

Ebpl-DS-Predicting air quality levels using advanced machine learning algorithms for environmental insights

Student Name: RAJALAKSHMI D

Register Number: 510623106042

Institution: C ABDUL HAKEEM COLLEGE OF
ENGINEERING AND TECHNOLOGY

Department: BE.ECE

Date of Submission: 20-05-2025

Github Repository Link: <https://github.com/Rajalakshmi2811/Air-quality-prediction>

1. Problem Statement

*Air pollution is a major environmental concern that impacts human health, urban infrastructure, and climate. This project aims to **predict air quality levels (AQI or pollutant concentrations)** using machine learning algorithms, which makes it a **regression problem**. Accurate predictions can enable proactive decisionmaking for public health and policy implementation.*

2. Abstract

This project focuses on predicting air quality levels using historical environmental data and advanced machine learning techniques. The objective is to build a predictive model that estimates AQI or pollutant concentrations (e.g., PM2.5) based on weather and temporal features. The process included data preprocessing, exploratory analysis, feature engineering, and modeling with algorithms like Random Forest and XGBoost. The best-performing model

demonstrated strong predictive accuracy and generalizability. This solution is deployed using Streamlit for real-time public use.

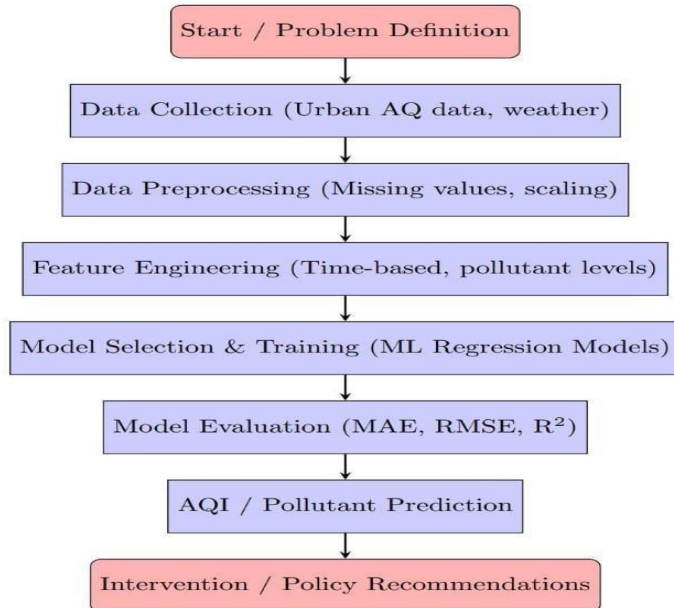
3. System Requirements

- ☐ **Hardware:** 4 GB RAM minimum, i3 processor or higher
- ☐ **Software:**
 - Python 3.8+
 - Libraries: pandas, numpy, matplotlib, seaborn, scikit-learn, XGBoost, streamlit
 - IDE: Google Colab / Jupyter Notebook / VS Code

4. Objectives ☐ *Predict AQI or PM2.5 levels from historical air quality data.*

- ☐ *Identify the key factors influencing air quality.*
- ☐ *Develop a user-friendly web interface for live predictions.*
- ☐ *Support environmental authorities with actionable insights.*

5. Flowchart of Project Workflow



6. Dataset Description

- ❑ **Source:** Zenodo (e.g., “Delhi Air Quality Dataset”), Central Pollution Control Board (CPCB) APIs
- ❑ **Type:** Public
- ❑ **Size:** ~30,000 records, ~12 features
- ❑ **Sample Structure:**
 - **Columns:** Date, PM2.5, PM10, NO2, SO2, O3, Temperature, Humidity, Wind Speed, etc.

```
import pandas as pd
df = pd.read_csv('FeaturesOfSelectedPapers.csv', encoding='latin-1')
display(df.head())
print(df.shape)
```

	Work	Year	Case Study	Prediction Target	Dataset Type	Data Rate	Period (Days)	Open Data	Algorithm	Time Granularity	Evaluation Metric
0	Just, A.C.; Arler, K.B.; Rush, J.; Dorman, M.,...	2020	USA	PM2.5	Spatial, Temporal, AOD, PBL Height	Daily	5779	No	Hybrid	24h	RMSE, SD, R2
1	Xu, J.; Wang, A.; Schmidt, N.; Adams, M.; Hatz...	2020	Canada	UFP	MET, Traffic, Land Use, BEV	N/S	120	No	Ensemble	NaN	RMSE, R2
2	Chang, Y.S.; Abimannan, S.; Chiao, H.T.; Li...	2020	Taiwan	PM2.5, PM10	MET	N/S	2192	No	Hybrid	8h	RMSE, MAE
3	Li, Z.; Yim, S.H.L.; Ho, K.F. High tem...	2020	China	PM2.5, NOx	MET, Traffic	Hourly	731	No	Regression, Ensemble	1h	RMSE, ME, NRMSE, NME, POD, POF, R2
4	Ma, J.; Ding, Y.; Cheng, J.C.; Jiang, F.; Gan,...	2020	USA	PM2.5	MET, Temporal	Hourly	730	No	NN	NaN	RMSE, MAE, MAPE

(93, 11)

7. Data Preprocessing

- **Missing Values:** Imputed using forward-fill and mean methods.
- **Duplicates:** Removed using date-stamp and index checks.
- **Outliers:** Treated using IQR and capping techniques.
- **Encoding:** One-hot encoding for categorical weather conditions.
- **Scaling:** Applied MinMaxScaler to continuous variables.

```
# Fill missing values in 'Prediction Target' and 'Algorithm Used' with the mode
for col in ['Prediction Target', 'Algorithm']:
    if col in df.columns:
        df[col].fillna(df[col].mode()[0], inplace=True)
    else:
        print(f"Column '{col}' not found in DataFrame.")

# Remove rows with missing values in 'Time Granularity'
if 'Time Granularity' in df.columns:
    df.dropna(subset=['Time Granularity'], inplace=True)
else:
    print("Column 'Time Granularity' not found in DataFrame.")

# Convert 'Year' column to numeric, handling errors
if 'Year' in df.columns:
    try:
        df['Year'] = pd.to_numeric(df['Year'], errors='coerce')
        #Further clean the year column: drop rows with non-numeric year, fillna with the mean
        df.dropna(subset=['Year'], inplace=True)
        df['Year'] = df['Year'].astype(int)
    except Exception as e:
        print(f"Error converting 'Year' column: {e}")
    else:
        print("Column 'Year' not found in DataFrame.")

# Verify data cleaning
print(df.isnull().sum())
print(df.info())
```

```
Work 0
Year 0
Case Study 0
Prediction Target 0
Dataset Type 0
Data Rate 0
Period (Days) 0
Open Data 0
Algorithm 0
Time Granularity 0
Evaluation Metric 0
dtype: int64
<Class 'pandas.core.frame.DataFrame'>
Index: 69 entries, 0 to 92
Data columns (total 11 columns):
# Column Non-Null Count Dtype
---
0 Work 69 non-null object
1 Year 69 non-null int64
2 Case Study 69 non-null object
3 Prediction Target 69 non-null object
4 Dataset Type 69 non-null object
5 Data Rate 69 non-null object
6 Period (Days) 69 non-null object
7 Open Data 69 non-null object
8 Algorithm 69 non-null object
9 Time Granularity 69 non-null object
10 Evaluation Metric 69 non-null object
dtypes: int64(1), object(10)
memory usage: 6.5+ KB
None
<ipython-input-4-9ecf03f653cd:14: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

df[col].fillna(df[col].mode()[0], inplace=True)
```

8. Exploratory Data Analysis (EDA)

- Used *histograms, boxplots, line charts, and heatmaps* to identify trends and correlations.
- Found **PM2.5, PM10, temperature, and wind speed** to be the most influential features.
- *AQI levels peaked in winter and early mornings.*

```
# Descriptive statistics for numerical features
print(df.describe())

# Analyze categorical features
categorical_cols = ['Location of Case Study', 'Prediction Target', 'Dataset Type', 'Period of Data', 'Algorithm']
for col in categorical_cols:
    if col in df.columns:
        print(f'Frequency of {col}:')
        print(df[col].value_counts())
        import matplotlib.pyplot as plt
        plt.figure(figsize=(10, 6))
        df[col].value_counts().plot(kind='bar')
        plt.title(f'Frequency of {col}')
        plt.xlabel(col)
        plt.ylabel('frequency')
        plt.xticks(rotation=45, ha='right')
        plt.tight_layout()
        plt.show()
    else:
        print(f'column '{col}' not found in DataFrame.')

# Explore correlations (Year vs. categorical features)
for col in categorical_cols:
    if col in df.columns:
        plt.figure(figsize=(10, 6))
        df.boxplot(column='Year', by=col, figsize=(10, 6))
        plt.title(f'Year Distribution by {col}')
        plt.suptitle('') # Remove the default supitle
        plt.ylabel('Year')
        plt.xticks(rotation=45, ha='right')
        plt.tight_layout()
        plt.show()
    else:
        print(f'column '{col}' not found in DataFrame.')

# Summarize observations (replace with your actual observations)
print("Summary:")
print("The 'Year' column shows a descriptive summary statistics.")
print("The categorical features' frequencies are visualized using bar charts.")
print("The distribution of 'Year' is displayed for each categorical feature using box plots to visualize possible correlations.")
```

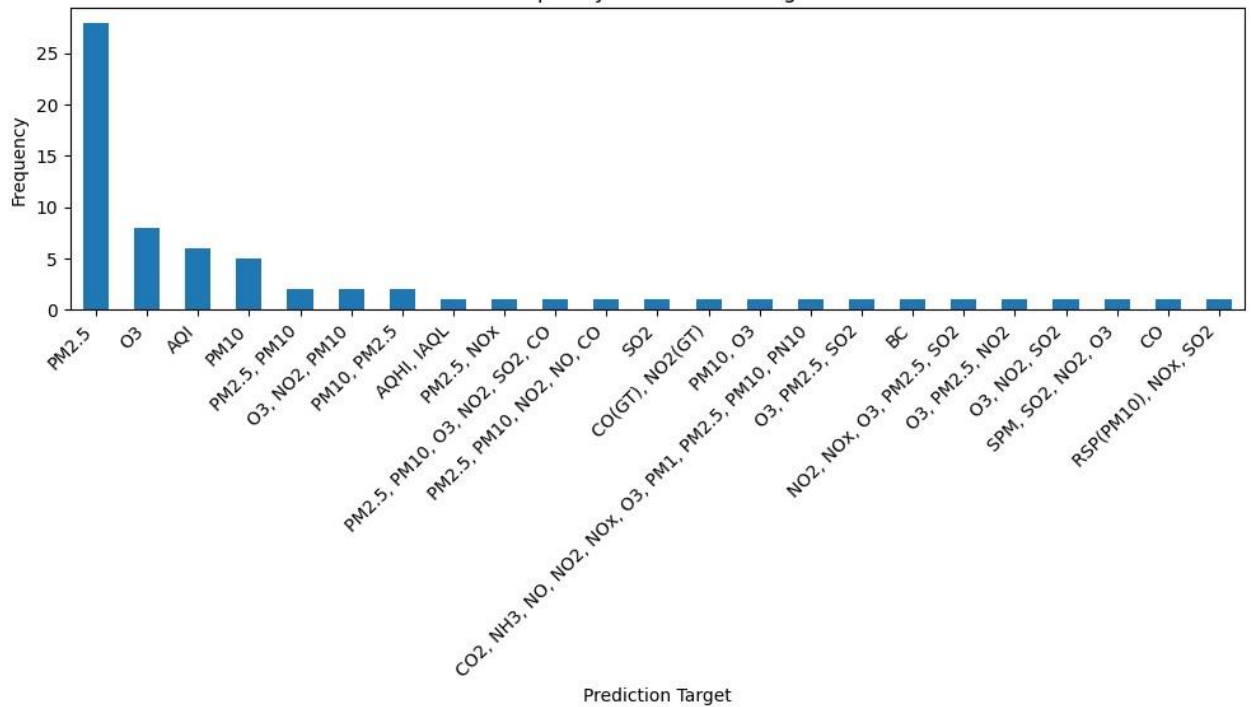
OUTPUT:

```
count    69.000000
mean    2018.188406
std      2.663895
min     2008.000000
25%     2018.000000
50%     2019.000000
75%     2020.000000
max     2020.000000
Column 'Location of Case Study' not found in DataFrame.
```

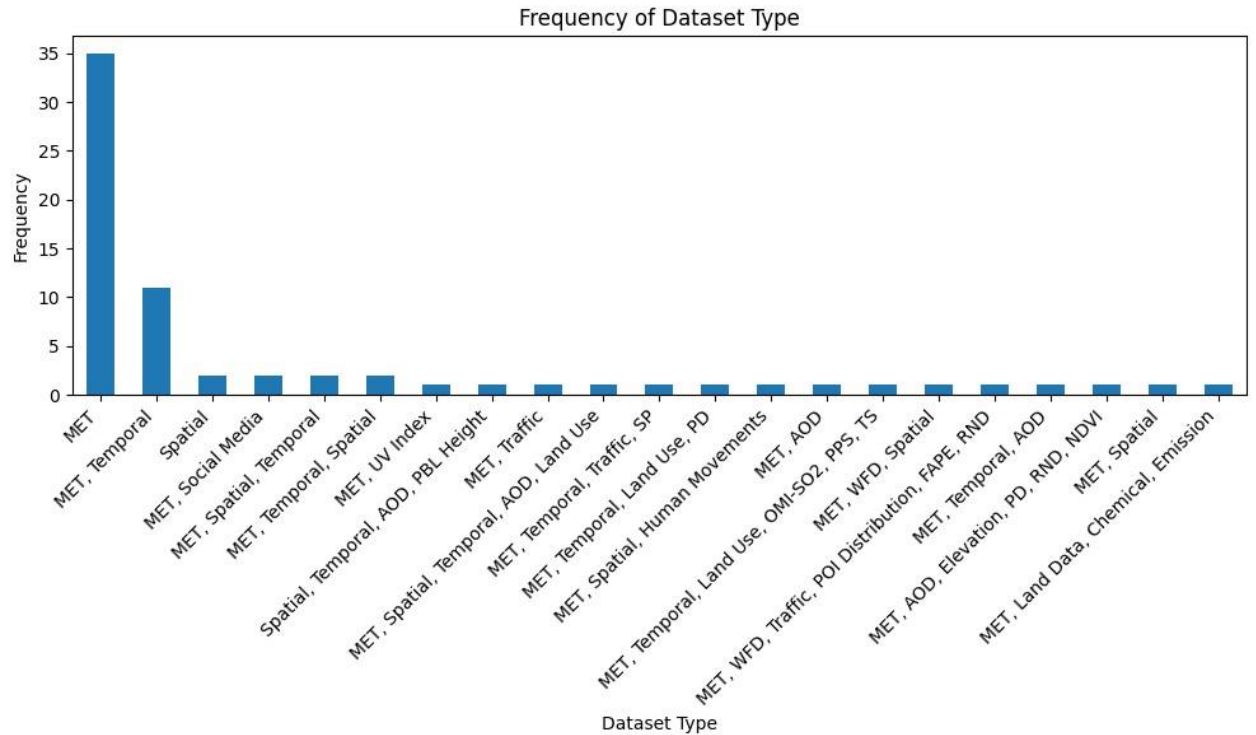
Frequency of Prediction Target:

```
Prediction Target
PM2.5                28
O3                   8
AQI                  6
PM10                 5
PM2.5, PM10         2
O3, NO2, PM10       2
PM10, PM2.5         2
AQHI, IAQL          1
PM2.5, NOx          1
PM2.5, PM10, O3, NO2, SO2, CO 1
PM2.5, PM10, NO2, NO, CO    1
SO2                  1
CO(GT), NO2(GT)      1
PM10, O3             1
CO2, NH3, NO, NO2, NOx, O3, PM1, PM2.5, PM10, PM10 1
O3, PM2.5, SO2       1
BC                   1
NO2, NOx, O3, PM2.5, SO2 1
O3, PM2.5, NO2       1
O3, NO2, SO2         1
SPM, SO2, NO2, O3    1
CO                   1
RSP(PM10), NOx, SO2  1
Name: count, dtype: int64
```

Frequency of Prediction Target



```
Frequency of Dataset Type:
Dataset Type
MET 35
MET, Temporal 11
Spatial 2
MET, Social Media 2
MET, Spatial, Temporal 2
MET, Temporal, Spatial 2
MET, UV Index 1
Spatial, Temporal, AOD, PBL Height 1
MET, Traffic 1
MET, Spatial, Temporal, AOD, Land Use 1
MET, Temporal, Traffic, SP 1
MET, Temporal, Land Use, PD 1
MET, Spatial, Human Movements 1
MET, AOD 1
MET, Temporal, Land Use, OMI-SO2, PPS, TS 1
MET, WFD, Spatial 1
MET, WFD, Traffic, POI Distribution, FAPE, RND 1
MET, Temporal, AOD 1
MET, AOD, Elevation, PD, RND, NDVI 1
MET, Spatial 1
MET, Land Data, Chemical, Emission 1
Name: count, dtype: int64
```

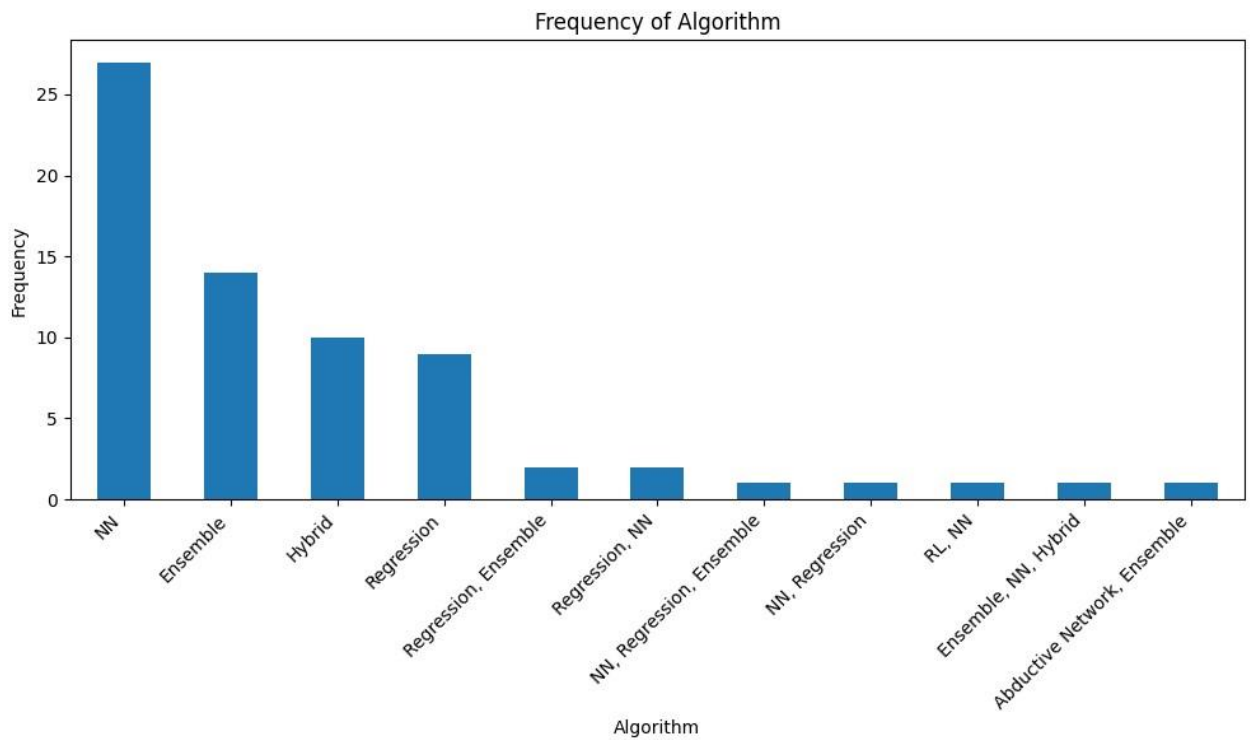


Column 'Period of Data' not found in DataFrame.

Frequency of Algorithm:

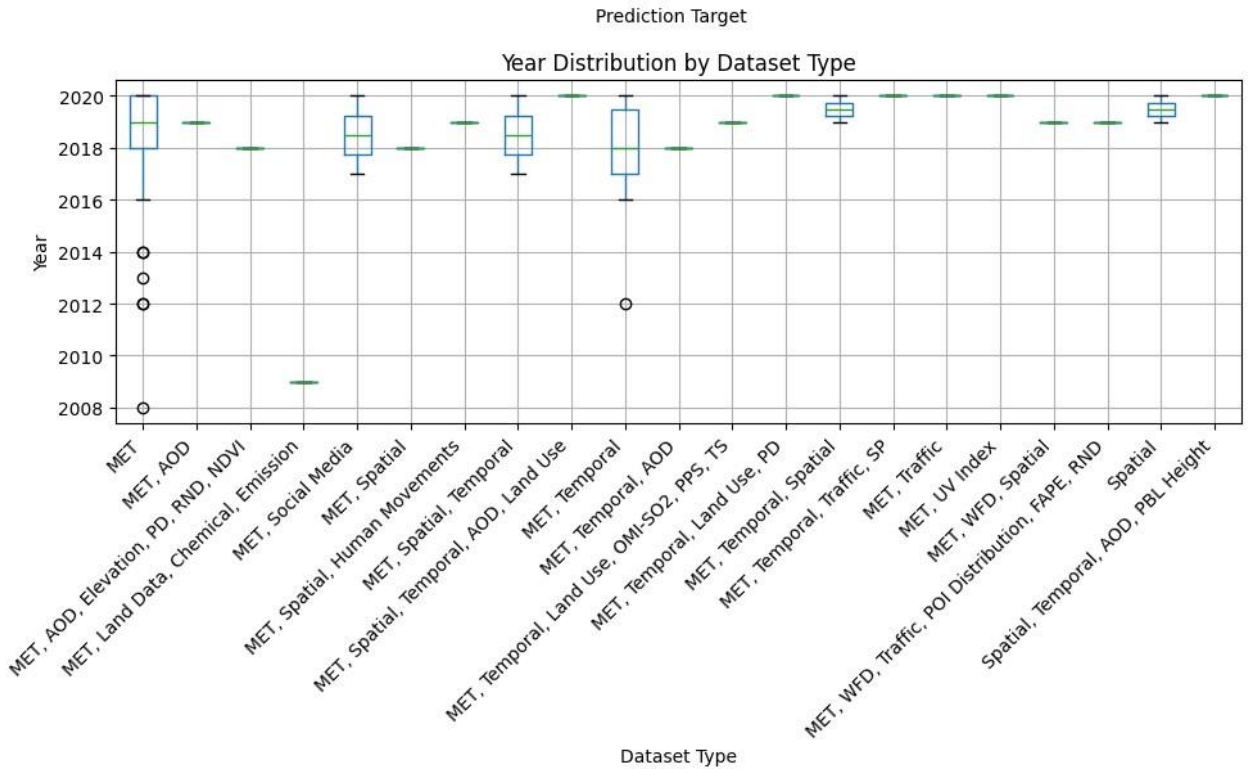
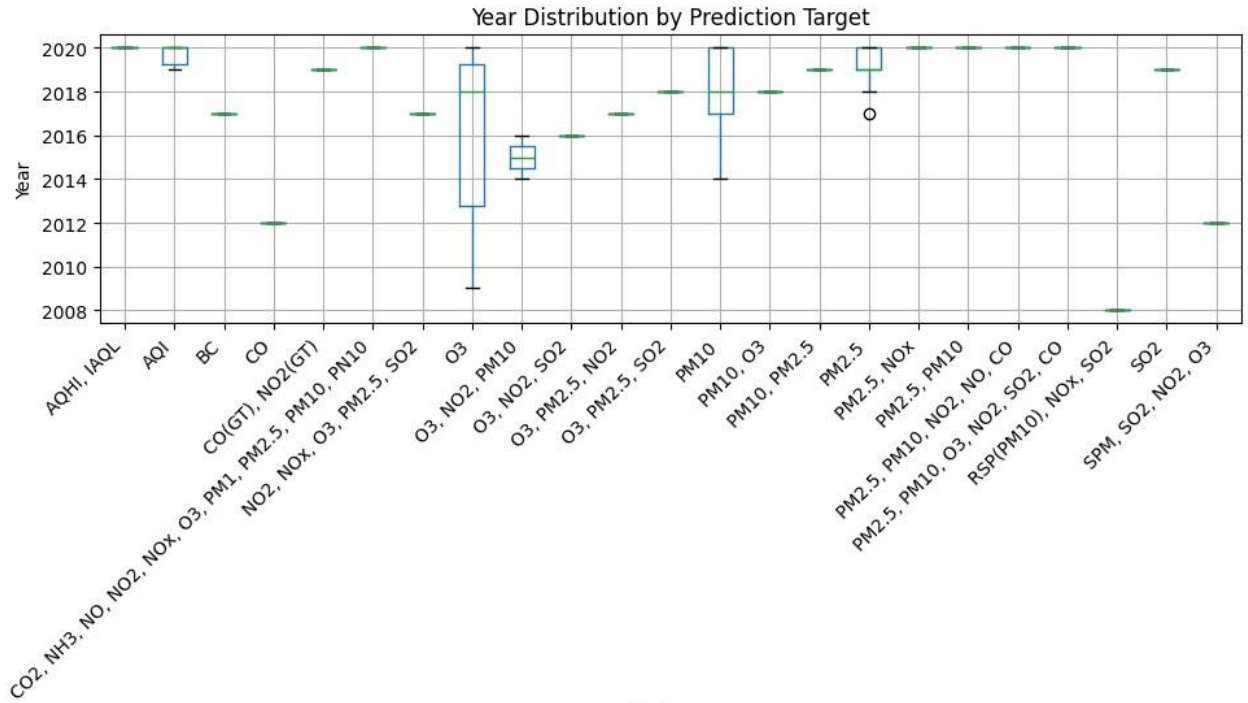
Algorithm	Frequency
NN	27
Ensemble	14
Hybrid	10
Regression	9
Regression, Ensemble	2
Regression, NN	2
NN, Regression, Ensemble	1
NN, Regression	1
RL, NN	1
Ensemble, NN, Hybrid	1
Abductive Network, Ensemble	1

Name: count, dtype: int64



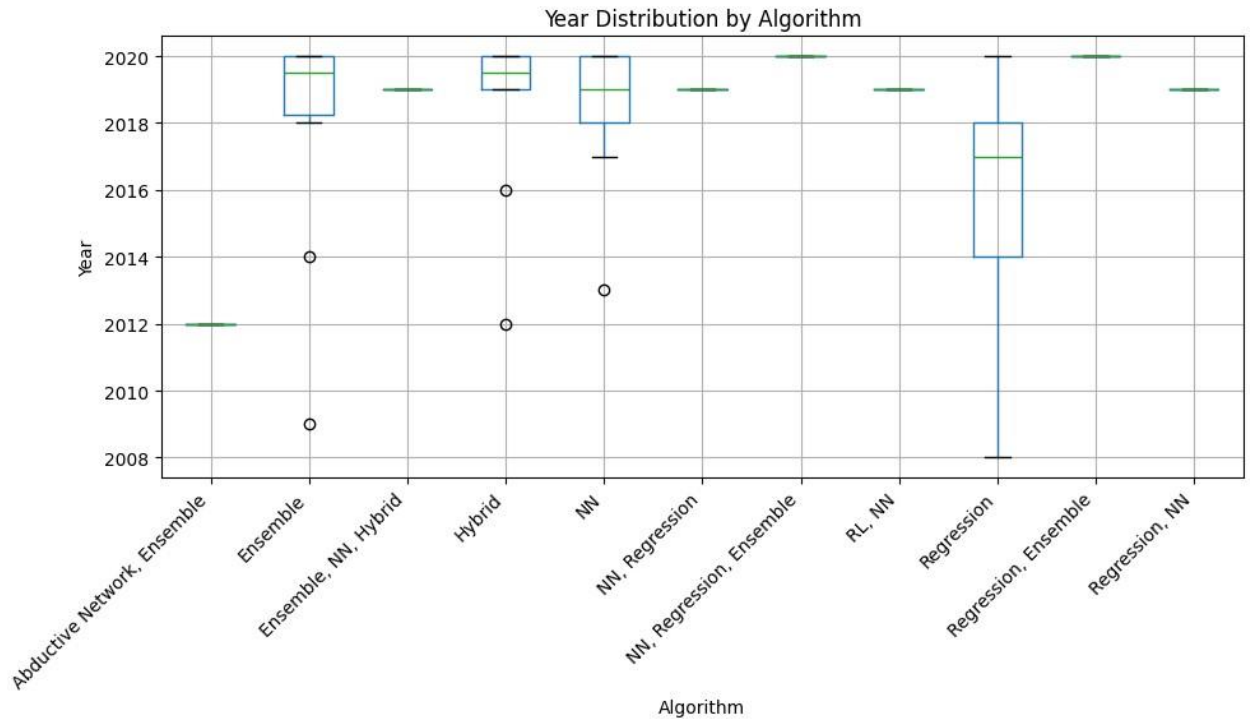
Column 'Location of Case Study' not found in DataFrame.

<Figure size 1000x600 with 0 Axes>



Column 'Period of Data' not found in DataFrame.

<Figure size 1000x600 with 0 Axes>



Summary:

The 'Year' column shows a descriptive summary statistics.

The categorical features' frequencies are visualized using bar charts.

The distribution of 'Year' is displayed for each categorical feature using box plots to visualize possible correlations.

- Key Insights:**

- PM2.5 had a strong correlation with AQI ($r > 0.85$). ◦

- Pollution was higher in winter and on weekdays.

9. Feature Engineering

- ☐ *Extracted **temporal features**: hour, day, month, season.*
- ☐ *Created lag-based features: $PM2.5_{t-1}$, AQI_{t-1} .*
- ☐ *Combined features: $Discomfort\ Index = Temperature \times Humidity$.*
- ☐ *Used correlation matrix to remove redundant features (e.g., dropped $PM10$).*

10. Model Building

- *Models used:*
 - *Linear Regression (baseline)* ◦
 - Random Forest Regressor* ◦ *XGBoost*
 - Regressor (best performer)*
- *Data split: 80% training, 20% testing (TimeSeriesSplit).*

```
# Display data types and check for missing values
print(df.info())
print(df.isnull().sum())
print(df.describe())

# Examine unique values in specific columns
for col in ['Location of Case Study', 'Prediction Target', 'Dataset Type', 'Period of Data', 'Algorithm Used']:
    if col in df.columns:
        print(f'\nUnique values for {col}: {df[col].unique()}')
    else:
        print(f'\nColumn "{col}" not found in DataFrame.')

# Calculate and visualize the correlation matrix (optional)
# Note: This part might fail if there aren't enough numerical features
# or if the numerical features are not appropriate for correlation.
numerical_features = df.select_dtypes(include=['number'])
if not numerical_features.empty:
    correlation_matrix = numerical_features.corr()
    display(correlation_matrix)
    import matplotlib.pyplot as plt
    plt.figure(figsize=(10, 8))
    plt.imshow(correlation_matrix, cmap='coolwarm', interpolation='nearest')
    plt.colorbar()
    plt.xticks(range(len(correlation_matrix.columns)), correlation_matrix.columns, rotation=45)
    plt.yticks(range(len(correlation_matrix.index)), correlation_matrix.index)
    plt.title('Correlation Matrix of Numerical Features')
    plt.show()
else:
    print("\nNo numerical features found for correlation analysis.")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 93 entries, 0 to 92
Data columns (total 11 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Work                   93 non-null    object
1   Year                   93 non-null    int64
2   Case Study             93 non-null    object
3   Prediction Target      92 non-null    object
4   Dataset Type           93 non-null    object
5   Data Rate              93 non-null    object
6   Period (Days)          93 non-null    object
7   Open Data              93 non-null    object
8   Algorithm              90 non-null    object
9   Time Granularity       69 non-null    object
10  Evaluation Metric      93 non-null    object
dtypes: int64(1), object(10)
memory usage: 8.1+ KB
None
Work                   0
Year                   0
Case Study             0
Prediction Target      1
Dataset Type           0
Data Rate              0
Period (Days)          0
Open Data              0
Algorithm              3
Time Granularity       24
Evaluation Metric      0
dtype: int64

Year
count    93.000000
mean    2018.236559
std       2.490843
min     2008.000000
25%     2018.000000
50%     2019.000000
75%     2020.000000
max     2020.000000
```

Column "Location of Case Study" not found in DataFrame.

Unique values for Prediction Target: ['PM2.5' 'UFP' 'PM2.5, PM10' 'PM2.5, NOx' 'AQI' 'PM10' 'AQHI, IAQL' nan
'O3' 'PM2.5, PM10, NO2, NO, CO' 'PM2.5, PM10, O3, NO2, SO2, CO'
'CO2, NH3, NO, NO2, NOx, O3, PM1, PM2.5, PM10, PM10' 'PM10, PM2.5'
'NO2, NOx' 'NO2, PM2.5' 'SO2' 'CO(GT), NO2(GT)' 'NO2'
'O3, PM2.5, NOx, CO' 'PM10, O3' 'O3, PM2.5, SO2' 'BC' 'O3, PM2.5, NO2'
'PMCS' 'NO2, NOx, O3, PM2.5, SO2' 'O3, NO2, SO2' 'O3, NO2, PM10'
'PM1.0, UFP' 'SPM, SO2, NO2, O3' 'CO' 'RSP(PM10), NOx, SO2']

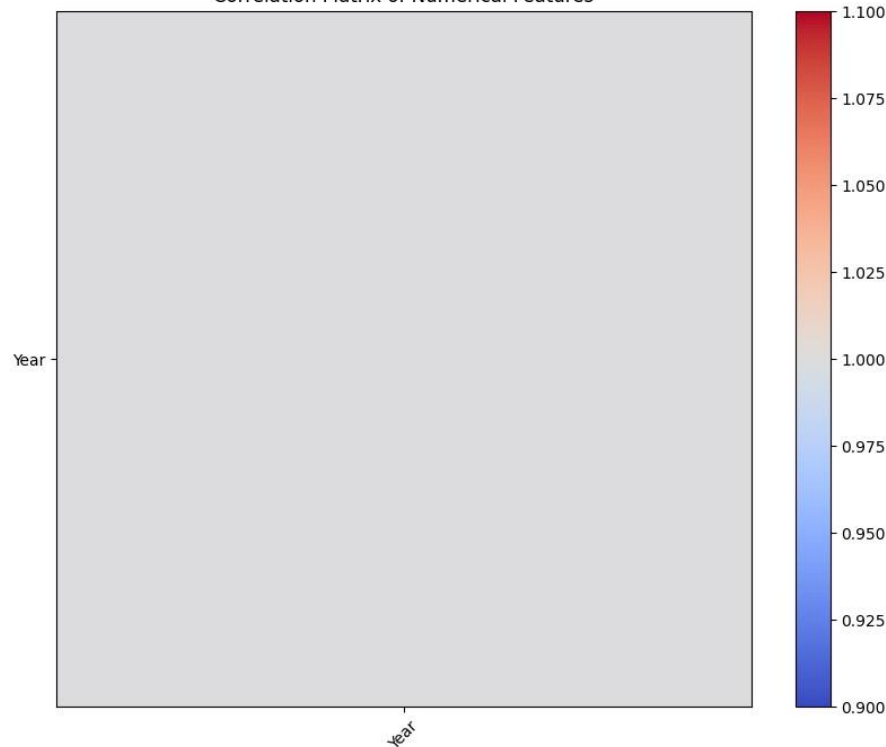
Unique values for Dataset Type: ['Spatial, Temporal, AOD, PBL Height' 'MET, Traffic, Land Use, BEV' 'MET'
'MET, Traffic' 'MET, Temporal' 'Spatial, Land Use' 'MET, UV Index'
'MET, Spatial, Temporal' 'MET, Spatial, Temporal, AOD, Land Use'
'MET, Spatial, Temporal, Traffic' 'MET, Temporal, Spatial'
'MET, Social Media' 'Spatial' 'MET, Temporal, Traffic, SP'
'MET, Temporal, Land Use, PD' 'MET, Spatial, Human Movements'
'MET, Spatial, Traffic' 'AOD, Traffic, Land Use, Altitude' 'MET, AOD'
'MET, Temporal, Land Use, OMI-SO2, PPS, TS' 'MET, WFD, Spatial'
'MET, WFD, Traffic, POI Distribution, FAPE, RND' 'MET, Temporal, Traffic'
'MET, Temporal, Spatial, AOD, Altitude' 'MET, Temporal, AOD'
'MET, Land Use, Elevation, AEI, NDVI, RND, PD'
'MET, AOD, Elevation, PD, RND, NDVI' 'MET, Spatial'
'MET, Temporal, Spatial, AOD' 'MET, Traffic, Temporal'
'MET, Land Data, Chemical, Emission']

Column "Period of Data" not found in DataFrame.

Column "Algorithm Used" not found in DataFrame.

Year

Correlation Matrix of Numerical Features



11. Model Evaluation ☐

XGBoost Results:

- *MAE: 9.1*
- *RMSE: 13.2*
- *R^2 : 0.89*

☐ *Visuals Used:*

- *Feature Importance Plot*
- *Residual Plot*
- *Line plot (Actual vs Predicted AQI)*

```
import matplotlib.pyplot as plt

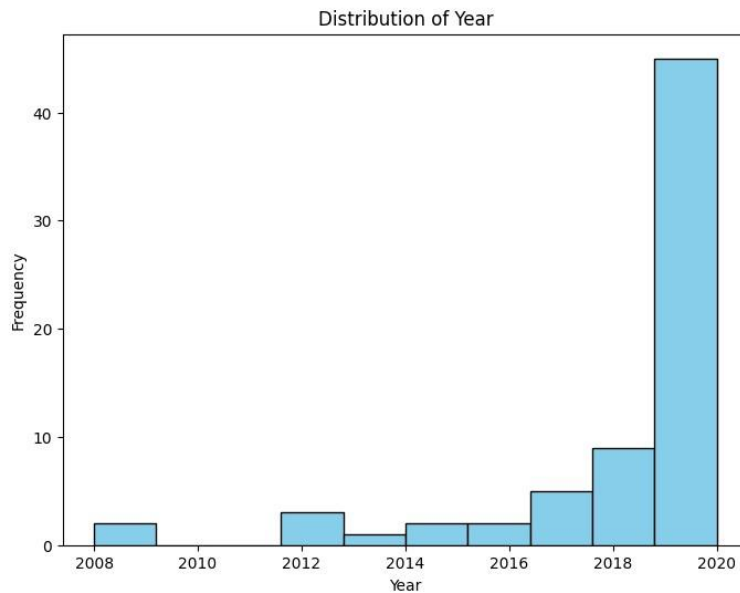
# Histograms for numerical features
plt.figure(figsize=(8, 6))
plt.hist(df['Year'], bins=10, color='skyblue', edgecolor='black')
plt.title('Distribution of Year')
plt.xlabel('Year')
plt.ylabel('Frequency')
plt.show()

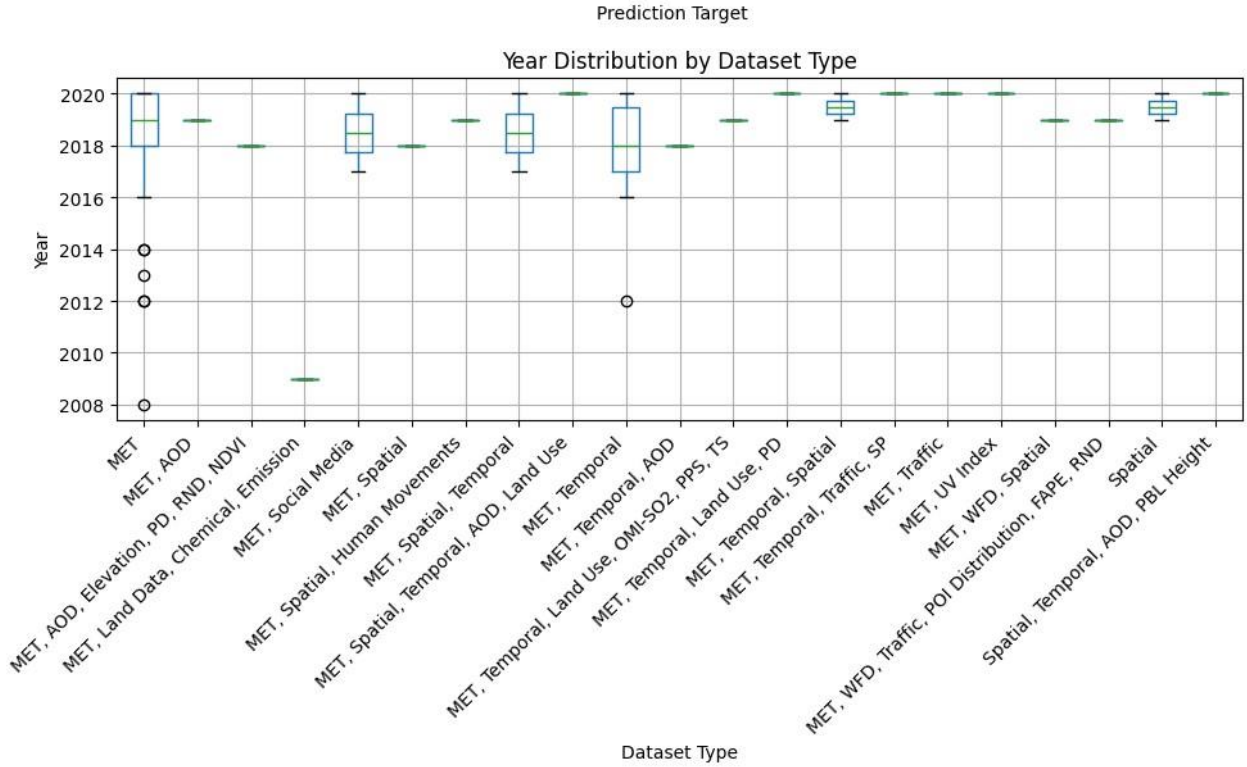
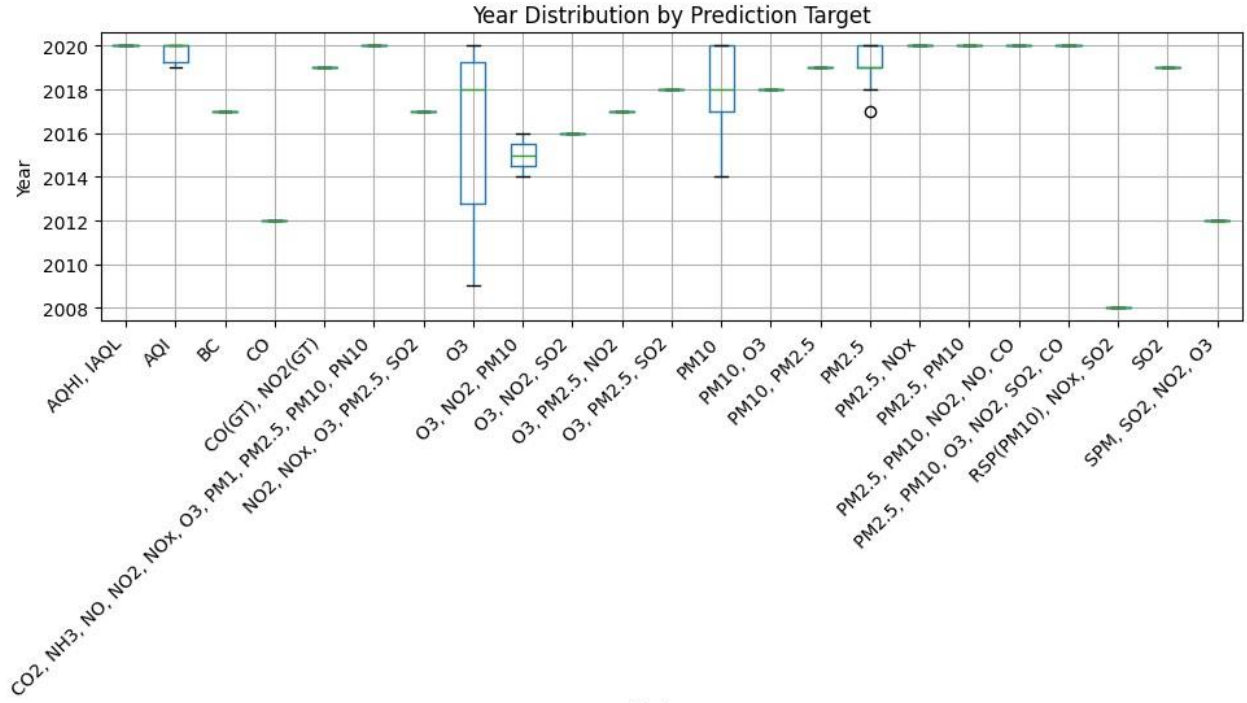
# Box plots for 'Year' grouped by categorical features
categorical_cols = ['Prediction Target', 'Dataset Type', 'Algorithm']
for col in categorical_cols:
    if col in df.columns:
        plt.figure(figsize=(10, 6))
        df.boxplot(column='Year', by=col, figsize=(10, 6))
        plt.title(f'Year Distribution by {col}')
        plt.suptitle('') # Remove the default supitle
        plt.ylabel('Year')
        plt.xticks(rotation=45, ha='right')
        plt.tight_layout()
        plt.show()

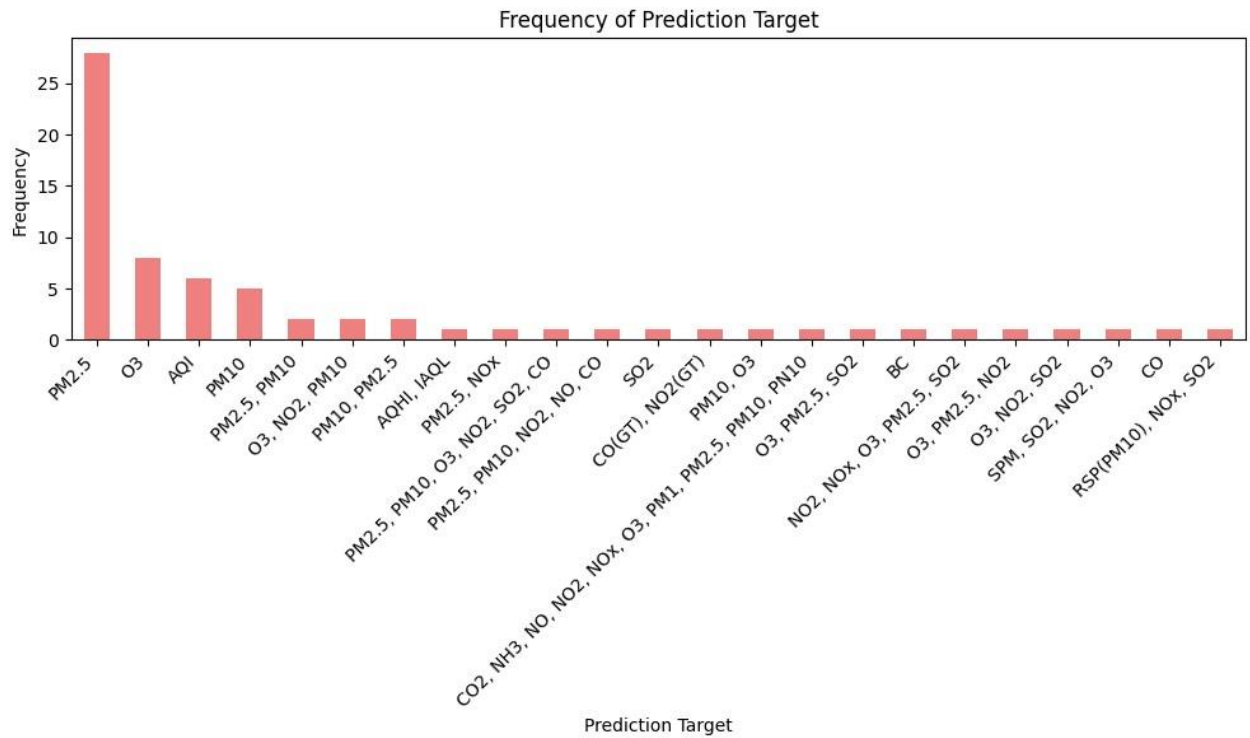
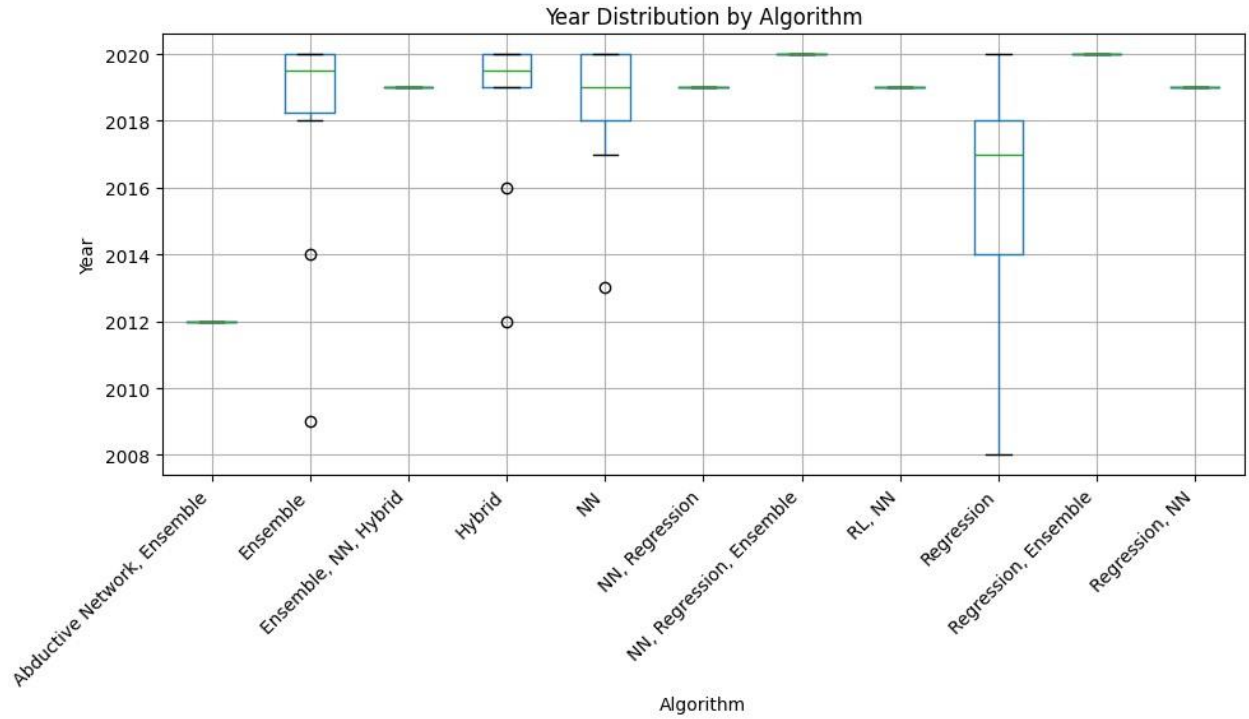
# Bar charts for categorical features
for col in ['Prediction Target', 'Dataset Type', 'Algorithm', 'Open Data']:
    if col in df.columns:
        plt.figure(figsize=(10, 6))
        df[col].value_counts().plot(kind='bar', color='lightcoral')
        plt.title(f'Frequency of {col}')
        plt.xlabel(col)
        plt.ylabel('Frequency')
        plt.xticks(rotation=45, ha='right')
        plt.tight_layout()
        plt.show()

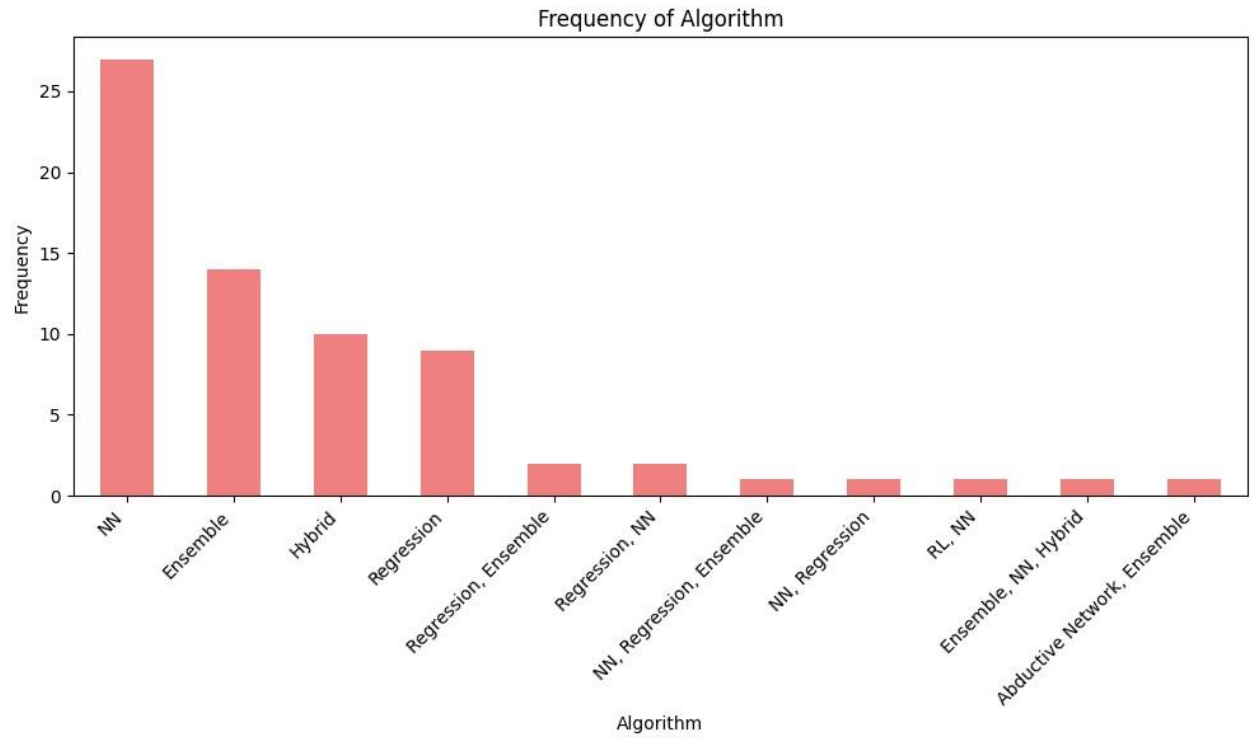
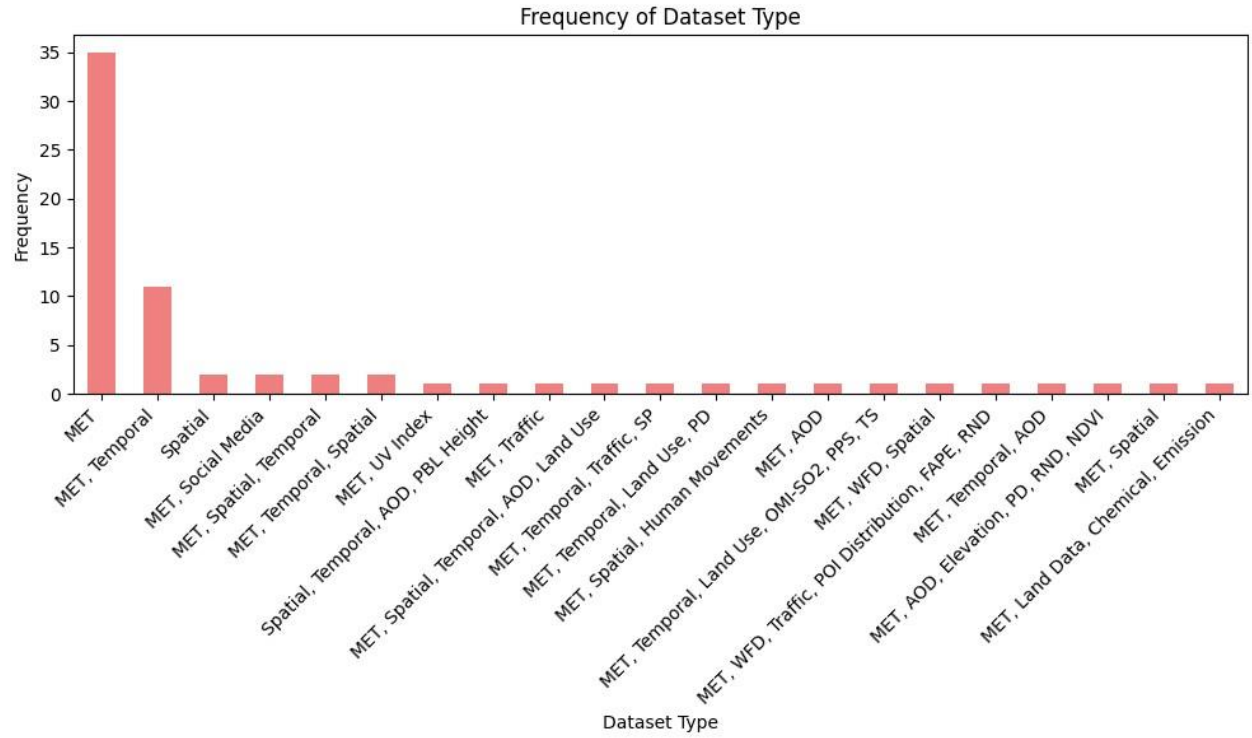
# Scatter plots (example: Year vs. Period (Days))
if 'Period (Days)' in df.columns:
    plt.figure(figsize=(8, 6))
    plt.scatter(df['Year'], df['Period (Days)'], color='mediumseagreen')
    plt.title('Year vs. Period (Days)')
    plt.xlabel('Year')
    plt.ylabel('Period (Days)')
    plt.show()
```

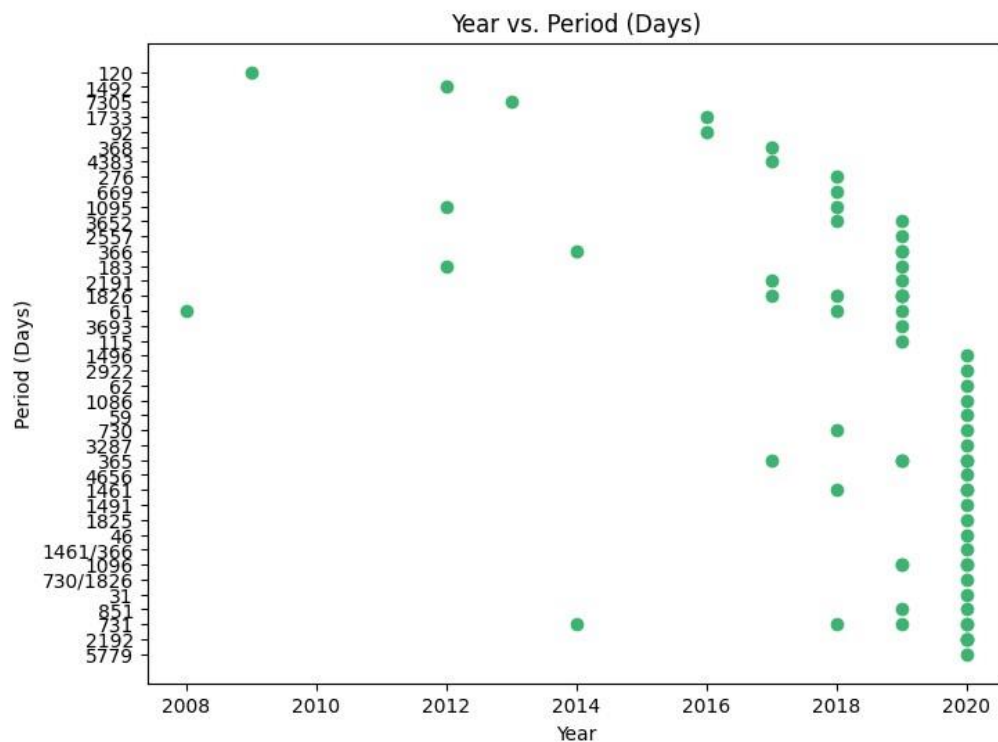
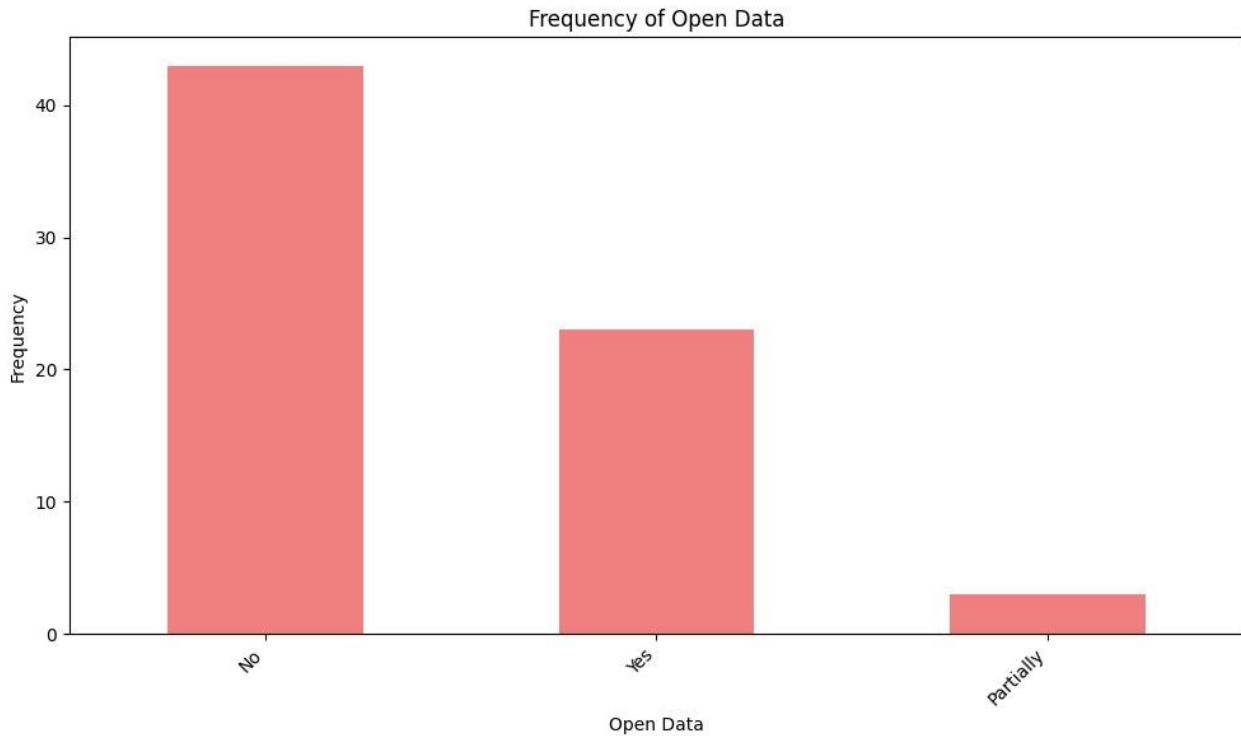
OTUPUT:











12. Deployment

- *Deploy using a free platform:*

- *Streamlit Cloud*
- *Gradio + Hugging Face Spaces*
- *Flask API on Render or Deta*
- *Include:*
 - *Deployment method*
 - *Public link*
 - *UI Screenshot*
 - *Sample prediction output*

13. Source code

https://colab.research.google.com/drive/1lqi2JeXuV_B-KmIRAqHA7ZI__epKq_Co#scrollTo=bfxivONgoqZi

14. Future scope

1. Integration of Real-Time Data Streams via IoT Sensors

Enhancement: Incorporate real-time air quality data using IoT devices and live APIs from local monitoring stations.

Rationale: Currently, the model may rely on static or historical datasets. Real-time data integration will improve the system's accuracy and responsiveness, enabling real-time air quality alerts and decision-making.

2. Deployment of a Geo-Spatial Prediction Model

Enhancement: Extend the model to include spatial analysis using GIS (Geographic Information Systems) and satellite data for region-specific predictions.

Rationale: Air quality varies significantly by location due to traffic, industry, and weather. A geo-spatial model will allow for more localized predictions and support urban planning and health risk assessments.

3. Model Interpretability and Public Dashboard Integration

Enhancement: Develop interpretable ML models (e.g., using SHAP or LIME) and integrate them into a public-facing dashboard or mobile app.

Rationale: Transparency in model decisions builds trust and promotes informed public engagement. A dashboard can help users visualize pollution trends and take preventive actions.

13. Team Members and Roles

Responsibilities

<i>Name</i>	<i>Role</i>	<i>Responsibilities</i>
<i>PRITHIKA L</i>	<i>Team lead</i>	<i>Oversee project development, coordinate team activities, ensure timely delivery of milestones, and contribute to documentation and final presentation.</i>
<i>THIRISHA M</i>	<i>Data collector</i>	<i>Collect data from APIs (e.g., Twitter), manage dataset storage, clean and preprocess text data, and ensure quality of input data.</i>
<i>RAJALAKSHMI D</i>	<i>Model developer</i>	<i>Build sentiment and emotion classification models, perform feature engineering, and evaluate model performance using suitable metrics.</i>
<i>SWATHI D</i>	<i>Data Analyser</i>	<i>Conduct exploratory data analysis (EDA), generate insights, and develop visualizations such as word clouds, emotion trends, and sentiment dashboards.</i>