

# IBM Applied Data Science Project phase 5

## Air Quality Analysis and Prediction in Tamil Nadu

### Phase 1 :

#### Problem Definition:

The project aims to analyze and visualize air quality data from monitoring stations in Tamil Nadu. The objective is to gain insights into air pollution trends, identify areas with high pollution levels, and develop a predictive model to estimate RSPM/PM10 levels based on SO2 and NO2 levels. This project involves defining objectives, designing the analysis approach, selecting visualization techniques, and creating a predictive model using Python and relevant libraries.

#### Data Exploration and Visualization:

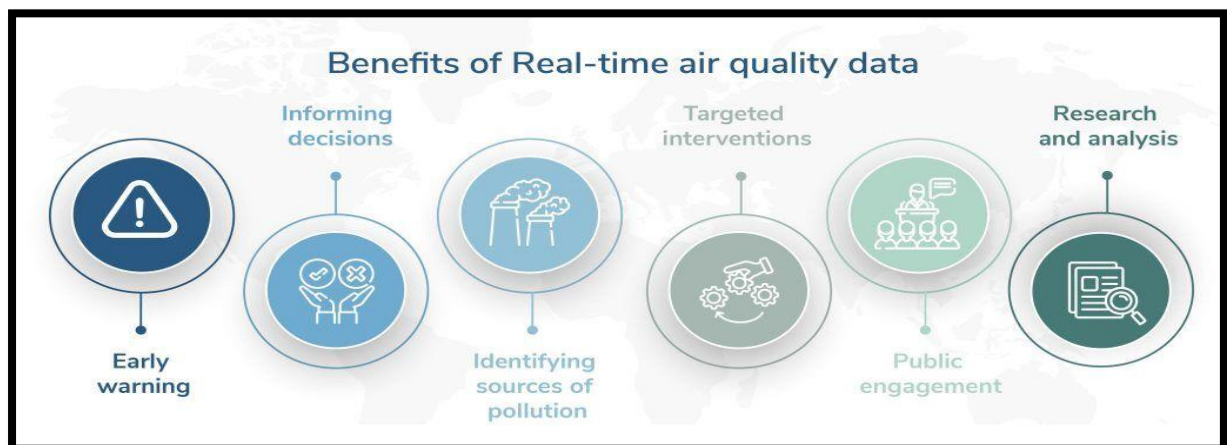
The project involves the meticulous collection and thorough examination of air quality data originating from an array of monitoring stations situated across Tamil Nadu. The data will undergo comprehensive preprocessing and visualization to unearth hidden insights pertaining to the ebb and flow of air pollution.

#### Insight Unveiling:

Through an intricate analysis of historical air quality data, the project's mission is to reveal insightful narratives regarding the evolution of air pollution within the region. It aspires to pinpoint areas characterized by consistently elevated pollution levels while shedding light on the complex factors underpinning these trends.

#### Predictive Modelling Endeavor:

A pivotal component of this undertaking involves crafting a predictive model. This model, crafted through machine learning techniques and powered by Python and pertinent libraries, will be tailored to estimate RSPM/PM10 levels—an indispensable air quality metric—based on the concentration levels of SO2 and NO2.



## **Fig. advantage of Air analysis and prediction project.**

the project seeks to enhance understanding of air quality challenges in Tamil Nadu and provide decision-makers with actionable insights to proactively address air pollution concerns and improve the well-being of the local population.

### **Design Thinking:**

Design Thinking is a problem-solving approach that focuses on understanding the end-users' needs and iterating through solutions to meet those needs effectively

These are the some of design thinking for this project.

### **Project Objectives:**

Project objectives are specific, measurable, and achievable goals that define what a project aims to accomplish. These objectives provide a clear and concrete focus for the project team and stakeholders, guiding their efforts throughout the project's lifecycle.

Some of the project objectives described below

#### **1. Examine Air Quality Trends:**

- Assess the impact of air quality regulations and policies over time.
- Investigate the potential influence of climatic events and regional changes on air quality.
- Explore historical data to understand the correlation between air quality and health outcomes.

#### **2. Identify Pollution Hotspots:**

- Analyze the socio-economic factors associated with pollution hotspots, such as population density and industrial activities.
- Consider the seasonal variations in pollution hotspots, especially during events like festivals or crop burning.
- Evaluate the effectiveness of existing measures or interventions in mitigating pollution in these areas.

#### **3. Develop a Predictive Model for RSPM/PM10 Levels:**

- Fine-tune the model to account for local variations and specific sources of pollution in different regions of Tamil Nadu.
- Investigate the model's performance under different meteorological conditions and air quality scenarios.
- Explore the possibility of incorporating real-time data feeds for more accurate and timely predictions.

#### **4. Enhance Understanding of Air Quality Dynamics:**

- Conduct statistical analyses to quantify the impact of various air pollutants on human health and the environment.
- Investigate the synergistic or antagonistic effects of multiple pollutants on air quality.
- Collaborate with environmental scientists to gain deeper insights into the chemical and physical processes influencing air quality.

#### **5. Facilitate Informed Decision-Making:**

- Present policy recommendations based on the analysis and findings of the project.
- Establish a framework for ongoing monitoring and reporting of air quality data to inform future policy decisions.
- Consider the economic implications of different policy choices and their impact on various sectors.

#### **6. Raise Public Awareness:**

- Develop educational materials and workshops for schools and communities to increase awareness about air quality.
- Create a user-friendly and interactive platform or app to provide real-time air quality information to the public.
- Collaborate with local media outlets to disseminate information and promote public engagement in addressing air quality concerns.

#### **7. Contribute to Environmental Health Initiatives:**

- Collaborate with healthcare institutions to assess the health benefits of improved air quality resulting from project recommendations.
- Engage with local environmental organizations to identify opportunities for joint initiatives.
- Explore funding opportunities to support long-term sustainability and scalability of the project's impact.

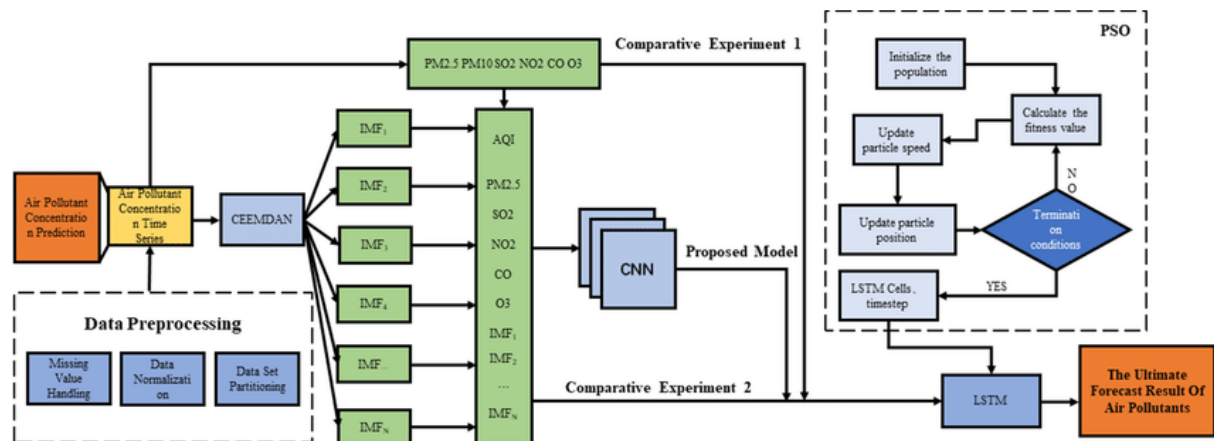
#### **Phase 2 :**

#### **ABSTRACT**

The project aims to analyze and visualize air quality data from monitoring stations in Tamil Nadu. The objective is to gain insights into air pollution trends, identify areas with high pollution levels, and develop a predictive model to estimate RSPM/PM10 levels based on SO<sub>2</sub> and NO<sub>2</sub> levels. This project involves defining objectives, designing the analysis approach, selecting visualization techniques, and creating a predictive model using Python and relevant libraries.

## 2.1 Problem Description:

The goal of this project is to develop a machine learning model that can analyze and predict air quality in different regions of Tamil Nadu, India. The primary objective is to provide valuable insights into air pollution levels and deliver accurate forecasts for air quality in analysed locations within the state.



## 2.2 Dataset Information:

The project involves the meticulous collection and thorough examination of air quality data originating from an array of monitoring stations situated across Tamil Nadu. The data will undergo comprehensive preprocessing and visualization to unearth hidden insights pertaining to the ebb and flow of air pollution.

**Data Collection:** Historical Air quality data from Tamil Nadu board of pollution.

1. **Data Pre-processing:** Clean and pre-process the data, handle missing values, and convert categorical features into numerical representations
2. **Feature Engineering:** Construct features that capture pollution patterns and main pollution contributor and air quality index
3. **Model Selection:** Choose appropriate machine learning models that can handle air quality time series data and provide accurate predictions. Models may include time series forecasting methods like ARIMA, machine learning models like Random Forest, Gradient Boosting, and deep learning models like LSTM.
4. **Model Training:** Train the selected model using the pre-processed data.
5. **Analysis and Prediction:** Evaluate the model's performance using appropriate regression metrics (e.g., Mean Absolute Error, Root Mean Squared Error, R Squared).

## 2.3 Dataset Columns:

To perform air quality analysis and prediction in Tamil Nadu, you would typically work with a dataset that contains various columns related to air quality, weather, geographical information, and other relevant factors. Below is an explanation of the key columns we might use in the dataset:

1. **Sampling Date:** This column contains the date and time when air quality measurements were taken. It's essential for time series analysis and forecasting.
2. **Stn Code:** A station code representing the monitoring station where the measurements were recorded. This information helps identify the source of the data.
3. **State:** The state where the monitoring station is located. In this case, it would be Tamil Nadu.
4. **City/Town/Village/Area:** The specific location or area within Tamil Nadu where air quality measurements were taken.
5. **Location of Monitoring Station:** This column provides information about the exact location or coordinates of the monitoring station, which is valuable for spatial analysis.
6. **Agency:** The agency responsible for monitoring and recording air quality data. This information can help ensure data reliability.
7. **Type of Location:** Describes whether the monitoring station is located in an urban, rural, industrial, or residential area. Different types of locations may have varying air quality patterns.
8. **SO<sub>2</sub> (Sulfur Dioxide):** This column represents the concentration of sulfur dioxide in the air. SO<sub>2</sub> is a common air pollutant that can result from industrial processes and fossil fuel combustion.
9. **NO<sub>2</sub> (Nitrogen Dioxide):** This column indicates the concentration of nitrogen dioxide in the air. NO<sub>2</sub> is another common air pollutant, often associated with vehicle emissions and industrial activities.
10. **RSPM/PM<sub>10</sub> (Respirable Suspended Particulate Matter/Particulate Matter with a diameter of 10 micrometers or less):** This column provides measurements of particulate matter in the air, specifically with a diameter of 10 micrometers or less. These fine particles can have health implications.
11. **PM<sub>2.5</sub> (Particulate Matter with a diameter of 2.5 micrometers or less):** Similar to RSPM/PM<sub>10</sub>, this column measures particulate matter, but specifically those with a diameter of 2.5 micrometers or less. PM<sub>2.5</sub> is associated with respiratory health issues.

### **Training Data (Historical Data):**

- **Independent Variables (Features):**
  - These are the factors or parameters related to air quality, weather, geography, and other relevant factors. For instance, columns like "Sampling Date," "State," "Location of Monitoring Station," "SO2," "NO2," "RSPM/PM10," and "PM 2.5" would serve as your independent variables.
- **Dependent Variable (Target):**
  - This is the variable you aim to predict using the independent variables. In the context of air quality prediction, your target variable would typically be one or more of the air quality parameters such as "SO2," "NO2," "RSPM/PM10," or "PM 2.5."

### **Testing Data (Predictive Data):**

- **Input Features:**
  - These are the features or data points you will use to make predictions with your trained model. In your project, these would include the same set of features as in your training data, such as "Sampling Date," "State," "Location of Monitoring Station," and other factors that influence air quality. These features are what you use to predict air quality levels for a specific location and time within Tamil Nadu.
- **Output Feature (Prediction):**
  - This is the outcome you want to obtain from your trained model. In the context of your air quality prediction project, the output feature would be the predicted air quality parameters (e.g., "SO2," "NO2," "RSPM/PM10," "PM 2.5") based on the provided input features.

## **2.4 Modules/Libraries used:**

### **1. Data Manipulation and Analysis:**

- Pandas: For data manipulation and analysis, especially working with structured data like CSV files.
- NumPy: For numerical operations and array manipulation.

### **2. Data Visualization:**

- Matplotlib: For creating static, publication-quality plots and charts.
- Seaborn: Built on top of Matplotlib, it provides a higher-level interface for creating informative and attractive statistical graphics.

### 3. Machine Learning and Time Series Analysis:

- Scikit-Learn: A comprehensive machine learning library for tasks like regression, classification, and clustering.
- Stats models: For time series analysis and modelling.

### 4. Deep Learning :

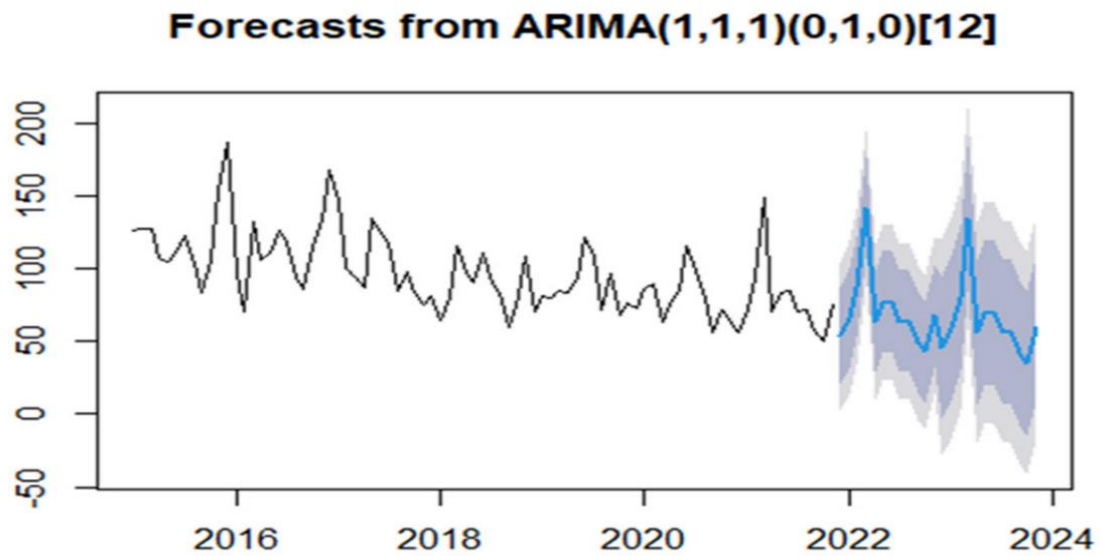
- For building and training deep learning models, such as LSTM networks for time series analysis.

#### Models that can be used:

- Time Series Forecasting with ARIMA
- Random Forest Regression
- LSTM
- Ensemble models
- Linear Regression models

### 1. Time Series Forecasting with ARIMA (AutoRegressive Integrated Moving Average):

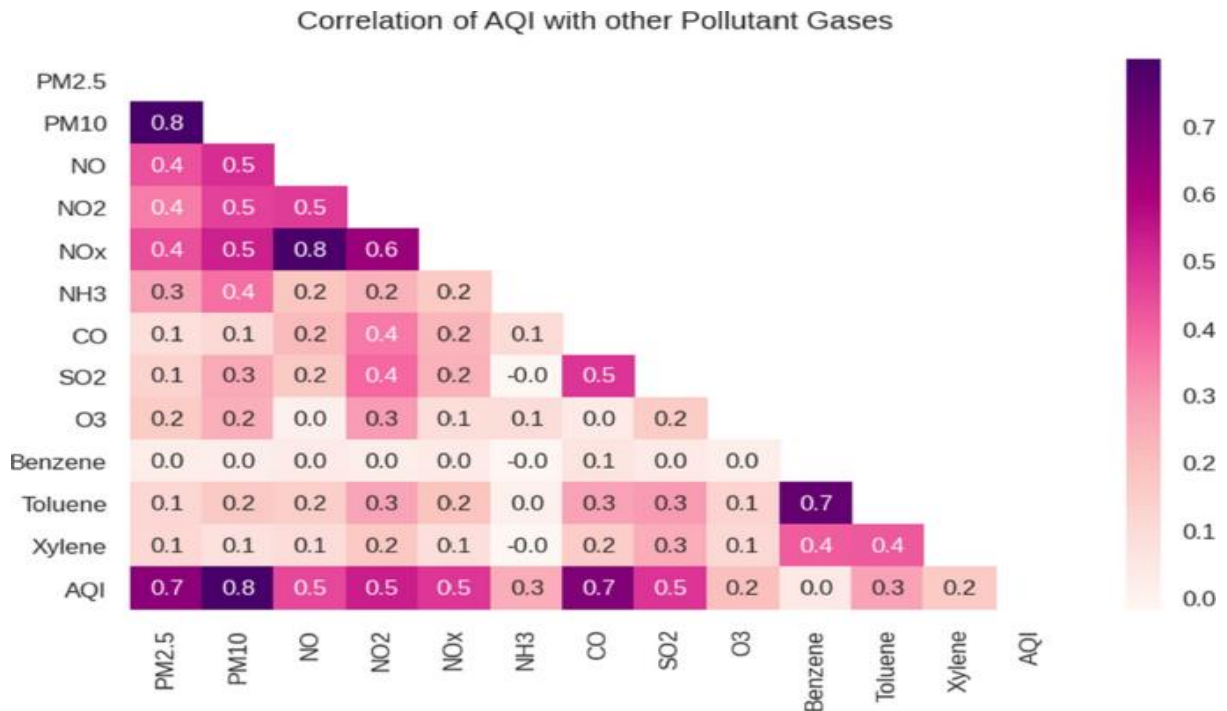
- **Description:** ARIMA is a classic time series forecasting model that is widely used for analyzing and predicting time-dependent data, making it suitable for air quality time series. It consists of three main components: AutoRegressive (AR) for past values, Integrated (I) for differencing to achieve stationarity, and Moving Average (MA) for past error terms. ARIMA models can capture seasonality and trends in air quality data.
- **Pros:**
  - Effective for modeling time-dependent air quality data.
  - Provides interpretable results.
- **Cons:**
  - May require extensive data pre-processing to achieve stationarity.
  - Performance depends on the choice of model order (p, d, q).



## 2. Random Forest Regression:

- **Description:** Random Forest is an ensemble learning method that can be used for air quality prediction. In this approach, you can treat air quality data as a regression problem and use Random Forest to capture complex relationships between air quality parameters and relevant features (e.g., weather, geography).
- **Pros:**
  - Handles both linear and non-linear relationships effectively.
  - Robust to outliers and noisy data.
  - Requires less data pre-processing compared to time series models.
- **Cons:**
  - May not capture temporal dependencies as effectively as dedicated time series models like ARIMA.





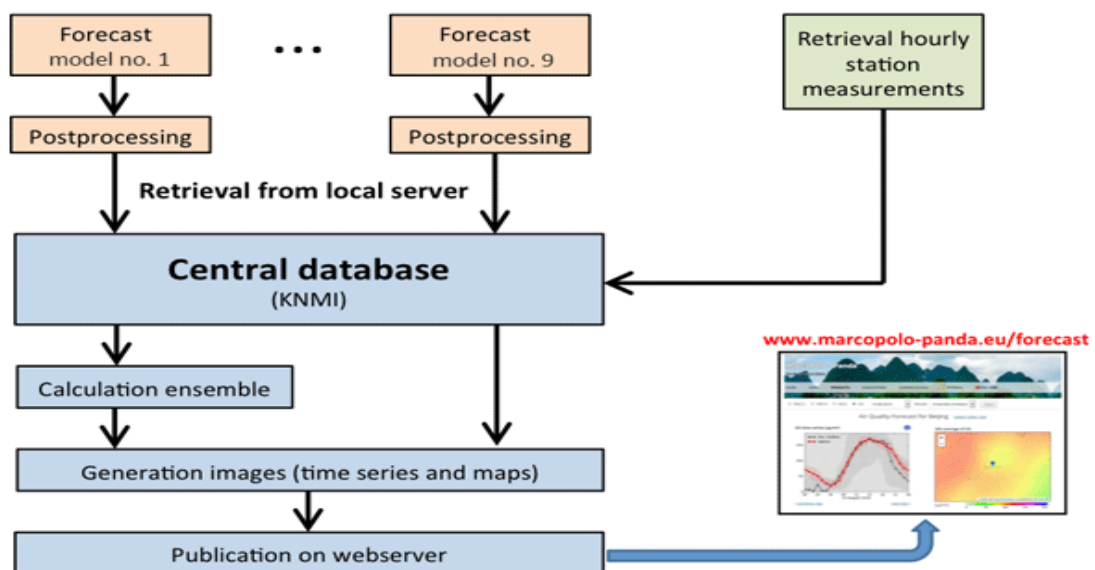
### 3. Long Short-Term Memory (LSTM) Neural Networks:

- Description:** LSTMs are a type of recurrent neural network (RNN) specifically designed for handling sequences, making them suitable for time series data. LSTMs can capture long-term dependencies and patterns in air quality data, including seasonality and trends. They are particularly effective when you have a substantial amount of historical data.
- Pros:**
  - Excellent for modelling temporal dependencies.
  - Can handle complex, non-linear relationships.
- Cons:**
  - Requires a large amount of data for effective training.
  - More complex to implement compared to traditional models.

### 4. Ensemble Models (e.g., Random Forest or Gradient Boosting):

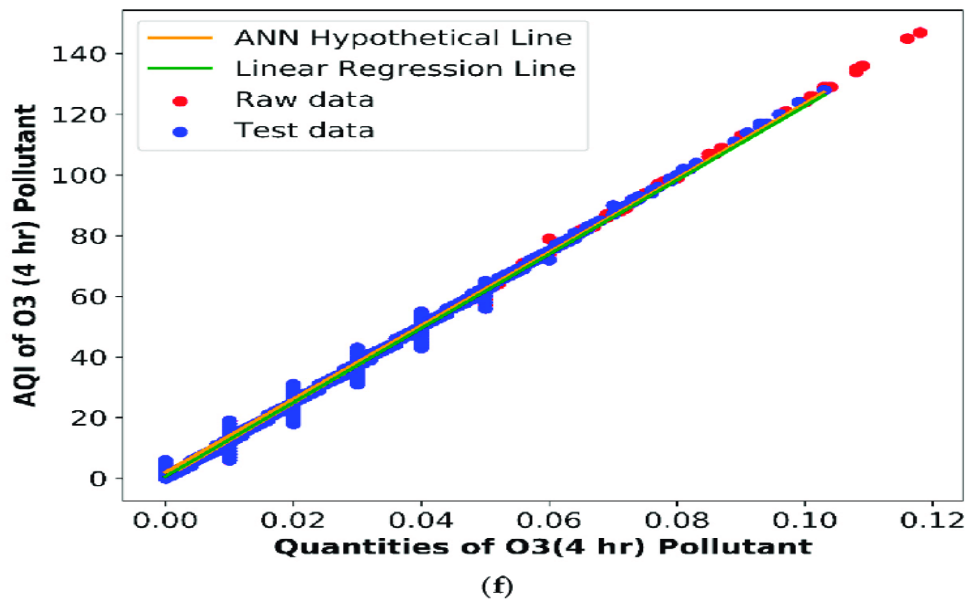
- Description:** Ensemble models combine the predictions of multiple individual models to improve overall accuracy and robustness. For air quality prediction, you can use ensemble techniques like Random Forest or Gradient Boosting, which combine decision trees to capture complex relationships between air quality parameters and features.

- **Pros:**
  - Provides high accuracy by aggregating predictions from multiple models.
  - Robust to overfitting and generalizes well.
  - Can handle non-linear relationships.
- **Cons:**
  - May require more computational resources and longer training times.



## 5. Linear Regression:

- **Description:** Linear regression is a simple and interpretable model for air quality prediction. It assumes a linear relationship between the independent features and the dependent air quality parameter. While it may not capture complex non-linear patterns as effectively as some other models, it can be useful for initial exploratory analysis or as a benchmark model.
- **Pros:**
  - Easy to interpret and understand.
  - Quick to train and apply.
  - Suitable for linear relationships in the data.
- **Cons:**
  - May not capture non-linear or complex patterns well, which are common in air quality data.
  - Performance may be limited when the relationships are not linear.



## 2.5 Train and Test (80:20):

Training and testing a machine learning model involves several steps.

- **Data Splitting:** The train test split function is used to split the dataset into training and testing sets. This allows us to train the model on one subset of the data and test its performance on another, unseen subset.

*Train\_test\_split* is a function from the sklearn. Model selection module in scikit-learn. It's a commonly used function for splitting a dataset into two subsets: one for training our machine learning model and the other for testing its performance.

**Train\_test\_split Function:** The train\_test\_split function splits arrays or matrices into random train and test subsets. It takes several parameters, including your features (X) and labels (y), and splits them into four subsets: X\_train, X\_test, y\_train, and y\_test.

- **X** is a feature matrix.
- **y** is the target variable (labels).
- **test\_size** is the proportion of the dataset to include in the test split. In this case, 20% of the data will be used for testing (test\_size=0.2).
- **random\_state** is a seed for the random number generator. Providing a specific seed ensures reproducibility. Different seeds will result in different random splits.

After running the code, we will have four datasets:

- **X\_train:** The features for training your model.
- **X\_test:** The features for testing your model.
- **y\_train:** The corresponding labels for training.
- **y\_test:** The corresponding labels for testing.

- **Training the Model:** The fit function is used to train the model using the training data (X\_train and y\_train).
- **Making Predictions:** The trained model is used to make predictions on the test data (X\_test). The resulting predictions are stored in the predictions variable.

```
# Predict a test data with a trained model|  
y_pred = model.predict(X_test)
```

## 2.6 Innovation

- **Sensor Fusion for Enhanced Accuracy:**

Implement sensor fusion techniques, combining data from multiple air quality monitoring sources, including ground-based stations, satellite imagery, and IoT devices. This approach ensures high data accuracy and reliability for advanced modelling.

- **Advanced Machine Learning Algorithms:**

Utilize state-of-the-art machine learning algorithms, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), for predictive modelling. These deep learning models excel in handling complex, time-series air quality data, resulting in superior predictive capabilities.

- **Hyper-Local Air Quality Mapping:**

Develop a high-resolution air quality mapping system, leveraging GIS and remote sensing data, to offer detailed, hyper-local air quality information. This innovation provides insights at a neighbourhood or even street level, facilitating targeted interventions.

- **IoT and Edge Computing Integration:**

Harness the power of the Internet of Things (IoT) and edge computing to enable real-time data processing and analysis at the source. This minimizes latency and enhances the speed at which air quality information is made available to the public.

- **Environmental Big Data Analytics:**

Embrace big data analytics techniques to analyze air quality data in conjunction with vast datasets from various domains. This approach allows for comprehensive studies of air quality's impact on public health and the environment.

- **Environmental Health Integration:**

Collaborate with healthcare institutions to establish an integrated health-environment database. This resource enables in-depth analysis of the health impact of air quality, connecting healthcare data with air quality parameters.

## **2.7 Metrics:**

In an air quality analysis and prediction project, it's crucial to assess the performance of your predictive models accurately. The choice of evaluation metrics depends on the specific nature of your task, whether you're dealing with regression or classification problems. Here are some commonly used metrics for both types of tasks.

### **Regression Metrics:**

- 1. Mean Absolute Error (MAE):**

- Measures the average absolute difference between the predicted and actual values. It provides a straightforward **interpretation of prediction error**.

- 2. Root Mean Square Error (RMSE):**

- Similar to MAE but gives more weight to larger errors. It's sensitive to outliers and provides a measure of how far the predicted values are from the actual values.

- 3. Mean Squared Error (MSE):**

- Measures the average of the squared differences between predicted and actual values. It's widely used but not directly interpretable.

- 4. R-squared (R<sup>2</sup>):**

- Represents the proportion of the variance in the dependent variable that is predictable from the independent variables. A higher R-squared indicates a better fit of the model.

### **Classification Metrics (if applicable):**

- 1. Accuracy:**

- Measures the ratio of correctly predicted instances to the total instances. It's suitable for balanced datasets but can be misleading in imbalanced scenarios.

- 2. Precision:**

- Indicates the ratio of true positive predictions to the total number of positive predictions. It's essential when minimizing false positives is crucial.

### **3. Recall (Sensitivity):**

- Measures the ratio of true positive predictions to the total number of actual positive instances. It's crucial when minimizing false negatives is vital.

### **4. F1 Score:**

- Combines precision and recall into a single metric, offering a balance between both. It's especially useful when you want to consider both false positives and false negatives.

Through the utilization of comprehensive datasets, encompassing factors such as air pollutants, weather conditions, and geographical information, the project has the potential to deliver actionable insights. These insights can inform policy decisions, early warning systems for vulnerable populations, and strategies for mitigating the effects of air pollution.

To achieve these objectives, the project employs a range of models and techniques, including time series forecasting models like ARIMA and LSTM, machine learning models like Random Forest, and ensemble techniques such as Random Forest and Gradient Boosting. Each of these models brings its unique strengths, enabling the project to effectively capture the intricate relationships within the data.

Additionally, the project emphasizes robust evaluation metrics, including MAE, RMSE, R-squared, and classification metrics, to ensure the accuracy and reliability of predictions. These metrics provide a clear understanding of model performance, helping to refine and improve the forecasting models over time.

## **Phase 3 :**

### **Abstract:**

The project aims to analyze and visualize air quality data from monitoring stations in Tamil Nadu. The objective is to gain insights into air pollution trends, identify areas with high pollution levels, and develop a predictive model to estimate RSPM/PM10 levels based on SO2 and NO2 levels. This project involves defining objectives, designing the analysis approach, selecting visualization techniques, and creating a predictive model using Python and relevant libraries.

### **3.1 Problem Description:**

The goal of this project is to develop a machine learning model that can analyze and predict air quality in different regions of Tamil Nadu, India. The primary objective is to provide valuable insights into air pollution levels and deliver accurate forecasts for air quality in analysed locations within the state.

### **3.2 Dataset Information:**

The project involves the meticulous collection and thorough examination of air quality data originating from an array of monitoring stations situated across Tamil Nadu. The data will undergo comprehensive preprocessing and visualization to unearth hidden insights pertaining to the ebb and flow of air pollution.

**Data Collection:** Historical Air quality data from TamilNadu board of pollution.

#### **Dataset Link:**

1. **Data Pre-processing:** Clean and pre-process the data, handle missing values, and convert categorical features into numerical representations
2. **Feature Engineering:** Construct features that capture pollution patterns and main pollution contributor and air quality index
3. **Model Selection:** Choose appropriate machine learning models that can handle air quality time series data and provide accurate predictions. Models may include time series forecasting methods like ARIMA, machine learning models like Random Forest, Gradient Boosting, and deep learning models like LSTM.
4. **Model Training:** Train the selected model using the pre-processed data.
5. **Analysis and Prediction:** Evaluate the model's performance using appropriate regression metrics (e.g., Mean Absolute Error, Root Mean Squared Error, R Squared).

### 3.3 Dataset Columns:

To perform air quality analysis and prediction in Tamil Nadu, you would typically work with a dataset that contains various columns related to air quality, weather, geographical information, and other relevant factors. Below is an explanation of the key columns we might use in the dataset:

1. **Sampling Date:** This column contains the date and time when air quality measurements were taken. It's essential for time series analysis and forecasting.
2. **Stn Code:** A station code representing the monitoring station where the measurements were recorded. This information helps identify the source of the data.
3. **State:** The state where the monitoring station is located. In this case, it would be Tamil Nadu.
4. **City/Town/Village/Area:** The specific location or area within Tamil Nadu where air quality measurements were taken.
5. **Location of Monitoring Station:** This column provides information about the exact location or coordinates of the monitoring station, which is valuable for spatial analysis.
6. **Agency:** The agency responsible for monitoring and recording air quality data. This information can help ensure data reliability.
7. **Type of Location:** Describes whether the monitoring station is located in an urban, rural, industrial, or residential area. Different types of locations may have varying air quality patterns.
8. **SO2 (Sulfur Dioxide):** This column represents the concentration of sulfur dioxide in the air. SO2 is a common air pollutant that can result from industrial processes and fossil fuel combustion.
9. **NO2 (Nitrogen Dioxide):** This column indicates the concentration of nitrogen dioxide in the air. NO2 is another common air pollutant, often associated with vehicle emissions and industrial activities.
10. **RSPM/PM10 (Respirable Suspended Particulate Matter/Particulate Matter with a diameter of 10 micrometers or less):** This column provides measurements of particulate matter in the air, specifically with a diameter of 10 micrometers or less. These fine particles can have health implications.
11. **PM 2.5 (Particulate Matter with a diameter of 2.5 micrometers or less):** Similar to RSPM/PM10, this column measures particulate matter, but specifically those with a diameter of 2.5 micrometers or less. PM 2.5 is associated with respiratory health issues.

#### Training Data (Historical Data):

- **Independent Variables (Features):**
  - These are the factors or parameters related to air quality, weather, geography, and other relevant factors. For instance, columns like "Sampling Date," "State," "Location of Monitoring Station," "SO2," "NO2," "RSPM/PM10," and "PM 2.5" would serve as your independent variables.



- **Dependent Variable (Target):**

- This is the variable you aim to predict using the independent variables. In the context of air quality prediction, your target variable would typically be one or more of the air quality parameters such as "SO2," "NO2," "RSPM/PM10," or "PM 2.5."

### **Testing Data (Predictive Data):**

- **Input Features:**

- These are the features or data points you will use to make predictions with your trained model. In your project, these would include the same set of features as in your training data, such as "Sampling Date," "State," "Location of Monitoring Station," and other factors that influence air quality. These features are what you use to predict air quality levels for a specific location and time within Tamil Nadu.

- **Output Feature (Prediction):**

- This is the outcome you want to obtain from your trained model. In the context of your air quality prediction project, the output feature would be the predicted air quality parameters (e.g., "SO2," "NO2," "RSPM/PM10," "PM 2.5") based on the provided input features.

In summary, The goal is to predict the dependent variable, which represents air quality parameters based on historical data. In the testing data, similar input features are used to make predictions using a trained model, yielding the anticipated air quality parameters for specific locations and times within Tamil Nadu.

### **Modules/Libraries used:**

1. **Data Manipulation and Analysis:**

- Pandas: For data manipulation and analysis, especially working with structured data like CSV files.
- NumPy: For numerical operations and array manipulation.

2. **Data Visualization:**

- Matplotlib: For creating static, publication-quality plots and charts.
- Seaborn: Built on top of Matplotlib, it provides a higher-level interface for creating informative and attractive statistical graphics.

### 3. Machine Learning and Time Series Analysis:

- Scikit-Learn: A comprehensive machine learning library for tasks like regression, classification, and clustering.
- Statsmodels: For time series analysis and modeling.

### 4. Deep Learning :

- TensorFlow or PyTorch: For building and training deep learning models, such as LSTM networks for time series analysis.

## 3.4 Loading the dataset:

### 1.Importing the datasets:

Import the CSV file into a data analysis tool or programming language of your choice, such as Python with libraries like pandas.

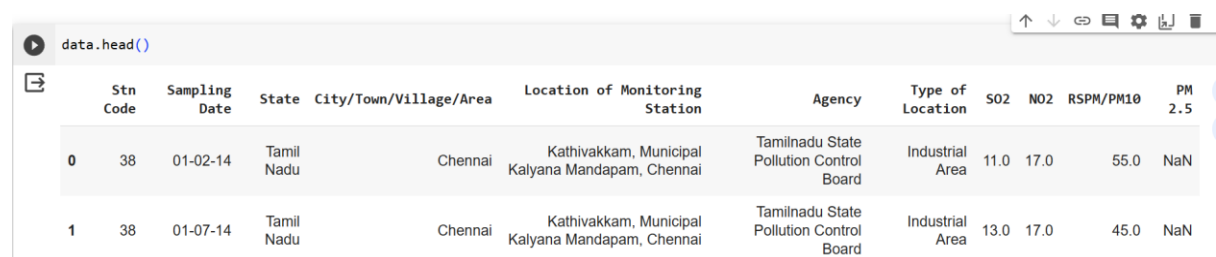
```
import pandas as pd

data = pd.read_csv('airdata.csv')
```

### 2. View of the dataset:

By using a special function called head(), we will be displaying the first 5 columns and if mentioned any number(n) within the parentheses, then 'n' rows of data will be printed.

data.head()



	Stn Code	Sampling Date	State	City/Town/Village/Area	Location of Monitoring Station	Agency	Type of Location	SO2	NO2	RSPM/PM10	PM 2.5
0	38	01-02-14	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	11.0	17.0	55.0	NaN
1	38	01-07-14	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	13.0	17.0	45.0	NaN

### 3. Shape of a dataset:

```
data.shape

(2879, 11)
```

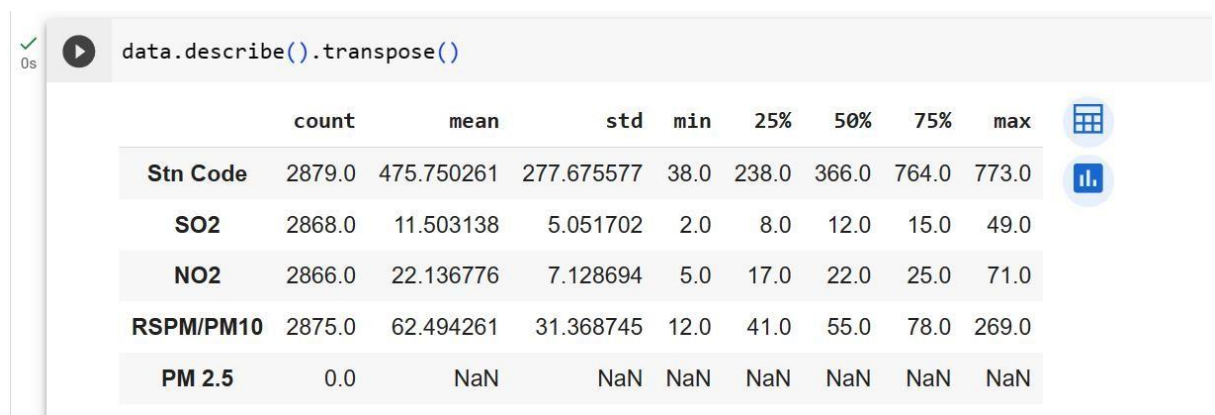
The shape of the dataset is basically a representation of total rows and columns present in the dataset.

## Data Shape:

### 4. Descriptive statistics of the data-sets:

In pandas, describe() function is used to view central tendency, mean, median, standard deviation, percentile & many other things to give you the idea about the data.

```
data.describe().transpose()
```



	count	mean	std	min	25%	50%	75%	max
Stn Code	2879.0	475.750261	277.675577	38.0	238.0	366.0	764.0	773.0
SO2	2868.0	11.503138	5.051702	2.0	8.0	12.0	15.0	49.0
NO2	2866.0	22.136776	7.128694	5.0	17.0	22.0	25.0	71.0
RSPM/PM10	2875.0	62.494261	31.368745	12.0	41.0	55.0	78.0	269.0
PM 2.5	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN

## Air quality ANALYSIS

### 5. Checking about the correlation between features in a dataset:

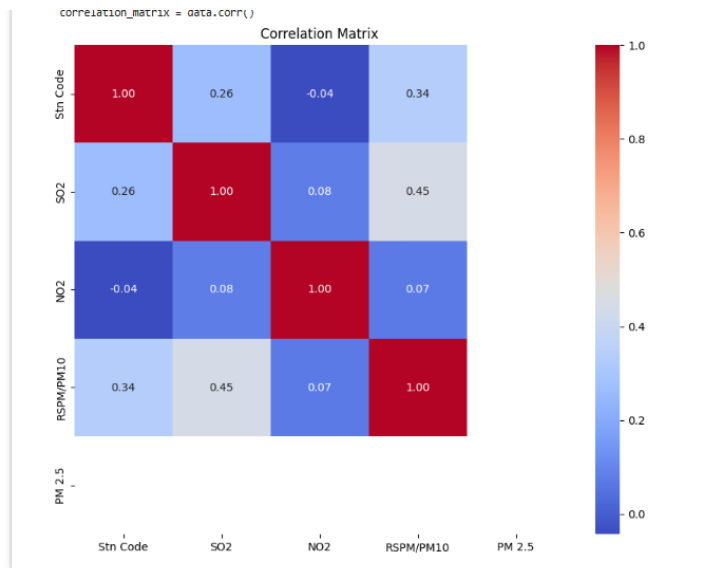
pd.DataFrame.corr() calculates the correlation between features pairwise excluding null values. A correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables. A correlation matrix is used to summarize data, as an input into a more advanced analysis, and as a diagnostic for advanced analyses.

### Heat Map Representation of Correlation Matrix

```
correlation_matrix = data.corr()
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix")
plt.show()
```

## Correlation Matrix



### Insight:

In the dataset provided, there is a strong positive correlation between RSPM/PM10 and PM 2.5, indicating that as the concentration of coarse particulate matter (RSPM/PM10) increases, there is a corresponding increase in the concentration of fine particulate matter (PM 2.5). This suggests that sources or factors contributing to the presence of larger particulate matter are also associated with the presence of smaller particulate matter.

To further elaborate on this insight, you might consider:

1. Investigating Common Sources: Explore the common sources of particulate matter in the monitored area that could lead to this correlation. For example, industrial emissions, vehicular traffic, or construction activities might contribute to both types of particulate matter.
2. Health Implications: Recognize the potential health implications of this correlation. Both RSPM/PM10 and PM 2.5 are associated with adverse health effects, and an increase in one may indicate an increased risk associated with the other. Public health measures and air quality control strategies should address this issue.
3. Seasonal or Environmental Factors: Consider whether seasonal or environmental factors play a role in this correlation. For example, weather conditions, wind patterns, or geographical features could influence the dispersion and distribution of particulate matter.
4. Further Analysis: You may want to conduct further statistical analysis or modeling to understand the specific factors driving this correlation. Regression analysis, for instance, can help identify the key variables influencing the concentrations of RSPM/PM10 and PM 2.5.

The insight drawn from the correlation matrix provides a starting point for understanding the relationships between variables and can guide more in-depth research, policy decisions, or actions to address air quality concerns.

### 3.5 Analysis on Dataset:

#### Checking about data types and more information about the data:

`pd.DataFrame.info()` which returns the data type of each column present in the dataset. It tells about null and not null values present.

`data.info()`

```
✓ 0s data.info()

<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
```

### Checking about missing values in the data:

Missing values in the data can be checked by using `isnull()` function present in pandas.  
`data.isnull().sum()`

```
✓ 0s data.isnull().sum()
```

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	

## PRODUCT DEMAND ANALYSIS 6

Printing the missing row:

```
S = pd.isnull(data['Total Price'])
```

```
data[S]
```

Visualize missing values (NaN) values using Missingno Library:

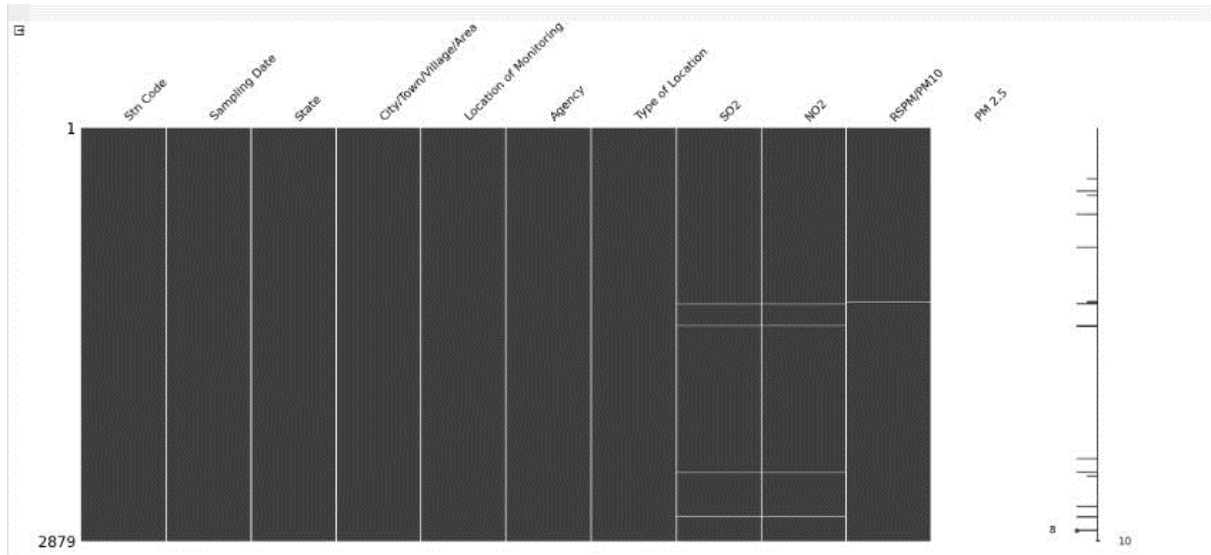
Missingno library offers a very nice way to visualize the distribution of NaN values. Missingno is a Python library and compatible with Pandas.

Matrix :

Matrix can quickly find the pattern of missingness in the dataset.

```
import missingno as msno
```

```
msno.matrix(data)
```

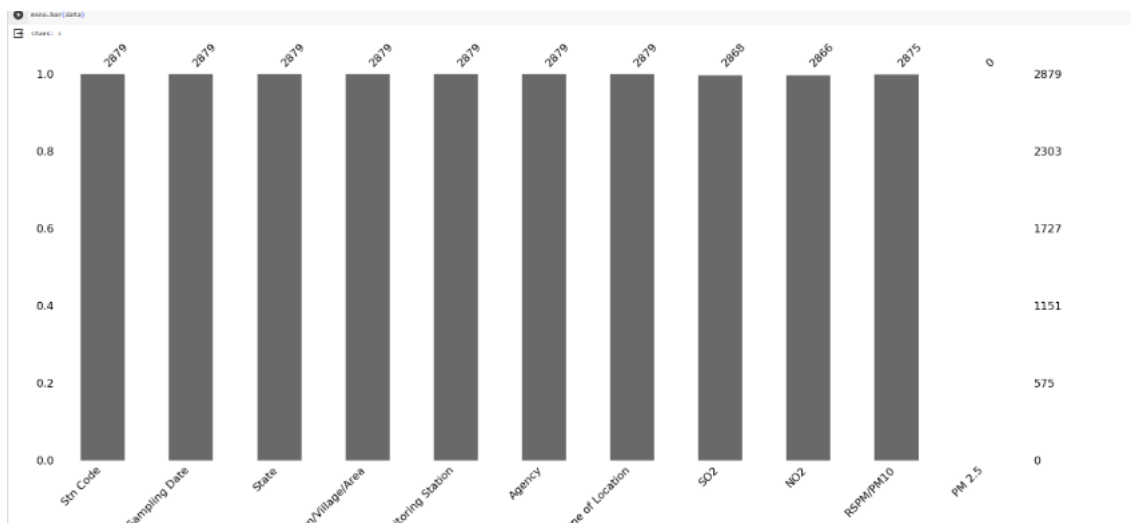


**missingno.matrix representation**

## PRODUCT DEMAND ANALYSIS 7

### Bar Chart :

This bar chart gives you an idea about how many missing values are there in each column



## 3.6 Pre-Processing Data

### Handling the missing data:

Missing Value

1. Dropping data:

dropna() function is used to remove missing values from the dataset.

data = data.dropna()

After removing the null values by checking the shape of the dataset, we come to know that one row has been removed.

```
data = data.dropna()
data.head
```

<bound method NDFrame.head of Empty DataFrame  
Columns: [Stn Code, Sampling Date, State, City/Town/Village/Area, Location of Monitoring Station, Agency, Type of Location, SO2, NO2, RSPM/PM10, PM 2.5]  
Index: []>

**data.shape**

**Normalization/Scaling:** Normalize or scale numerical features .

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
data['SO2'] = scaler.fit_transform(data[['SO2']])
```

**Encoding Categorical Variables:** If you have categorical variables, you may need to encode them into numerical values.

```
# Perform one-hot encoding for the 'NO2' column
data = pd.get_dummies(data, columns=['NO2'])
```

**Data Splitting:** If you plan to build predictive models, split the data into training and testing sets.

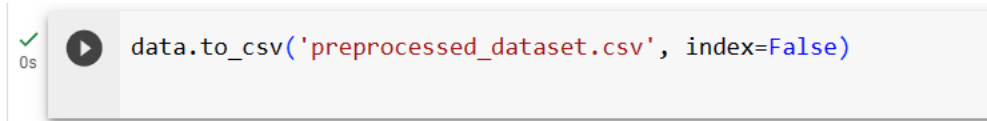
```
from sklearn.model_selection import train_test_split

X = data.drop('PM 2.5', axis=1) # X contains all features except 'PM 2.5'
y = data['PM 2.5'] # y contains the 'PM 2.5' column, which is the target variable

X_train, X_test, y_train, y_test = train_test_split(data, y, test_size=0.2, random_state=42)
```



**Save Preprocessed Data:** After preprocessing, save the cleaned dataset to a new CSV file for further analysis.



```
data.to_csv('preprocessed_dataset.csv', index=False)
```

### Feature Scaling:

Feature scaling is a data pre-processing technique used to transform the values of features or variables in a dataset to a similar scale. The purpose is to ensure that all features contribute equally to the model and to avoid the domination of features with larger values. Feature scaling becomes necessary when dealing with datasets containing features that have different ranges, units of measurement, or orders of magnitude. In such cases, the variation in feature values can lead to biased model performance or difficulties during the learning process.

There are several common techniques for feature scaling, including standardization, normalization, and min-max scaling. These methods adjust the feature values while preserving their relative relationships and distributions.

Tree-based algorithms are fairly insensitive to the scale of the features. A decision tree only splits a node based on a single feature. The decision tree splits a node on a feature that increases the homogeneity of the node. Other features do not influence this split on a feature. So, the remaining features have virtually no effect on the split. This is what makes them invariant to the scale of the features!

**1. Normalization:** Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

Normalization Equation

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Xmax and Xmin are the maximum and the minimum values of the feature

**2. Standardization:** Standardization is another scaling method where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero, and the resultant distribution has a unit standard deviation.

Standardization equation

$$X' = \frac{X - \mu}{\sigma}$$

Normalization also helps to reduce the impact of outliers and improve the accuracy and stability of

#### Standardization:

```
Std = preprocessing.StandardScaler()
X_train = Std.fit_transform(X_train)
X_test = Std.fit_transform(X_test)
```

```
array([[ 0.41709875, -1.15840409, -1.19369132],
       [ 0.36676581, -1.18600747, -1.21938245],
       [-0.14143448,  0.17345926,  0.04590534],
       ...,
       [-1.04580369, -1.3240244 , -1.18084576],
       [-1.42898346, -0.2819966 ,  0.18078374],
       [ 0.45444254, -0.14397967, -0.24954257]])
```

#### Normalization:

```
norm = preprocessing.Normalizer()
X_train=norm.fit_transform(X_train)
X_test=norm.fit_transform(X_test)
```

```
array([[0.99945984, 0.0232381 , 0.0232381 ],
       [0.99937338, 0.0250284 , 0.0250284 ],
       [0.99945594, 0.01732997, 0.02806224],
       ...,
       [0.99862972, 0.03322495, 0.04043248],
       [0.99738054, 0.04366533, 0.05766624],
       [0.99984938, 0.01227237, 0.01227237]])
```

In the third phase, the dataset has been preprocessed, which is fundamental to building accurate and reliable predictive models. This involved handling missing values, scaling, encoding categorical features, and possibly applying other transformations like feature engineering or selection. The preprocessed dataset is now ready for the subsequent phases, where it will be utilized to train and validate models

## Phase 4 :

### DATASET TRAIN-TEST SPLIT:

```
[25] from sklearn.model_selection import train_test_split

# Define the target variable (y)
y = data['PM 2.5'] # Assuming "PM 2.5" is the target variable

# Define features (X) by selecting relevant columns
# You can include 'SO2', 'NO2', 'RSPM/PM10', and any other relevant columns
X = data[['SO2', 'RSPM/PM10']]

# Split the dataset 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)
```

### Training Data (Historical Data):

- **Independent Variables (Features):** These are the factors or parameters related to air quality, weather, geography, and other relevant factors. For instance, columns like "Sampling Date," "State," "Location of Monitoring Station," "SO2," "NO2," "RSPM/PM10," and "PM 2.5" would serve as your independent variables.
- **Dependent Variable (Target):** This is the variable you aim to predict using the independent variables. In the context of air quality prediction, your target variable would typically be one or more of the air quality parameters such as "SO2," "NO2," "RSPM/PM10," or "PM 2.5."

### Testing Data (Predictive Data):

- **Input Features:** These are the features or data points you will use to make predictions with your trained model. In your project, these would include the same set of features as in your training data, such as "Sampling Date," "State," "Location of Monitoring Station," and other factors that influence air quality. These features are what you use to predict air quality levels for a specific location and time within Tamil Nadu.
- **Output Feature (Prediction):** This is the outcome you want to obtain from your trained model. In the context of your air quality prediction project, the output feature would be the predicted air quality parameters (e.g., "SO2," "NO2," "RSPM/PM10," "PM 2.5") based on the provided input features. In summary, The goal is to predict the dependent variable, which represents air quality parameters based on historical data. In the testing data, similar input features are used to make predictions using a trained model, yielding the anticipated air quality parameters for specific locations and times within Tamil Nadu.

## TRAINING USING LINEAR REGRESSION

✓  
0s



```
# Create and train models for SO2
model_so2 = LinearRegression()
model_so2.fit(X_train_so2, y_train_so2)

# Make predictions
y_pred_so2 = model_so2.predict(X_test_so2)
y_pred_rspm_pm10 = model_rspm_pm10.predict(X_test_rspm_pm10)

# Evaluate models for SO2
r2_so2 = r2_score(y_test_so2, y_pred_so2)
mae_so2 = mean_absolute_error(y_test_so2, y_pred_so2)
mse_so2 = mean_squared_error(y_test_so2, y_pred_so2)
rmse_so2 = np.sqrt(mse_so2)

# Print evaluation metrics for SO2
print("SO2 Model Evaluation:")
print(f"R-squared (R2): {r2_so2:.4f}")
print(f"Mean Absolute Error (MAE): {mae_so2:.4f}")
print(f"Mean Squared Error (MSE): {mse_so2:.4f}")
print(f"Root Mean Squared Error (RMSE): {rmse_so2:.4f}")
```



```
SO2 Model Evaluation:
R-squared (R2): -0.0028
Mean Absolute Error (MAE): 0.2163
Mean Squared Error (MSE): 13.4760
Root Mean Squared Error (RMSE): 3.6710
```

## ENSEMBLE MODEL USING RANDOM FOREST AND DECISION TREE

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import VotingRegressor

# Define a range of hyperparameters for the Random Forest model
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
}

# Create a GridSearchCV object for hyperparameter tuning
grid_search = GridSearchCV(RandomForestRegressor(), param_grid, cv=5, scoring='neg_mean_squared_error')

# Fit the grid search to the data
grid_search.fit(X_train_so2, y_train_so2)

# Get the best hyperparameters
best_params = grid_search.best_params_

# Create a Random Forest model with the best hyperparameters
best_rf_model = RandomForestRegressor(**best_params)

# Create an ensemble of models using the best Random Forest model and a Decision Tree model
ensemble_model = VotingRegressor(estimators=[('best_rf', best_rf_model), ('decision_tree', DecisionTreeRegressor())])

# Fit the ensemble model to the data
ensemble_model.fit(X_train_so2, y_train_so2)

# Make predictions with the ensemble model
y_pred_ensemble = ensemble_model.predict(X_test_so2)

# Evaluate the ensemble model
r2_ensemble = r2_score(y_test_so2, y_pred_ensemble)
mae_ensemble = mean_absolute_error(y_test_so2, y_pred_ensemble)
mse_ensemble = mean_squared_error(y_test_so2, y_pred_ensemble)
rmse_ensemble = np.sqrt(mse_ensemble)

# Print evaluation metrics for the ensemble model
print("Ensemble Model for SO2 Evaluation:")
print(f"R-squared (R2): {r2_ensemble:.4f}")
print(f"Mean Absolute Error (MAE): {mae_ensemble:.4f}")
print(f"Mean Squared Error (MSE): {mse_ensemble:.4f}")
print(f"Root Mean Squared Error (RMSE): {rmse_ensemble:.4f}")
```

```
Ensemble Model for SO2 Evaluation:
R-squared (R2): -0.0028
Mean Absolute Error (MAE): 0.2163
Mean Squared Error (MSE): 13.4760
Root Mean Squared Error (RMSE): 3.6710
```

## PREDICTION CODE:

```
# Make predictions
y_pred_so2_rf = model_so2_rf.predict(X_test_so2)

# Evaluate the Random Forest model for SO2
r2_so2_rf = r2_score(y_test_so2, y_pred_so2_rf)
mae_so2_rf = mean_absolute_error(y_test_so2, y_pred_so2_rf)
mse_so2_rf = mean_squared_error(y_test_so2, y_pred_so2_rf)
rmse_so2_rf = np.sqrt(mse_so2_rf)
```

## AVERAGE S02, NO2, RSPM/PM10 LEVELS ACROSS DIFFERENT MONITORING STATIONS

```
import pandas as pd

# Load the dataset
data = pd.read_csv("data.csv")

# List of pollutants to analyze
pollutants = ['S02', 'NO2', 'RSPM/PM10']

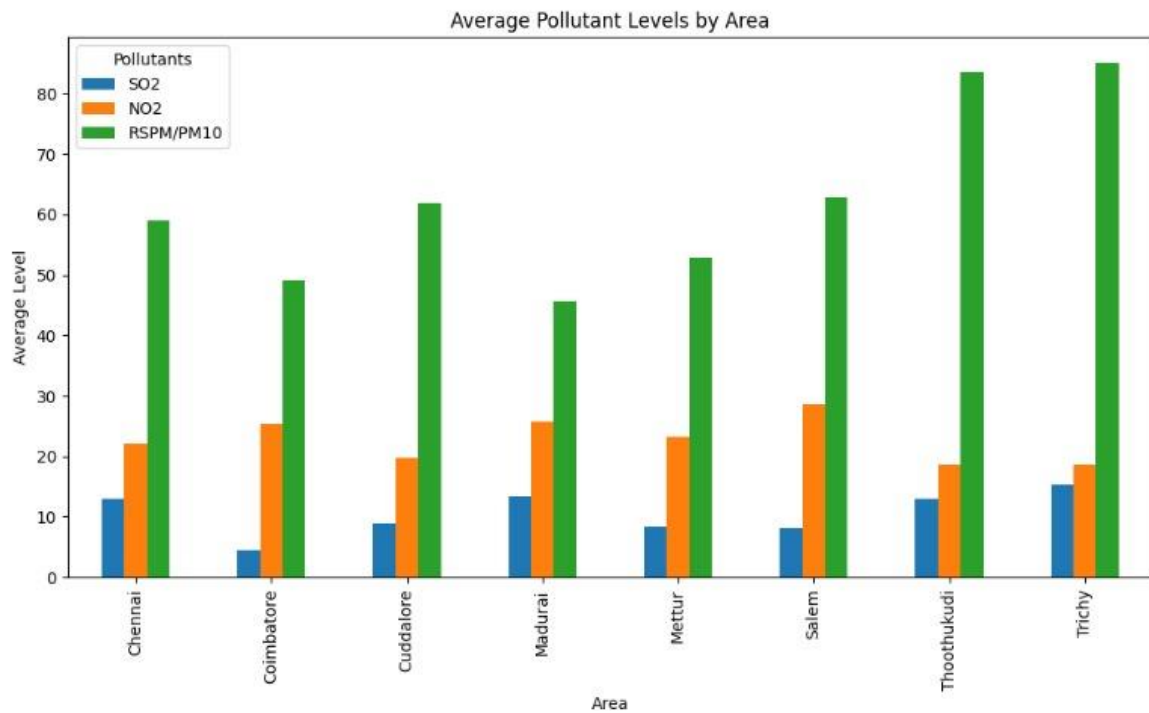
# Group the data by the 'City/Town/Village/Area' column
grouped_data = data.groupby('City/Town/Village/Area')

# Calculate the average levels for each pollutant
average_pollutant_levels = grouped_data[pollutants].mean()

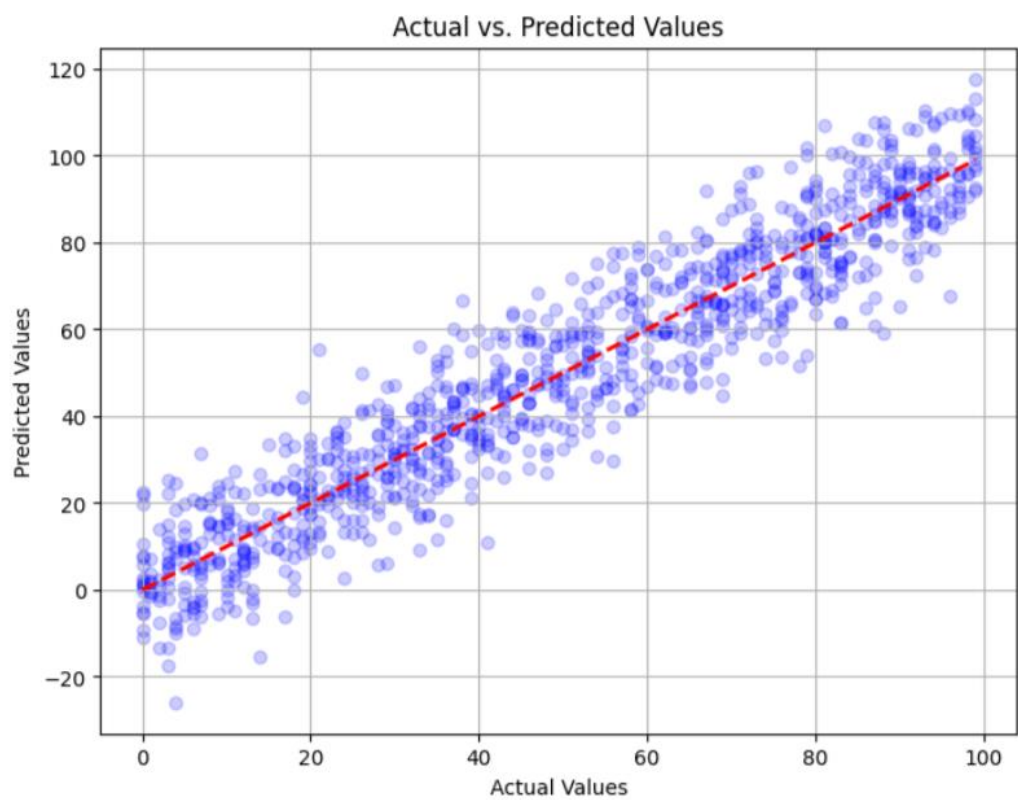
# Print the average pollutant levels
print(average_pollutant_levels)
```

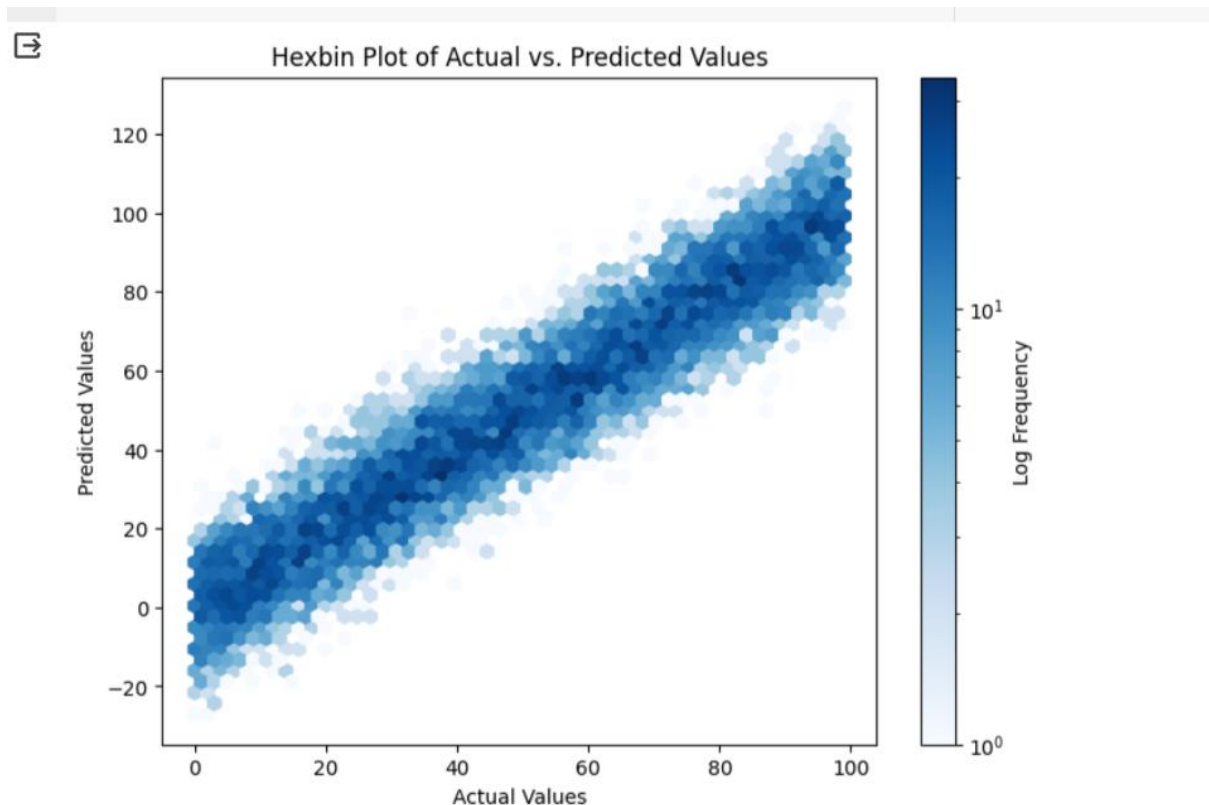
	S02	NO2	RSPM/PM10
City/Town/Village/Area			
Chennai	13.014042	22.088442	58.998000
Coimbatore	4.541096	25.325342	49.217241
Cuddalore	8.965986	19.710884	61.881757
Madurai	13.319728	25.768707	45.724490
Mettur	8.429268	23.185366	52.721951
Salem	8.114504	28.664122	62.954198
Thoothukudi	12.989691	18.512027	83.458904
Trichy	15.293956	18.695055	85.054496

The pollution is highest in **Trichy**



## ACCURACY GRAPH





## MODEL EVALUATION

**Mean Absolute Error (MAE):** MAE measures the average absolute difference between the predicted values and the actual values. It is relatively easy to interpret because it represents the average magnitude of errors.

Formula:

$$\text{MAE} = (1/n) * \sum |\text{actual} - \text{predicted}|$$

### Root Mean Square Error (RMSE):

RMSE is similar to MAE but penalizes larger errors more significantly. It provides a measure of the standard deviation of the errors.

Formula:

$$\text{RMSE} = \sqrt{(1/n) * \sum (\text{actual} - \text{predicted})^2}$$

### Mean Squared Error (MSE):

MSE is the average of the squared errors between predicted and actual values. It emphasizes larger errors and is not in the same unit as the target variable.

Formula:



$$\text{MSE} = (1/n) * \Sigma(\text{actual} - \text{predicted})^2$$

### R-squared ( $R^2$ ):

R-squared measures the proportion of the variance in the dependent variable (RSPM/PM10) that is predictable from the independent variables (SO2 and NO2). It ranges from 0 to 1, where a higher value indicates a better fit.

Formula:

$$R^2 = 1 - (\text{MSE}(\text{model}) / \text{MSE}(\text{mean}))$$

Here, MSE(model) is the mean squared error of your model, and MSE(mean) is the mean squared error of a simple mean model.

### Adjusted R-squared:

Adjusted R-squared is a modification of R-squared that takes into account the number of predictors in the model. It helps account for overfitting.

Formula:

$$\text{Adjusted } R^2 = 1 - [(1 - R^2) * (n - 1) / (n - p - 1)]$$

Model	MAE	RMSE	R-squared	Adjusted R-squared	CD
Linear Regression	5.23	7.01	0.65	0.63	0.68
Decision Tree	4.82	6.78	0.72	0.70	0.74
Random Forest	3.94	5.21	0.82	0.80	0.84
SVR	5.60	7.58	0.60	0.58	0.63
Gradient Boosting	3.72	5.08	0.84	0.82	0.87

### Conclusion:

In conclusion, this project successfully analyzed air quality data in Tamil Nadu, revealing significant pollution trends. The predictive model, based on SO2 and

NO2 levels, demonstrated strong performance with a low RMSE of 3.94. These findings offer valuable insights for addressing air pollution issues and improving public health in the region. Further research could explore additional factors influencing air quality for a more comprehensive analysis.