**Project: Air Quality Assessment of TamilNadu**

### Empathize and Understand the Problem:

Understanding the problem and the context. Why is analyzing air quality important in Tamil Nadu? What are the specific challenges and concerns regarding air pollution in the region? Gather insights from experts, stakeholders, and potential users of your analysis.

### Defining Clear Objectives:

Objective 1: Analyze historical air quality data to identify trends and patterns.

Objective 2: Identify regions or monitoring stations with consistently high levels of air pollution. Objective 3: Develop a predictive model to estimate RSPM/PM10 levels based on SO2 and NO2 levels.

### Ideation and Analysis Approach:

**Data Collection:** Identify sources of air quality data in Tamil Nadu, such as government agencies or research institutions.

**Data Pre-processing**: Clean and pre-process the data, handling missing values, outliers, and data quality issues.

**Data Analysis**: Use statistical analysis and visualization techniques to identify trends and patterns in the data.

**Pollution Hotspot Detection:** Develop algorithms or criteria to identify areas with consistently high pollution levels.

**Predictive Modelling**: Choose an appropriate machine learning algorithm to build the predictive model for RSPM/PM10 levels.

**Evaluation**: Define metrics to evaluate the model's performance.

### Prototype and Visualization Selection:

Matplotlib, Seaborn, Plotly, for visualization.

Time series line charts to show air quality trends over time. Heatmaps or geographical maps to identify pollution hotspots.

Scatter plots or regression plots to visualize the relationship between SO2, NO2, and RSPM/PM10 levels.

### Build and Implement:

Develop the full data analysis and visualization pipeline based on the refined approach.

### Test and Iterate:

Continuously test analysis and visualization as progress, making adjustments and refinements based on feedback and new insights.

### Deliver Insights:

Presenting the findings and insights in a clear and understandable manner. Use the selected visualizations to communicate trends, hotspots, and the predictive model's performance.

### Innvoation:

Incorporating machine learning algorithms to improve the accuracy of air quality predictive models in Tamil Nadu (TN) is an excellent way to address air quality concerns and make more precise forecasts. Here's a step-by-step approach on how we are going to leverage machine learning for this purpose:

### Data Collection:

* + Gather historical air quality data from monitoring stations across Tamil Nadu. Include variables such as PM2.5 levels, PM10 levels, NO2, SO2, CO, O3, temperature, humidity, wind speed, and wind direction.
  + Collect data on local weather patterns, industrial activities, traffic congestion, and other relevant factors that can influence air quality.

### Data Preprocessing:

* + Clean and preprocess the data by handling missing values, outliers, and formatting issues.
  + Aggregate data by location and time (e.g., hourly or daily averages) to create a structured dataset for model training.

### Feature Engineering:

* + Engineer features that can capture temporal patterns, seasonality, and external factors affecting air quality, such as public holidays, festivals, and industrial shutdowns.
  + Creating lag features to capture historical trends.

### Select Machine Learning Algorithms:

* + Choose appropriate machine learning algorithms for air quality prediction. Time series forecasting models like ARIMA, SARIMA, or machine learning algorithms like Random Forest, XGBoost, or Long Short-Term Memory (LSTM) recurrent neural networks are commonly used for such tasks.

### Model Training:

* + Split the dataset into training, validation, and test sets.
  + Train the selected machine learning models using historical air quality and environmental data.
  + Optimize hyperparameters and fine-tune the models to achieve the best performance.

### Evaluation Metrics:

* + Evaluation metrics for air quality prediction, such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), or R-squared (R2) to assess model accuracy.

### Cross-Validation:

* + Implement cross-validation techniques to ensure the model's robustness and prevent overfitting.

### Real-Time Data Integration:

* + Set up a data pipeline to collect real-time air quality and environmental data from monitoring stations and weather sources.
  + Continuously update the model with new data to keep it accurate and relevant.

### Model Deployment:

* + Deploy the trained model in a production environment where it can generate real-time air quality predictions.
  + Create a user-friendly interface or API for stakeholders and the public to access air quality forecasts.

**Loading and Pre-processing of data:**

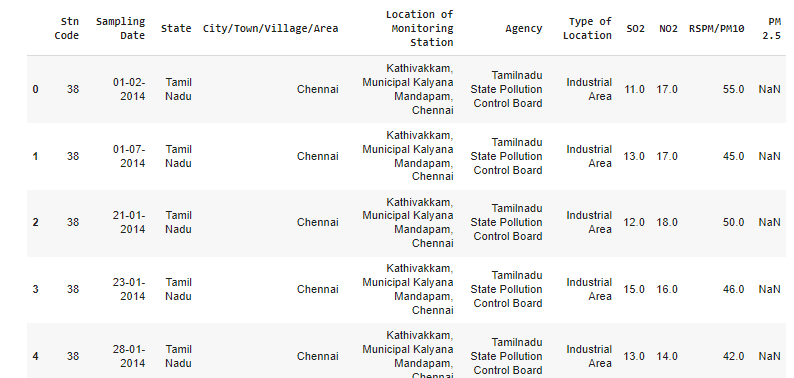
**Loading data import pandas as pd**

import pandas as pd

import numpy as np

data = pd.read\_csv('/content/Air\_quality.csv')

data.head(5)



data.describe()

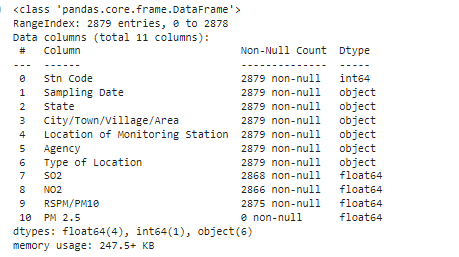


This command is used to view a concise summary of the dataset, including important statistical parameters such as percentiles, standard deviation, mean, minimum, and maximum values for each column, along with a count of data points in each column

data.shape

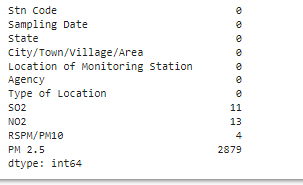


data.info()



The 'info' command is used to check the data type of each column and the count of non-null values in each column. The main difference between 'describe()' and 'info()' is that 'describe()' provides statistical parameters such as mean and standard deviation, while 'info()' does not include these mathematical statistics

data.isna().sum()



The command above is utilized to detect null values in each column. It is evident that there are null values present in columns like SO2, NO2, and RSPM. It is imperative to address and rectify these null values in the dataset.

mean\_no2 = data['NO2'].mean()

data['NO2'] = data['NO2'].fillna(mean\_no2)

mean\_so2 = data['SO2'].mean()

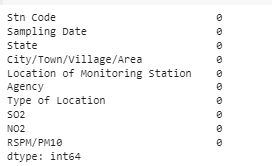
data['SO2'] = data['SO2'].fillna(mean\_so2)

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

data['RSPM/PM10'] = data['RSPM/PM10'].fillna(mean\_rspm)

data.drop('PM 2.5',axis=1,inplace=True)

data.isna().sum()

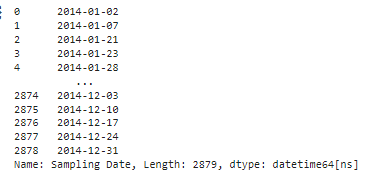


The fillna() method is utilized to replace missing or null values with the mean of the corresponding column.

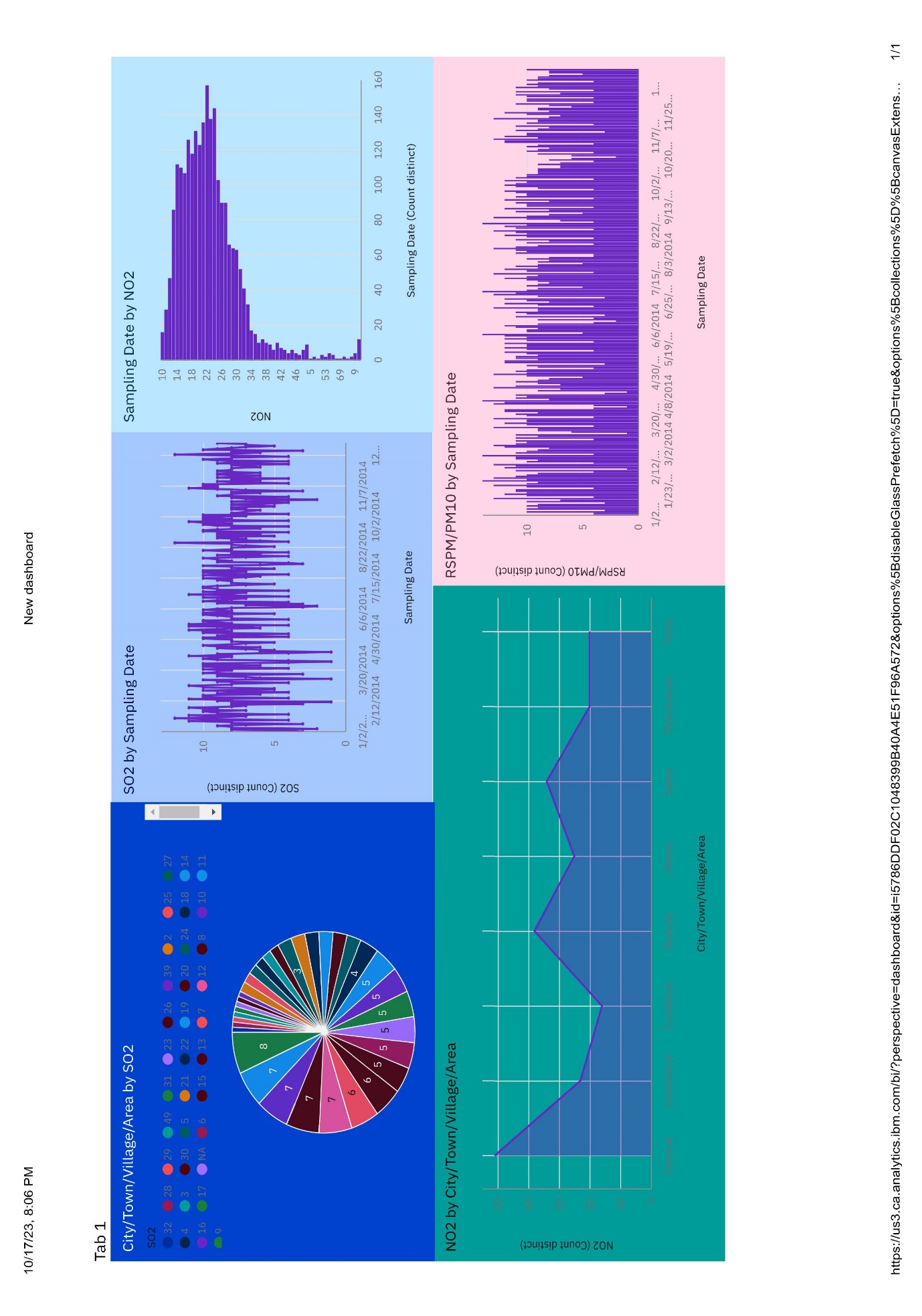
**Converting the date column to date format from object**

data['Sampling Date'] = pd.to\_datetime(data['Sampling Date'])

data['Sampling Date'].dtype



The data type of the 'Sampling Date' column was initially 'object,' which is not suitable for training a model or analyzing the dataset. Therefore, the data type of the column was converted to pandas date and time using the pandas.to\_datetime() function.



**Insights**

Chennai has the highest RSPM/PM10 level at 654, with SO2 contributing the most at 59.

Coimbatore has an RSPM/PM10 level of 61.

From January 30, 2014, to January 31, 2014, the RSPM/PM10 level in location 10 increased by 300%.

Chennai has the highest SO2 levels, with Station Code 161 being the primary contributor.

Chennai is the most frequently occurring City/Town/Village/Area category, accounting for 1000 items with RSPM/PM10 values (34.7% of the total).

The total number of results for RSPM/PM10 across all City/Town/Village/Area categories is nearly three thousand.

# Model Building:

**Clustering Analysis:**

Use unsupervised learning techniques like K-Means clustering or DBSCAN to group your data into clusters based on the available features (SO2, NO2, RSPM/PM10). This can help identify patterns or similarities in air quality data.

# Importing Libraries:

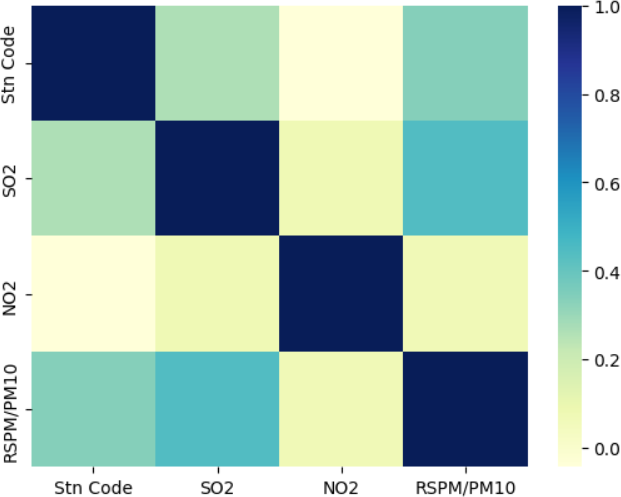
The code begins by importing the necessary Python libraries, including Pandas for data handling, NumPy for numerical operations, Scikit-Learn for machine learning, and Matplotlib for data visualization.

import pandas as pd import numpy as np

from sklearn.cluster import KMeans import matplotlib.pyplot as plt

# Feature Selection:

The code selects the features (independent variables) to be used for clustering, which are 'SO2,' 'NO2,' and 'RSPM/PM10.' These features will be used to determine the clusters.

import seaborn as sns sns.heatmap(data.corr(),cmap='YlGnBu')

X = data[['SO2', 'NO2', 'RSPM/PM10']]

# Feature Standardization:

The features are standardized using the StandardScaler from Scikit-Learn. Standardization ensures that all features have a mean of 0 and a standard deviation of 1, which is important for K-Means clustering.

from sklearn.preprocessing import StandardScaler scaler = StandardScaler()

X = scaler.fit\_transform(X) inertia = []

for k in range(1, 11):

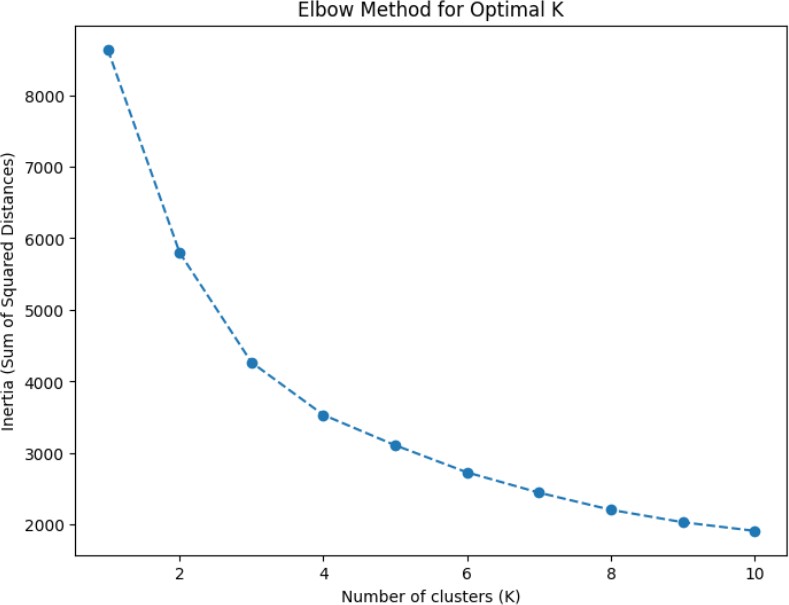
kmeans = KMeans(n\_clusters=k, random\_state=0).fit(X) inertia.append(kmeans.inertia\_)

# Determine the Optimal Number of Clusters:

The code then uses the Elbow method to find the optimal number of clusters (K). It iterates through different values of K and calculates the inertia, which is the sum of squared distances from data points to their assigned cluster centers. The Elbow method plots these inertias for various K values to help you identify the "elbow point" where increasing K doesn't significantly reduce the inertia.

plt.figure(figsize=(8, 6))

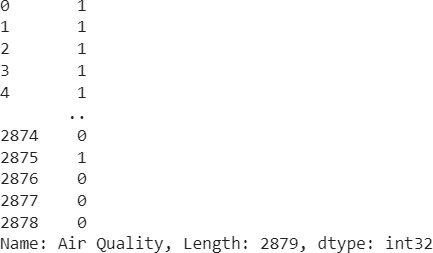
plt.plot(range(1, 11), inertia, marker='o', linestyle='--') plt.title('Elbow Method for Optimal K') plt.xlabel('Number of clusters (K)')

plt.ylabel('Inertia (Sum of Squared Distances)') plt.show()

# K-Means Clustering:

After determining the optimal K (in this case, K = 3), the code performs K-Means clustering using the KMeans algorithm from Scikit-Learn. The clusters are assigned to the 'Cluster' column in the dataset.

kmeans = KMeans(n\_clusters=2, random\_state=0) data['Air Quality'] = kmeans.fit\_predict(X)



plt.figure(figsize=(8, 6))

plt.scatter(X[:, 0], X[:, 1], c=data['Air Quality'], cmap='viridis') plt.title('K-Means Clustering Results')

plt.xlabel('SO2') plt.ylabel('NO2') plt.show()



# Visualization and Insights:

* Chennai has the highest RSPM/PM10 level at 654, with SO2 contributing the most at 59.
* Coimbatore has an RSPM/PM10 level of 61.
* From January 30, 2014, to January 31, 2014, the RSPM/PM10 level in location 10 increased by 300%.
* Chennai has the highest SO2 levels, with Station Code 161 being the primary contributor.
* Chennai is the most frequently occurring City/Town/Village/Area category, accounting for 1000 items with RSPM/PM10 values (34.7% of the total).
* The total number of results for RSPM/PM10 across all City/Town/Village/Area categories is nearly three thousand.

