

**COLLEGE CODE: 5113**

# APPLIED DATA SCIENCE-phase 5

**AIR QUALITY ANALYSIS AND**

**PREDICTION IN TAMILNADU-project9**

***TEAM MEMBERS:***

* R.AFREEN TAJ-au511321104003
* B.DEEPIKA-au511321104017
* J.DELSI JULIE-au511321104019
* J.DHANU PRIYA-au511321104020

***INTRODUCTION:***

Air quality analysis and prediction are crucial components of environmental science and public health management. These fields focus on monitoring, assessing, and forecasting the quality of the air we breathe, with the ultimate goal of improving air quality and minimizing its adverse effects on human health and the environment.

Air quality analysis involves the collection and examination of data related to various pollutants present in the Earth's atmosphere, such as particulate matter (PM2.5 and PM10), ground-level ozone (O3), nitrogen dioxide (NO2), sulfur dioxide (SO2), carbon monoxide (CO), and volatile organic compounds (VOCs). Monitoring stations, satellite technology, and specialized instruments continuously measure these pollutants to generate real-time data.

***ABOUT DATASET:***

Where did we get the dataset?

**Kaggle:**

* we got our ‘Air quality analysis and prediction in Tamil Nadu’ dataset from Kaggle. [www.kaggle.com/data](http://www.kaggle.com/data).

Dataset link: [**https://tn.data.gov.in/resource/location-wise-daily-ambient-air-quality-tamil-nadu-year-2014**](https://tn.data.gov.in/resource/location-wise-daily-ambient-air-quality-tamil-nadu-year-2014)

* Kaggle is a popular platform for sharing datasets and hosting data science competitions.

**Details:**

* The dataset contains all details about the quality of air in Tamil Nadu which is for predicting the quality of Air.
* Where the air contains S02,NO2,etc., which helps us to analyze the Air quality in various cities of Tamil Nadu.

The dataset contains the following details,

* Sample: Unique identifiers for data measurements.
* State: Denotes Tamil Nadu as the data's geographical origin.
* City: Names of cities in Tamil Nadu where monitoring occurred.
* Location of Monitoring Station: Specific addresses or locations of monitoring stations.
* Agency: Organizations responsible for data collection and analysis.
* SO2 (Sulfur Dioxide): Concentrations of sulfur dioxide, a harmful gas from combustion.
* NO2 (Nitrogen Dioxide): Levels of nitrogen dioxide, an air pollutant from various sources.
* RSPM10 (PM10): Measurements of particulate matter with a diameter of 10 micrometers.
* PM 2.5 (Particulate Matter 2.5): Data on fine particulate matter with a diameter of 2.5 micrometers, posing health risks.

***COLUMNS THAT WE USED:***

**Sample:**

This column likely contains identifiers or labels for individual data samples or measurements. Each row in your dataset represents a specific data sample, and the "Sample" column helps distinguish one sample from another. It may consist of unique numerical or alphanumeric identifiers.

**State:**

This column indicates the state within India where the air quality measurements were taken. In your case, it is Tamil Nadu, which is a state in the southern part of India.

**City:**

This column specifies the name of the city or urban area within Tamil Nadu where the air quality monitoring was conducted. Different cities within Tamil Nadu may have varying levels of air pollution due to factors like industrial activity, traffic, and geography.

**Location of Monitoring Station:**

This column provides information about the specific location or address of the monitoring station within the city. Monitoring stations are strategically placed to capture air quality data from different parts of the city, including industrial, residential, and commercial areas.

**Agency:**

This column likely mentions the organization or agency responsible for conducting the air quality monitoring and data collection. Various government agencies and environmental organizations are typically involved in managing and reporting air quality data.

**SO2 (Sulfur Dioxide):**

This column contains measurements of sulfur dioxide (SO2) concentrations in the air. SO2 is a gaseous air pollutant primarily produced by the burning of fossil fuels, such as coal and oil. It can have adverse health effects and contribute to acid rain and smog formation.

**NO2 (Nitrogen Dioxide):**

This column contains measurements of nitrogen dioxide (NO2) concentrations in the air. NO2 is a common air pollutant produced by combustion processes, including vehicle emissions and industrial activities. High levels of NO2 can irritate the respiratory system and contribute to the formation of ground-level ozone.

**RSPM10 (Respirable Suspended Particulate Matter - PM10):**

This column includes measurements of particulate matter with a diameter of 10 micrometers or less (PM10). RSPM10 consists of tiny solid particles or liquid droplets suspended in the air. These particles can come from various sources, including dust, construction, and industrial emissions, and can have adverse health effects when inhaled.

**PM 2.5 (Particulate Matter 2.5):**

This column contains measurements of fine particulate matter with a diameter of 2.5 micrometers or less (PM2.5). PM2.5 is even smaller and more harmful than PM10, as it can penetrate deeper into the lungs and pose significant health risks. It originates from similar sources as PM10 but has a more significant impact on respiratory and cardiovascular health.

***DETAILS OF LIBRARIES USED AND WAY TO DOWNLOAD:***

**Pandas (Data Manipulation):**

Pandas is a fundamental library for data manipulation and analysis, offering data structures like Data Frames and Series.

**Installation:**

Use the following command to install Pandas: pip install pandas

**NumPy (Numerical Computing):**

NumPy is essential for numerical operations, providing support for arrays and matrices.

**Installation:**

Install NumPy with the following command: pip install numpy

**Matplotlib (Data Visualization):**

Matplotlib is a widely used library for creating static and interactive data visualizations.

**Installation:**

Use this command to install Matplotlib:

pip install matplotlib

**Seaborn (Statistical Data Visualization):**

Seaborn is built on top of Matplotlib and simplifies the creation of informative statistical graphics.

**Installation:**

Install Seaborn with pip: pip install seaborn.

**Scikit-learn (Machine Learning):**

Scikit-learn is a powerful library for machine learning and data mining tasks.

**Installation:**

Use this command to install Scikit-learn: pip install scikit-learn

***BEGINNING WITH THE PROJECT***

To begin building a project for air quality analysis and prediction, we first need to load the dataset.

We have a dataset file in a common format like CSV, here are the steps to load the dataset:

**1.Importing the required Libraries(data.csv):**

In this step, we import the necessary Python libraries and modules to work with our data and perform various data processing and machine learning tasks.eg,

## import pandas as pd

## import numpy as np

## from sklearn.preprocessing import Imputer

## from sklearn.model\_selection import train\_test\_split

## from sklearn.preprocessing import StandardScaler

## from sklearn.preprocessing import OneHotEncoder

**2.Importing the data set(read data set; create matrix ):**

This step involves loading our dataset into memory. We use libraries like pandas to read data from a CSV file or other formats. After loading, we create a feature matrix (often denoted as X) and a target vector (often denoted as Y).eg,

## dataset = pd.read\_csv('data.csv')

## X = dataset.iloc[:, :-1].values

## Y = dataset.iloc[:, -1].values

**3.Handling the Missing Data.(sklearn.preprocessing library contains class called imputer, helps in missing data):**

Datasets often have missing values. The sklearn.preprocessing.Imputer class is used to address this issue. You can specify a strategy for imputing missing values, such as replacing them with the mean, median, or mode of the column.eg,

## imputer = Imputer(missing\_values='NaN', strategy='mean', axis=0)

## imputer = imputer.fit(X[:, columns\_with\_missing\_data])

## X[:, columns\_with\_missing\_data] = imputer.transform(X[:, columns\_with\_missing\_data])

**4.Encoding Categorical Data.(one-hot encoding):**

One-hot encoding is a technique used to convert categorical data into a numerical format. Each category becomes a binary feature (0 or 1) in a new column, making it suitable for machine learning algorithms.eg,

## Encode=OneHotEncode(categoricalfeatures=categoricalcolumn)

## X = encode.fit\_transform(X).toarray()

**5.Splitting the data set into test set and training set.( import train train\_test\_split)(X\_train,X\_test, Y\_train,Y\_test):**

Before building a machine learning model, it's essential to divide our dataset into two sets: a training set and a test set. The training set is used to train the model, while the test set is used to evaluate its performance.eg,

## X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=0)

**6.Feature Scaling.(import StandardScaler):**

Feature scaling ensures that all features have the same scale, typically with a mean of 0 and a standard deviation of 1.

## scaler = StandardScaler()

## X\_train = scaler.fit\_transform(X\_train)

## X\_test = scaler.transform(X\_test)

***HOW TO TRAIN AND TEST:***

Training and testing a dataset is a fundamental step in building and evaluating machine learning models. The process involves dividing your dataset into two parts: one for training the model and the other for testing the model's performance. Here are the general steps to train and test a dataset:

**Data Splitting:**

* Start by splitting your dataset into two subsets: a training set and a testing set.
* The common practice is to allocate a larger portion of the data for training, often around 70-80% of the dataset, and the remaining portion for testing.

**Data Preprocessing:**

* Before training or testing, preprocess the data. This may involve tasks such as data cleaning, feature selection, handling missing values, and scaling or normalizing features to ensure the data is in a suitable format for the machine learning algorithm.
* Select a Machine Learning Algorithm.
* Choose an appropriate machine learning algorithm based on your problem's characteristics. Common algorithms include regression, classification, clustering, and more specialized models for specific tasks.

**Training the Model:**

* Use the training data to fit the chosen machine learning model. This involves feeding the algorithm with features (input variables) and their corresponding labels (output or target variable).
* The model will learn patterns and relationships in the training data.

**Model Evaluation:**

* Once the model is trained, use the testing set to evaluate its performance. The model has not seen this data during training, making it a suitable benchmark for assessing generalization.
* Common evaluation metrics include accuracy, precision, recall, F1-score, mean squared error (MSE), etc., depending on the type of problem (classification, regression, etc.).

**Tuning and Validation (Optional):**

* If the model's performance is unsatisfactory, you can fine-tune hyperparameters or explore different algorithms. Cross-validation techniques can help optimize the model further.

**Make Predictions (Inference):**

* After achieving a satisfactory model, you can use it to make predictions on new, unseen data.

**Assess Model on Real Data (Deployment):**

* if the model performs well during testing, you can deploy it for real-world applications, where it can make predictions based on incoming data.

**Monitoring and Maintenance:**

* Continuously monitor your model's performance in the real world and update it as necessary to adapt to changing data distributions or requirements.

***METRICS USED FOR ACCURACY CHECK:***

Metrics used for the analysis and prediction of a dataset depend on the type of problem you are addressing—whether it's a classification problem, regression problem, clustering problem, or any other specific task. Here are some commonly used metrics for different types of data analysis and prediction tasks:

* **Accuracy**: Measures the proportion of correctly classified instances. It's suitable when classes are balanced.
* **Mean Absolute Error (MAE):** Measures the average absolute difference between predicted and actual values.
* **Mean Squared Error (MSE):** Measures the average squared difference between predicted and actual values. It gives more weight to larger errors.
* **Root Mean Squared Error (RMSE):** The square root of MSE, providing a more interpretable measure.

**PERFORMING DIFFERENT ANALYSIS**:

Performing different types of analysis on a dataset depends on the goals of your analysis and the nature of the data. Here are some common types of analysis that you might perform on a dataset:

**Descriptive Analysis:**

Summarize and describe the main characteristics of the dataset, including measures of central tendency, dispersion, and visualizations such as histograms, box plots, and bar charts.

**Exploratory Data Analysis (EDA):**

Explore the dataset to uncover patterns, relationships, and anomalies. Visualize data using scatter plots, heatmaps, and correlation matrices.

**Statistical Analysis:**

Conduct hypothesis testing and statistical inference to make inferences about the data. Perform t-tests, ANOVA, chi-squared tests, and other statistical tests as appropriate.

**Regression Analysis:**

Build regression models to predict a continuous target variable based on one or more predictor variables. Evaluate model performance using metrics like R-squared, Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).

**Classification Analysis:**

Develop classification models to predict categorical outcomes or classes .Evaluate model performance using metrics such as accuracy, precision, recall, F1-score, and ROC curves.

**Clustering Analysis:**

Apply clustering algorithms to group similar data points together.

***PREPROCESSING THE DATASET:***

Preprocessing of data in a dataset refers to the various techniques and operations applied to the data before using it for analysis, modeling, Here's a more detailed explanation of data preprocessing within the context of a dataset:

**1. Data Cleaning:**

* Handling Missing Values: Identify and deal with missing data, which may involve filling in missing values, removing rows with missing data, or using imputation techniques.
* Dealing with Duplicates: Detect and remove duplicate records to ensure data integrity.

**2. Data Transformation:**

* Feature Scaling: Normalize or standardize numerical features to bring them to a similar scale. This is important for algorithms sensitive to feature scales.
* Feature Encoding: Convert categorical variables into a numerical format using techniques like one-hot encoding or label encoding.
* Feature Engineering: Create new features or modify existing ones to capture relevant information and patterns in the data.
* Binning: Group continuous data into bins or categories to simplify analysis.
* Log Transformation: Apply logarithmic transformations to features when necessary to make their distribution more normal.

**3. Data Reduction:** Dimensionality Reduction: Reduce the number of features, often using techniques like Principal Component Analysis (PCA) or feature selection to select the most relevant variables.

Outlier Detection and Handling: Identify and deal with outliers, which can distort analysis and modeling results.

**4. Data Integration:**

Merge data from multiple sources or datasets to create a consolidated dataset for analysis.

**5. Data Formatting:**

Ensure data is in the appropriate format and structure for the specific analysis or modeling task.

**6. Data Splitting:**

Split the dataset into subsets, such as training, validation, and test sets, for model development, evaluation, and testing.

**7. Data Validation:**

Verify data consistency, correctness, and integrity by checking for data entry errors and anomalies.

***CALCULATE AVERAGE OF SO2,NO2,RSPM/PM 25:***

To calculate average SO2, NO2, and RSPM/PM10 levels across different monitoring stations, cities, or areas and identify pollution trends, we use Python and popular data manipulation libraries like Pandas for data processing and Matplotlib for data visualization**.**

To compute the average:

**Gather data:**

Collect data on SO2, NO2, and RSPM concentrations from various monitoring stations or areas,

typically recorded at specific time intervals.

**Summation:**

Sum up the values of SO2, NO2, and RSPM over the selected time period or areas.

**Division:**

Divide the total sum of each pollutant by the number of data points or monitoring stations.

This calculation yields the average concentration of each pollutant

**Interpretation:**

The resulting averages provide a quantitative assessment of the typical air quality for SO2, NO2, and RSPM in the specified region during the given time frame.

**ACTUAL CODE**

## import pandas as pd

## import matplotlib. pyplot as plt

## import seaborn as sns

## df =pd.read\_csv("C:/Users/WIN10/Downloads/cpcb\_dly\_aq\_tamil\_nadu-2014.csv")

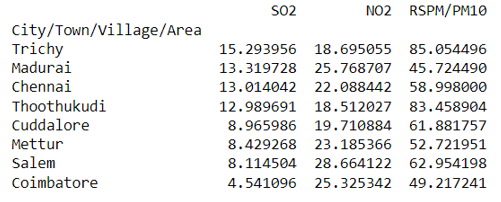
## average\_levels = df.groupby('City/Town/Village/Area')[['SO2', 'NO2', 'RSPM/PM10']].mean()

## print(average\_levels)

## sorted\_average\_levels = average\_levels.sort\_values(by=['SO2', 'NO2', 'RSPM/PM10'], ascending=False

## print(sorted\_average\_levels)

**OUTPUT**



The above is the output for calculating average for the given dataset

**CREATE VISUALIZATION**

We can also visualize the model prediction for the air quality analysis to gain insight into its performance.

Interactive graphs can be created which makes it easier to check air quality and increasingly diverse colors can visually highlight the air quality level

# Create bar plots for SO2, NO2, and RSPM/PM10 level

## plt.figure(figsize=(12, 6)

## plt.bar(sorted\_average\_levels.index, sorted\_average\_levels['SO2'], label='SO2')

## plt.bar(sorted\_average\_levels.index, sorted\_average\_levels['NO2'], bottom=sorted\_average\_levels['SO2'], label='NO2')

## plt.bar(sorted\_average\_levels.index,sorted\_average\_levels['RSPM/PM10'],bottom=sorted\_average\_levels['SO2']sorted\_average\_levels['NO2'], label='RSPM/PM10')

## plt.xlabel('City/Town/Village/Area')plt.ylabel('Average Levels')

## plt.title('Average Air Quality Levels by Area')

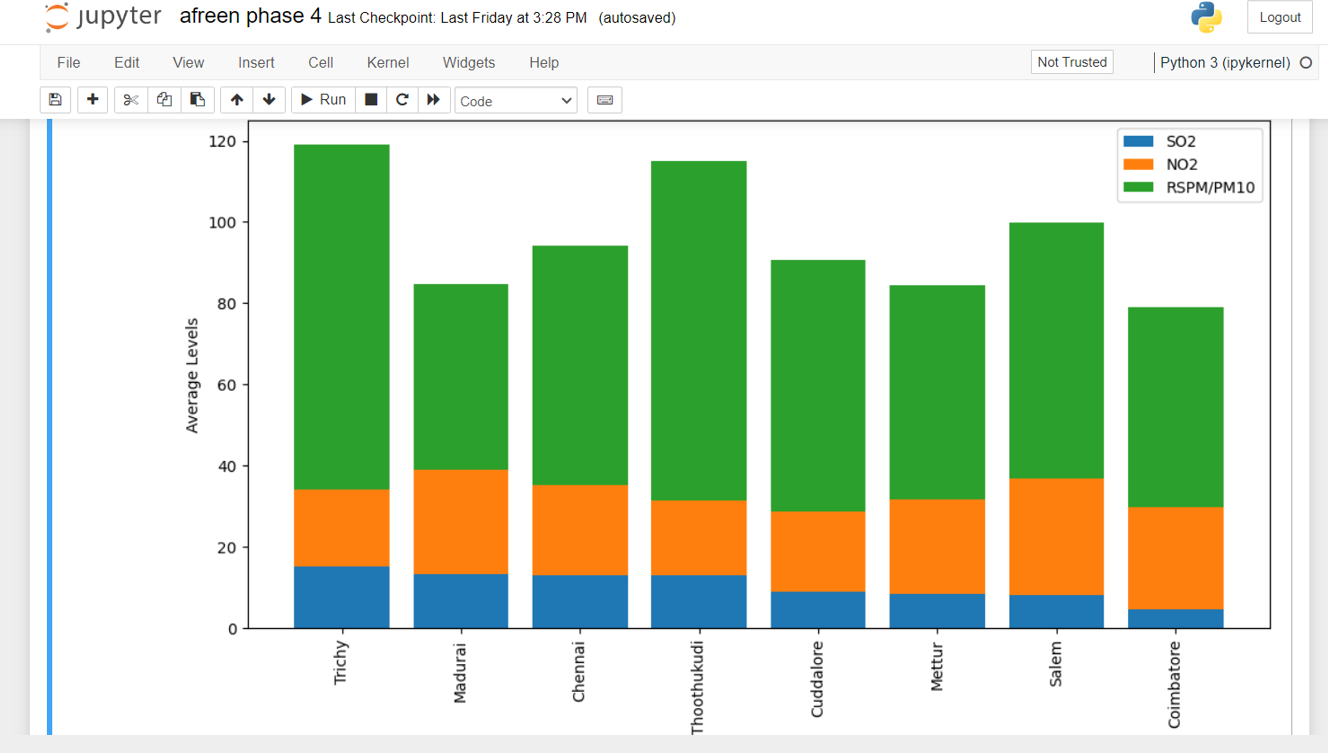
## plt.title('Average Air Quality Levels by Area')

## plt.xticks(rotation=90)

## plt.legend()

## plt.show()

***output***



**CODE FOR DISPLAYING HIGHLY POLLUTED AREA**

The code is to identify the which air quality is highly polluted in the given area .

Here SO2 is the highly polluted air in trichy

## import numpy as np # linear algebra

## import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

## import matplotlib.pyplot as plt

## import seaborn as sns

## import chart\_studio.plotly as py

## # Warnings

## import warnings

## warnings.filterwarnings('ignore')

## df = pd.read\_csv("C:/Users/haris/Downloads/cpcb\_dly\_aq\_tamil\_nadu-2014.csv")

## print(df.head(10))

## print(df.columns)

## print(df["City/Town/Village/Area"].nunique())

## print(df.isnull().sum())

## print(df.dtypes)

## data=df

## print(data.info())

## data[['SO2','City/Town/Village/Area']].groupby(["City/Town/Village/Area"]).median().sort\_values(by='SO2',ascending=False).head(10).plot.bar(color='r')

## plt.show()

## data[['SO2','City/Town/Village/Area']].groupby(["City/Town/Village/Area"]).median().sort\_values(by='SO2',ascending=False).tail(10).plot.bar(color='r')

## plt.show()

## data[['NO2','City/Town/Village/Area']].groupby(["City/Town/Village/Area"]).median().sort\_values(by='NO2',ascending=False).head(10).plot.bar(color='g')

## plt.show()

## data[['NO2','City/Town/Village/Area']].groupby(["City/Town/Village/Area"]).median().sort\_values(by='NO2',ascending=False).tail(10).plot.bar(color='g')

## plt.show()

## data[['RSPM/PM10','City/Town/Village/Area']].groupby(["City/Town/Village/Area"]).median().sort\_values(by='RSPM/PM10',ascending=False).head(10).plot.bar(color='b')

## plt.show()

## data[['RSPM/PM10','City/Town/Village/Area']].groupby(["City/Town/Village/Area"]).median().sort\_values(by='RSPM/PM10',ascending=False).tail(10).plot.bar(color='b')

## plt.show()

## sns.jointplot(x='SO2', y='NO2', data=data,kind='hex',color='k',xlim={0,100}, ylim={0,100})

## plt.show()

## splitted = df['Sampling Date'].str.split('-', expand=True)

## df['day'] = splitted[0].astype('int')

## df['month'] = splitted[1].astype('int')

## df['year'] = splitted[2].astype('int')

## print(data.year)

## data = data[(data['year']>0)]

## #Heatmap Pivot with State as Row, Year as Col, No2 as Value

## f, ax = plt.subplots(figsize=(15,15))

## ax.set\_title('{} by City/Town/Village/Area and year'.format('SO2'))

## sns.heatmap(data.pivot\_table('SO2', index='City/Town/Village/Area',

## columns=['year'],aggfunc='median',margins=True),

## annot=True,cmap="BuPu", linewidths=.5, ax=ax,cbar\_kws={'label': 'Annual Average'})

## plt.show()

## #Heatmap Pivot with State as Row, Year as Col, So2 as Value

## f, ax = plt.subplots(figsize=(15,15))

## ax.set\_title('{} by City/Town/Village/Area and year'.format('no2'))

## sns.heatmap(data.pivot\_table('NO2', index='City/Town/Village/Area',

## columns=['year'],aggfunc='median',margins=True),

## annot=True,cmap="BuPu", linewidths=.5

## ax=ax,cbar\_kws={'label': 'Annual Average'})

## plt.show()

**OUTPUT**

Stn Code Sampling Date State ... NO2 RSPM/PM10 PM 2.5

0 38 01-02-14 Tamil Nadu ... 17.0 55.0 NaN

1 38 01-07-14 Tamil Nadu ... 17.0 45.0 NaN

2 38 21-01-14 Tamil Nadu ... 18.0 50.0 NaN

3 38 23-01-14 Tamil Nadu ... 16.0 46.0 NaN

4 38 28-01-14 Tamil Nadu ... 14.0 42.0 NaN

5 38 30-01-14 Tamil Nadu ... 18.0 43.0 NaN

6 38 02-04-14 Tamil Nadu ... 17.0 51.0 NaN

7 38 02-06-14 Tamil Nadu ... 16.0 46.0 NaN

8 38 02-11-14 Tamil Nadu ... 19.0 50.0 NaN

9 38 13-02-14 Tamil Nadu ... 14.0 48.0 NaN

[10 rows x 11 columns]

Index(['Stn Code', 'Sampling Date', 'State', 'City/Town/Village/Area',

'Location of Monitoring Station', 'Agency', 'Type of Location', 'SO2',

'NO2', 'RSPM/PM10', 'PM 2.5'],

dtype='object')

8

Stn Code 0

Sampling Date 0

State 0

City/Town/Village/Area 0

Location of Monitoring Station 0

Agency 0

Type of Location 0

SO2 11

NO2 13

RSPM/PM10 4

PM 2.5 2879

dtype: int64

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

PM 2.5 float64

dtype: object

<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 Sampling 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

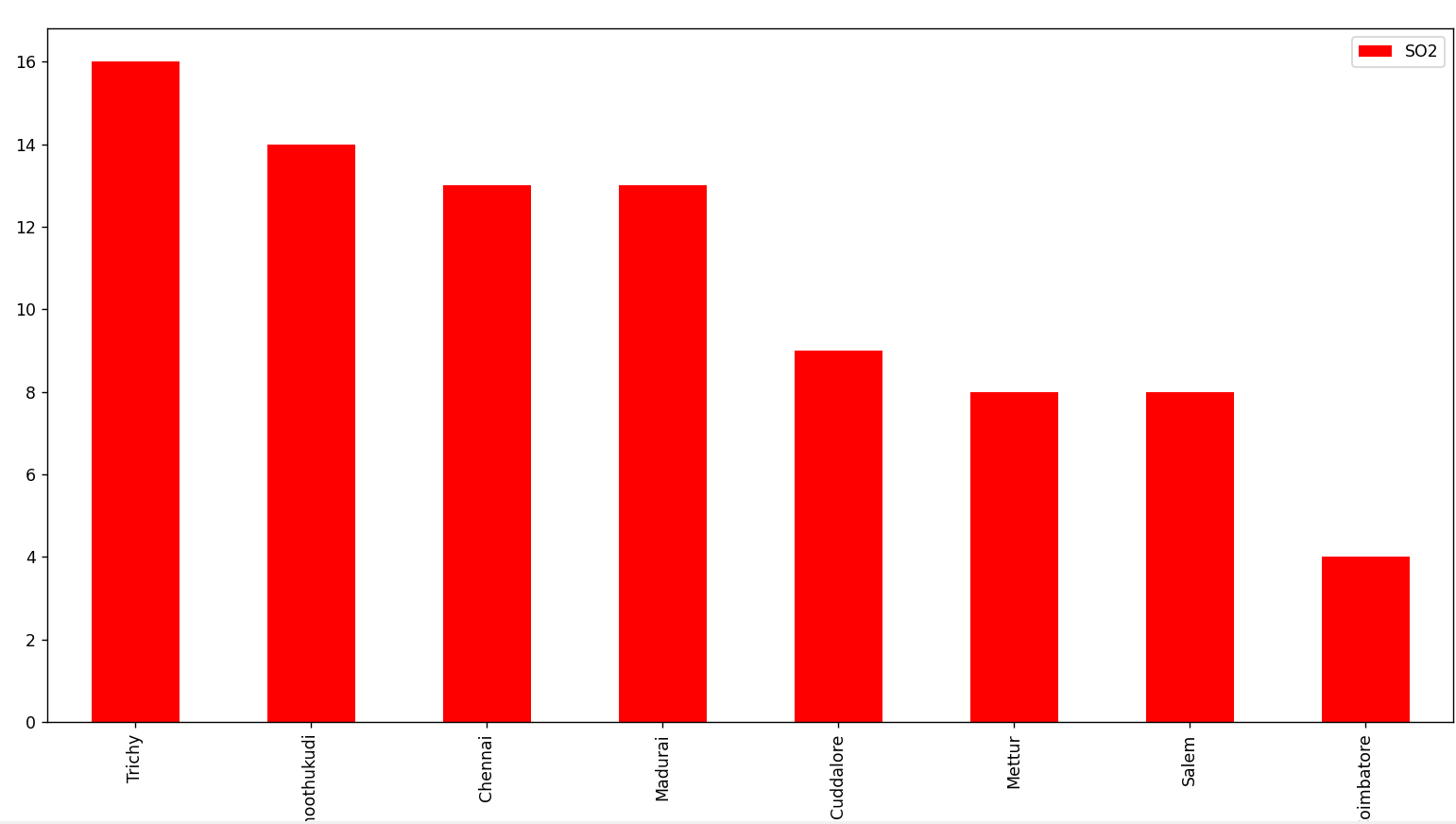
9 RSPM/PM10 2875 non-null float64

10 PM 2.5 0 non-null float64

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

memory usage: 247.5+ KB

None



**CONCLUSION**

In conclusion, the average concentrations of SO2, NO2, and RSPM/PM10 provide critical insights into the overall air quality conditions in a specific area.

Averaging these values helps in assessing the general air quality trend over a specific time period, enabling authorities to make informed decisions regarding pollution control measures and public health interventions

**THANK YOU**