

AIR QUALITY ANALYSIS USING MACHINE LEARNING

Project: *Air Quality Analysis*

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

OBJECTIVE:

The project objectives specifically focused on air quality analysis, including the analysis of air quality trends, identifying pollution hotspots, and building a predictive model for RSPM/PM10 levels:

1. Air Quality Trend Analysis:

- Analyze historical air quality data to identify long-term trends and variations in key pollutants, including RSPM/PM10, NO2, SO2, CO, O3, and VOCs.

2. Pollution Hotspot Identification:

- Identify geographical areas or specific sources within [Location] that consistently exhibit high levels of RSPM/PM10, pinpointing pollution hotspots.
- Determine the contributing factors and sources of RSPM/PM10 emissions in these hotspots.

3. Predictive Model Development:

- Develop a predictive model for RSPM/PM10 levels based on historical air quality data, meteorological parameters, and other relevant variables.
- Validate the model's accuracy and reliability through testing and validation datasets.

4. Spatial Mapping:

- Create spatial maps illustrating the spatial distribution of RSPM/PM10 levels across [Location], helping to visualize pollution patterns.

5. Temporal Analysis:

- Conduct temporal analysis to understand how RSPM/PM10 levels vary throughout the day, week, month, and season, and identify any cyclical patterns.

6. Correlation Analysis:

- Investigate potential correlations between RSPM/PM10 levels and other environmental factors, such as temperature, humidity, wind speed, and traffic volume.

7. Health Impact Assessment:

- Assess the potential health risks associated with elevated RSPM/PM10 levels, including respiratory and cardiovascular health impacts, and communicate findings to raise public awareness.

8. Emission Source Attribution:

- Use source apportionment techniques to attribute RSPM/PM10 emissions to specific sources or sectors, such as industrial activities, transportation, or natural sources.

9. Recommendation of Mitigation Strategies:

- Based on hotspot identification and source attribution results, provide recommendations for targeted mitigation strategies to reduce RSPM/PM10 emissions and improve air quality.

10. Data Visualization and Public Access:

- Create user-friendly data visualization tools and reports to present findings to the public, local authorities, and stakeholders, promoting transparency and understanding.

11. Long-Term Monitoring Plan:

- Propose a plan for ongoing monitoring of RSPM/PM10 levels and related pollutants to track the effectiveness of mitigation measures and adapt strategies as needed.

By achieving these objectives, the project aims to enhance our understanding of RSPM/PM10 pollution dynamics, contribute to better air quality management, and ultimately improve the health and well-being of residents in [Location].

Analysis Approach for Air Quality Data:

1. Data Collection:

- Gather air quality data from reliable sources, which may include government agencies, environmental monitoring stations, or sensor networks.
- Ensure that data is comprehensive, covering relevant pollutants (e.g., RSPM/PM10, NO2, SO2, CO, O3, VOCs) and includes metadata such as date, time, and location.
- Dataset Link:- (<https://tn.data.gov.in/resource/location-wise-daily-ambient-air-quality-tamil-nadu-year-2014>)

2. Data Cleaning and Preprocessing:

- Handle missing or incomplete data points through imputation or removal as appropriate.
- Perform data validation to identify outliers and erroneous readings and address them through interpolation or other data correction techniques.
- Convert data formats if needed (e.g., date/time formatting, unit conversions).

3. Exploratory Data Analysis (EDA):

- Conduct initial data exploration to gain insights into the dataset's characteristics.
- Generate summary statistics, histograms, and scatter plots to understand the distribution of pollutants and detect trends.
- Identify any seasonality or periodicity in the data through time series analysis.

4. Feature Engineering:

- Create additional features if necessary, such as rolling averages, moving aggregates, or lagged variables, to capture relevant temporal patterns.
- Incorporate meteorological data (e.g., temperature, humidity, wind speed) if available, as these can impact air quality.

5. Data Integration:

- Merge or join multiple datasets if there are data sources from various monitoring stations or sensors to create a unified dataset for analysis.

6. Statistical Analysis:

- Perform statistical tests, such as correlation analysis, to explore relationships between air pollutants and potential influencing factors.
- Conduct time series decomposition to separate trends, seasonal patterns, and residuals.

7. Spatial Analysis:

- If working with spatial data, use Geographic Information System (GIS) tools to analyze spatial patterns, create maps, and identify pollution hotspots.

8. Machine Learning and Predictive Modeling:

- Develop predictive models for air quality using machine learning algorithms like regression, time series forecasting, or deep learning.
- Split the dataset into training and testing subsets for model validation.

9. Model Evaluation:

- Evaluate model performance using appropriate metrics (e.g., Mean Absolute Error, R-squared) and cross-validation techniques.
- Refine models and hyperparameters as needed for better predictive accuracy.

10. Visualization:

- Create informative and visually appealing charts and graphs to present the analysis results. Common visualization tools include Matplotlib, Seaborn, or specialized GIS software.
- Generate time series plots, heatmaps, scatter plots, and geographic maps to visualize air quality trends, correlations, and spatial variations.

11. Reporting and Documentation:

- Document the entire analysis process, including data sources, preprocessing steps, analysis techniques, and model details.
- Summarize key findings, insights, and any actionable recommendations for stakeholders.

12. Continuous Monitoring and Updates:

- Establish a plan for ongoing data collection, analysis, and reporting to ensure that air quality trends and mitigation efforts are monitored over time.

This analysis approach provides a structured framework for loading, preprocessing, analyzing, and visualizing air quality data to derive meaningful insights of it.

Visualization Selection for Air Quality Analysis:

When selecting visualization techniques to effectively represent air quality trends and pollution levels, it's important to choose methods that provide clarity, facilitate insights, and communicate findings to a broad audience. Here are some visualization techniques commonly used in air quality analysis:

1. Time Series Plots (Line Charts):

- Use line charts to depict the temporal trends of air quality parameters (e.g., RSPM/PM10, NO2, CO) over a specific time period.
- Multiple pollutants can be plotted on the same chart for easy comparison.
- Highlighting pollution thresholds and regulatory limits can provide context.

2. Heatmaps:

- Create heatmaps to visualize spatial and temporal variations in air quality across different locations and time intervals.
- Color coding can represent pollutant concentrations, with gradients indicating intensity.
- Heatmaps are useful for identifying pollution hotspots.

3. Scatter Plots:

- Utilize scatter plots to show relationships and correlations between air quality parameters and other variables (e.g., temperature, humidity, wind speed).
- Each data point represents a measurement at a specific time, allowing for trend identification and outlier detection.

4. Bar Charts:

- Bar charts can be used to compare pollutant concentrations across different monitoring stations or regions.
- Stacked bar charts can show the contribution of different pollutant sources to the overall air quality.

5. Box Plots:

- Box plots provide a compact summary of pollutant distributions, including median values, quartiles, and outliers.
- They are effective for visualizing the spread and skewness of data.

6. Time Series Decomposition:

- Decompose time series data into trend, seasonal, and residual components using techniques like seasonal decomposition of time series (STL).
- Visualization can include subplots showing each component separately, helping to understand underlying patterns.

7. Geospatial Maps:

- Geographic Information System (GIS) tools can be used to create maps that display air quality data spatially.
- Color-coded markers or choropleth maps can represent pollutant concentrations in different regions or locations.

8. Animated Visualizations:

- Animation can be employed to display temporal changes in air quality, creating dynamic visualizations that highlight trends and fluctuations over time.
- Animated line charts, heatmaps, or maps can be effective for this purpose.

9. Polar Plots:

- Polar plots can be used to display diurnal or seasonal variations in air quality, especially when cyclic patterns are observed.
- Concentric rings represent time intervals, and radial lines show pollutant levels.

10. Interactive Dashboards:

- Develop interactive dashboards using tools like Tableau, Power BI, or custom web applications to allow users to explore and filter data themselves.
- Users can select pollutants, time ranges, and regions for personalized analysis.

The choice of visualization techniques should align with the specific objectives of the analysis and the needs of the target audience. Effective visualizations can enhance understanding, facilitate decision-making, and communicate the complexities of air quality data clearly.

Python Program for Visualizing(EDA):-

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns, numpy as np, os

from scipy.stats import pearsonr, spearmanr

# Load the dataset

df = pd.read_excel(r'C:\Users\HP\Downloads\ -Machine-Learning-2023-Air-Quality-
Prediction-main\ -Machine-Learning-2023-Air-Quality-Prediction-main\dataset\data.xlsx',
parse_dates= True)

df.set_index('datetime', inplace=True)

print(df.head())

print(df.info())

print(df.describe())

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

df.hist(bins=50, figsize=(20,15))

plt.show()

sns.heatmap(df.corr(), cmap='coolwarm', vmin=-0.9, vmax=0.9, annot=True, fmt='.2f')

plt.show()

sns.pairplot(df, x_vars=['ws', 'wd', 'temp', 'dew_temp', 'pressure', 'wv', 'blh', 'bcaod550',
'duaod550', 'omaod550', 'ssaod550', 'suaod550', 'aod469', 'aod550', 'aod670', 'aod865',
'aod1240'], y_vars=['pm2p5'], height=7, aspect=0.7)

plt.show()

# , height=8, aspect=0.5

from sklearn.feature_selection import mutual_info_regression

X = df.copy()

y = X.pop("pm2p5")

def make_mi_scores(X, y):

    mi_scores = mutual_info_regression(X, y)
```

```
mi_scores = pd.Series(mi_scores, name="MI Scores", index=X.columns)

mi_scores = mi_scores.sort_values(ascending=False)

return mi_scores

mi_scores = make_mi_scores(X, y)

mi_scores[::3] # show a few features with their MI scores

def plot_mi_scores(scores):

    scores = scores.sort_values(ascending=True)

    width = np.arange(len(scores))

    ticks = list(scores.index)

    plt.barh(width, scores)

    plt.yticks(width, ticks)

    plt.title("Mutual Information Scores")

plt.figure(dpi=100, figsize=(8, 5))

plot_mi_scores(mi_scores)

# Calculate Pearson correlation coefficient

corr_p, p_val_p = pearsonr(df['pm2p5'], df['pressure'])

print("Pearson correlation coefficient:", corr_p)

print("p-value:", p_val_p)

# Calculate Spearman correlation coefficient

corr_s, p_val_s = spearmanr(df['pm2p5'], df['pressure'])

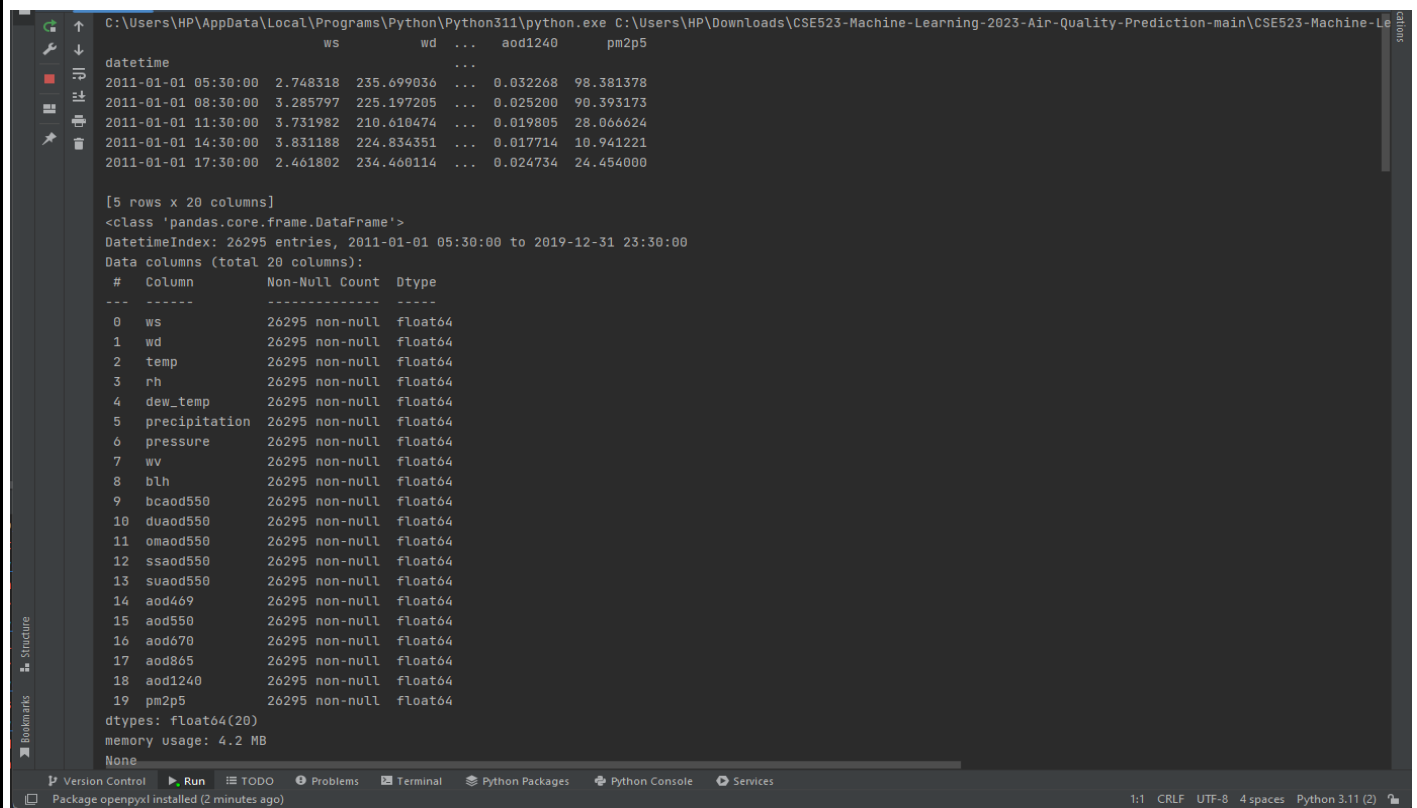
print("Spearman correlation coefficient:", corr_s)

print("p-value:", p_val_s)
```


Output:-

```
Pearson correlation coefficient: 0.46912222965661793
p-value: 0.0
Spearman correlation coefficient: 0.4690099375988372
p-value: 0.0

Process finished with exit code 0
```

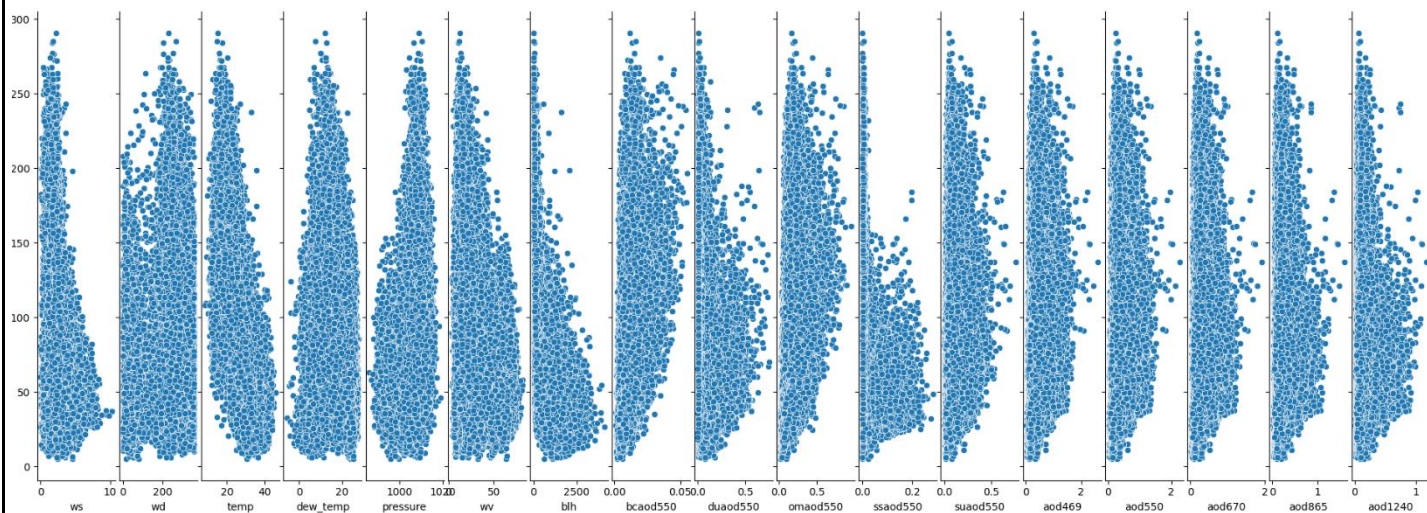
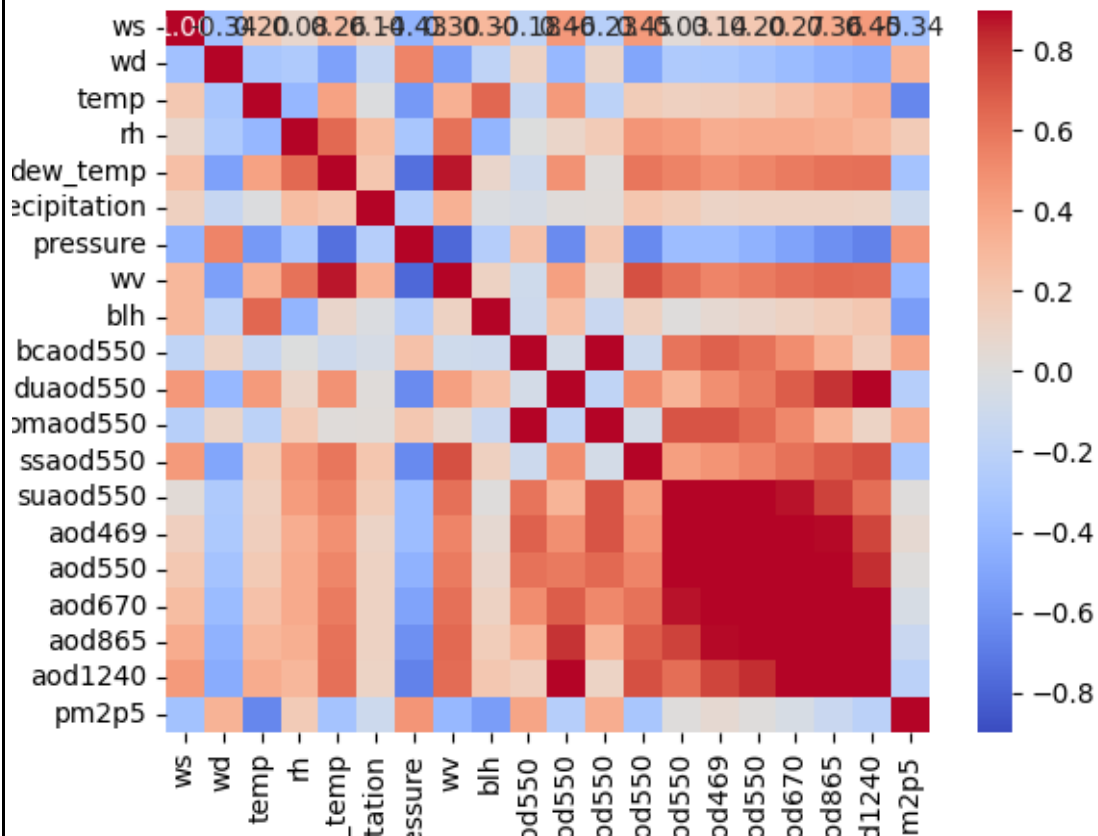


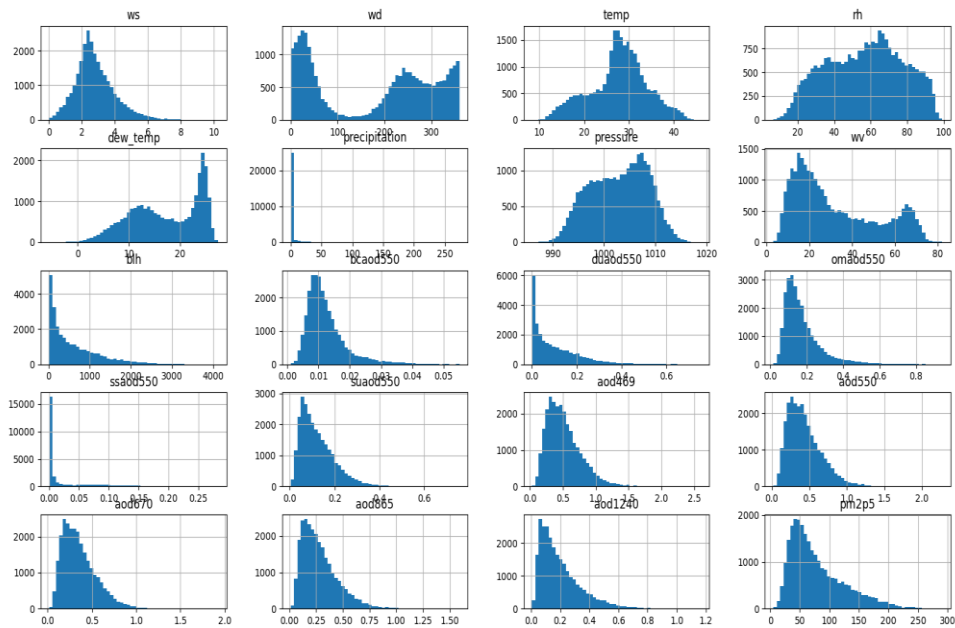
The screenshot shows a Jupyter Notebook interface with a dark theme. The main area displays the output of a pandas DataFrame operation. At the top, a preview of 5 rows is shown with columns: datetime, ws, wd, ..., aod1240, pm2p5. Below this, the DataFrame's metadata is displayed: [5 rows x 20 columns], <class 'pandas.core.frame.DataFrame'>, DatetimeIndex: 26295 entries, 2011-01-01 05:30:00 to 2019-12-31 23:30:00, and Data columns (total 20 columns):. A table follows, listing columns 0 through 19, their non-null counts (all 26295), and their dtypes (all float64). The dtypes are listed as float64(20). The memory usage is 4.2 MB. The status bar at the bottom shows 'Package openpyxl installed (2 minutes ago)', '1:1 CRLF UTF-8 4 spaces Python 3.11 (2)', and icons for Version Control, Run, TODO, Problems, Terminal, Python Packages, Python Console, and Services.

```
C:\Users\HP\AppData\Local\Programs\Python\Python311\python.exe C:\Users\HP\Downloads\CSE523-Machine-Learning-2023-Air-Quality-Prediction-main\CSE523-Machine-Le
ws wd ... aod1240 pm2p5
datetime
2011-01-01 05:30:00 2.748318 235.699036 ... 0.032268 98.381378
2011-01-01 08:30:00 3.285797 225.197205 ... 0.025200 90.393173
2011-01-01 11:30:00 3.731982 210.610474 ... 0.019805 28.066624
2011-01-01 14:30:00 3.831188 224.834351 ... 0.017714 10.941221
2011-01-01 17:30:00 2.461802 234.460114 ... 0.024734 24.454000

[5 rows x 20 columns]
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 26295 entries, 2011-01-01 05:30:00 to 2019-12-31 23:30:00
Data columns (total 20 columns):
# Column Non-Null Count Dtype
---
0 ws 26295 non-null float64
1 wd 26295 non-null float64
2 temp 26295 non-null float64
3 rh 26295 non-null float64
4 dew_temp 26295 non-null float64
5 precipitation 26295 non-null float64
6 pressure 26295 non-null float64
7 wv 26295 non-null float64
8 blh 26295 non-null float64
9 bcaod550 26295 non-null float64
10 duaod550 26295 non-null float64
11 omaod550 26295 non-null float64
12 ssaod550 26295 non-null float64
13 suaod550 26295 non-null float64
14 aod469 26295 non-null float64
15 aod550 26295 non-null float64
16 aod670 26295 non-null float64
17 aod865 26295 non-null float64
18 aod1240 26295 non-null float64
19 pm2p5 26295 non-null float64
dtypes: float64(20)
memory usage: 4.2 MB
None

Version Control Run TODO Problems Terminal Python Packages Python Console Services
Package openpyxl installed (2 minutes ago) 1:1 CRLF UTF-8 4 spaces Python 3.11 (2)
```





CONCLUSION:

In Phase 4, we have established a clear understanding of our goal: to predict Air quality analysis using Visualization techniques. We outlined a structured approach that includes the steps to load, preprocess, analyze, and visualize the air quality and we also done the data. Visualization using visualization techniques (e.g., line charts, heatmaps) to effectively represent air quality trends and pollution levels.