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:

1. **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.
2. **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.
3. **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.
4. **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.
5. **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.

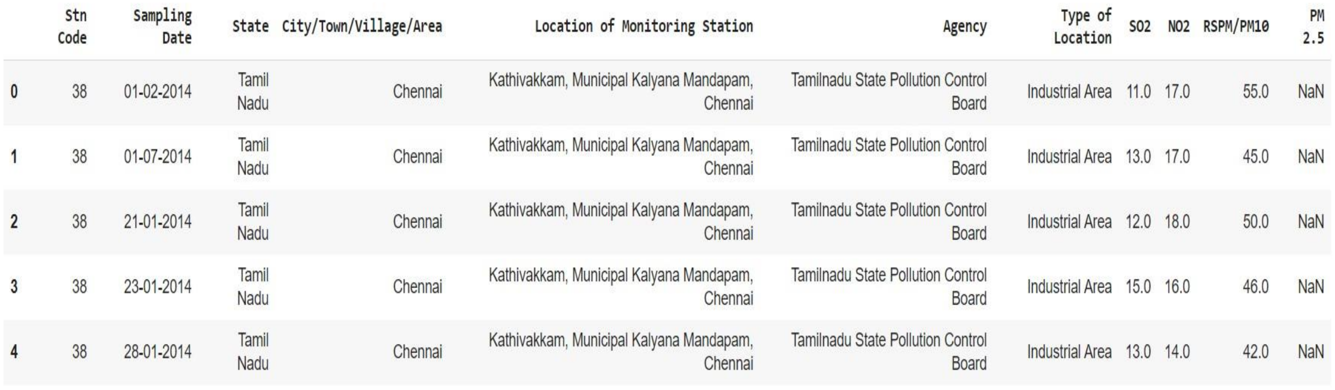
6. **Evaluation Metrics:**
    - Evaluation metrics for air quality prediction, such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), or R-squared (R2) to assess model accuracy.
7. **Cross-Validation:**
    - Implement cross-validation techniques to ensure the model's robustness and prevent overfitting.
8. **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.
9. **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:

```
from google.colab import drive

drive.mount('/content/drive')
```

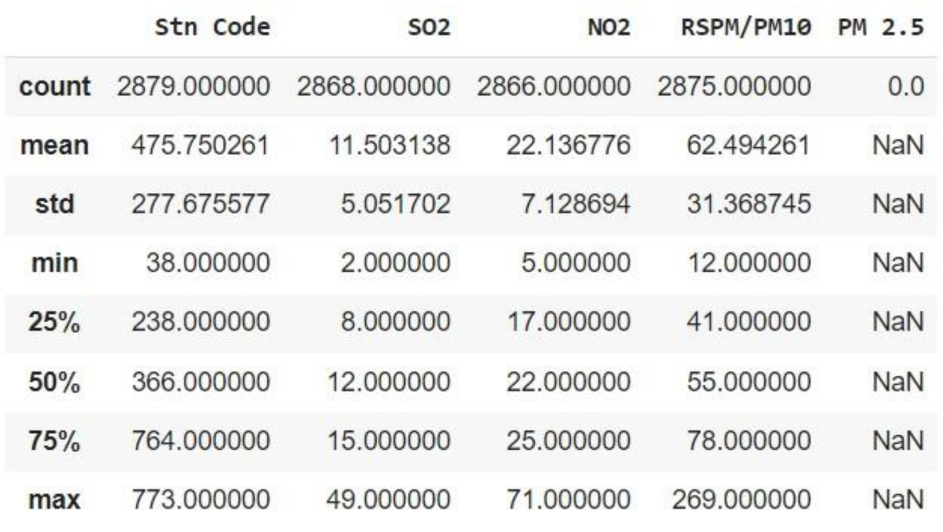
Loading data

```
import pandas as pd import
numpy as np

data = pd.read_csv('/content/drive/MyDrive/datasets/datasets/Air_quality.csv') data.head(5)
```



	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	01-02-2014	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	11.0	17.0	55.0	NaN
1	38	01-07-2014	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	13.0	17.0	45.0	NaN
2	38	21-01-2014	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	12.0	18.0	50.0	NaN
3	38	23-01-2014	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	15.0	16.0	46.0	NaN
4	38	28-01-2014	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	13.0	14.0	42.0	NaN

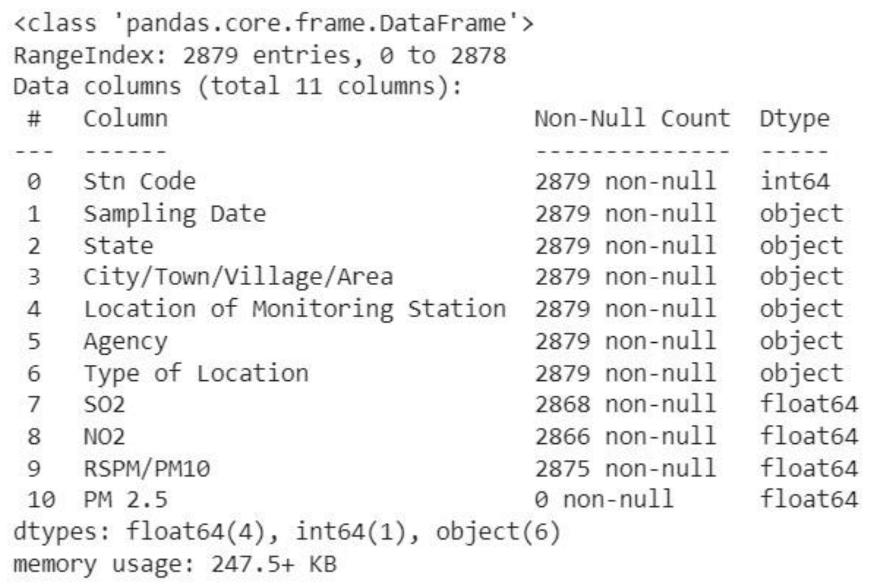
data.describe()



	Stn Code	SO2	NO2	RSPM/PM10	PM 2.5
count	2879.000000	2868.000000	2866.000000	2875.000000	0.0
mean	475.750261	11.503138	22.136776	62.494261	NaN
std	277.675577	5.051702	7.128694	31.368745	NaN
min	38.000000	2.000000	5.000000	12.000000	NaN
25%	238.000000	8.000000	17.000000	41.000000	NaN
50%	366.000000	12.000000	22.000000	55.000000	NaN
75%	764.000000	15.000000	25.000000	78.000000	NaN
max	773.000000	49.000000	71.000000	269.000000	NaN

This command is used to view the brief summary of the dataset. We can see the mathematical parameters such as percentiles, standard deviation , mean, minimum and maximum values and count of each column.

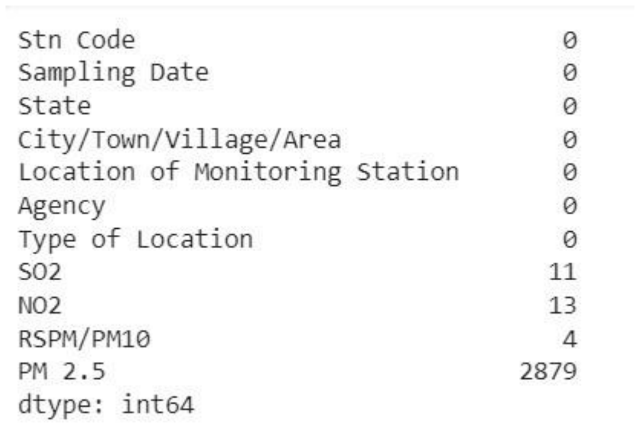
data.info()



```
<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
```

Info command is used check the datatype of every column and the count of each column. The difference between the describe() and info() is that describe command will give the mathematical parameters but info command will not give the mathematical parameters such as mean and standard deviation

data.isna().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
```

The above command is used to check for null values in each column. We can see that there are null values in the columns such as SO2,NO2,RSPM. It is very necessary to take action to clear the null values in the data set

mean_so2 = data['SO2'].mean()

data['SO2'] = data['SO2'].fillna(mean_so2)

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

data['NO2'] = data['NO2'].fillna(mean_no2)

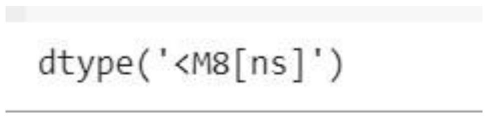
mean_rspm = data['RSPM/PM10'].mean()

data['RSPM/PM10'] = data['RSPM/PM10'].fillna(mean_rspm) data.drop('PM 2.5',axis=1,inplace=True)

Here fillna() method is used to fill the null values by the mean of the particular column

Converting the date column to date format from object

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



```
dtype('<M8[ns]')
```

Initially the data type of the ‘Sampling Date’ column was object this will not be suitable to train a model or analyse the data set , so the data type of the column is converted to pandas date and time using pandas.to_datetime()

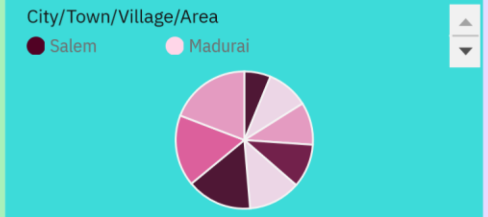
Tab 1

SO2

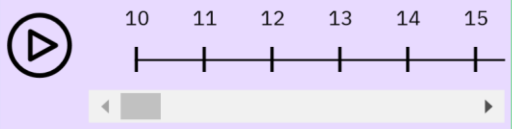
34

SO2

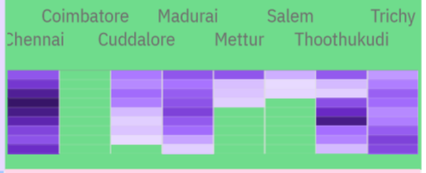
RSPM/PM10 by City/Town/Village/Area



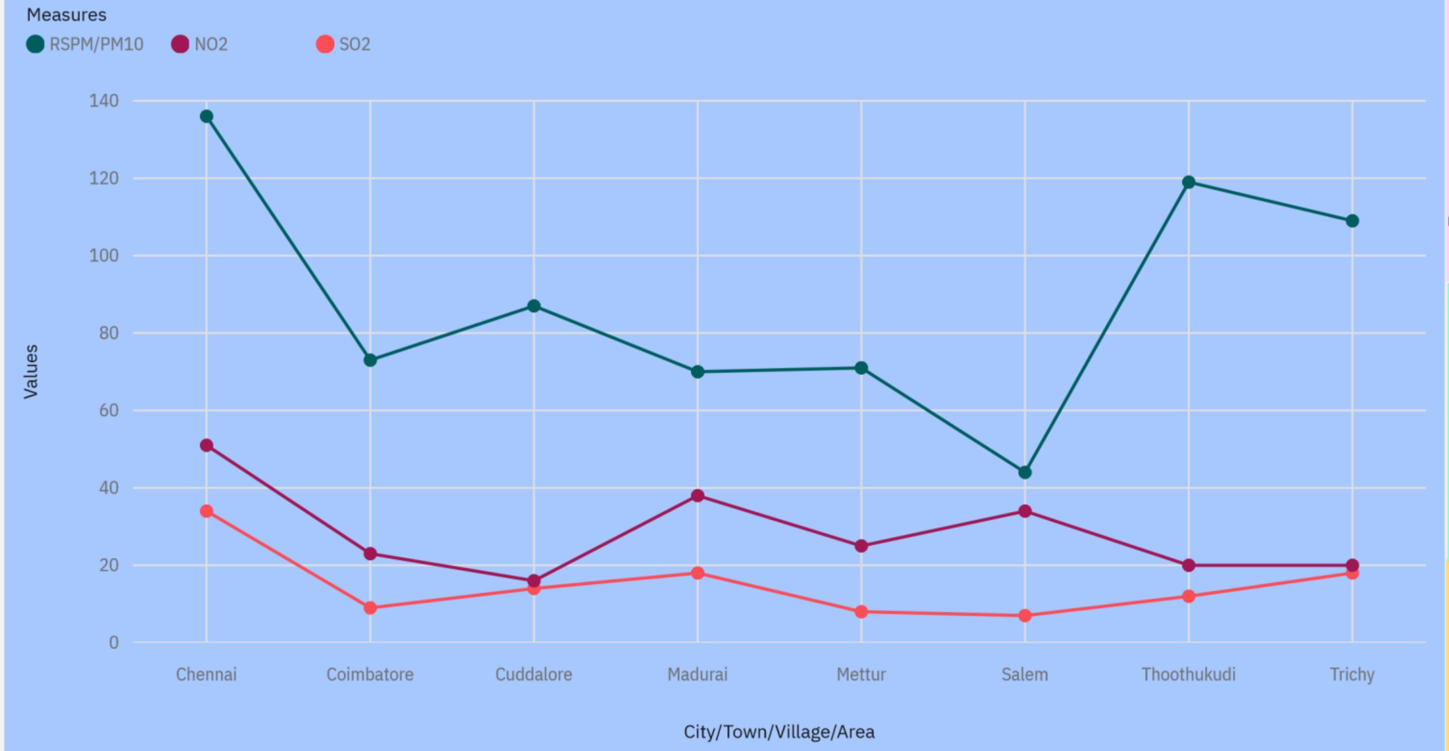
NO2



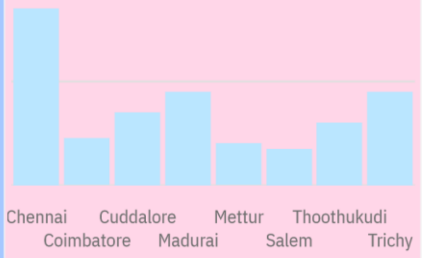
RSPM/PM10 by SO2 and City/Town/Village/Area



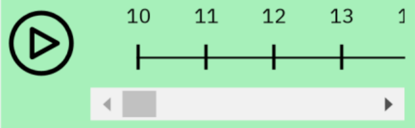
RSPM/PM10, NO2 and SO2 by City/Town/Village/Area



SO2 by City/Town/Village/Area



SO2



NO2

54

NO2

Insights:

1. Chennai has the highest RSPM/PM10 at 654, out of which SO2 13 contributed the most at 59.
2. 4 has a RSPM/PM10 of 61 for Coimbatore.
3. From 2014-01-30 to 2014-01-31, 10's RSPM/PM10 increased by 300%.
4. Chennai has the highest SO2 due to Stn Code 161.
5. Chennai is the most frequently occurring category of City/Town/Village/Area with a count of 1000 items with RSPM/PM10 values (34.7 % of the total).
6. The total number of results for RSPM/PM10, across all City/Town/Village/Area, is nearly three thousand.

Phase 4:

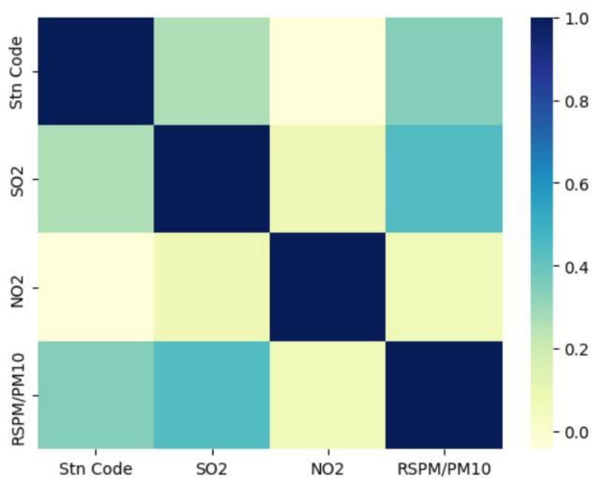
Model Building:
Clustering Analysis:
Use unsupervised learning techniques like K-Means clustering or DBSCAN to group your data into clusters based on the available features (SO2, NO2, RSPM/PM10). This can help identify patterns or similarities in air quality data.

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:

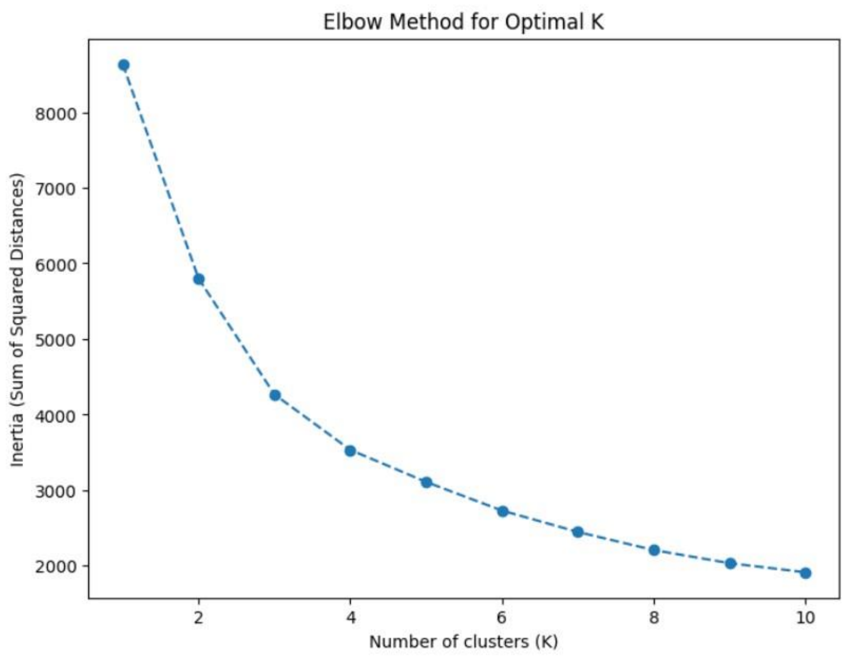
The features are standardized using the StandardScaler from Scikit-Learn. Standardization ensures that all features have a mean of 0 and a standard deviation of 1, which is important for K-Means clustering. from sklearn.preprocessing import StandardScaler

```
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.

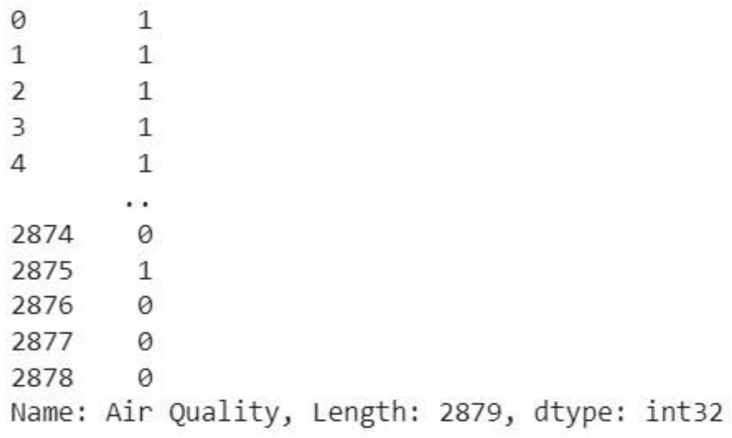
```
plt.figure(figsize=(8, 6))

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.

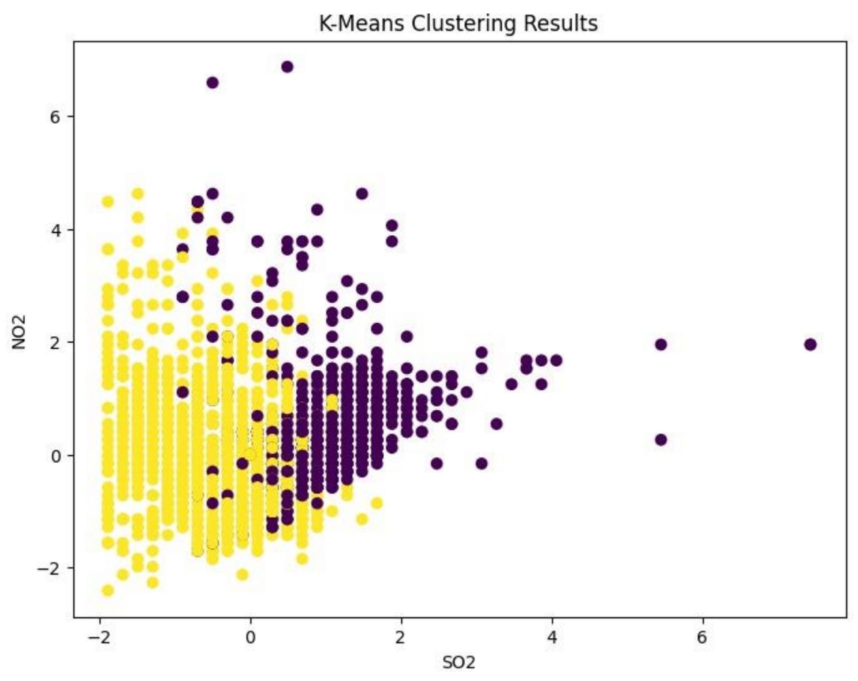
kmeans = KMeans(n_clusters=2, random_state=0) data['Air Quality'] = kmeans.fit_predict(X)



```
0       1
1       1
2       1
3       1
4       1
       ..
2874    0
2875    1
2876    0
2877    0
2878    0
Name: Air Quality, Length: 2879, dtype: int32
```

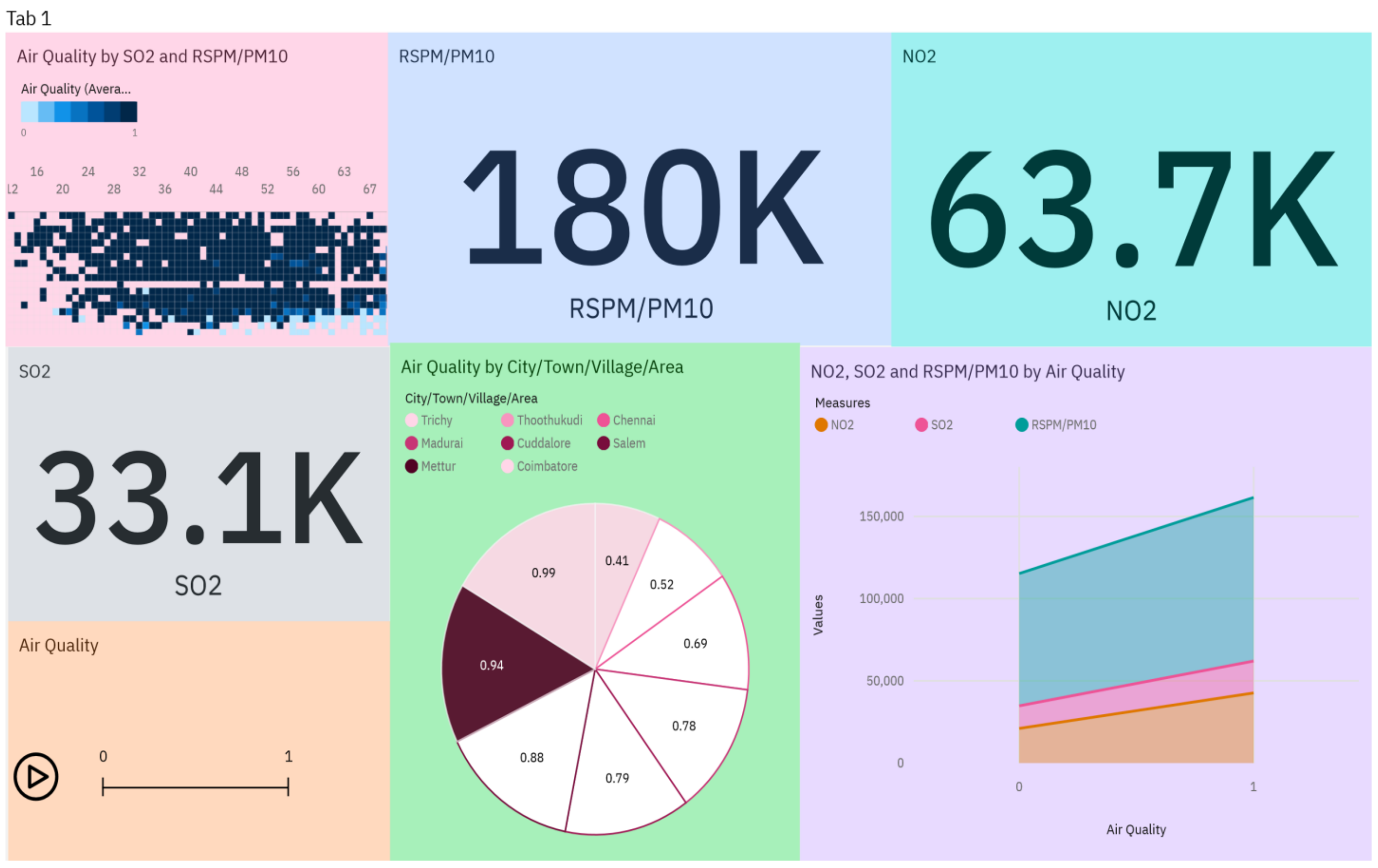
plt.figure(figsize=(8, 6))

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:

1. The RSPM/PM10 and Air Quality relationship is weakly influenced by SO2.
2. RSPM/PM10 44 has the highest Air Quality at 16.63, out of which SO2 2 contributed the most at 1.
3. Air Quality is most unusual when City/Town/Village/Area is Trichy, Coimbatore and Mettur.
4. Chennai is the most frequently occurring category of City with a count of 1000 items with Air Quality values (34.7 % of the total).
5. Over all air qualities, the average of NO2 is 22.14, the average of RSPM/PM10 is 62.49,the average of SO2 is 11.5.


Tab 1
Air Quality by SO2 and RSPM/PM10
Air Quality (Avera...
0
1
12 16 20 24 28 32 36 40 44 48 52 56 60 63 67
RSPM/PM10
180K
RSPM/PM10
NO2
63.7K
NO2
SO2
33.1K
SO2
Air Quality
0
1
Air Quality by City/Town/Village/Area
City/Town/Village/Area
Trichy
Thoothukudi
Chennai
Madurai
Cuddalore
Salem
Mettur
Coimbatore
0.41
0.52
0.69
0.78
0.79
0.88
0.94
0.99
NO2, SO2 and RSPM/PM10 by Air Quality
Measures
NO2
SO2
RSPM/PM10
150,000
100,000
50,000
0
Values
0
1
Air Quality