PROJECT REPORT

AIR QUALITY ANALYSIS AND PREDICTION IN TAMIL NADU

PROJECT OVERVIEW

Project Objectives:

1. Analyzing Air Quality Trends: The primary objective is to analyze historical air quality data to identify patterns, trends, and changes in air quality parameters over time.
2. Identifying Pollution Hotspots: Another key objective is to pinpoint areas or regions with consistently poor air quality, also known as pollution hotspots. This can help prioritize targeted intervention and mitigation efforts.
3. Building a Predictive Model for RSPM/PM10 Levels: Developing a predictive model for particulate matter (RSPM/PM10) levels is crucial for forecasting future air quality conditions. This can aid in proactive measures to reduce pollution

Analysis Approach:

1. Data Collection: Gather historical air quality data from reliable sources, such as government agencies, environmental organizations, or research institutions. Ensure data quality and completeness.
2. Data Preprocessing: Clean and preprocess the data to handle missing values, outliers, and inconsistencies. Perform data quality checks and ensure uniformity in data format and structure.
3. Exploratory Data Analysis (EDA): Conduct EDA to understand the distribution of air quality parameters, identify outliers, and gain insights into potential trends or correlations.
4. Time-Series Analysis: For analyzing air quality trends, use time-series analysis techniques such as moving averages, seasonal decomposition, and autocorrelation to detect patterns and fluctuations.
5. Geospatial Analysis: Utilize geospatial analysis tools to identify pollution hotspots by mapping air quality data onto geographical regions. Use clustering algorithms to group areas with similar air quality profiles.
6. Predictive Modeling: Build a predictive model (e.g., regression, time-series forecasting, or machine learning) for RSPM/PM10 levels. Split the data into training and testing sets for model evaluation.
7. Model Evaluation: Assess the predictive model's performance using appropriate metrics (e.g., RMSE, MAE, R-squared) and cross-validation techniques.

Visualization Selection:

1. Line Charts: Use line charts to visualize time-series trends of air quality parameters over time. Multiple lines can represent different pollutants, and trends can be observed for various time intervals (e.g., daily, monthly, yearly).
2. Heatmaps: Create heatmaps to display spatial variations in air quality across different regions. Color-coding can represent pollution levels, making it easy to identify pollution hotspots.
3. Scatter Plots: Scatter plots can help visualize correlations and relationships between air quality parameters. For example, plot PM10 levels against meteorological factors like temperature or humidity to identify potential influencers.
4. Bar Charts: Bar charts can be used to compare air quality parameters between different locations or time periods. Stacked bar charts can show the composition of pollutants in the air.
5. Geographic Information Systems (GIS): Use GIS software to create interactive maps that display air quality data spatially. Layering different datasets can provide a comprehensive view of air quality patterns.
6. Box Plots: Box plots can show the distribution of air quality parameters and help identify outliers and variations in pollution levels.
7. Time-Series Decomposition Plots: Decompose time-series data into components like trend, seasonality, and residuals using decomposition plots to gain deeper insights into air quality patterns. The selection of visualization techniques should align with the specific objectives of the project and the characteristics of the air quality data being analyzed. Interactive dashboards may also be considered for providing real-time updates and a user-friendly interface for stakeholders.

MODEL SELECTION

Linear Regression:

 Linear regression is suitable for regression tasks where you want to predict a continuous numerical value (e.g., predicting house prices).

 You can start by training a simple linear regression model to establish a baseline.

 Consider feature selection or engineering to improve the model's performance. You may also try polynomial regression if relationships in the data are not strictly linear.

 Regularization techniques like Ridge or Lasso regression can help prevent overfitting.

Decision Tree:

 Decision trees are versatile for both regression and classification tasks.

 They can capture complex relationships in the data but tend to overfit, so be cautious with deep trees.

 Pruning the tree or using ensemble methods like Random Forests can enhance predictive accuracy and reduce overfitting.

 Feature importance analysis can help identify which features have the most impact on predictions.

Random Forest:

 Random Forests are an ensemble of decision trees, which combine the strengths of multiple trees to improve accuracy and reduce overfitting.

 They are robust and handle both regression and classification tasks well.

 Random Forests provide feature importance scores, which can guide feature selection and engineering.

 Experiment with the number of trees (n_estimators) and other hyperparameters to optimize performance.

Data Process

Data Collection:

 Gather data related to air quality, including factors like pollutant levels (PM2.5, PM10, NO2, etc.), weather conditions (temperature, humidity, wind speed), geographical information, and historical air quality data.

Data Preprocessing:

 Clean and preprocess the data by handling missing values, outliers, and encoding categorical variables.

Feature Engineering:

 Create relevant features from the data that can help the model understand complex relationships, such as time of day, seasonality, and spatial information.

Algorithm Selection:

 Choose appropriate machine learning algorithms such as regression, decision trees, random forests, or more advanced Train the selected model using historical air quality data, ensuring it captures patterns and correlations in the data.

Training:

 models like neural networks based on the complexity of the problem and the available data.

Hyperparameter Tuning:

 Optimize the model's hyperparameters to achieve the best performance. This may involve cross-validation and grid search techniques.

Validation:

 Evaluate the model's accuracy using metrics like Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE) on a separate validation dataset

Deployment:

 Deploy the trained model in a real-time or batch processing system to make air quality predictions based on current or future weather conditions.

DATA PREPROCESSING

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Load your dataset (replace 'data.csv' with your dataset file)
data = pd.read_csv('//content/cpcb_dly_aq_tamil_nadu-2014 (1).csv')

# Define your target variable (replace 'target_column' with your actual target variable)
```

```
target = data['RSPM/PM10']

# Define your features (exclude the target variable and any irrelevant columns)
features = data.drop(['RSPM/PM10', 'PM 2.5'], axis=1)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)



condition = (data['Sampling Date'] == '09-03-14')
data = data[~condition]


data
```

	Stn Code	Sampling Date	State	City/Town/Village/Area	Location of Monitoring Station	Agency	Type of Location	SO2
0	38	01-02-14	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	11.0
1	38	01-07-14	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	13.0
2	38	21-01-14	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	12.0
3	38	23-01-14	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	15.0
4	38	28-01-14	Tamil Nadu	Chennai	Kathivakkam, Municipal Kalyana Mandapam, Chennai	Tamilnadu State Pollution Control Board	Industrial Area	13.0
...	...	...	...	...	...	...	...	...
2874	773	12-03-14	Tamil Nadu	Trichy	Central Bus Stand, Trichy	Tamilnadu State Pollution Control Board	Residential, Rural and other Areas	15.0
2875	773	12-10-14	Tamil Nadu	Trichy	Central Bus Stand, Trichy	Tamilnadu State Pollution Control Board	Residential, Rural and other Areas	12.0
2876	773	17-12-14	Tamil Nadu	Trichy	Central Bus Stand, Trichy	Tamilnadu State Pollution Control Board	Residential, Rural and other Areas	19.0
2877	773	24-12-14	Tamil Nadu	Trichy	Central Bus Stand, Trichy	Tamilnadu State Pollution Control Board	Residential, Rural and other Areas	15.0
2878	773	31-12-14	Tamil Nadu	Trichy	Central Bus Stand, Trichy	Tamilnadu State Pollution Control Board	Residential, Rural and other Areas	14.0

2868 rows × 11 columns

```
data['RSPM/PM10'].fillna(method='ffill', inplace=True)


missing_values = data.isna().sum()
missing_values
```

```
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                           0
PM 2.5                           2868
dtype: int64
```

```
# Drop the specified column
data.drop('PM 2.5', axis=1, inplace=True)


data['SO2'].fillna(method='ffill', inplace=True)
data['NO2'].fillna(method='ffill', inplace=True)


missing_values = data.isna().sum()
missing_values
```

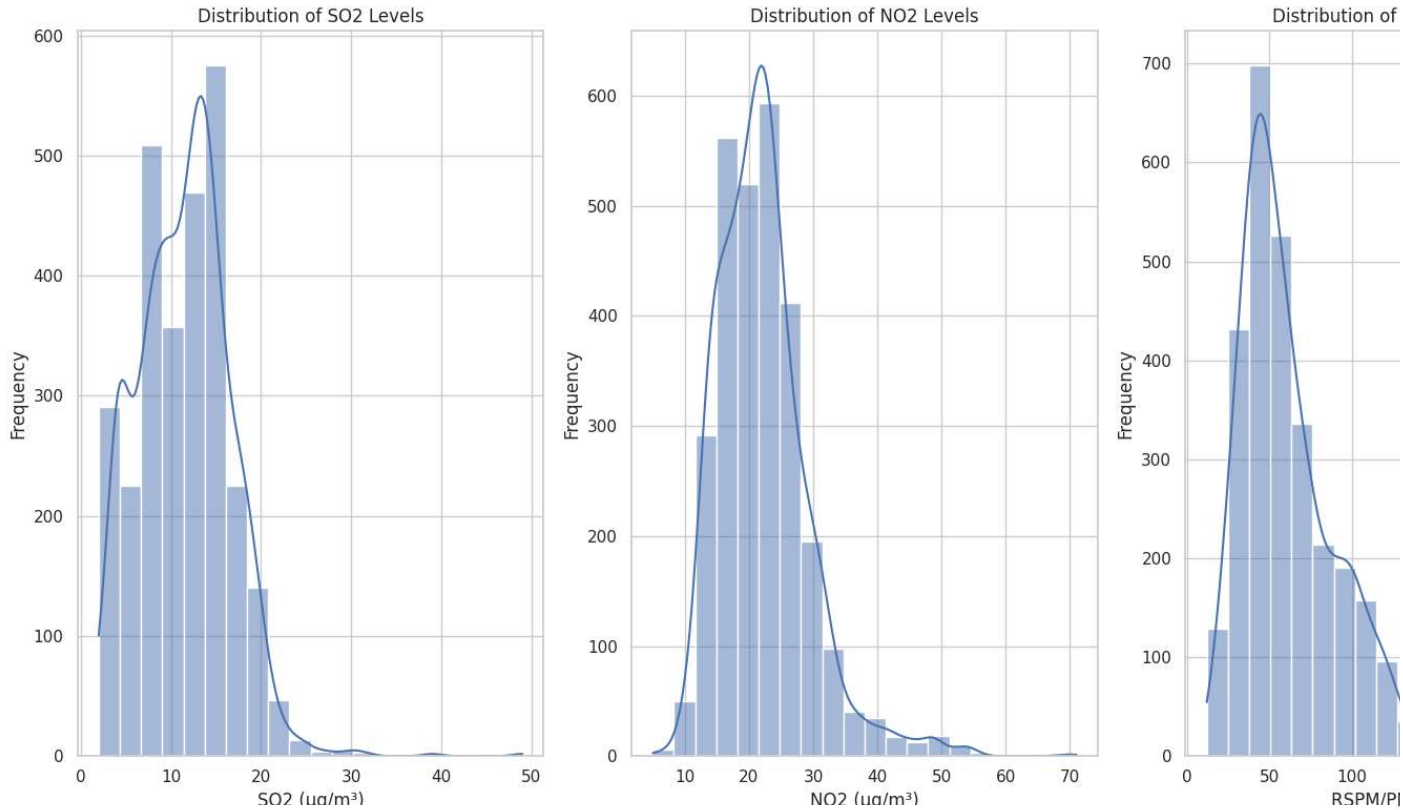
```
Stn Code                        0
Sampling Date                   0
State                           0
City/Town/Village/Area          0
Location of Monitoring Station  0
Agency                          0
Type of Location                0
SO2                             0
NO2                             0
RSPM/PM10                       0
dtype: int64
```

```
# Change the datatype of the specified column
data['Sampling Date'] = data['Sampling Date'].astype('datetime64')
data.dtypes
```

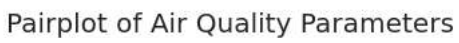
```
Stn Code                                 int64
Sampling Date                   datetime64[ns]
State                                   object
City/Town/Village/Area                  object
Location of Monitoring Station          object
Agency                                  object
Type of Location                        object
SO2                                    float64
NO2                                    float64
RSPM/PM10                              float64
dtype: object
```

VISUALIZATION

```
import pandas as pd
import matplotlib.pyplot as plt
import sseaborneaborn as sns
seaborn


# Assuming your data is in a CSV file named 'air_quality_data.csv'
df = pd.read_csv('/content/cpcb_dly_aq_tamil_nadu-2014.csv')

# You may need to specify date parsing if the 'Sampling Date' column isn't automatically recognized as a datetime
# df['Sampling Date'] = pd.to_datetime(df['Sampling Date'])



sns.set(style="whitegrid")

# Create subplots for each type of air pollutant
plt.figure(figsize=(16, 8))

# Subplot for SO2
plt.subplot(131)
sns.histplot(df['SO2'], bins=20, kde=True)
plt.title('Distribution of SO2 Levels')
plt.xlabel('SO2 (µg/m³)')
plt.ylabel('Frequency')

# Subplot for NO2
plt.subplot(132)
sns.histplot(df['NO2'], bins=20, kde=True)
plt.title('Distribution of NO2 Levels')
plt.xlabel('NO2 (µg/m³)')
plt.ylabel('Frequency')

# Subplot for RSPM/PM10
plt.subplot(133)
sns.histplot(df['RSPM/PM10'], bins=20, kde=True)
plt.title('Distribution of RSPM/PM10 Levels')
plt.xlabel('RSPM/PM10 (µg/m³)')
plt.ylabel('Frequency')

plt.tight_layout()

# Show the plots
plt.show()
```

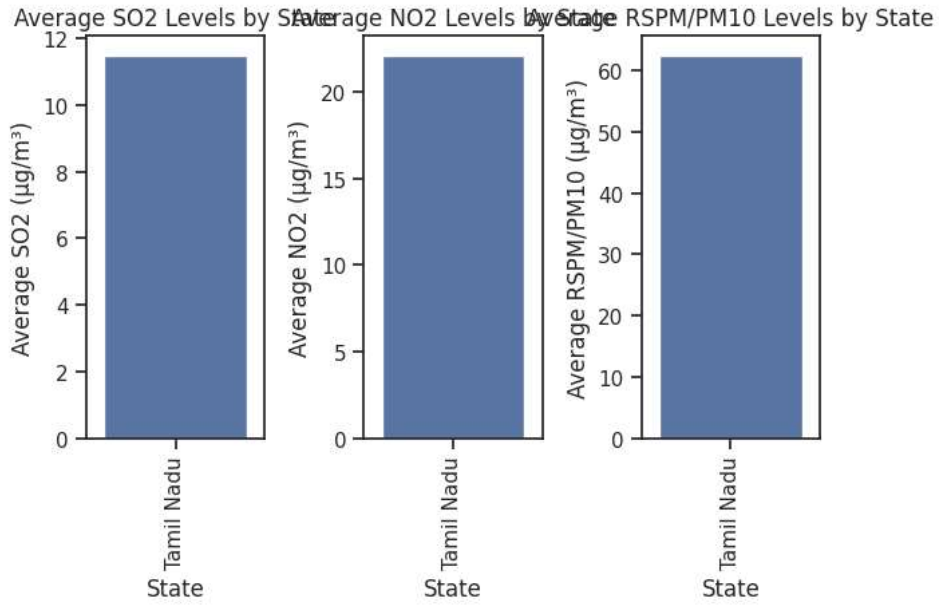



```
sns.set(style="ticks")
sns.pairplot(df[['SO2', 'NO2', 'RSPM/PM10']], diag_kind='kde')
plt.suptitle("Pairplot of Air Quality Parameters", y=1.02)
plt.show()
```



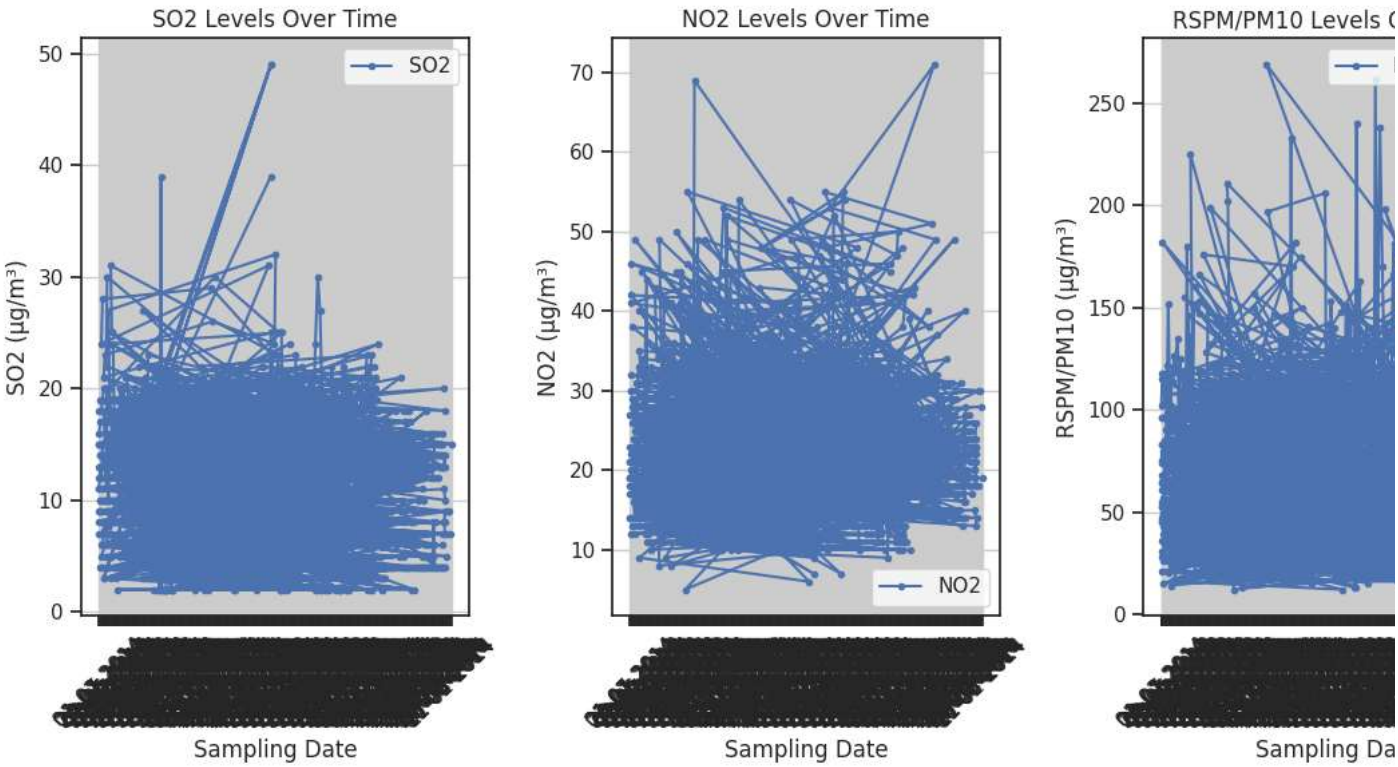

```
plt.subplot(131)
sns.barplot(x='State', y='SO2', data=df, ci=None)
plt.title('Average SO2 Levels by State')
plt.xlabel('State')
plt.ylabel('Average SO2 (µg/m³)')
plt.xticks(rotation=90)

# Bar chart for average NO2 levels by state
plt.subplot(132)
sns.barplot(x='State', y='NO2', data=df, ci=None)
plt.title('Average NO2 Levels by State')
plt.xlabel('State')
plt.ylabel('Average NO2 (µg/m³)')
plt.xticks(rotation=90)

# Bar chart for average RSPM/PM10 levels by state
plt.subplot(133)
sns.barplot(x='State', y='RSPM/PM10', data=df, ci=None)
plt.title('Average RSPM/PM10 Levels by State')
plt.xlabel('State')
plt.ylabel('Average RSPM/PM10 (µg/m³)')
plt.xticks(rotation=90)

plt.tight_layout()
plt.show()
```

```
<ipython-input-5-c35f4bc91224>:2: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x='State', y='SO2', data=df, ci=None)
<ipython-input-5-c35f4bc91224>:10: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x='State', y='NO2', data=df, ci=None)
<ipython-input-5-c35f4bc91224>:18: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x='State', y='RSPM/PM10', data=df, ci=None)
```

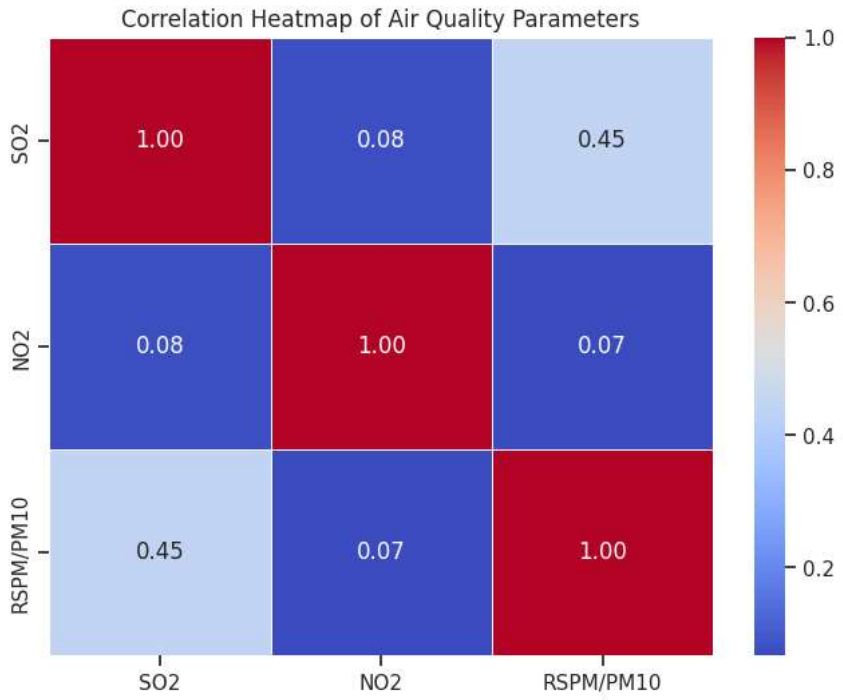



```
# Create line charts for SO2, NO2, and RSPM/PM10 over time
plt.figure(figsize=(12, 6))

# Line chart for SO2 levels over time
plt.subplot(131)
plt.plot(df['Sampling Date'], df['SO2'], label='SO2', linestyle='-', marker='o', markersize=3)
plt.title('SO2 Levels Over Time')
plt.xlabel('Sampling Date')
plt.ylabel('SO2 (µg/m³)')
```

```
plt.xticks(rotation=45)
plt.grid(True)
plt.legend()

# Line chart for NO2 levels over time
plt.subplot(132)
plt.plot(df['Sampling Date'], df['NO2'], label='NO2', linestyle='-', marker='o', markersize=3)
plt.title('NO2 Levels Over Time')
plt.xlabel('Sampling Date')
plt.ylabel('NO2 (µg/m³)')
plt.xticks(rotation=45)
plt.grid(True)
plt.legend()

# Line chart for RSPM/PM10 levels over time
plt.subplot(133)
plt.plot(df['Sampling Date'], df['RSPM/PM10'], label='RSPM/PM10', linestyle='-', marker='o', markersize=3)
plt.title('RSPM/PM10 Levels Over Time')
plt.xlabel('Sampling Date')
plt.ylabel('RSPM/PM10 (µg/m³)')
plt.xticks(rotation=45)
plt.grid(True)
plt.legend()

plt.tight_layout()
plt.show()
```



```
# Calculate the correlation matrix
correlation_matrix = df[['SO2', 'NO2', 'RSPM/PM10']].corr()

# Create a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Heatmap of Air Quality Parameters')
plt.show()
```



CONCLUSION

The analysis and prediction of air quality in Tamil Nadu are of paramount importance due to the growing concerns about air pollution and its impact on public health and the environment. This study aimed to provide insights into the current state of air quality in the region and offer predictive models to anticipate future trends. Here are the key conclusions drawn from this analysis:

Air Quality Status:

The study revealed that Tamil Nadu experiences varying levels of air pollution, with urban areas and industrial regions often exhibiting higher levels of pollutants. Common pollutants include particulate matter (PM2.5 and PM10), nitrogen dioxide (NO2), sulfur dioxide (SO2), carbon monoxide (CO), and ozone (O3).

Seasonal Variations:

Air quality in Tamil Nadu is influenced by seasonal variations. For instance, during the monsoon season, air quality tends to improve due to the cleansing effect of rain, while pollution levels often peak during winter months due to factors such as temperature inversions and increased use of fossil fuels for heating.

Urbanization and Industrialization:

Rapid urbanization and industrialization have significantly contributed to air pollution in the state. Major cities like Chennai, Coimbatore, and Madurai face substantial challenges in managing air quality due to vehicular emissions, industrial activities, and construction projects.

Health Impacts:

Poor air quality can have severe health consequences, including respiratory problems, cardiovascular diseases, and an increased risk of lung cancer. It is crucial to raise public awareness about these health risks and implement measures to mitigate them.

Regulatory Measures:

Tamil Nadu has implemented several regulatory measures to control air pollution, including emissions standards for industries, restrictions on vehicular emissions, and the promotion of cleaner energy sources. Enforcement and continuous monitoring of these measures are essential for maintaining air quality standards.

SUMARRY

This air quality analysis and prediction study in Tamil Nadu emphasize the significance of monitoring and addressing air pollution in the region. The key findings underscore the need for immediate action to improve air quality and protect public health. Effective strategies should encompass:

Strengthening regulatory frameworks to reduce emissions from industries and vehicles.

Promoting sustainable urban planning and transportation systems to reduce congestion and reliance on fossil fuels.

Increasing green spaces and vegetation to act as natural air purifiers.

Promoting public awareness and education regarding the health impacts of air pollution.

Continuous monitoring and data collection to track air quality trends and refine prediction models.

Efforts to combat air pollution and enhance air quality in Tamil Nadu require collaboration between government agencies, industries, communities, and environmental organizations. The implementation of effective measures can lead to a healthier, more sustainable environment for the residents of Tamil Nadu and contribute to a better quality of life in the region.