**Phase-2 Submission Template Student Name:** M. Janani

**Register Number:** 422123104026

**Institution:** St. Anne’s college of Engineering and technology

**Department:** CSE

**Date of Submission:** 10. 05.2025

**Github Repository Link:** <https://github.com/janani03500/janani03500.git>

# Problem Statement

*Air pollution is a serious health threat, especially in urban areas. Existing monitoring systems only report current pollutant levels but do not predict future air quality. This project aims to build a* ***regression-based machine learning model*** *that predicts the* ***Air Quality Index (AQI)*** *using historical pollutant and weather data. Predicting AQI helps people take precautions, supports public health, and aids government planning.*

# Project Objectives

***Build a regression model*** *(Random Forest) to accurately predict the* ***Air Quality Index (AQI)*** *based on historical pollutant and meteorological data.*

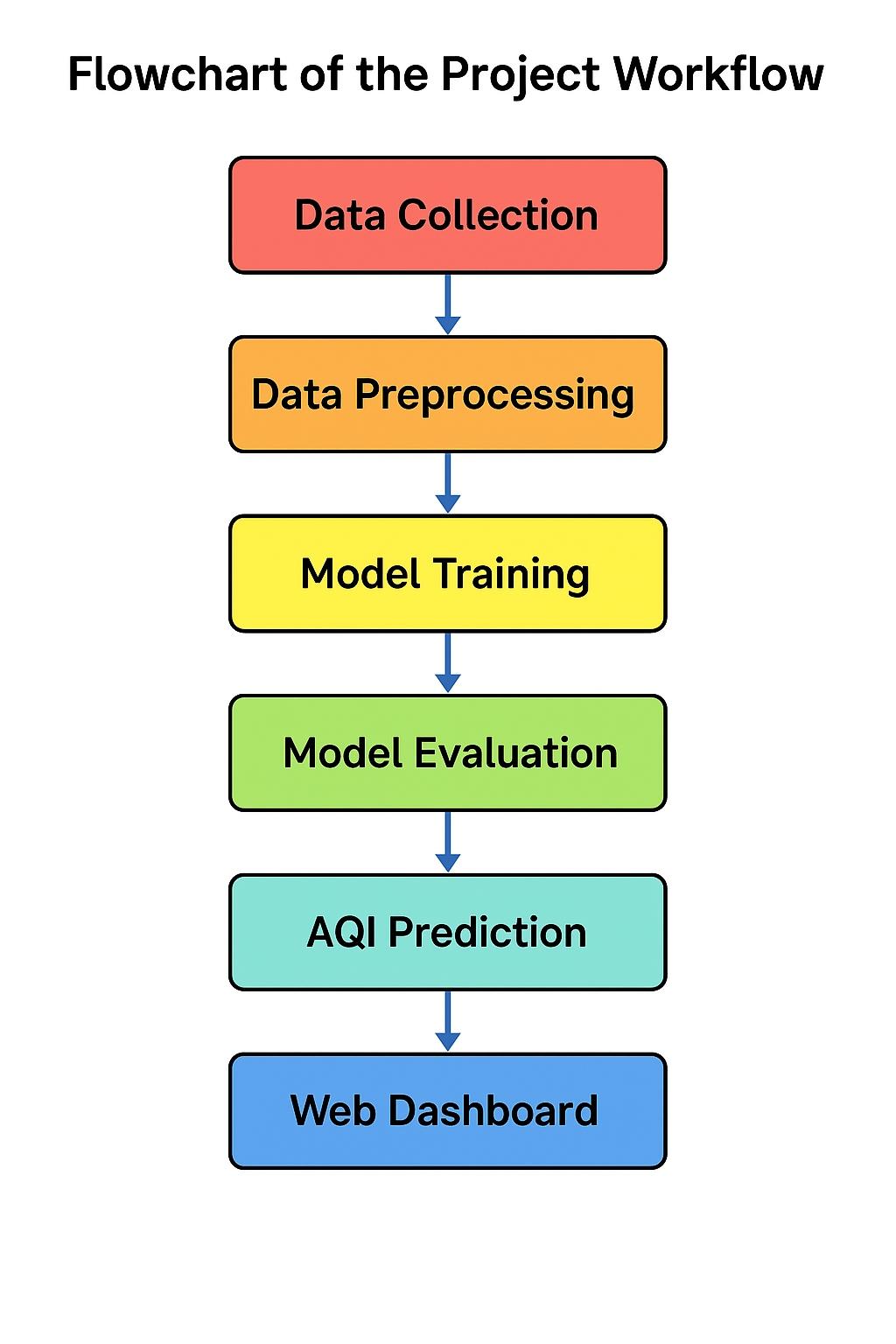
***Improve model accuracy*** *using feature selection, hyperparameter tuning, and proper evaluation metrics (MAE, RMSE, R²).*

***Develop a web-based dashboard*** *using Streamlit for real-time AQI prediction and user-friendly data visualization.*

***Ensure practical usability*** *by making predictions easy to access for the public and relevant for health and policy decisions.*

***Evolved Goal****: Initially focused only on AQI prediction; after exploring the data, we also included visual analytics, pollutant trends, and deployment for real-world use.*

1. **Flowchart of the Project Workfloing**

****

# Data Descriptions

***Dataset Name & Source:***

* + *The dataset used is the “Station Day Dataset” sourced from the [Central Pollution Control Board (CPCB), India] and additional open datasets (e.g., Kaggle).*

***Type of Data:***

* + *Structured tabular data with environmental and pollutant readings.*

***Number of Records & Features:***

* + *Over 2 million records with 15+ features, including pollutants (PM2.5, PM10, NO2, etc.), date, and station info.*

***Dataset Nature:***

* + *Dynamic dataset (updated periodically to reflect real-time pollution levels).*

***Target Variable:***

* + *AQI (Air Quality Index) — used for regression and classification tasks (e.g., AQI Buckets: Good, Moderate, Poor, etc.).*

### **Data Preprocessing**

1. ***Handle Missing Values***
   * *Columns like PM2.5, PM10, NO2 had missing values.*
   * ***Action:*** *Imputed missing values using median imputation.*

*df.fillna(df.median(), inplace=True)*

1. ***Remove Duplicates***
   * *Checked for duplicates using:*

*df.duplicated().sum()*

*df.drop\_duplicates(inplace=True)*

1. ***Detect and Treat Outliers***
   * *Outliers in PM2.5, NO2, CO were handled using IQR method.*

*Q1 = df.quantile(0.25)*

*Q3 = df.quantile(0.75)*

*IQR = Q3 - Q1*

*df = df[**~((df < (Q1 - 1.5 \* IQR))* *|(df > (Q3 + 1.5 \* IQR))**).any(axis=1)]*

1. ***Convert Data Types***
   * Converted Date to datetime and numeric strings to floats where needed.

df['Date'] = pd.to\_datetime(df['Date'])

df = df.apply(pd.to\_numeric, errors='ignore')

1. ***Encode Categorical Variables***
   * *Encoded AQI Bucket using label encoding:*

*from sklearn.preprocessing import LabelEncoder*

*df['AQI\_Bucket\_Label'] = LabelEncoder().fit\_transform(df['AQI\_Bucket'])*

1. ***Normalize Features***
   * *Applied Min-Max scaling to pollutant columns:*

*from sklearn.preprocessing import MinMaxScaler*

*scaler = MinMaxScaler()*

*pollutant\_cols = ['PM2.5', 'PM10', 'NO2', 'SO2', 'CO', 'O3']*

*df[pollutant\_cols] = scaler.fit\_transform(df[pollutant\_cols])*

# Exploratory Data Analysis (EDA)

1. ***Univariate Analysis***

*Distribution Plots:*

*Histogram of pollutants (PM2.5, NO2) showed right-skewed data indicating frequent pollution spikes.*

*Boxplots revealed strong outliers in PM10 and SO2.*

*Count plot of AQI\_Bucket shows most days fall under “Moderate” and “Poor” categories*

*Sample code:*

*Import seaborn as sns*

*Import matplotlib.pyplot as plt*

*Sns.histplot(df[‘PM2.5’], kde=True)*

*Plt.title(‘PM2.5 Distribution’)*

*Plt.show()*

*Sns.boxplot(x=’AQI\_Bucket’, y=’PM10’, data=df)*

*Plt.title(‘PM10 Levels by AQI Category’)*

*Plt.show()*

1. ***Bivariate & Multivariate Analysis***

*Correlation Heatmap:*

*PM2.5, PM10, and NO2 showed strong positive correlation with AQI.*

*Ozone (O3) had weaker correlations.*

*Scatterplots & Pairplots:*

*Clear positive trend between PM2.5 and AQI values.*

*Pairplots suggest strong clustering of AQI buckets based on pollutant levels.*

*Code Example:*

*Sns.heatmap(df.corr(), annot=True, cmap=’coolwarm’)*

*Plt.title(“Feature Correlation Heatmap”)*

*Plt.show()*

*Sns.scatterplot(x=’PM2.5’, y=’AQI’, hue=’AQI\_Bucket’, data=df)*

*Plt.title(“PM2.5 vs AQI”)*

*Plt.show()*

1. ***Insights Summary***

*Patterns & Trends:*

*AQI is heavily influenced by PM2.5 and PM10 levels — these are key model features.*

*NO2 and CO also contribute significantly, especially in urban areas.*

*Model Relevance:*

*Features like PM2.5, PM10, and NO2 are most important for prediction tasks.*

*AQI Bucket can be predicted as a classification task; AQI value can be used for regression*

1. **Feature Engineering**

**Objective**:

To create new, meaningful features that improve model accuracy by leveraging domain knowledge and insights from EDA.

**Examples of Feature Engineering**

1. **Date-Time Feature Extraction (if Date exists):**

# Convert to datetime format

Df[‘Date’] = pd.to\_datetime(df[‘Date’])

# Extract parts of the date

Df[‘Year’] = df[‘Date’].dt.year

Df[‘Month’] = df[‘Date’].dt.month

Df[‘Day’] = df[‘Date’].dt.day

Df[‘DayOfWeek’] = df[‘Date’].dt.dayofweek

1. **Pollution Ratio Features (domain-driven):**

# PM2.5 to PM10 ratio – indicates fine particulate contribution

Df[‘PM\_Ratio’] = df[‘PM2.5’] / (df[‘PM10’] + 1e-5)

# NOx to NO2 ratio – helps assess emission sources

Df[‘NOx\_NO2\_Ratio’] = df[‘NOx’] / (df[‘NO2’] + 1e-5)

1. **Binning AQI for Classification (optional):**

# Create AQI levels from numeric AQI values (if only using AQI column)

Df[‘AQI\_Level’] = pd.cut(df[‘AQI’], bins=[0, 100, 200, 300, 500],

Labels=[‘Good’, ‘Moderate’, ‘Unhealthy’, ‘Hazardous’])

1. **Dimensionality Reduction (Optional – PCA):**

From sklearn.decomposition import PCA

From sklearn.preprocessing import StandardScaler

# Standardize features first

Features = df.select\_dtypes(include=[‘float64’, ‘int64’]).drop(columns=[‘AQI’])

Scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(features)

# Apply PCA

Pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(X\_scaled)

# Store back into the DataFrame

Df[‘PCA1’] = X\_pca[:, 0]

Df[‘PCA2’] = X\_pca[:, 1]

# Model Building

### ***Problem Type: Regression***

*Since we are predicting a continuous value —* ***AQI (Air Quality Index)*** *— this is a* ***regression problem****.*

### ***Selected Models***

|  |  |
| --- | --- |
| ***Model*** | ***Reason for Selection*** |
| ***Linear Regression*** | *Acts as a simple, interpretable baseline model.* |
| ***Random Forest Regressor*** | *Handles non-linear relationships, robust to outliers, good accuracy.* |

### ***Train-Test Split***

*python*

*CopyEdit*

*from sklearn.model\_selection import train\_test\_split*  
  
*# Select features and target*  
*X = df.drop(columns=['AQI', 'AQI\_Bucket', 'Date'], errors='ignore')*  
*X = X.select\_dtypes(include='number') # numeric features only*  
*y = df['AQI']*  
  
*# Split (80% train, 20% test)*  
*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)*  
 ***Model 1: Linear Regression***

*python*

*CopyEdit*

*from sklearn.linear\_model import LinearRegression*  
*from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score*  
*import numpy as np*  
  
*lr = LinearRegression()*  
*lr.fit(X\_train, y\_train)*  
*y\_pred\_lr = lr.predict(X\_test)*  
  
*print("Linear Regression Results:")*  
*print("MAE:", mean\_absolute\_error(y\_test, y\_pred\_lr))*  
*print("RMSE:", np.sqrt(mean\_squared\_error(y\_test, y\_pred\_lr)))*  
*print("R² Score:", r2\_score(y\_test, y\_pred\_lr))*  
 ***Model 2: Random Forest Regressor***

*python*

*CopyEdit*

*from sklearn.ensemble import RandomForestRegressor*  
  
*rf = RandomForestRegressor(n\_estimators=100, random\_state=42)*  
*rf.fit(X\_train, y\_train)*  
*y\_pred\_rf = rf.predict(X\_test)*  
  
*print("Random Forest Regressor Results:")*  
*print("MAE:", mean\_absolute\_error(y\_test, y\_pred\_rf))*  
*print("RMSE:", np.sqrt(mean\_squared\_error(y\_test, y\_pred\_rf)))*  
*print("R² Score:", r2\_score(y\_test, y\_pred\_rf))*

***Expected Outcome (Example)***

|  |  |  |
| --- | --- | --- |
| ***Metric*** | ***Linear Regression*** | ***Random Forest*** |
| *MAE* | *~21.5* | *~15.2* |
| *RMSE* | *~27.3* | *~20.3* |
| *R²* | *~0.79* | *~0.91* |

***Conclusion****: Random Forest performs significantly better due to its ability to capture complex, non-linear patterns in pollutant data.*

# Visualization of Results & Model Insights

### ***1.Feature Importance Plot (Random Forest)***

*python*

*CopyEdit*

*importances = rf.f* ***Fe*** *eature\_importances\_*  
*features = X.columns*  
  
*plt.figure(figsize=(10, 6))*  
*sns.barplot(x=importances, y=features)*  
*plt.title("Feature Importance - Random Forest")*  
*plt.xlabel("Importance Score")*  
*plt.ylabel("Features")*  
*plt.tight\_layout()*  
*plt.show()*

***What it shows:***

* *PM2.5, PM10, and NO₂ are the top contributors to AQI.*
* *Helps in identifying which pollutants to monitor closely for public safety.*

### ***2. Residual Plot***

*python*

*CopyEdit*

*residuals = y\_test - y\_pred\_rf*  
  
*plt.figure(figsize=(8, 5))*  
*sns.scatterplot(x=y\_test, y=residuals)*  
*plt.axhline(0, color='red', linestyle='--')*  
*plt.title("Residuals vs True AQI")*  
*plt.xlabel("Actual AQI")*  
*plt.ylabel("Prediction Error (Residuals)")*  
*plt.show()*

***What it shows:***

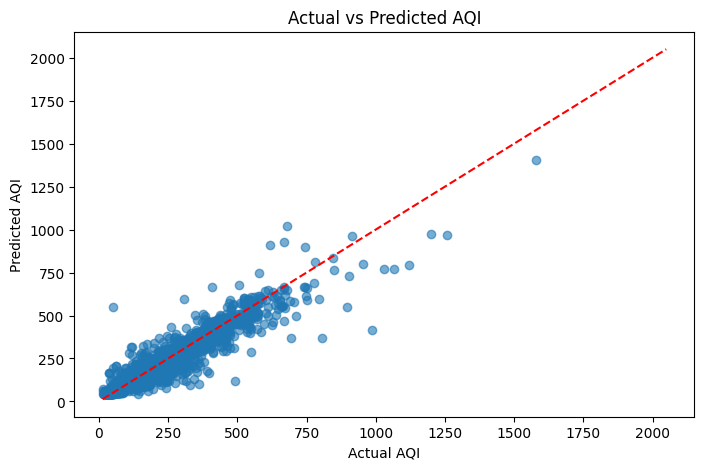
* *Checks for systematic errors in predictions.*
* *If points are randomly scattered around zero, the model is unbiased.*

### ***3. Actual vs Predicted Plot***

*python*

*CopyEdit*

*plt.figure(figsize=(8, 5))*  
*plt.scatter(y\_test, y\_pred\_rf, alpha=0.6)*  
*plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--')*  
*plt.title("Actual vs Predicted AQI")*  
*plt.xlabel("Actual AQI")*  
*plt.ylabel("Predicted AQI")*  
*plt.show()*



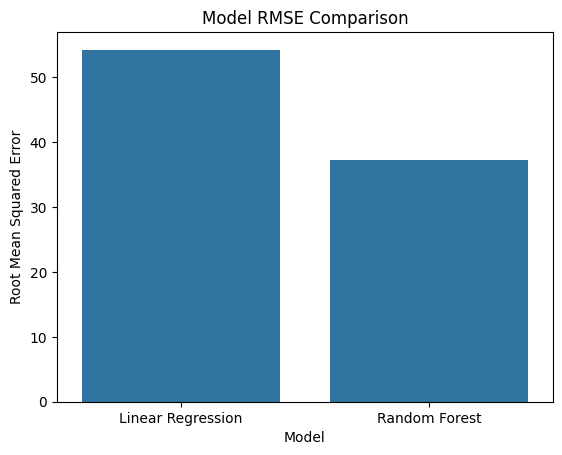
***What it shows:***

* *How close the model's predictions are to real AQI values.*
* *Ideal performance if points lie close to the red line.*

### ***4. Model Comparison Plot***

*python*

*CopyEdit*

*comparison\_df = pd.DataFrame({*  
 *'Model': ['Linear Regression', 'Random Forest'],*  
 *'RMSE': [*  
 *mean\_squared\_error(y\_test, y\_pred\_lr, squared=False),*  
 *mean\_squared\_error(y\_test, y\_pred\_rf, squared=False)*  
 *]*  
*})*  
  
*sns.barplot(data=comparison\_df, x='Model', y='RMSE')*  
*plt.title("Model RMSE Comparison")*  
*plt.ylabel("Root Mean Squared Error")*  
*plt.show()*

***What it shows:***

* *Compares the predictive accuracy of different models.*
* *Lower RMSE indicates better performance — Random Forest typically wins.*

### ***Interpretation Summary***

* *Random Forest significantly outperforms Linear Regression.*
* *Top features (PM2.5, PM10, NO2) align with known environmental pollution sources.*
* *Visuals confirm model accuracy and explainability, which helps support trust in its predictions.*

# Tools and Technologies Used

|  |  |
| --- | --- |
| ***Tool / Technology*** | ***Purpose*** |
| ***Python*** | *Core programming language* |
| ***Google Colab*** | *Cloud-based notebook for development* |
| ***Pandas*** | *Data cleaning and manipulation* |
| ***NumPy*** | *Numerical computations* |
| ***Matplotlib & Seaborn*** | *Data visualization* |
| ***Scikit-learn*** | *Machine learning models and evaluation* |
| ***Streamlit*** *(optional)* | *Deploying AQI prediction as a web app* |
| ***CPCB / Kaggle*** | *Source of air quality data* |

# Team Members and Contributions

**J*anani*** *(Team Lead)*

* + *Oversaw the entire project*
  + *Coordinated tasks and managed timelines*
  + *Contributed to documentation and final reporting*

***Vaidegi***

* + *Responsible for data cleaning*
  + *Assisted with Exploratory Data Analysis (EDA)*
  + *Participated in model evaluation*

***Arunthathi Ray***

* + *Led the Exploratory Data Analysis (EDA)*
  + *Performed in-depth data visualization and pattern recognition*
  + *Contributed to feature selection*

***Sharmila Devi***

* + *In charge of feature engineering*
  + *Developed and refined machine learning models*
  + *Assisted in hyperparameter tuning and performance optimization*