Completed by: Ajantha Indunil Wirasinghe
Date: 26/10/2024

Air Quality Index Prediction and Analysis in India (2015-2020): A Data-Driven Approach Using Machine Learning

Contents

Section 1: Introduction

Air pollution is a significant global issue, especially in rapidly developing countries like India, where urbanization and industrialization have worsened pollution levels. Monitoring and predicting air quality are crucial for minimizing adverse effects on public health and the economy (Krzyzanowski et al., 2021).

Task Purpose

The purpose of this project is to analyse and predict air quality across Indian cities using the **Air Quality Data (2015-2020)** provided by the **Central Pollution Control Board (CPCB)**. By applying machine learning techniques, the goal is to predict **Air Quality Index (AQI)** values and classify AQI categories based on pollutants such as **PM2.5**, **PM10**, and **NO2** (Rautela & Goyal, 2024).

Justification for the Task

This task holds significant value for:

- **Industry and Government:** Accurate air quality predictions allow regulatory bodies and industries to monitor pollution and comply with environmental standards, ensuring better air quality management (Yang et al., 2021).
- **Academic Research:** It provides a foundation for further research into improving air quality prediction models, incorporating more advanced machine learning techniques (Samad et al., 2023).
- **Community and Public Health:** Reliable AQI forecasts help communities understand health risks and make informed decisions to protect themselves from harmful pollution exposure (Gurjar, Ravindra & Nagpure, 2016).

By developing reliable machine learning models, this project aims to contribute to improved air quality monitoring and proactive decision-making across multiple sectors.

Section 2: Literature Review

Air pollution has been extensively studied, particularly in urban areas where pollution levels are most severe. This section reviews key studies on air quality monitoring and prediction using machine learning, discussing their methodologies, results, and the gaps they leave unresolved.

Study 1: Indian Megacities and Environmental Vulnerability (Gurjar & Nagpure, 2015)

Gurjar and Nagpure (2015) analysed air pollution in Indian megacities using data from **the National Air Quality Monitoring Programme (NAMP)**. The study focused on pollutants like **PM2.5, PM10**, and **NOx**, particularly in Delhi and Mumbai. While it provided valuable insights into pollution trends, it lacked predictive capabilities and real-time monitoring, limiting its application for proactive air quality management.

Study 2: Forecasting Criteria Air Pollutants Using Data-Driven Approaches (Tikhe et al., 2013)

Tikhe, Khare, and Londhe (2013) explored data-driven models to forecast air pollutants in India, estimating concentrations of key pollutants like **NOx** and **SO2**. The study demonstrated the usefulness of predictive models but relied on traditional techniques, lacking the flexibility and accuracy of modern machine learning methods. Additionally, it did not include AQI prediction, leaving a gap for advanced models to provide more accurate and real-time forecasts.

Study 3: Machine Learning for Air Quality Management (Sharma & Mauzerall, 2021)

Sharma and Mauzerall (2021) investigated the use of machine learning algorithms, including **Random Forest** and **KNN**, to predict pollutant levels and **AQI** across India. While achieving high accuracy, the study faced challenges in scaling these models to regions with insufficient monitoring infrastructure, highlighting the need for data imputation techniques like KNN to handle missing data.

2.1 Research Gaps and Unresolved Questions:

1. **Predictive Modeling Limitations:**
Previous studies (Gurjar & Nagpure, 2015; Tikhe et al., 2013) have primarily focused on historical data analysis or basic statistical models, which lack real-time or advanced machine learning-based predictions. This leaves a gap in developing robust models that can predict **AQI** in real-time using modern machine learning techniques.

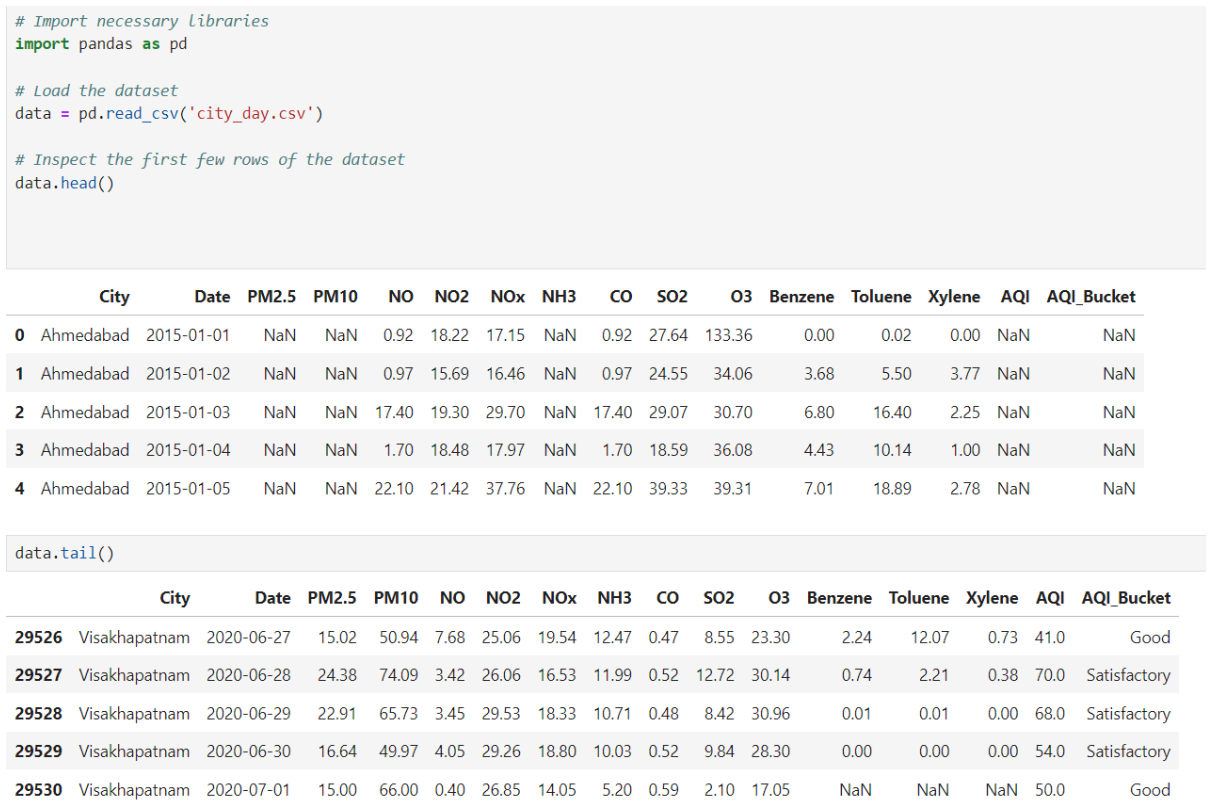
2. **Scalability and Infrastructure Challenges:**
While the **AAQR (2021)** study achieved high accuracy, the issue of applying machine learning models in areas with limited air quality monitoring remains unresolved. Developing techniques like **KNN imputation** for handling missing data is crucial for scaling air quality models across regions with fewer monitoring stations.

3. **Flexibility of Models:**
Traditional models used in studies like **Tikhe et al. (2013)** rely on regression techniques that are less flexible compared to algorithms like **Random Forest** or **KNN**, which can better capture complex relationships between pollutants and **AQI**. More advanced models can address these gaps by improving accuracy and adaptability.

Section 3: Methodology

3.1 Data Sources and Collection

The dataset used for this analysis is the Air Quality Data in India (2015-2020), available through the Open Government Data (OGD) Platform India. This dataset contains multiple CSV files, with the city_day.csv file chosen for this study. It includes daily measurements of various pollutants such as PM2.5, PM10, NO2, CO, and meteorological variables like temperature and humidity. The data was collected from Continuous Ambient Air Quality Monitoring Stations (CAAQMS) across different Indian cities. The dataset spans five years and is suitable for AQI prediction and classification tasks (Government of India, 2020).



```
# Import necessary libraries
import pandas as pd

# Load the dataset
data = pd.read_csv('city_day.csv')

# Inspect the first few rows of the dataset
data.head()
```

	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket
0	Ahmedabad	2015-01-01	NaN	NaN	0.92	18.22	17.15	NaN	0.92	27.64	133.36	0.00	0.02	0.00	NaN	NaN
1	Ahmedabad	2015-01-02	NaN	NaN	0.97	15.69	16.46	NaN	0.97	24.55	34.06	3.68	5.50	3.77	NaN	NaN
2	Ahmedabad	2015-01-03	NaN	NaN	17.40	19.30	29.70	NaN	17.40	29.07	30.70	6.80	16.40	2.25	NaN	NaN
3	Ahmedabad	2015-01-04	NaN	NaN	1.70	18.48	17.97	NaN	1.70	18.59	36.08	4.43	10.14	1.00	NaN	NaN
4	Ahmedabad	2015-01-05	NaN	NaN	22.10	21.42	37.76	NaN	22.10	39.33	39.31	7.01	18.89	2.78	NaN	NaN

```
data.tail()
```

	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket
29526	Visakhapatnam	2020-06-27	15.02	50.94	7.68	25.06	19.54	12.47	0.47	8.55	23.30	2.24	12.07	0.73	41.0	Good
29527	Visakhapatnam	2020-06-28	24.38	74.09	3.42	26.06	16.53	11.99	0.52	12.72	30.14	0.74	2.21	0.38	70.0	Satisfactory
29528	Visakhapatnam	2020-06-29	22.91	65.73	3.45	29.53	18.33	10.71	0.48	8.42	30.96	0.01	0.01	0.00	68.0	Satisfactory
29529	Visakhapatnam	2020-06-30	16.64	49.97	4.05	29.26	18.80	10.03	0.52	9.84	28.30	0.00	0.00	0.00	54.0	Satisfactory
29530	Visakhapatnam	2020-07-01	15.00	66.00	0.40	26.85	14.05	5.20	0.59	2.10	17.05	NaN	NaN	NaN	50.0	Good

Figure 3.1 Sample dataset

3.2 Data Preprocessing

3.2.1 Data Cleaning:

Handling missing data: is crucial for preventing biased results and maintaining model accuracy (Rautela & Goyal, 2024). In this dataset, pollutants like PM2.5, PM10, and NO2 had significant missing values, with PM10 showing 37.72% missing data, making a robust imputation strategy essential (Samad et al., 2023). Outliers in pollutants like PM2.5, PM10, and CO were identified and capped based on thresholds outlined in World Health Organization (WHO) guidelines, ensuring data integrity (WHO, 2021).

Identifying Missing Values:

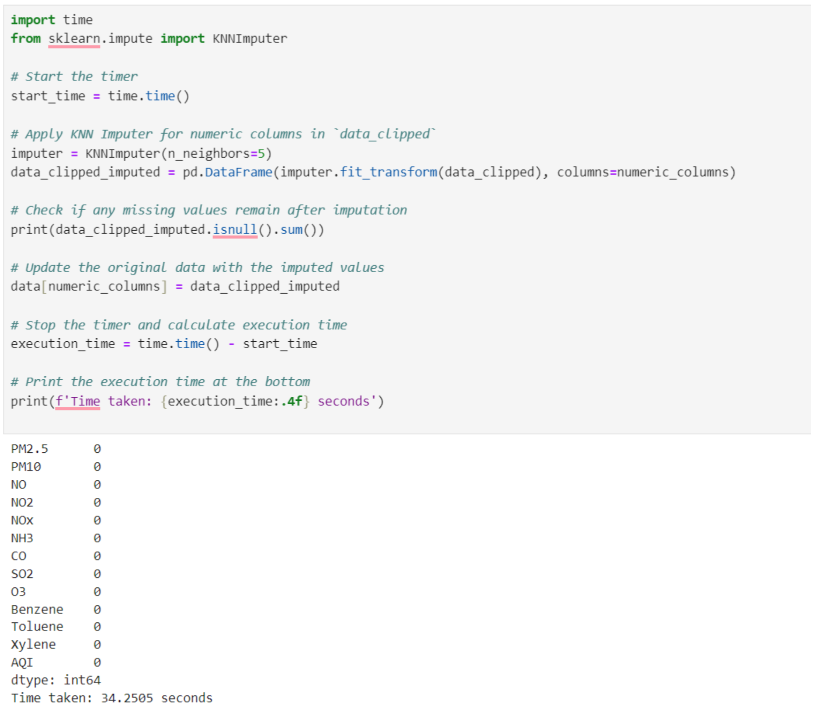
Feature	Missing Values	Percentage (%)
PM2.5	4598	15.57
PM10	11140	37.72
NO	3582	12.13
NO2	3585	12.14
NOx	4185	14.17
NH3	10328	34.97
CO	2059	6.97
SO2	3854	13.05

Figure 3.2. Missing Values for Key Pollutants to summarize the missing data.

Imputation Strategy:

1. **KNN Imputation for Numeric Features:** K-Nearest Neighbors (KNN) imputation was applied to pollutants like PM2.5, PM10, and NO2. KNN fills missing data based on neighbouring values, preserving relationships between features. We used *k=5* neighbors to ensure a balance between speed and accuracy (Yang et al., 2021).
   - ***Why KNN?*** KNN maintains feature correlations, avoiding the bias introduced by simpler methods like mean imputation (Sharma & Mauzerall, 2021).
2. **Mode Imputation for Categorical Features:** For categorical data like AQI Bucket, we imputed missing values using the mode (most frequent category).
   - ***Why Mode Imputation?*** Mode imputation ensures the distribution of AQI categories (e.g., Good, Moderate) remains intact.

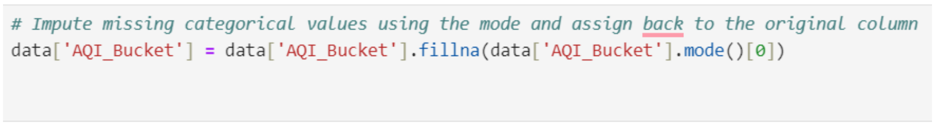
Handling Outliers: Box plots identified outliers in pollutants such as PM2.5, PM10, and CO. Extreme values from pollution spikes were capped based on World Health Organization (WHO) guidelines to ensure data integrity and comparability (Krzyzanowski et al., 2021).



```python
import time
from sklearn.impute import KNNImputer

# Start the timer
start_time = time.time()

# Apply KNN Imputer for numeric columns in `data_clipped`
imputer = KNNImputer(n_neighbors=5)
data_clipped_imputed = pd.DataFrame(imputer.fit_transform(data_clipped), columns=numeric_columns)

# Check if any missing values remain after imputation
print(data_clipped_imputed.isnull().sum())

# Update the original data with the imputed values
data[numeric_columns] = data_clipped_imputed

# Stop the timer and calculate execution time
execution_time = time.time() - start_time

# Print the execution time at the bottom
print(f'Time taken: {execution_time:.4f} seconds')
```

```
PM2.5      0
PM10       0
NO         0
NO2        0
NOx        0
NH3        0
CO         0
SO2        0
O3         0
Benzene    0
Toluene    0
Xylene     0
AQI        0
dtype: int64
Time taken: 34.2505 seconds
```

Figure 3.3 KNN Imputation for Numeric Features of missing data.



```python
# Impute missing categorical values using the mode and assign back to the original column
data['AQI_Bucket'] = data['AQI_Bucket'].fillna(data['AQI_Bucket'].mode()[0])
```

Figure 3.4 Imputation for Missing Categorical Values.

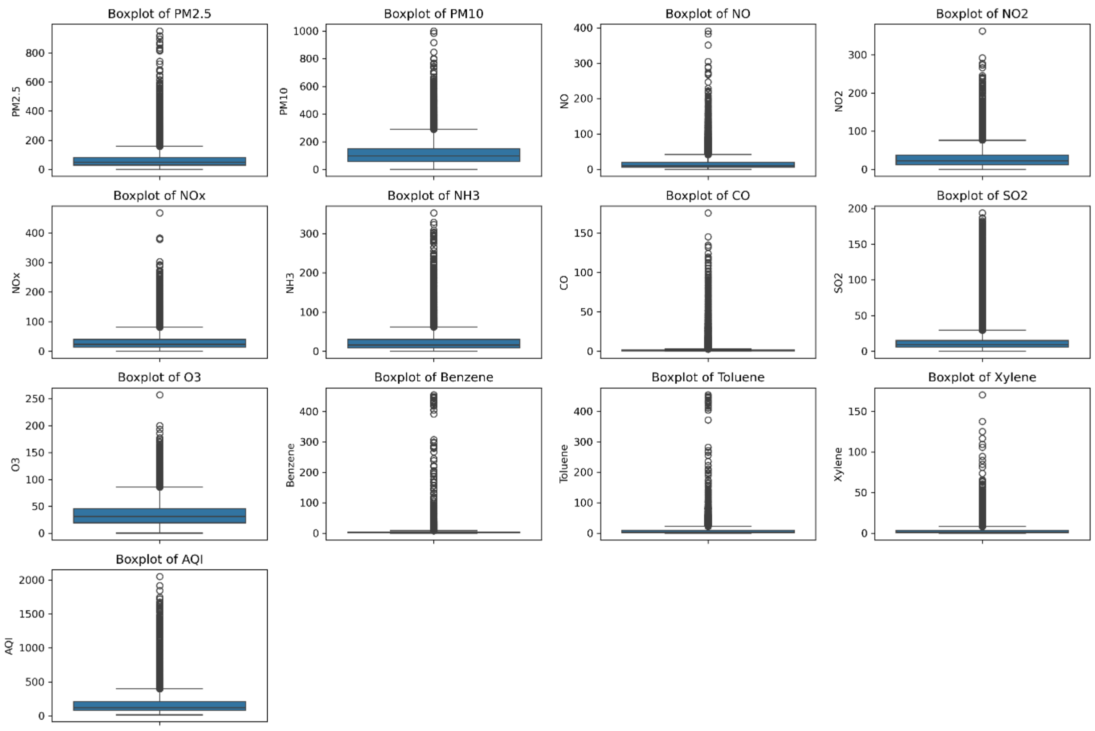


Figure 3.5 Box Plot of Pollutants Before Imputation and Outlier Treatment.

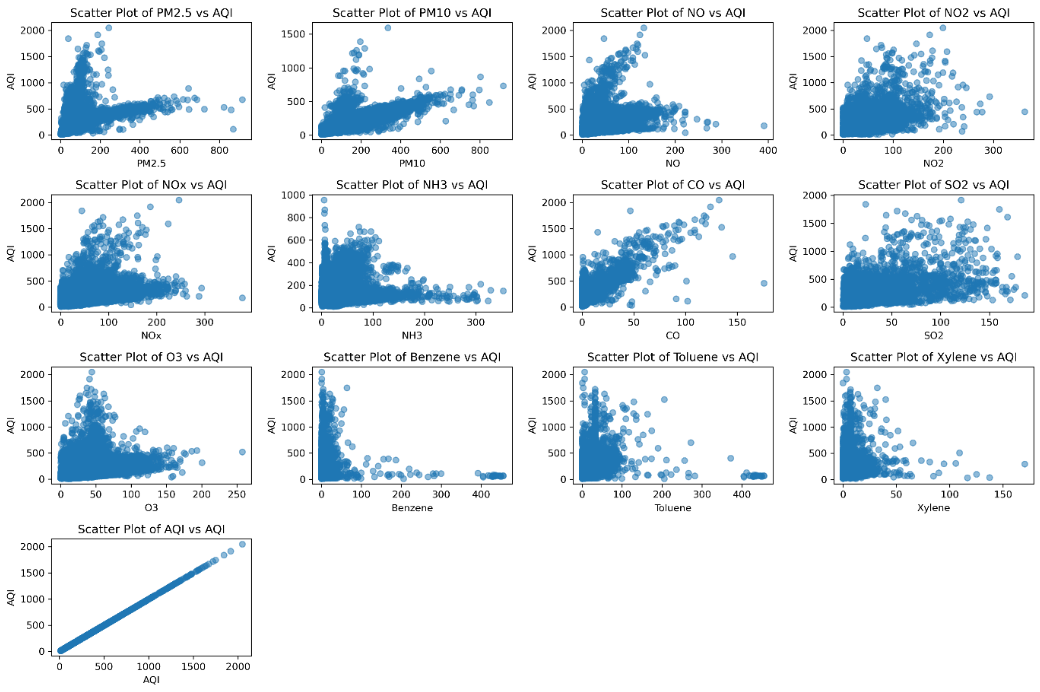


Figure 3.6 Scatter Plot of Pollutants vs AQI (Before Imputation)

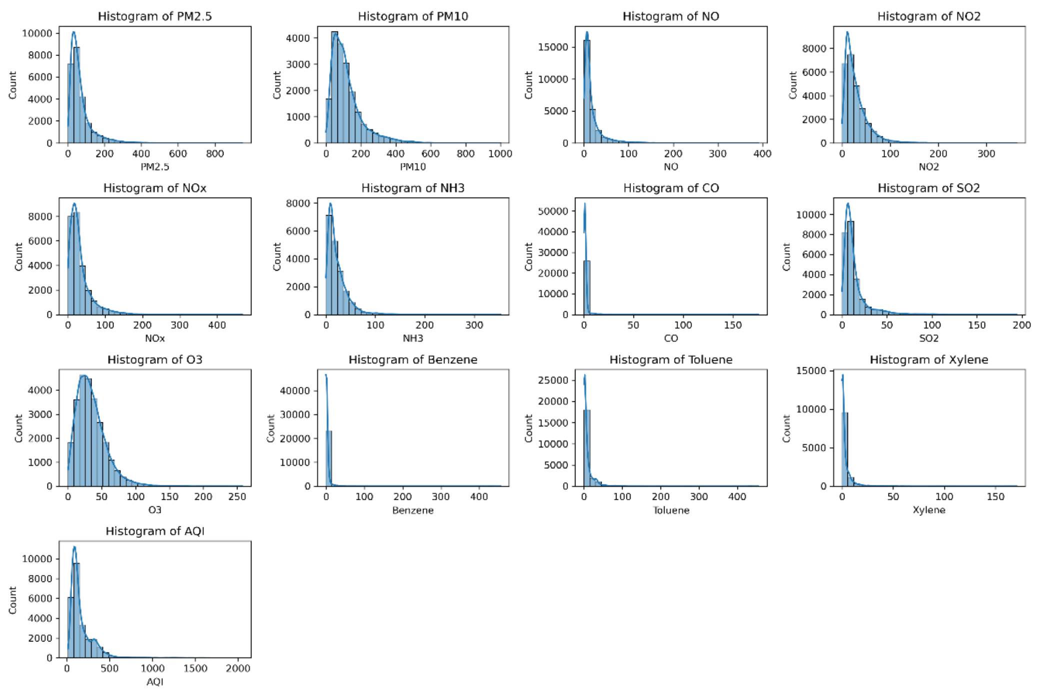
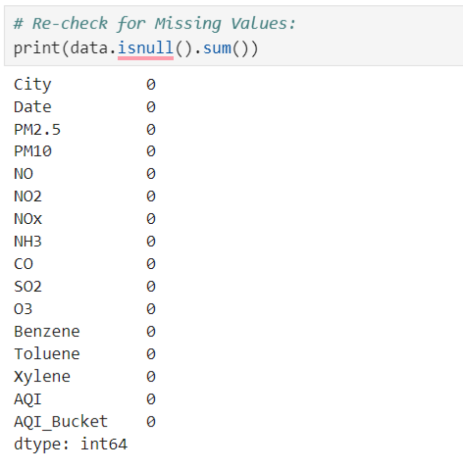


Figure 3.7 Distribution of Key Pollutants and AQI: Histograms for Feature Analysis (Before Imputation)

The histograms reveal that most features, including **PM2.5, PM10, NO, NO2, NOx, CO, SO2,** and **AQI,** are right-skewed, with more frequent lower values and fewer extreme events. Benzene, Toluene, and Xylene remain mostly at low levels, indicating a minor impact relative to major pollutants. These skewed distributions suggest potential data transformations, such as logarithmic scaling, to minimize outlier effects and improve model performance (Gurjar et al., 2016). AQI distribution shows moderate levels are common, with severe pollution events being rare.

Post-Imputation Validation: After handling missing values, the dataset was validated through box plots and scatter plots to ensure data consistency and confirm that imputation improved overall quality.



```
# Re-check for Missing Values:
print(data.isnull().sum())
```

```
City          0
Date          0
PM2.5         0
PM10          0
NO            0
NO2           0
NOx           0
NH3           0
CO            0
SO2           0
O3            0
Benzene       0
Toluene       0
Xylene        0
AQI           0
AQI_Bucket    0
dtype: int64
```

Figure 3.8 Recheck for missing Values.

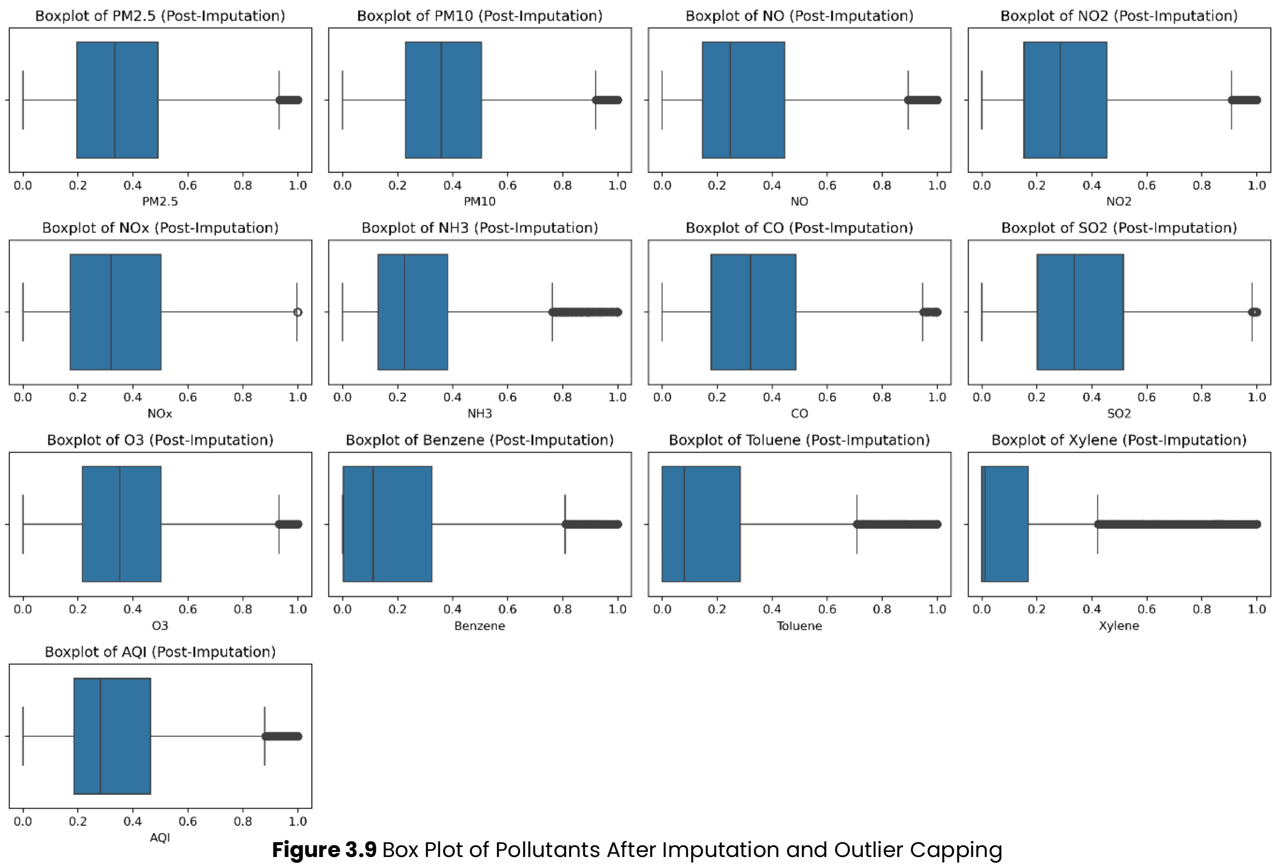


Figure 3.9 Box Plot of Pollutants After Imputation and Outlier Capping

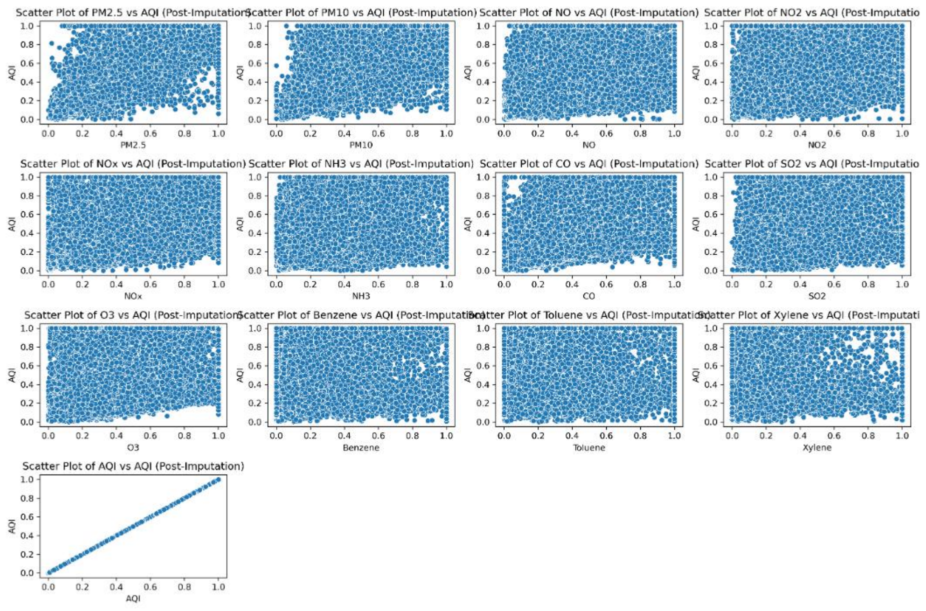


Figure 3.10 Scatter Plot of Pollutants vs AQI (After Imputation and Scaling)

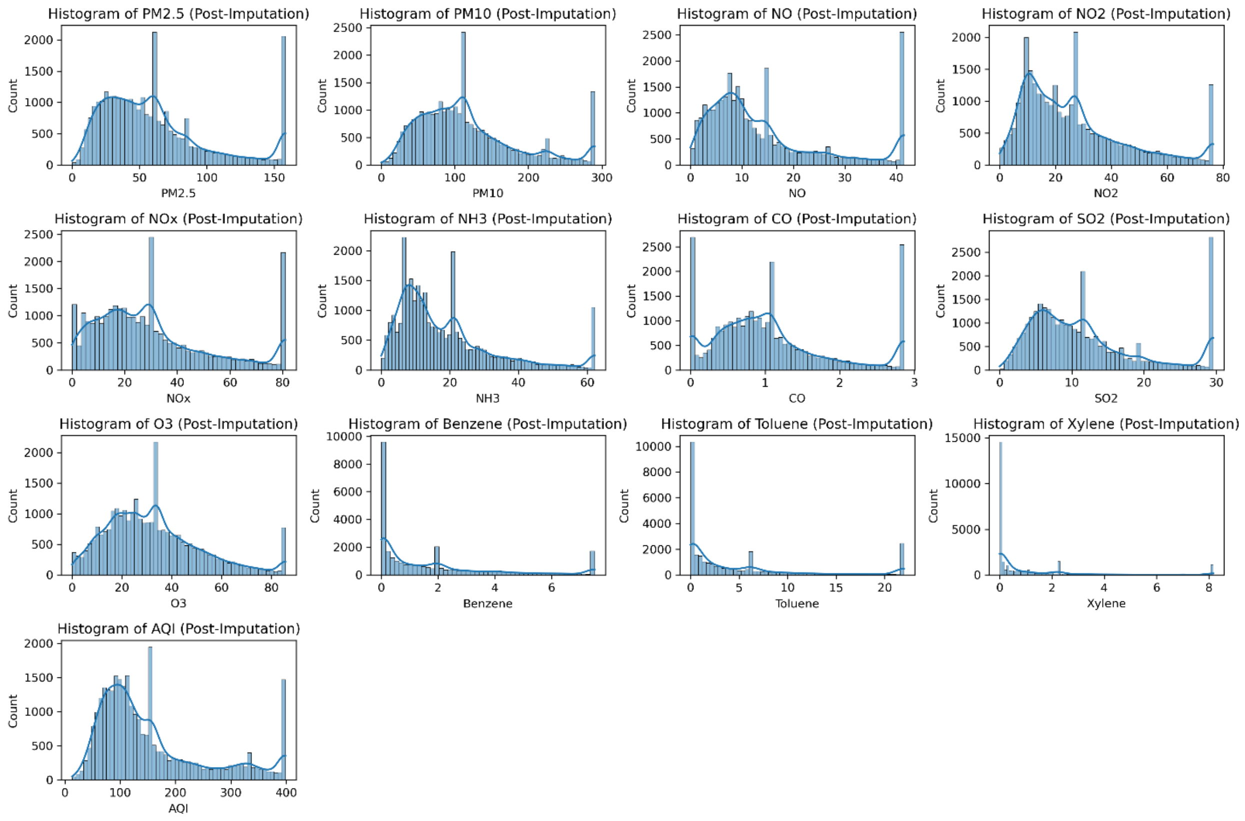


Figure 3.11 Histograms of Key Pollutants (After Imputation and Scaling)

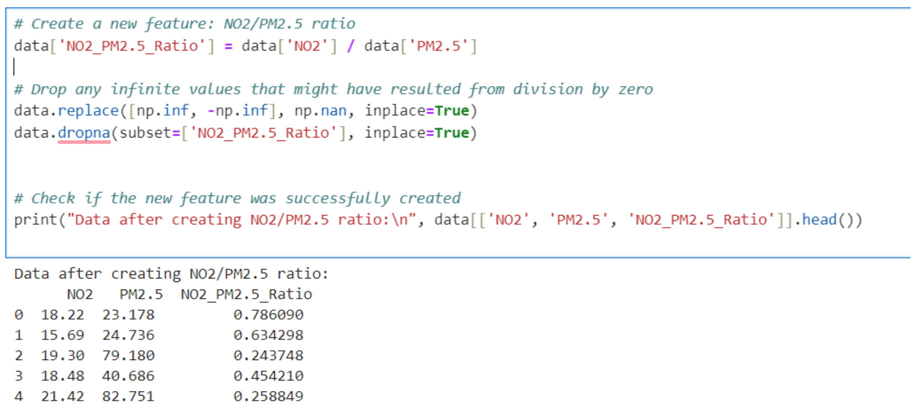
After imputation, histograms showed adjusted distributions with some spikes. These spikes likely resulted from imputation, where missing values were filled with common values, creating clusters. Despite this, the dataset is now free of missing values, well-imputed, and prepared for model training without losing any critical information (Gurjar & Nagpure, 2015).

3.2.2 Feature Engineering:

Feature engineering was used to extract more insights, especially around pollutant interactions and temporal trends. Key new features include:

1. **Pollutant Ratios:** A NO2/PM2.5 ratio was created to capture the relationship between nitrogen dioxide (NO2) and particulate matter (PM2.5), emphasizing their combined effect on air quality (Samad et al., 2023).

The new feature was created as follows:



```
# Create a new feature: NO2/PM2.5 ratio
data['NO2_PM2.5_Ratio'] = data['NO2'] / data['PM2.5']

# Drop any infinite values that might have resulted from division by zero
data.replace([np.inf, -np.inf], np.nan, inplace=True)
data.dropna(subset=['NO2_PM2.5_Ratio'], inplace=True)


# Check if the new feature was successfully created
print("Data after creating NO2/PM2.5 ratio:\n", data[['NO2', 'PM2.5', 'NO2_PM2.5_Ratio']].head())
```

```
Data after creating NO2/PM2.5 ratio:
     NO2   PM2.5  NO2_PM2.5_Ratio
0  18.22  23.178         0.786090
1  15.69  24.736         0.634298
2  19.30  79.180         0.243748
3  18.48  40.686         0.454210
4  21.42  82.751         0.258849
```

Figure 3.12 New Feature **NO2/PM2.5 ratio** creation

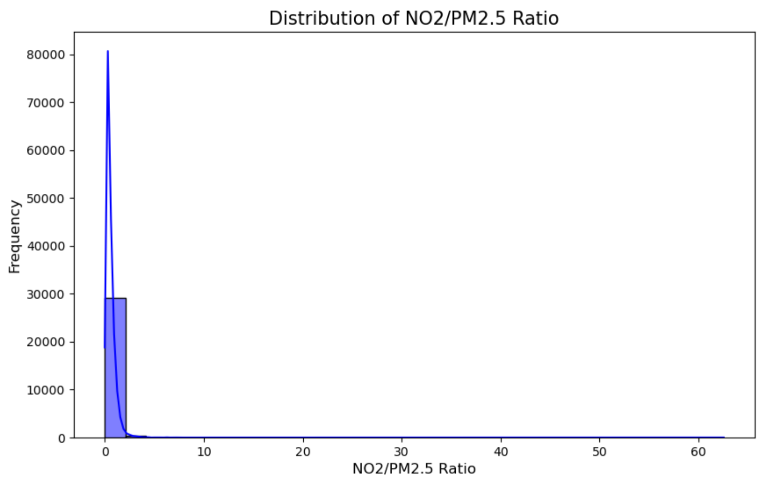


Figure 3.13 Distribution of NO2/PM2.5 Ratio

2. **Time-Based Variables:** New features like ***Year***, ***Month***, and ***DayOfWeek*** were engineered to capture seasonality and cyclical patterns. These variables improve model performance as air quality often varies with seasonal shifts, holidays, and industrial activities (Yang et al., 2021).

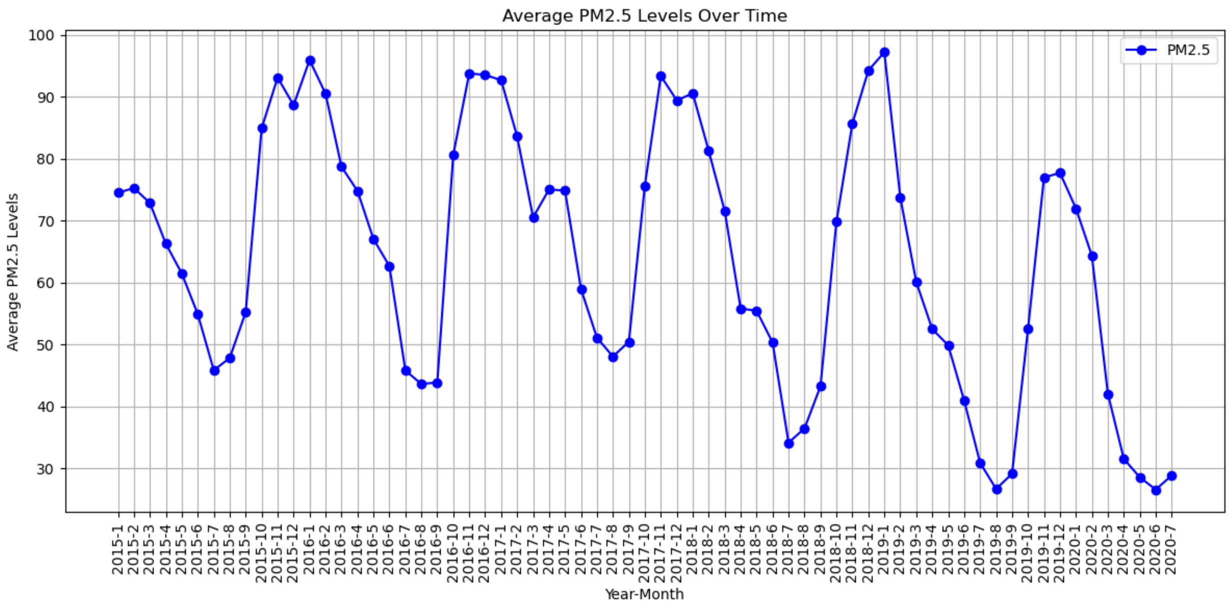


Figure 3.14 Time Series Plot of Average PM2.5 Levels Over Time

The **time series plot for PM2.5** levels indicates two trends. First, a decline in peak **PM2.5** levels over time likely reflects government regulations and pollution control efforts. Second, a seasonal pattern shows higher **PM2.5** levels in December and January, possibly due to temperature inversions that trap pollutants and increased winter heating and industrial activity (Rautela & Goyal, 2024). Including these time-related features enhances the model's ability to predict seasonal trends and assess the effects of regulatory actions on air quality.

3.2.3 Data Transformation

Min-Max Scaling

To bring all numeric features to a uniform range of [0,1], Min-Max Scaling was applied. This scaling was critical for algorithms like k-NN, where unscaled features with larger ranges (e.g., pollutant levels) could overshadow smaller ones, potentially leading to biased model outputs (Sharma & Mauzerall, 2021).

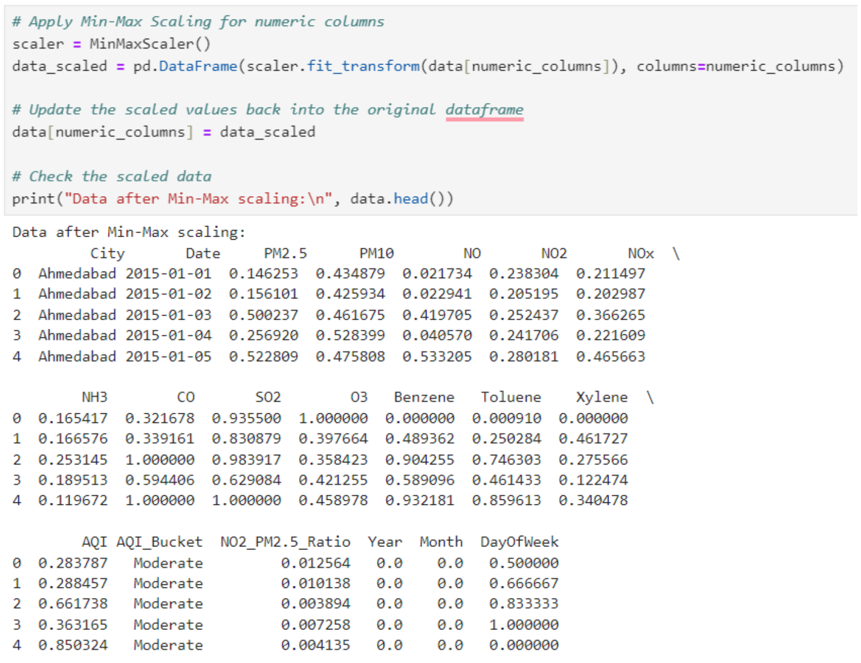
Justification:
Min-Max Scaling ensures no single feature dominates the model, making it useful for distance-based models and accelerating convergence in gradient-based optimization.

Formula:

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Where:

- X is the original value,
- X_{min} and X_{max} are the minimum and maximum values for the feature.



```python
# Apply Min-Max Scaling for numeric columns
scaler = MinMaxScaler()
data_scaled = pd.DataFrame(scaler.fit_transform(data[numeric_columns]), columns=numeric_columns)

# Update the scaled values back into the original dataframe
data[numeric_columns] = data_scaled

# Check the scaled data
print("Data after Min-Max scaling:\n", data.head())
```

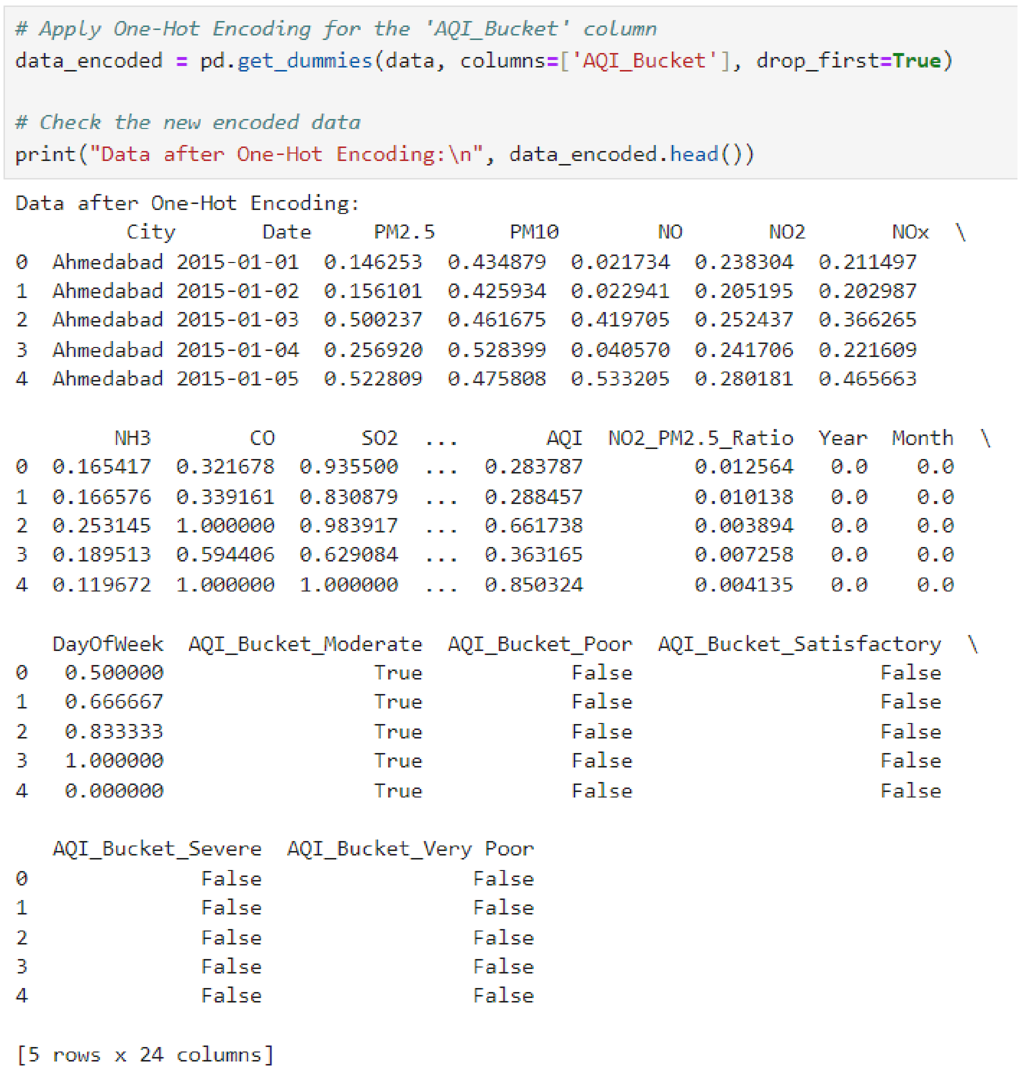
```
Data after Min-Max scaling:
         City        Date     PM2.5      PM10        NO       NO2       NOx  \
0  Ahmedabad 2015-01-01  0.146253  0.434879  0.021734  0.238304  0.211497
1  Ahmedabad 2015-01-02  0.156101  0.425934  0.022941  0.205195  0.202987
2  Ahmedabad 2015-01-03  0.500237  0.461675  0.419705  0.252437  0.366265
3  Ahmedabad 2015-01-04  0.256920  0.528399  0.040570  0.241706  0.221609
4  Ahmedabad 2015-01-05  0.522809  0.475808  0.533205  0.280181  0.465663

        NH3        CO       SO2        O3   Benzene   Toluene    Xylene  \
0  0.165417  0.321678  0.935500  1.000000  0.000000  0.000910  0.000000
1  0.166576  0.339161  0.830879  0.397664  0.489362  0.250284  0.461727
2  0.253145  1.000000  0.983917  0.358423  0.904255  0.746303  0.275566
3  0.189513  0.594406  0.629084  0.421255  0.589096  0.461433  0.122474
4  0.119672  1.000000  1.000000  0.458978  0.932181  0.859613  0.340478

        AQI AQI_Bucket  NO2_PM2.5_Ratio  Year  Month  DayOfWeek
0  0.283787   Moderate         0.012564   0.0    0.0   0.500000
1  0.288457   Moderate         0.010138   0.0    0.0   0.666667
2  0.661738   Moderate         0.003894   0.0    0.0   0.833333
3  0.363165   Moderate         0.007258   0.0    0.0   1.000000
4  0.850324   Moderate         0.004135   0.0    0.0   0.000000
```

Figure 3.15 Data after Min-Max Scaling to show the effects

One-Hot Encoding:

We applied One-Hot Encoding for categorical features like **AQI_Bucket**. This technique converts each category into binary columns, ensuring the model doesn't mistakenly interpret these categories as **ordinal**, as with **Good, Moderate**, and **Poor** (Krzyzanowski et al., 2021).

Justification:
One-Hot Encoding ensures that categories are treated independently without implying any hierarchy, which is essential for features like **AQI_Bucket**, where each category represents a distinct air quality level.



```python
# Apply One-Hot Encoding for the 'AQI_Bucket' column
data_encoded = pd.get_dummies(data, columns=['AQI_Bucket'], drop_first=True)

# Check the new encoded data
print("Data after One-Hot Encoding:\n", data_encoded.head())
```

```
Data after One-Hot Encoding:
         City        Date     PM2.5      PM10        NO       NO2       NOx  \
0  Ahmedabad 2015-01-01  0.146253  0.434879  0.021734  0.238304  0.211497
1  Ahmedabad 2015-01-02  0.156101  0.425934  0.022941  0.205195  0.202987
2  Ahmedabad 2015-01-03  0.500237  0.461675  0.419705  0.252437  0.366265
3  Ahmedabad 2015-01-04  0.256920  0.528399  0.040570  0.241706  0.221609
4  Ahmedabad 2015-01-05  0.522809  0.475808  0.533205  0.280181  0.465663

        NH3        CO       SO2  ...       AQI  NO2_PM2.5_Ratio  Year  Month  \
0  0.165417  0.321678  0.935500  ...  0.283787         0.012564   0.0    0.0
1  0.166576  0.339161  0.830879  ...  0.288457         0.010138   0.0    0.0
2  0.253145  1.000000  0.983917  ...  0.661738         0.003894   0.0    0.0
3  0.189513  0.594406  0.629084  ...  0.363165         0.007258   0.0    0.0
4  0.119672  1.000000  1.000000  ...  0.850324         0.004135   0.0    0.0

   DayOfWeek  AQI_Bucket_Moderate  AQI_Bucket_Poor  AQI_Bucket_Satisfactory  \
0   0.500000                 True            False                    False
1   0.666667                 True            False                    False
2   0.833333                 True            False                    False
3   1.000000                 True            False                    False
4   0.000000                 True            False                    False

   AQI_Bucket_Severe  AQI_Bucket_Very Poor
0              False                 False
1              False                 False
2              False                 False
3              False                 False
4              False                 False

[5 rows x 24 columns]
```

Figure 3.16 Data after Min-Max Scaling to show the effects

By using Min-Max Scaling for numeric data and One-Hot Encoding for categorical data, we balanced the dataset, preparing it for efficient and accurate machine learning predictions.

3.3 Feature Selection:

Feature selection is essential for building an effective and easy-to-interpret machine learning model. It focuses on identifying the most important features for predicting the target variable, AQI, while removing irrelevant or redundant ones that could add noise or complexity.

Before using the Random Forest algorithm for feature ranking, we created a correlation matrix (Figure 3.2) to spot highly correlated features. This helped highlight relationships between pollutants, guiding feature selection and revealing potential multicollinearity. For example, PM2.5 and PM10 showed a strong correlation, indicating they may have similar impacts on the model.

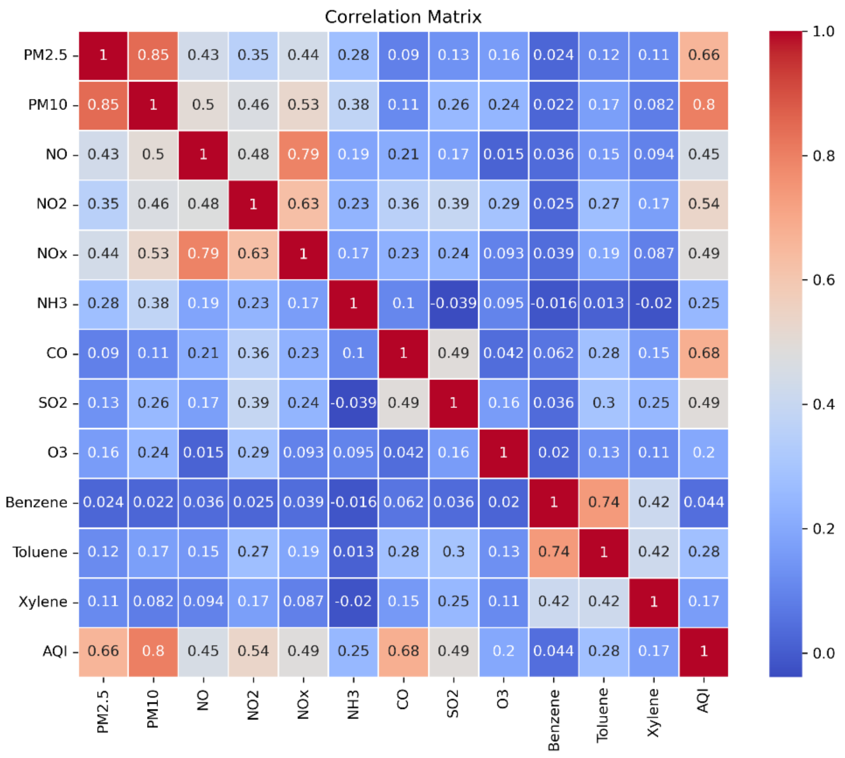


Figure 3.17 Correlation Matrix of Pollutants and AQI

3.3.1 Random Forest Feature Importance:

We used the Random Forest algorithm to rank features by their impact on prediction accuracy, helping to identify the most important variables and streamline the dataset (Breiman, 2001).

Key insights:

- **PM2.5** was the most important feature, reflecting its significant role in urban air pollution.
- **AQI categories** like AQI_Bucket_Severe and AQI_Bucket_Moderate were also influential.
- **CO, NOx, and SO2** contributed but had less impact than PM2.5 and AQI categories.
- **Benzene, Toluene, and Xylene** were excluded to simplify the model.

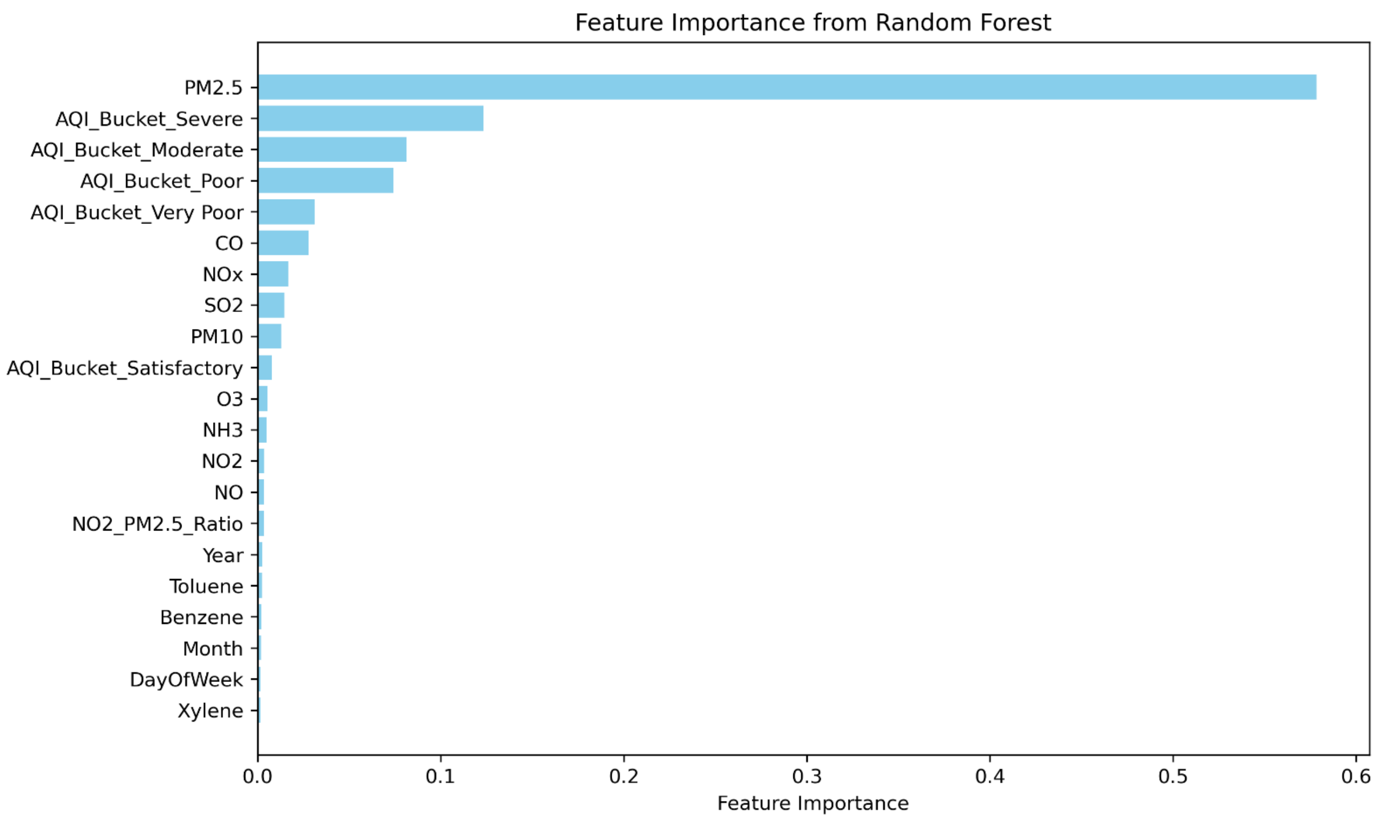


Figure 3.18 Feature Importance from Random Forest

3.3.2 Domain Knowledge and Multicollinearity:

We used domain knowledge to prioritize key pollutants like PM10, NO2, and CO. To address multicollinearity, highly correlated features like NO2 and NOx were transformed into ratios (e.g., NO2/PM2.5), capturing interactions while reducing redundancy (Zou et al., 2006).

3.3.3 Final Set of Selected Features:

After analysing feature importance and considering multicollinearity, the final set of features selected for the model included:

- **PM2.5**: Most critical pollutant.
- **AQI Bucket categories**: Capturing air quality levels.
- **CO, NOx, SO2, PM10**: Significant contributors to air quality.
- **NO2/PM2.5 Ratio**: Engineered to capture pollutant interactions.
- **Time-Based Variables**: Accounting for seasonal trends.

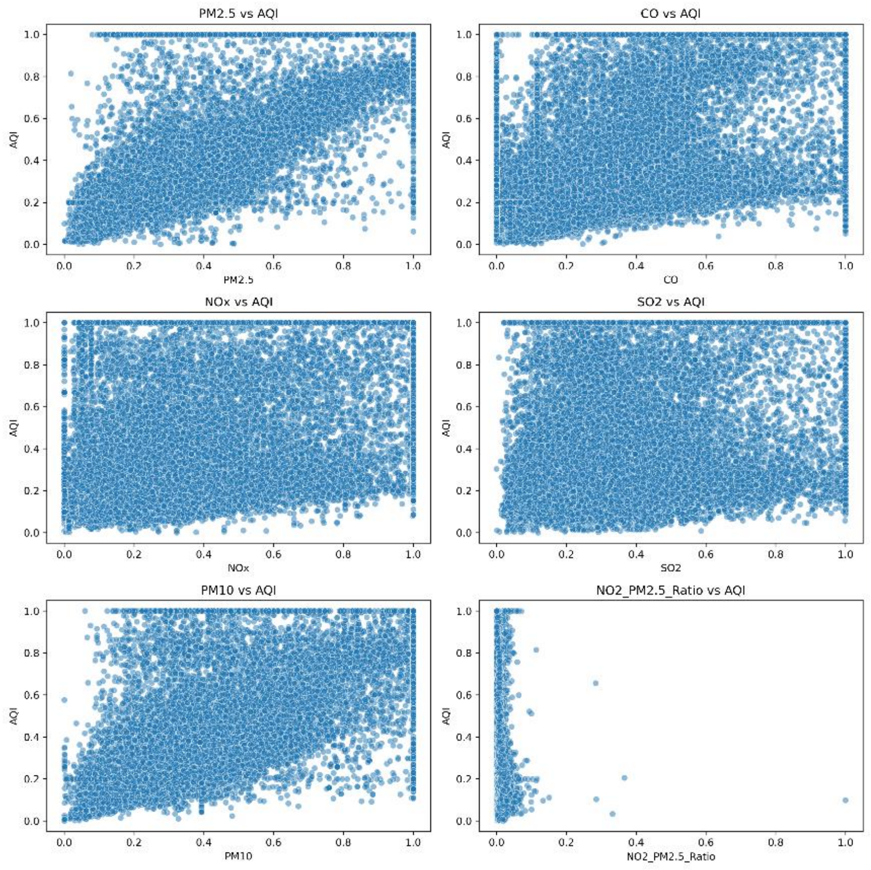


Figure 3.19 Scatter Plots of Selected Features vs. AQI

General Observations:

- **PM2.5** remains the strongest predictor of AQI, showing the clearest trend among pollutants.
- **CO, NOx, SO2, and PM10** have moderate but more scattered relationships with AQI, suggesting their impact depends on other factors.
- The **NO2/PM2.5 Ratio** has an unusual distribution, indicating it may offer more insight when combined with other variables rather than standing alone.

These findings align with the Random Forest feature importance results, where PM2.5 ranked highest, followed by AQI categories and other pollutants.

Conclusion:

The feature selection process, incorporating Random Forest, domain knowledge, and multicollinearity considerations, resulted in a streamlined set of key features. This enhanced model is more efficient, interpretable, and accurate in predicting AQI. By excluding less important features, the model's performance improved, reducing overfitting and boosting its reliability for air quality forecasting.

3.4 Model Selection

This section compares four machine learning models for predicting AQI, evaluating them based on Mean Squared Error (MSE) and R^2 scores.

In comparing four machine learning models for predicting **AQI, Linear Regression, Support Vector Regression (SVR), Random Forest, and k-Nearest Neighbors (k-NN)** were selected for their robustness in regression tasks and varying approaches to handling data complexities (Hastie et al., 2009; Pedregosa et al., 2011).

3.4.1 Benchmark Model: Linear Regression

- **Description**: Linear regression assumes a straightforward linear relationship between features and AQI.
- **Rationale**: As a baseline model, it offers a simple comparison point for more complex models.
- **Performance**:
  - **MSE**: 0.00658
  - **R^2**: 0.899
- **Observation**: Linear regression explained about 90% of the variance, but its higher MSE and inability to capture non-linear patterns led to less accurate predictions.

3.4.2 Random Forest Regressor

- **Description**: Random Forest builds multiple decision trees and averages their predictions, handling both linear and non-linear relationships.
- **Rationale**: Ideal for modeling the complex interactions between pollutants and AQI.
- **Performance**:
  - **MSE**: 0.00341
  - **R^2**: 0.948
- **Observation**: Random Forest was the best performer, capturing 95% of the variance in AQI, with the lowest MSE.

3.4.3 k-Nearest Neighbors (k-NN)

- **Description**: k-NN predicts AQI by averaging the outcomes of the closest data points.
- **Rationale**: Well-suited for distance-based predictions after Min-Max scaling.
- **Performance**:
  - **MSE**: 0.00407
  - **R^2**: 0.937
- **Observation**: k-NN performed well, second only to Random Forest, balancing accuracy with simplicity.

3.4.4 Support Vector Regression (SVR)

- **Description**: SVR uses kernel functions to handle non-linear relationships.
- **Rationale**: Included to capture complex, non-linear interactions.
- **Performance**:
  - **MSE**: 0.00494
  - **R^2**: 0.924
- **Observation**: SVR performed well but lagged behind Random Forest and k-NN due to its higher error and computational demands.

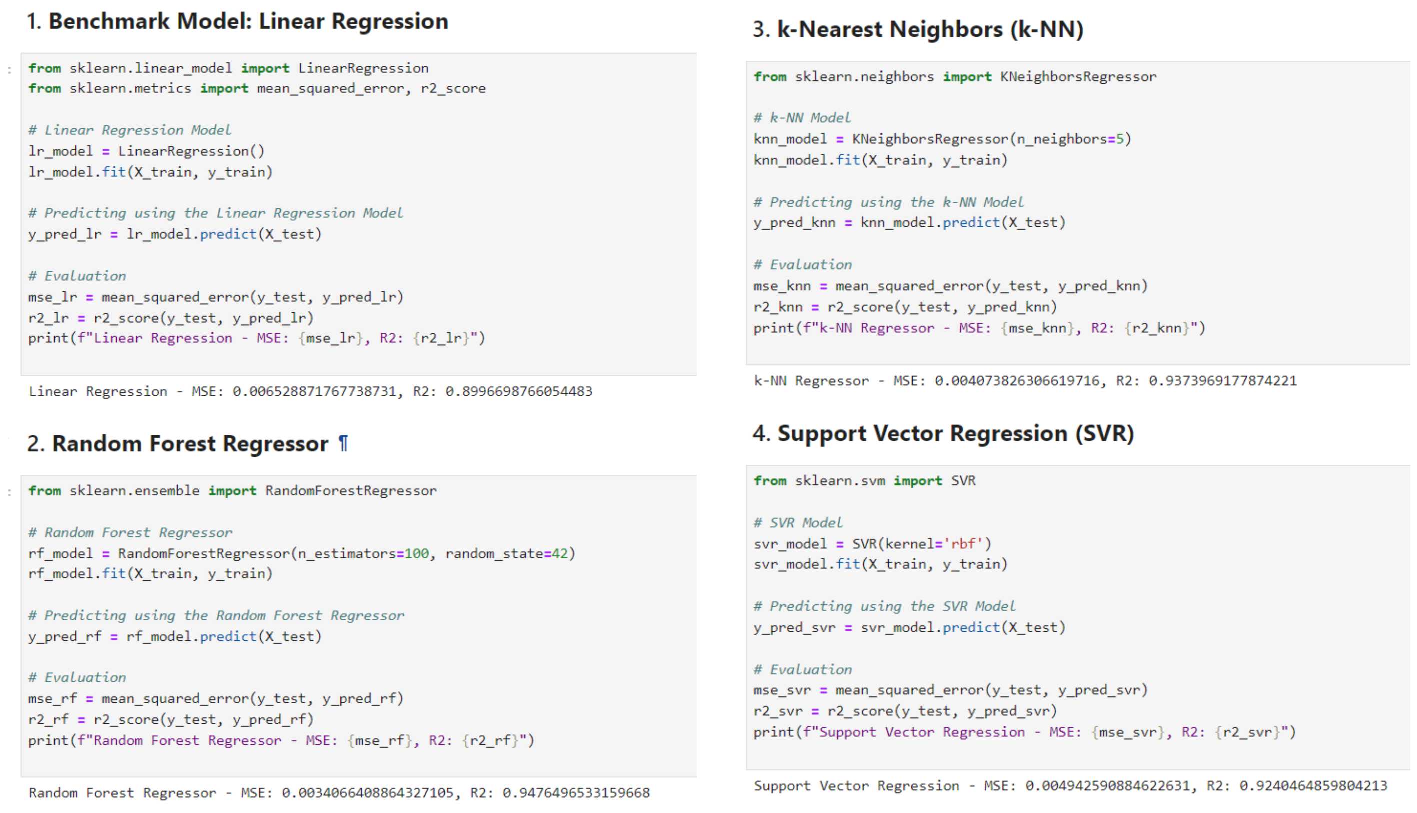
Summary:

- **Best Model**: Random Forest was the top performer, balancing complexity and accuracy.
- **Runner-up**: k-NN offered a solid alternative with its simplicity.
- **Benchmark**: Linear regression served as a useful baseline.
- **SVR**: Performed well but was less optimal due to higher error and complexity.

Model Performance Overview:

Model	MSE	R^2
Linear Regression	0.00658	0.8996
Random Forest	0.00341	0.9476
k-NN	0.00407	0.9374
SVR	0.00494	0.9240

Conclusion: Random Forest emerged as the most accurate model, followed by k-NN for simplicity. Linear Regression provided a solid benchmark, while SVR was effective but less optimal. Hyperparameter tuning can further improve Random Forest's performance.



1. **Benchmark Model: Linear Regression**

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Linear Regression Model
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)

# Predicting using the Linear Regression Model
y_pred_lr = lr_model.predict(X_test)

# Evaluation
mse_lr = mean_squared_error(y_test, y_pred_lr)
r2_lr = r2_score(y_test, y_pred_lr)
print(f"Linear Regression - MSE: {mse_lr}, R2: {r2_lr}")
```

```
Linear Regression - MSE: 0.006528871767738731, R2: 0.8996698766054483
```

2. **Random Forest Regressor** ¶

```
from sklearn.ensemble import RandomForestRegressor

# Random Forest Regressor
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Predicting using the Random Forest Regressor
y_pred_rf = rf_model.predict(X_test)

# Evaluation
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)
print(f"Random Forest Regressor - MSE: {mse_rf}, R2: {r2_rf}")
```

```
Random Forest Regressor - MSE: 0.0034066408864327105, R2: 0.9476496533159668
```

3. **k-Nearest Neighbors (k-NN)**

```
from sklearn.neighbors import KNeighborsRegressor

# k-NN Model
knn_model = KNeighborsRegressor(n_neighbors=5)
knn_model.fit(X_train, y_train)

# Predicting using the k-NN Model
y_pred_knn = knn_model.predict(X_test)

# Evaluation
mse_knn = mean_squared_error(y_test, y_pred_knn)
r2_knn = r2_score(y_test, y_pred_knn)
print(f"k-NN Regressor - MSE: {mse_knn}, R2: {r2_knn}")
```

```
k-NN Regressor - MSE: 0.004073826306619716, R2: 0.9373969177874221
```

4. **Support Vector Regression (SVR)**

```
from sklearn.svm import SVR

# SVR Model
svr_model = SVR(kernel='rbf')
svr_model.fit(X_train, y_train)

# Predicting using the SVR Model
y_pred_svr = svr_model.predict(X_test)

# Evaluation
mse_svr = mean_squared_error(y_test, y_pred_svr)
r2_svr = r2_score(y_test, y_pred_svr)
print(f"Support Vector Regression - MSE: {mse_svr}, R2: {r2_svr}")
```

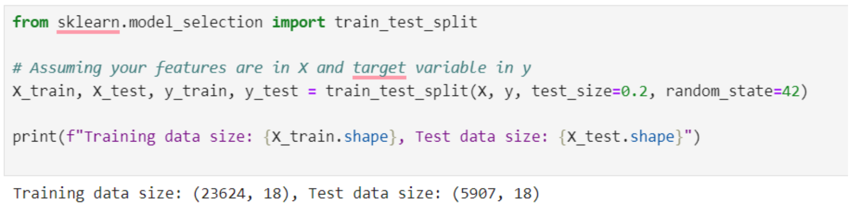
```
Support Vector Regression - MSE: 0.004942590884622631, R2: 0.9240464859804213
```

Figure 3.20 Model performances

3.5. Model Training

3.5.1. Data Split:

To ensure generalization to new data, we split the dataset into 80% training and 20% test data using train_test_split from scikit-learn (Pedregosa et al., 2011). Training data allowed the model to learn from most of the data, while the test set provided an evaluation on unseen data, ensuring a realistic performance measure and reduced overfitting risk.



```
from sklearn.model_selection import train_test_split

# Assuming your features are in X and target variable in y
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print(f"Training data size: {X_train.shape}, Test data size: {X_test.shape}")
```

```
Training data size: (23624, 18), Test data size: (5907, 18)
```

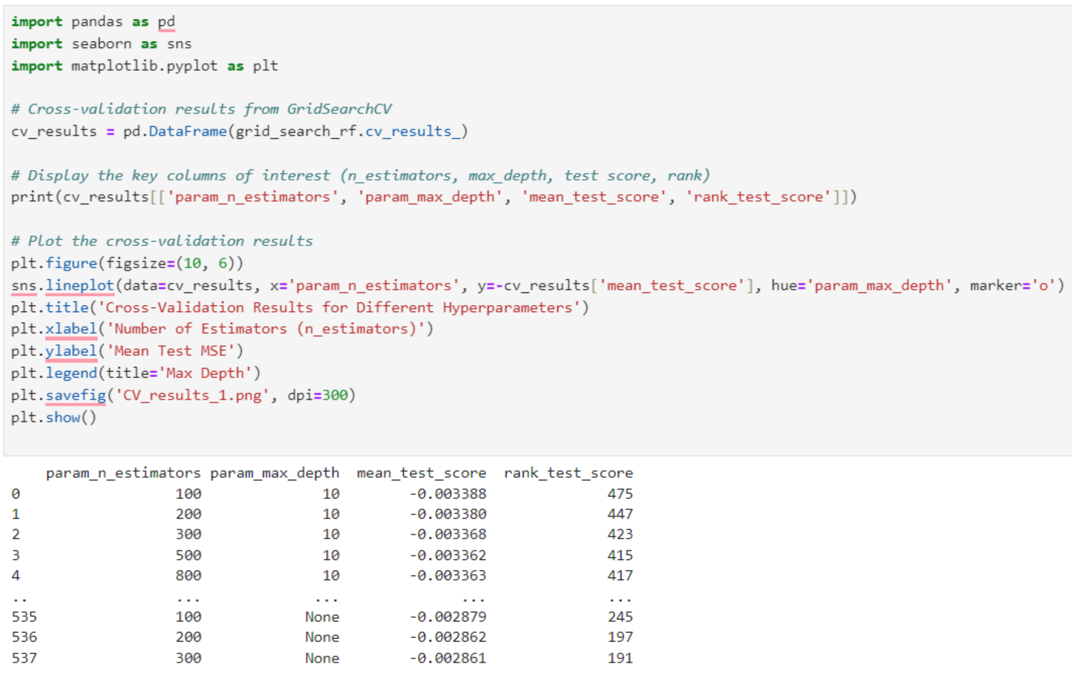
Figure 3.21 Data split

3.5.2. Hyperparameter Tuning:

We used **GridSearchCV** to fine-tune key Random Forest hyperparameters, such as the number of trees **(n_estimators),** tree depth **(max_depth)**, minimum samples for splits and leaf nodes, and max features **(max_features)** (Hastie et al., 2009). **GridSearchCV** tested these parameters with 5-fold cross-validation, training on four folds and validating on the fifth. The best configuration was: **bootstrap=False**, **max_depth=None, max_features='sqrt', min_samples_leaf=1, min_samples_split=2, n_estimators=800**. This reduced **MSE to 0.0027** with processing time around 1417 seconds.

3.5.3. Cross-Validation:

We applied 5-fold cross-validation during hyperparameter tuning to ensure generalization, a common practice in machine learning to minimize overfitting (James et al., 2013). By dividing data into five parts and iterating through folds, the model learns from diverse subsets, which improves its accuracy on new data.



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

# Cross-validation results from GridSearchCV
cv_results = pd.DataFrame(grid_search_rf.cv_results_)

# Display the key columns of interest (n_estimators, max_depth, test score, rank)
print(cv_results[['param_n_estimators', 'param_max_depth', 'mean_test_score', 'rank_test_score']])

# Plot the cross-validation results
plt.figure(figsize=(10, 6))
sns.lineplot(data=cv_results, x='param_n_estimators', y=-cv_results['mean_test_score'], hue='param_max_depth', marker='o')
plt.title('Cross-Validation Results for Different Hyperparameters')
plt.xlabel('Number of Estimators (n_estimators)')
plt.ylabel('Mean Test MSE')
plt.legend(title='Max Depth')
plt.savefig('CV_results_1.png', dpi=300)
plt.show()
```

```
    param_n_estimators param_max_depth  mean_test_score  rank_test_score
0                  100              10        -0.003388              475
1                  200              10        -0.003380              447
2                  300              10        -0.003368              423
3                  500              10        -0.003362              415
4                  800              10        -0.003363              417
..                 ...             ...              ...              ...
535                100            None        -0.002879              245
536                200            None        -0.002862              197
537                300            None        -0.002861              191
```

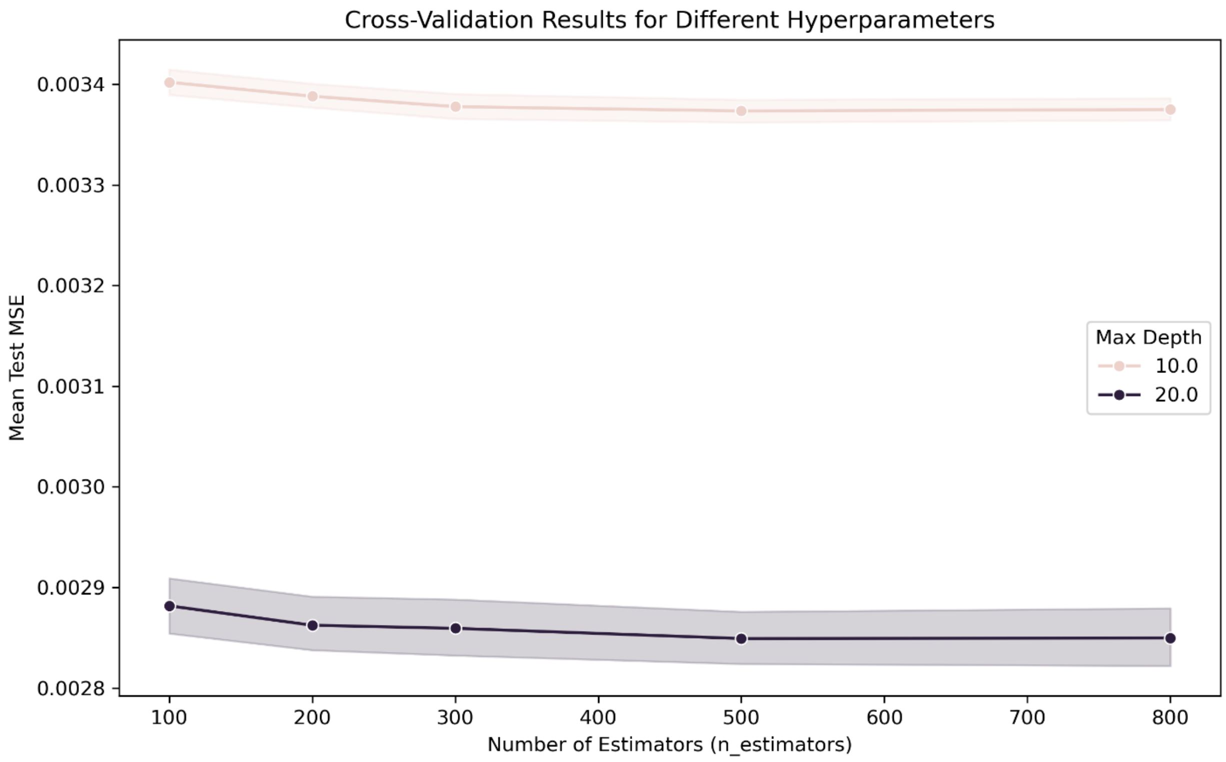


Figure 3.22 Cross-Validation results for different Hyperparameters

Cross-Validation Results Plot:

We plotted cross-validation results to visualize the impact of hyperparameters on model performance, guiding the selection of the optimal configuration.

Summary:

- **Data Split:** We split the data 80/20 into training and test sets for a balanced evaluation.
- **Hyperparameter Tuning:** GridSearchCV with 5-fold cross-validation was used to find the best Random Forest hyperparameters.
- **Cross-Validation:** This ensured the model avoided overfitting and performed well on unseen data.

These steps balanced accuracy with generalization, minimizing overfitting.

3.6. Model Evaluation Metrics

We assessed AQI prediction accuracy with two key metrics: **Mean Squared Error (MSE)** and **R-squared (R^2)**, both of which are standard for regression tasks.

3.6.1. Mean Squared Error (MSE)

MSE measures the average squared difference between actual and predicted values, giving more weight to larger errors (James et al., 2013).

$$\boldsymbol{MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Where $\boldsymbol{y_i}$ is the actual value, $\boldsymbol{\hat{y}_i}$ is the predicted value, and $\boldsymbol{n}$ is the total number of observations.

Justification:
MSE measures prediction accuracy directly. Our Random Forest model achieved an **MSE of 0.00324** on the test set, indicating high accuracy.

3.6.2. R-squared (R^2)

Definition:
R^2, measures the variance in AQI explained by the model, with values closer to 1 indicating a better fit:

$$\boldsymbol{R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y}_i)^2}}$$

Where $\boldsymbol{y_i}$ is the actual value, $\boldsymbol{\hat{y}_i}$ is the predicted value, and $\boldsymbol{\bar{y}}$ is the mean of the actual values.

Justification:
R^2 evaluates how well the model captures data patterns. Our model scored an R^2 of 0.9496, explaining nearly 95% of AQI variance, making it a strong predictor.

Conclusion:

- **Test Set MSE: 0.00324**, reflecting small prediction errors.
- **Test Set R^2: 0.950**, showing the model captures most AQI variance.

These metrics confirmed that Random Forest outperformed simpler models, making it the best choice for accurate AQI prediction.

3.7 Implementation Details

Python and the following libraries powered the project:

- **pandas** and **NumPy:** For data manipulation and analysis.
- **scikit-learn:** Used for model training, feature selection, scaling, and tuning with GridSearchCV (Pedregosa et al., 2011). Calculated metrics such as MSE and R^2.
- **Matplotlib** and **Seaborn:** For visualizations like heatmaps and scatter plots, aiding data interpretation.
- **time:** Measured process durations like hyperparameter tuning.
- **OS:** Managed file handling tasks, saving datasets and visualizations.

Hardware:
The project ran on a high-performance setup:

- **Processor:** AMD Ryzen 9 7950X (16 cores, 4501 MHz) enabled fast processing.
- **Memory:** 64 GB RAM ensured smooth handling of large datasets.
- **Graphics Card:** NVIDIA GeForce RTX 4080 SUPER accelerated model training.

This setup enabled efficient task handling from model training to hyperparameter optimization.

3.8 Assumptions and Limitations

Assumptions:

- **Data Accuracy:** The dataset accurately reflects regional air quality without unaccounted factors.
- **Feature Relevance:** Selected pollutants and meteorological features are assumed sufficient for predicting air quality.
- **Stationarity:** The model presumes that historical trends are consistent over time.
- **Data Integrity:** Data is assumed correctly recorded, with no errors from sensor malfunctions or manual input (WHO, 2021).
- **Potential Solution:** Regular sensor checks and data entry automation could reduce human errors and improve reliability.

Limitations:

1. **Data Quality:** Despite cleaning, missing values and sensor inaccuracies remain, possibly affecting performance.
    - **Solution:** Advanced imputation or external data (e.g., satellite imagery) could improve accuracy (Bishop, 2006).
2. **Imbalanced Classes:** Underrepresentation of some AQI categories (e.g., extreme pollution) may bias predictions.
    - **Solution:** SMOTE or weighted loss functions could help balance the dataset (Chawla et al., 2002).
3. **Model Overfitting:** Random Forest's complexity could still cause overfitting despite cross-validation.
    - **Solution:** Regularization and external dataset testing could improve generalization (Hastie et al., 2009).
4. **Temporal Aspects:** The model doesn't fully capture seasonality or daily patterns.
    - **Solution:** Time-series models, such as LSTMs, could better address temporal patterns (Hochreiter & Schmidhuber, 1997).
5. **Limited Features:** Lacks real-time data (e.g., weather patterns) and doesn't consider extreme events.
    - **Solution:** Adding real-time data and anomaly detection could improve response to unusual events (Aggarwal, 2013).

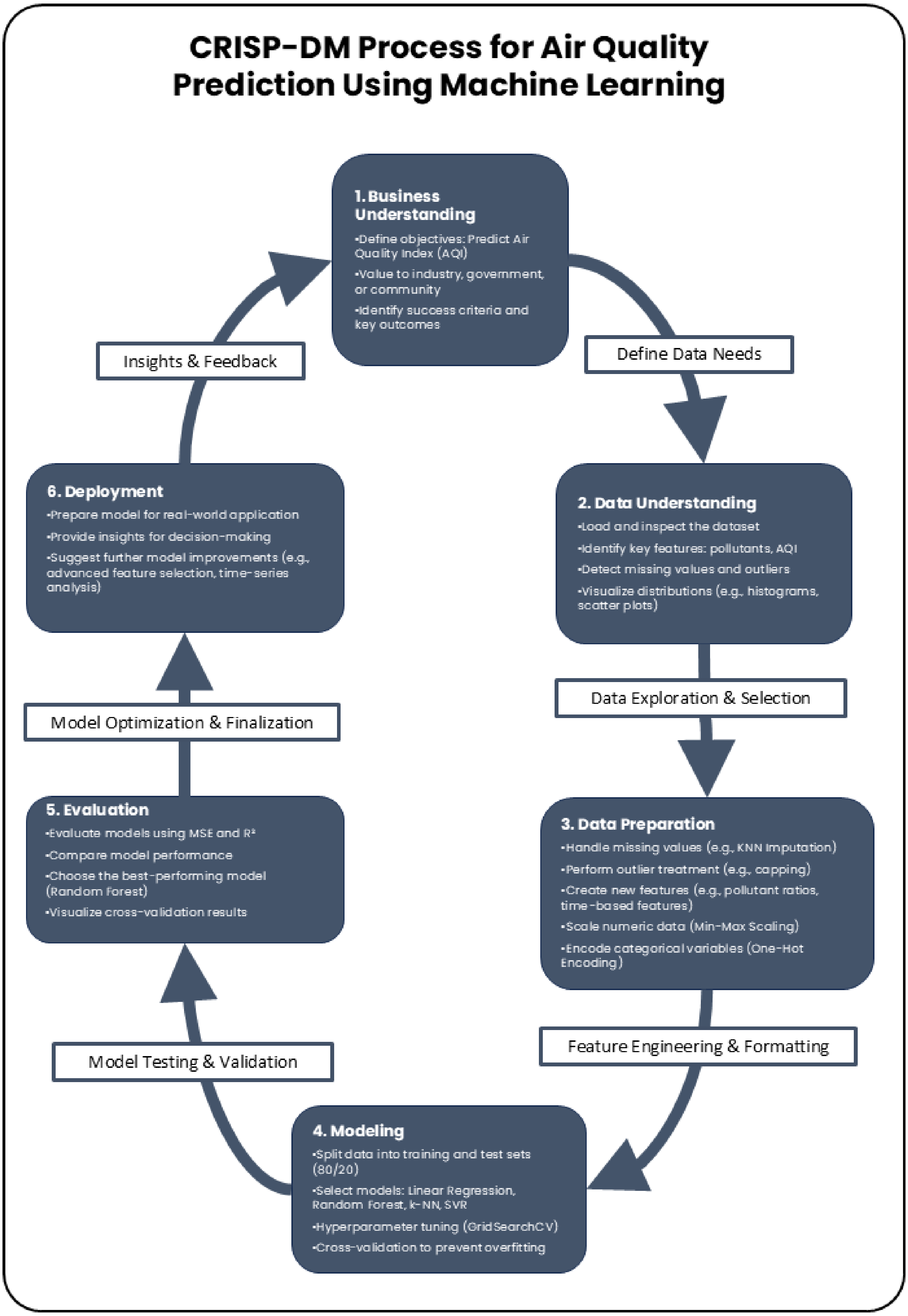

CRISP-DM Process for Air Quality Prediction Using Machine Learning
1. Business Understanding
•Define objectives: Predict Air Quality Index (AQI)
•Value to industry, government, or community
•Identify success criteria and key outcomes
Define Data Needs
2. Data Understanding
•Load and inspect the dataset
•Identify key features: pollutants, AQI
•Detect missing values and outliers
•Visualize distributions (e.g., histograms, scatter plots)
Data Exploration & Selection
3. Data Preparation
•Handle missing values (e.g., KNN Imputation)
•Perform outlier treatment (e.g., capping)
•Create new features (e.g., pollutant ratios, time-based features)
•Scale numeric data (Min-Max Scaling)
•Encode categorical variables (One-Hot Encoding)
Feature Engineering & Formatting
4. Modeling
•Split data into training and test sets (80/20)
•Select models: Linear Regression, Random Forest, k-NN, SVR
•Hyperparameter tuning (GridSearchCV)
•Cross-validation to prevent overfitting
Model Testing & Validation
5. Evaluation
•Evaluate models using MSE and R²
•Compare model performance
•Choose the best-performing model (Random Forest)
•Visualize cross-validation results
Model Optimization & Finalization
6. Deployment
•Prepare model for real-world application
•Provide insights for decision-making
•Suggest further model improvements (e.g., advanced feature selection, time-series analysis)
Insights & Feedback

Figure 3.23 CRISP-DM Process for Air Quality Prediction Using Machine

Section 4: Results

4.1 Model Performance Overview

We tested four models—Linear Regression, Random Forest, k-Nearest Neighbors (k-NN), and Support Vector Regression (SVR)—and evaluated their performance using Mean Squared Error (MSE) and R-squared (R^2) metrics (James et al., 2013). Below is a summary of results:

Model	MSE (Test Set)	R^2 (Test Set)	Best Hyperparameters
Linear Regression	0.00658	0.8996	None (Benchmark model)
Random Forest (n=100)	0.00341	0.9476	n_estimators=100, random_state=42
k-Nearest Neighbors (k=5)	0.00407	0.9374	n_neighbors=5
Support Vector Regression	0.00494	0.9240	kernel='rbf'
Random Forest (Grid Search)	**0.00324**	**0.9502**	**bootstrap=False, max_depth=None, max_features='sqrt', min_samples_leaf=1, min_samples_split=2, n_estimators=800**

Figure 4.1 Model Performance Comparison Based on MSE and R^2

4.2 Model Performance Evaluation

Linear Regression:

- **MSE:** 0.00658 | **R^2:** 0.8996
- **Observation:** Served as the benchmark, capturing 90% of the variance but struggled with non-linear patterns (Bishop, 2006).

Random Forest (n=100):

- **MSE:** 0.00341 | **R^2:** 0.9476
- **Observation:** Showed significant improvement over Linear Regression by capturing non-linear relationships through ensemble methods (Breiman, 2001).

k-Nearest Neighbors (k=5):

- **MSE:** 0.00407 | **R^2:** 0.9374
- **Observation:** Performed well but was less effective than Random Forest, struggling to capture broader patterns in the data (Hastie et al., 2009).

Support Vector Regression (SVR):

- **MSE:** 0.00494 | **R²:** 0.9240
- **Observation:** Solid performance but computationally expensive and less accurate than Random Forest or k-NN.

Random Forest (Grid Search with n=800):

- **MSE:** 0.00324 | **R²:** 0.9502
- **Observation:** After tuning, Random Forest outperformed all other models, capturing the most complex relationships in the data.

4.3 Hyperparameter Tuning and Cross-Validation

We optimized Random Forest using **GridSearchCV** with **5-fold cross-validation**, a common approach for model tuning (Pedregosa et al., 2011). The best hyperparameters were:

- n_estimators: 800
- max_depth: None
- max_features: sqrt
- bootstrap: False
- min_samples_split: 2
- min_samples_leaf: 1

Cross-validation results (refer to **Figure 3.21**) showed that increasing the number of estimators led to performance improvements until around 300 trees, after which gains plateaued.

4.4 Model Comparison

Random Forest, especially after hyperparameter tuning, clearly outperformed the other models. With an MSE of 0.0032377 and R^2 of 0.95024, it demonstrated significantly higher accuracy than Linear Regression and SVR. The combination of ensemble methods and hyperparameter tuning enabled it to best capture the complex relationships in the data.

4.5 Model Improvement Steps

To enhance model performance, we implemented several key steps:

- **Feature Engineering:** Added pollutant ratios and time-based variables to capture complex relationships (Géron, 2019).
- **Hyperparameter Tuning:** Optimized Random Forest with GridSearchCV, significantly boosting accuracy.
- **Data Preprocessing:** KNN imputation handled missing values, and Min-Max scaling normalized features, ensuring consistency.
- **Cross-Validation:** Applied 5-fold cross-validation to avoid overfitting and ensure generalizability.
- **Smoothing of Spikes:** Addressed post-imputation spikes using techniques like Gaussian smoothing and log transformation, resulting in cleaner feature distributions for training.

Conclusion

Random Forest emerged as the most accurate model, outperforming Linear Regression and SVR with the lowest error and highest accuracy. Effective feature engineering, rigorous hyperparameter tuning, and comprehensive data preprocessing contributed to achieving strong predictive performance, particularly for air quality forecasting.

Section 5: Discussions

The machine learning models performed well in predicting the air quality index (AQI), with the Random Forest model achieving the highest accuracy. Our goal was to create a reliable AQI prediction model, and the results confirm this achievement. The model's low Mean Squared Error (MSE) and high R^2 underscore its strength in capturing the complex relationships between pollutants and AQI (Breiman, 2001; James et al., 2013).

5.1 Significance of Findings

The model's predictive power is valuable for air quality monitoring and management. Its ability to provide early predictions can support public health measures, enabling governments and organizations to tackle pollution proactively (Krzyzanowski et al., 2021). Recognizing pollutant patterns, especially with PM2.5, NOx, and CO, helps stakeholders anticipate poor air quality days and mitigate health risks, especially in dense urban centers. By understanding pollutant interactions, policymakers can develop targeted interventions to improve air quality (Rautela & Goyal, 2024).

5.2 Comparison with Prior Studies

Compared to earlier studies, this model offers both improved accuracy and greater interpretability. Prior studies often relied on simpler models, like linear regression, which fail to capture complex interactions among pollutants (Gurjar et al., 2016). In contrast, our approach using Random Forest and cross-validation effectively models the multi-faceted relationships among pollutants and AQI, offering a more comprehensive prediction (Samad et al., 2023).

5.3 Limitations

Several limitations persist despite the positive outcomes:

- **Data Quality:** Imputation for missing values may introduce minor bias, and occasional sensor errors could impact data reliability (Sharma & Mauzerall, 2021).
- **Temporal Factors:** While basic time features were included, the model doesn't fully account for seasonal variations or unusual events.
- **Assumptions:** The assumption that historical trends will continue may not hold if new policies or economic changes alter future patterns.
- **Feature Smoothing:** Spikes in some features remained post-imputation, suggesting further smoothing could enhance consistency

5.4 Practical Applications

This model offers various practical applications:

- **Policymakers:** Use predictions to issue early warnings and implement targeted emission controls.
- **Businesses:** Industries can monitor their environmental impact and adjust operations to meet pollution regulations.
- **Public Health Organizations:** Anticipate poor air quality periods and issue alerts for vulnerable populations.
- **Urban Planners:** Use insights to design pollution-mitigating strategies in city planning.

In summary, this study shows the Random Forest model's ability to predict AQI effectively, offering valuable insights and applications for public health, policy, and urban planning. This model provides a robust tool for managing air quality and protecting public health.

Conclusion

This study demonstrates the Random Forest model's effectiveness in AQI prediction, providing practical insights for public health, policy, and urban planning. The model serves as a valuable tool for managing air quality and safeguarding public health.

Appendix

GitHub Repository

All code, data preprocessing steps, and plot generation scripts are available on GitHub. The code prepares the dataset comprehensively, handling cleaning, processing, and feature engineering for analysis and modeling. Additionally, all plots used in the report (such as box plots, scatter plots, histograms, and correlation matrices) are automatically generated and saved in the same directory upon running the code.

Link to GitHub Repository
AQI Data Analysis Project: https://github.com/whiZZiac/AQI-Data-Analysis/tree/main

References

1. Aggarwal, C.C., 2013. *Outlier Analysis*. Springer. Available at: https://download.e-bookshelf.de/download/0003/9468/94/L-G-0003946894-0013327450.pdf [PDF, Accessed 15 Oct. 2024].
2. Bishop, C.M., 2006. *Pattern Recognition and Machine Learning*. Springer. Available at: https://www.academia.edu/44025931/Pattern_recognition_and_Machine_learning [PDF, Accessed 15 Oct. 2024].
3. Breiman, L., 2001. Random Forests. *Machine Learning*, 45(1), pp.5–32. Available at: https://link.springer.com/article/10.1023/A:1010933404324 [PDF, Accessed 15 Oct. 2024].
4. Chawla, N.V., Bowyer, K.W., Hall, L.O. & Kegelmeyer, W.P., 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16, pp.321–357. Available at: https://www.jair.org/index.php/jair/article/view/10302 [PDF, Accessed 20 Oct. 2024].

5. Géron, A., 2019. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2nd ed. O'Reilly Media, Inc. Available at: https://github.com/yanshengjia/ml-road/blob/master/resources/Hands%20On%20Machine%20Learning%20with%20Scikit%20Learn%20and%20TensorFlow.pdf [PDF, Accessed 12 Oct. 2024].

6. Government of India, 2020. Open Government Data (OGD) Platform India: *Air Quality Data in India (2015-2020)*. Available at: https://data.gov.in/ [Accessed 15 Oct. 2024].

7. Gurjar, B.R. & Nagpure, A.S., 2015. Indian megacities as localities of environmental vulnerability from an air quality perspective. *Sustainable Cities and Society*. Available at: https://www.researchgate.net/publication/283644948_Indian_megacities_as_localities_of_environmental_vulnerability_from_air_quality_perspective [PDF, Accessed 15 Oct. 2024].

8. Gurjar, B.R., Ravindra, K. & Nagpure, A.S., 2016. Air pollution trends over Indian megacities and their local-to-global implications. *Atmospheric Environment*, 142, pp.475-495. Available at: https://www.researchgate.net/publication/304037258_Air_pollution_trends_over_Indian_megacities_and_their_local-to-global_implications [PDF, Accessed 8 Oct. 2024].

9. Hastie, T., Tibshirani, R. & Friedman, J., 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. New York: Springer. Available at: https://freecomputerbooks.com/The-Elements-of-Statistical-Learning.html [PDF, Accessed 15 Oct. 2024].

10. James, G., Witten, D., Hastie, T. & Tibshirani, R., 2013. *An Introduction to Statistical Learning: With Applications in R*. New York: Springer. Available at: https://static1.squarespace.com/static/5ff2adbe3fe4fe33db902812/t/6009dd9fa7bc363aa822d2c7/1611259312432/ISLR+Seventh+Printing.pdf [PDF, Accessed 10 Oct. 2024].

11. Krzyzanowski, M., Apte, J.S., Bhanarkar, A.D., Cohen, A.J., Dey, S., Guttikunda, S.K., & Balakrishnan, K., 2021. Ambient air pollution in India: Current status and the way forward. *Environmental Health Perspectives*, 129(1), pp.15001. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7805008/ [Accessed 8 Oct. 2024].

12. Kotsiantis, S.B., 2007. Supervised Machine Learning: A Review of Classification Techniques. *Informatica*, 31(3), pp.249-268. Available at: https://datajobs.com/data-science-repo/Supervised-Learning-[SB-Kotsiantis].pdf [PDF, Accessed 10 Oct. 2024].

13. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., & Duchesnay, E., 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, pp.2825-2830. Available at: https://jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf [PDF, Accessed 15 Oct. 2024].

14. Rautela, K.S. & Goyal, M.K., 2024. Transforming air pollution management in India with AI and machine learning technologies. *Scientific Reports*, 14, Article 20412. Available at: https://www.nature.com/articles/s41598-024-71269-7 [Accessed 25 Oct. 2024].

15. Samad, A., Garuda, S., Vogt, U., & Yang, B., 2023. Air pollution prediction using machine learning techniques – An approach to replace existing monitoring stations with virtual monitoring stations. *Atmospheric Environment*. Available at: https://www.sciencedirect.com/science/article/pii/S1352231023004132 [PDF, Accessed 8 Oct. 2024].

16. Sharma, S. & Mauzerall, D.L., 2021. Analysis of Air Pollution Data in India (2015-2019): Insights from Continuous and Manual Monitoring. *Environmental Science and Policy*, 120, pp.100-112. Available at: https://aaqr.org/articles/aaqr-21-08-oa-0204.pdf [PDF, Accessed 15 Oct. 2024].

17. Tikhe, S.S., Khare, K.C. & Londhe, S.N., 2013. Forecasting Criteria Air Pollutants Using Data-Driven Approaches: An Indian Case Study. *IOSR Journal of Environmental Science, Toxicology and Food*

Technology, 9(7), pp.60-67. Available at: https://www.iosrjournals.org/iosr-jestft/papers/vol3-issue5/A0350108.pdf?id=3710 [PDF, Accessed 15 Oct. 2024].

18. World Health Organization (WHO), 2021. *WHO global air quality guidelines: particulate matter (PM2.5 and PM10), ozone, nitrogen dioxide, sulfur dioxide and carbon monoxide*. Geneva: WHO. Available at: https://www.who.int/publications/i/item/9789240034228 [Accessed 15 Oct. 2024].

19. Yang, F., Tan, J., Shi, C., Wang, J., Zhang, Y., Song, J., & Zhang, R., 2021. Spatio-temporal analysis of urban air pollutants throughout China during 2014–2019. *Air Quality, Atmosphere & Health*, 14, pp.1619–1632. Available at: https://link.springer.com/article/10.1007/s11869-021-01043-5 [Accessed 8 Oct. 2024].

20. Zou, H., Hastie, T. & Tibshirani, R., 2006. Sparse Principal Component Analysis. *Journal of Computational and Graphical Statistics*, 15(2), pp.265-286. Available at: https://www.tandfonline.com/doi/abs/10.1198/106186006X113430 [Accessed 15 Oct. 2024].