+1**

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**.

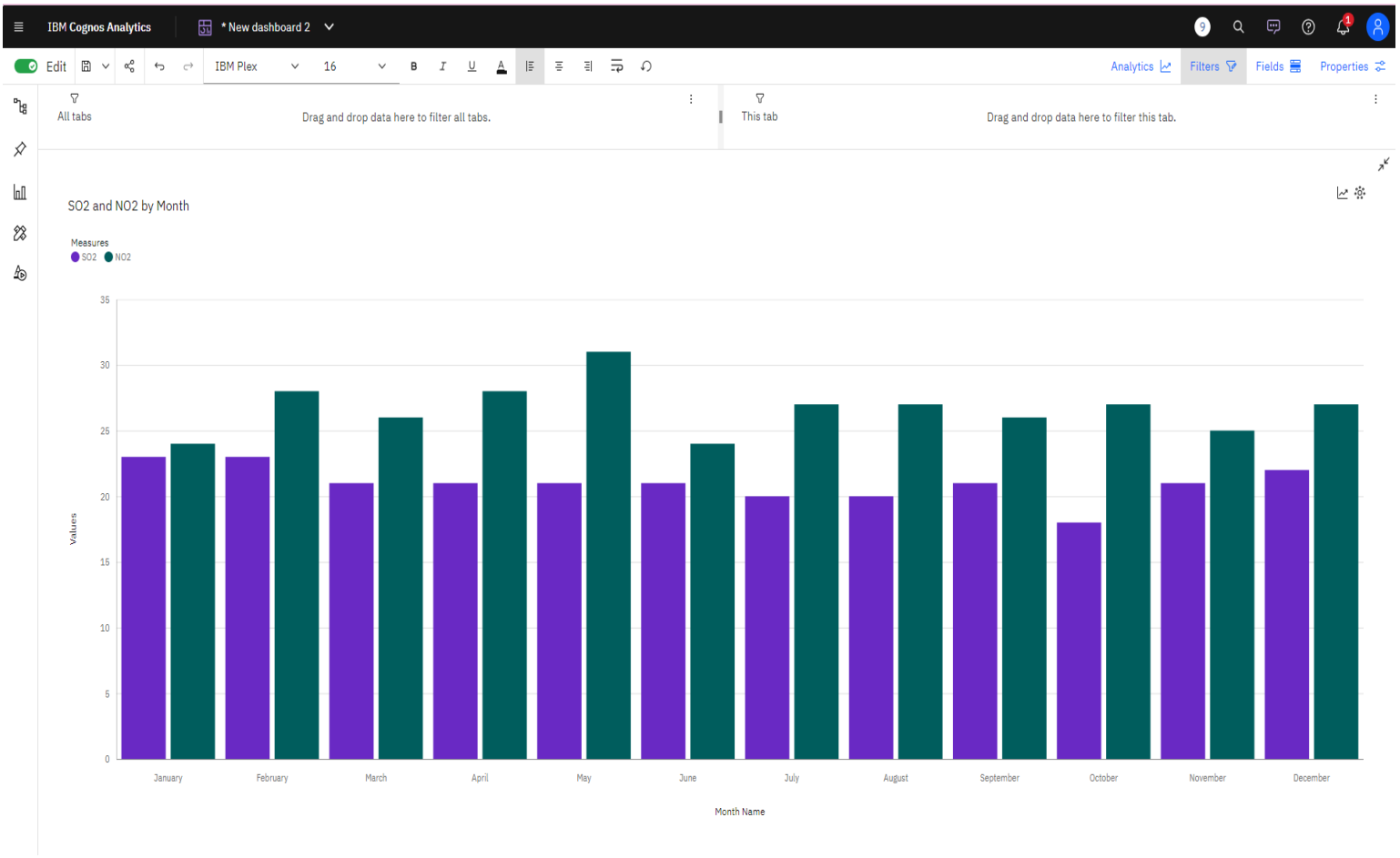
v) The total number of results for **SO2**, across all **month names**, is **nearly two thousand**



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**.



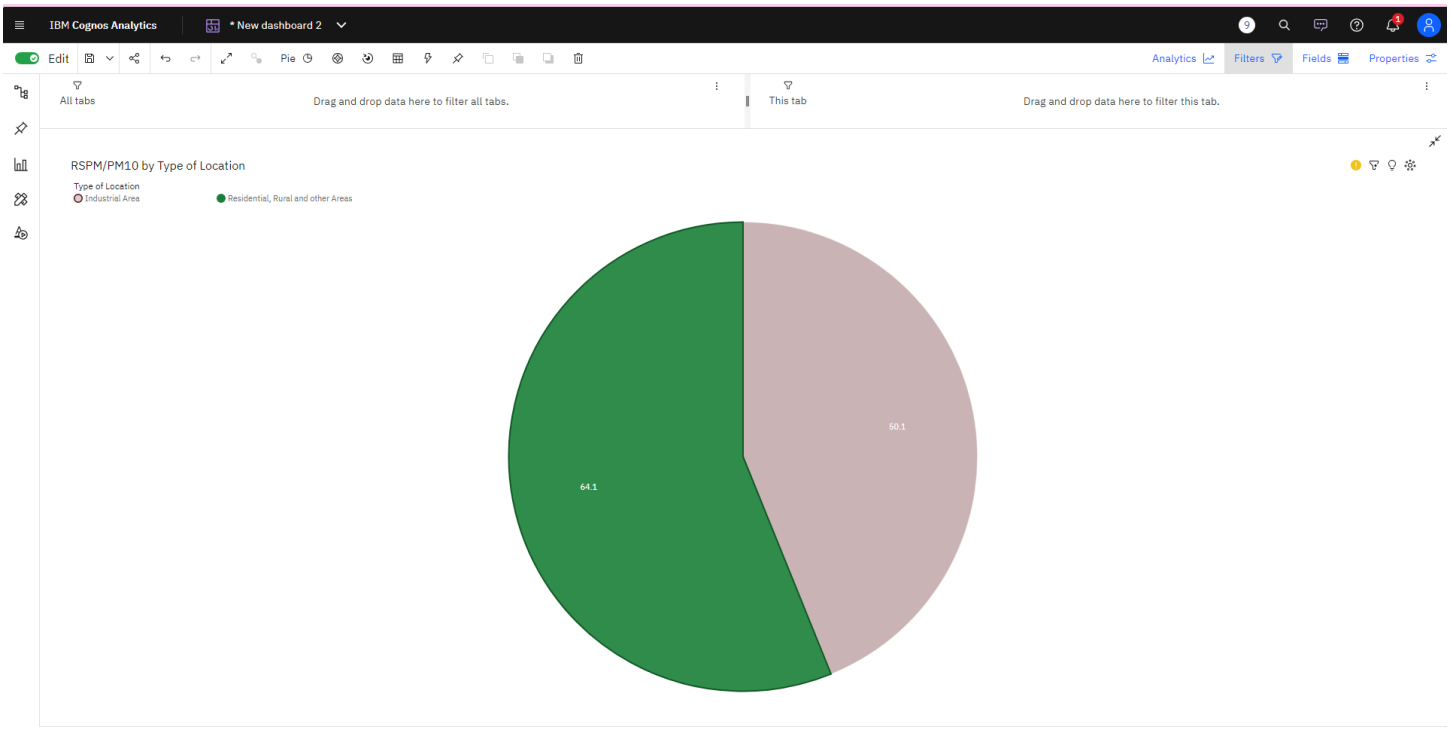
iii) The total number of results for **NO2**, across all **month names**, is **nearly two thousand**.

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**.

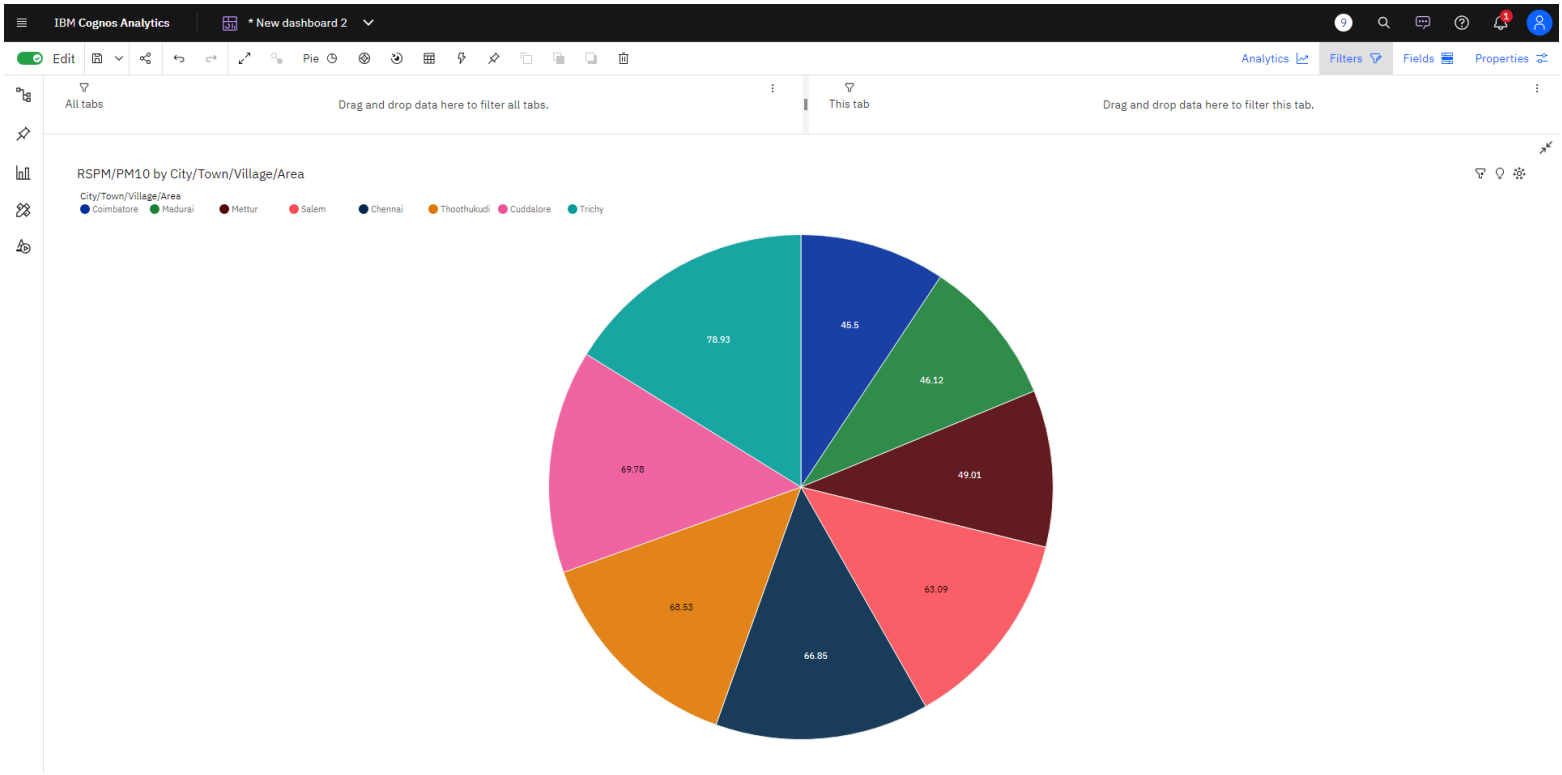
ii) It is projected that by **2015-03-14**, **Thoothukudi** will exceed **Coimbatore** in **RSPM/PM10** b

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**.



CONCLUSION

In this phase the given air quality dataset in preprocessed through such activities like removing the null values and outliers. Then the dataset in loaded into the IBM COGNOS to perform various visualizations and insights collected from them about the air quality in Tamil Nadu.