PROBLEM STATEMENT:

The problem statement of air quality prediction involves developing models or systems to forecast and monitor air quality in specific locations. This typically includes predicting air pollutants like PM2.5, PM10, ozone, nitrogen dioxide, sulfur dioxide, and carbon monoxide. The goal is to provide real-time or future air quality information to help individuals, communities, and authorities make informed decisions related to health, the environment, and public safety. Various data sources, including sensors, satellites, and historical data, are used to create these predictive models.

THE PROBLEM DEFINITION:

Problem: To develop a reliable system for predicting air quality levels, with a focus on specific pollutants and geographical regions, in order to provide timely and accurate information for public health, environmental protection, and policy decision-making.

Key Aspects:

Scope: Determine the scope of the prediction task, specifying the pollutants of interest (e.g., PM2.5, ozone, nitrogen dioxide), the target geographical area (e.g., a city or region), and the time frame for predictions (e.g., daily, weekly, or long-term).

Objective: Define the primary goal of the prediction system, whether it's real-time monitoring, short-term forecasting, or long-term trend analysis, and clarify the intended use of the predictions.

Data Sources: Identify the sources of data, including ground-based monitoring stations, satellite data, meteorological information, and any other relevant data sources. Explain how and where this data will be collected or obtained.

Prediction Model: Specify the prediction model or methodology to be employed, such as machine learning models, numerical simulations, or statistical techniques. Describe how the model will be developed, trained, and implemented.

THE DESIGN THINKING PROCESS FOR AIR QUALITY PREDICTION INVOLVES SEVERAL KEY STEPS:

Empathize: Understand the needs and concerns of the community and individuals affected by air quality. This may involve gathering data, conducting surveys, and speaking with stakeholders to gain insights.

Define: Clearly define the problem and the specific goals of the air quality prediction system. What pollutants are you trying to predict, and for whom (e.g., residents, policymakers, healthcare professionals)?

Ideate: Brainstorm potential solutions and prediction models. Explore various data sources, machine learning algorithms, and sensor technologies that can be used to forecast air quality.

Prototype: Create a prototype or proof-of-concept of the air quality prediction system. This may involve developing a predictive model and a user interface for accessing air quality information.

Test: Test the prototype with real data and user feedback. Evaluate its accuracy, usability, and effectiveness in providing meaningful air quality predictions.

Iterate: Based on the feedback and test results, make necessary improvements to the prediction system. This may involve refining the model, enhancing data collection methods, or improving the user interface.

Implement: Once the prototype is refined and proven effective, implement the air quality prediction system in real-world settings. Ensure that it can provide ongoing, reliable predictions.

THERE ARE SEVERAL INNOVATIVE SOLUTIONS THAT CAN BE IMPLEMENTED FOR AIR QUALITY PREDICTION IN TAMIL NADU:

Mobile Sensor Networks: Deploying mobile sensor networks on vehicles or drones to collect real-time air quality data across different locations in Tamil Nadu. This approach can provide fine-grained and dynamic data for more accurate predictions.

Fusion of Satellite Data: Integrating satellite imagery and remote sensing data with ground-based measurements to improve the accuracy of air quality predictions. This can provide a broader spatial perspective.

Machine Learning and AI: Utilizing advanced machine learning and AI techniques, including deep learning and neural networks, to develop more accurate and adaptive air quality prediction models. These models can learn from historical data and adapt to changing conditions.

IoT and Edge Computing: Leveraging the Internet of Things (IoT) devices and edge computing for real-time data collection, processing, and prediction at the local level. This can enable faster response to air quality changes.

DATASET LINK:

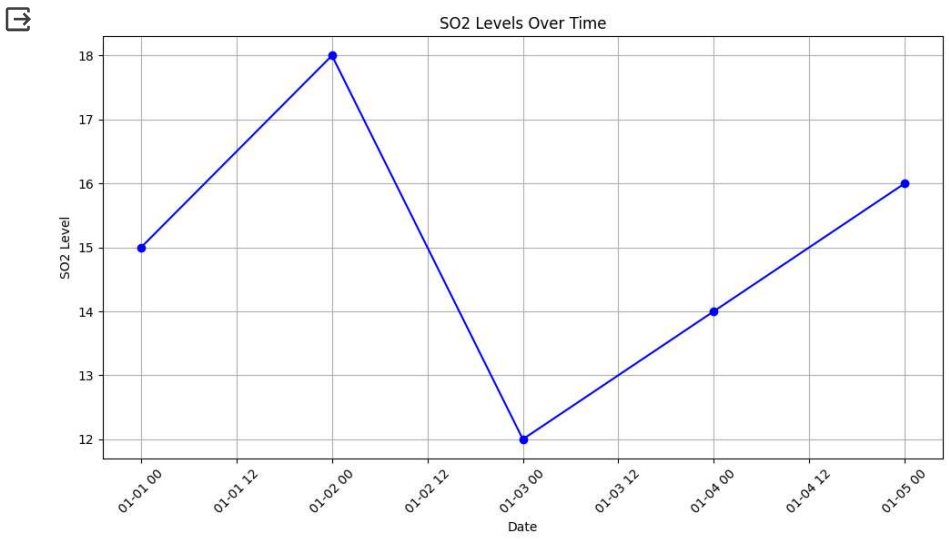
https://www.kaggle.com/datasets/seshupavan/air-pollution-data-of-india-2020-2023/

PROJECT TIMELINE:


Different Zones of India
PM2.5
NOx
Difference in Concentration (%)
10
0
-10
-20
-30
-40
-50
-60
-70
North
21 Cities
-34
-49
East
5 Cities
-17
-57
Centre
7 Cities
1.7
-46
West
6 Cities
-31
-60
South
7 Cities
-26
-32

STEP 1: create visualization for SO2

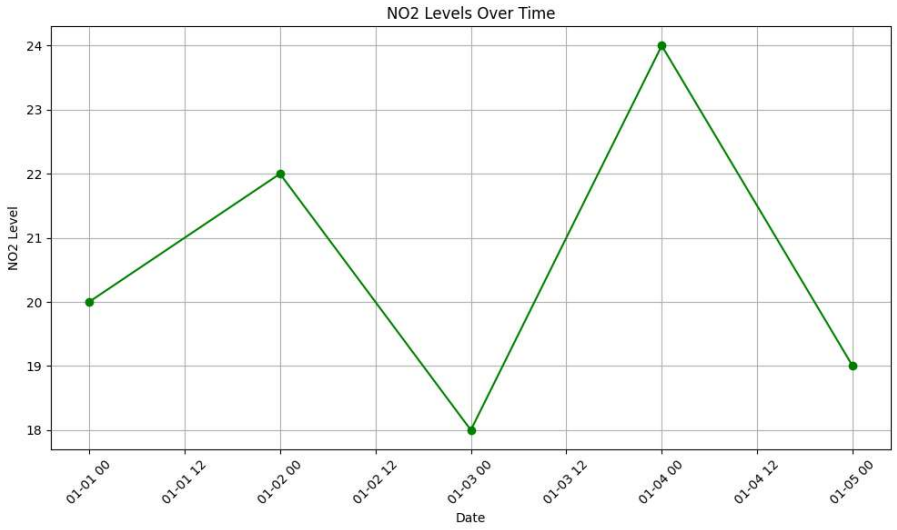
```
import matplotlib.pyplot as plt
import pandas as pd

# Sample data (replace this with your actual SO2 dataset)
data = {
    'Date': ['2023-01-01', '2023-01-02', '2023-01-03', '2023-01-04', '2023-01-05'],
    'SO2_Level': [15, 18, 12, 14, 16]  # Replace with your SO2 data
}

# Create a DataFrame from the data
df = pd.DataFrame(data)
df['Date'] = pd.to_datetime(df['Date'])  # Convert Date column to datetime format

# Plot the SO2 levels
plt.figure(figsize=(10, 6))
plt.plot(df['Date'], df['SO2_Level'], marker='o', linestyle='-', color='b')
plt.title('SO2 Levels Over Time')
plt.xlabel('Date')
plt.ylabel('SO2 Level')
plt.grid(True)

# Format the x-axis to display dates nicely
plt.xticks(rotation=45)

# Show the plot
plt.tight_layout()
plt.show()
```



STEP 2: create visualization for NO2

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

# Sample data (replace this with your actual NO2 dataset)
data = {
    'Date': ['2023-01-01', '2023-01-02', '2023-01-03', '2023-01-04', '2023-01-05'],
    'NO2_Level': [20, 22, 18, 24, 19]  # Replace with your NO2 data
}
```

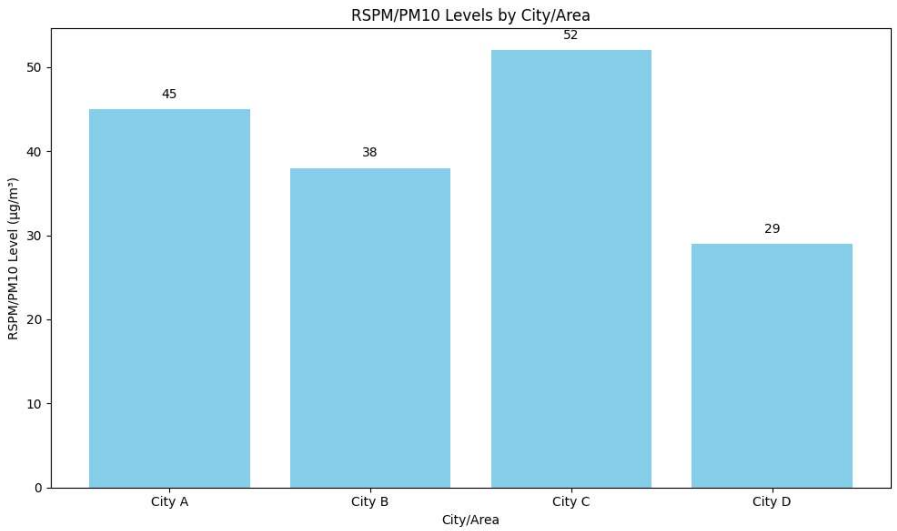
```
# Create a DataFrame from the data
df = pd.DataFrame(data)
df['Date'] = pd.to_datetime(df['Date'])  # Convert Date column to datetime format

# Plot the NO2 levels
plt.figure(figsize=(10, 6))
plt.plot(df['Date'], df['NO2_Level'], marker='o', linestyle='-', color='g')
plt.title('NO2 Levels Over Time')
plt.xlabel('Date')
plt.ylabel('NO2 Level')
plt.grid(True)

# Format the x-axis to display dates nicely
plt.xticks(rotation=45)

# Show the plot
plt.tight_layout()
plt.show()
```



STEP 3: create visualization for RSPM/PM10

```
import matplotlib.pyplot as plt

# Sample data (replace this with your actual RSPM/PM10 dataset)
categories = ['City A', 'City B', 'City C', 'City D']
rspm_pm10_levels = [45, 38, 52, 29]  # Replace with your RSPM/PM10 data

# Create a bar chart to visualize RSPM/PM10 levels
plt.figure(figsize=(10, 6))
plt.bar(categories, rspm_pm10_levels, color='skyblue')
plt.title('RSPM/PM10 Levels by City/Area')
plt.xlabel('City/Area')
plt.ylabel('RSPM/PM10 Level (µg/m³)')

# Add data labels above each bar
for i, level in enumerate(rspm_pm10_levels):
    plt.text(i, level + 1, str(level), ha='center', va='bottom')

# Show the plot
plt.tight_layout()
```

```
plt.show()
```




```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

STEP 4: grouping data

```
import pandas as pd

# Load your data into a Pandas DataFrame (replace 'data.csv' with your data file)
df = pd.read_csv('/content/drive/MyDrive/Excel 4.csv')

# Group the data by the desired column (e.g., 'City' or 'Monitoring Station')
grouped_data = df.groupby('CITIES')
```

STEP 5: calculating average

```
# Calculate the average SO2, NO2, and RSPM/PM10 levels for each group
averages = grouped_data[['SO2', 'NO2', 'RSPM/PM10']].mean()

# Display the calculated averages
print(averages)
```

```
                     SO2        NO2   RSPM/PM10
CITIES
Chennai         9.433333  23.036667  145.640000
Dindigul       22.880000  23.753333  129.780000
Erode          18.900000  27.065000  165.580000
Kanniyakumari  27.790000  34.330000  148.680000
Kodaikanal     12.130000  38.050000  171.910000
Madurai        13.897500  26.727500  156.572500
Salem          11.100000  30.060000  160.976667
Thanjavur       3.650000  30.790000  145.180000
Vellore        16.726667  23.116667  141.240000
madurai        10.220000  30.910000  114.270000
salem          30.990000  29.050000  128.150000
```

```
# Calculate the average SO2, NO2, and RSPM/PM10 levels for each group
averages = grouped_data[[ 'RSPM/PM10']].mean()

# Display the calculated averages
print(averages)
```

```
                    RSPM/PM10
CITIES
Chennai        145.640000
Dindigul       129.780000
Erode          165.580000
Kanniyakumari  148.680000
Kodaikanal     171.910000
Madurai        156.572500
Salem          160.976667
Thanjavur      145.180000
Vellore        141.240000
madurai        114.270000
salem          128.150000
```