****

**Assessment Report**

on

**“Air Quality Index Prediction”**

submitted as partial fulfillment for the award of

**BACHELOR OF TECHNOLOGY**

**DEGREE**

SESSION 2024-25

in

**CSE(AIML) – GROUP 3**

Bhavya arora-72

Aditya Kumar-15

Abdul Rafey-3

Ayush Raj-66

Aditya Kumar-16

**Introduction**

**Air pollution is one of the most critical environmental challenges faced by modern urban societies, and India is no exception. Rapid industrialization, population growth, and increased vehicular emissions have led to a significant decline in air quality across many Indian cities. Prolonged exposure to polluted air is directly linked to respiratory diseases, cardiovascular conditions, and other serious health issues, making air quality monitoring and forecasting a national priority.**

**Project Objective**

**The objective of this project is to build a regression model using machine learning techniques, specifically the Random Forest algorithm, to predict AQI values based on environmental features collected from Indian cities. In addition to prediction, the project also includes visual analysis to illustrate the regional distribution of air pollution and the influence of individual pollutants on air quality.**

**This project is based on data from the Central Pollution Control Board (CPCB), consisting of daily air quality measurements from multiple cities. By applying machine learning models, we aim to derive meaningful insights and build an effective predictive tool that can aid in both analysis and policy planning.**

**Methodology**

**To predict the Air Quality Index (AQI) effectively, a systematic data science workflow was followed. The methodology includes multiple phases, each designed to prepare the data, build a robust model, and extract meaningful insights.**

**1. Data Collection**

**The dataset used in this project, city\_day.csv, was sourced from the Central Pollution Control Board (CPCB) of India. It contains daily air quality measurements for multiple Indian cities, including concentrations of key pollutants and AQI values.**

**2. Data Preprocessing**

**Preprocessing was a crucial step to ensure data quality and model accuracy. The following actions were taken:**

**Missing Value Handling: Records with missing AQI values were removed since AQI is the target variable. Remaining missing pollutant values were also excluded to avoid bias.**

**Feature Selection: Nine environmental features were selected as predictors based on their relevance:**

**PM2.5, PM10, NO, NO₂, NOx, NH₃, CO, SO₂, and O₃**

**Data Cleaning: Irrelevant columns such as Date, City, and AQI\_Bucket were dropped to focus on numerical features.**

**3. Model Selection**

**The project uses the Random Forest Regression algorithm, a powerful ensemble learning technique known for its robustness, accuracy, and ability to handle non-linear data patterns.**

**Why Random Forest?**

**Handles large datasets with high dimensionality**

**Resistant to overfitting**

**Provides feature importance, which helps in interpretability**

**4. Model Training & Evaluation**

**The cleaned dataset was split into training (80%) and testing (20%) sets. The Random Forest model was trained on the training data, and predictions were made on the test set.**

**Evaluation Metrics:**

**Mean Absolute Error (MAE): Measures average absolute difference between predicted and actual AQI values.**

**R² Score: Measures how well the model explains the variability of the AQI.**

**The model achieved:**

**MAE: ≈ 14.48**

**R² Score: ≈ 0.935**

**These metrics indicate that the model performs well and can reliably predict AQI values.**

**CODE :**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**# Preprocessing**

**from sklearn.preprocessing import LabelEncoder, StandardScaler**

**from sklearn.model\_selection import train\_test\_split**

**# Models**

**from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier**

**# Metrics**

**from sklearn.metrics import mean\_squared\_error, r2\_score**

**from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, classification\_report, confusion\_matrix, ConfusionMatrixDisplay**

**# Load datasets**

**df\_city\_hour = pd.read\_csv('/content/city\_hour.csv')**

**df\_station\_hour = pd.read\_csv('/content/station\_hour.csv')**

**df\_station\_day = pd.read\_csv('/content/station\_day.csv')**

**df\_stations = pd.read\_csv('/content/stations.csv')**

**# Display initial rows**

**print(df\_station\_day.head())**

**print(df\_stations.head())**

**print(df\_city\_hour.head())**

**print(df\_station\_hour.head())**

**# Make a copy of the station\_day dataset**

**df = df\_station\_day.copy()**

**print("Initial shape:", df.shape)**

**# Drop rows with missing AQI**

**df.dropna(subset=['AQI'], inplace=True)**

**# Fill missing values with column means**

**df.fillna(df.mean(numeric\_only=True), inplace=True)**

**# Features and target**

**features = ['PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3',**

**'CO', 'SO2', 'O3', 'Benzene', 'Toluene']**

**X = df[features]**

**y = df['AQI']**

**# Train-test split**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

**# Train model**

**reg\_model = RandomForestRegressor(n\_estimators=100, random\_state=42)**

**reg\_model.fit(X\_train, y\_train)**

**# Predict**

**y\_pred = reg\_model.predict(X\_test)**

**# Evaluation**

**print("R² Score:", r2\_score(y\_test, y\_pred))**

**print("RMSE:", np.sqrt(mean\_squared\_error(y\_test, y\_pred)))**

**plt.figure(figsize=(8, 5))**

**plt.scatter(y\_test, y\_pred, alpha=0.5, color='blue')**

**plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'r--')**

**plt.xlabel("Actual AQI")**

**plt.ylabel("Predicted AQI")**

**plt.title("Actual vs Predicted AQI")**

**plt.grid(True)**

**plt.show()**

**# Select features and drop NA**

**df\_cls = df\_station\_day[['PM2.5', 'NO2', 'Toluene', 'AQI']].dropna()**

**# Categorize AQI into levels**

**def aqi\_category(aqi):**

**if aqi <= 100:**

**return 'low'**

**elif aqi <= 200:**

**return 'medium'**

**else:**

**return 'high'**

**df\_cls['quality\_level'] = df\_cls['AQI'].apply(aqi\_category)**

**# Encode category**

**le = LabelEncoder()**

**df\_cls['quality\_level\_encoded'] = le.fit\_transform(df\_cls['quality\_level'])**

**# Feature selection**

**X\_cls = df\_cls[['PM2.5', 'NO2', 'Toluene']]**

**y\_cls = df\_cls['quality\_level\_encoded']**

**# Normalize features**

**scaler = StandardScaler()**

**X\_cls\_scaled = scaler.fit\_transform(X\_cls)**

**# Split data**

**X\_train\_cls, X\_test\_cls, y\_train\_cls, y\_test\_cls = train\_test\_split(X\_cls\_scaled, y\_cls, test\_size=0.2, random\_state=42)**

**# Train classifier**

**cls\_model = RandomForestClassifier(random\_state=42)**

**cls\_model.fit(X\_train\_cls, y\_train\_cls)**

**# Predict**

**y\_pred\_cls = cls\_model.predict(X\_test\_cls)**

**# Evaluation metrics**

**accuracy = accuracy\_score(y\_test\_cls, y\_pred\_cls)**

**precision = precision\_score(y\_test\_cls, y\_pred\_cls, average='macro')**

**recall = recall\_score(y\_test\_cls, y\_pred\_cls, average='macro')**

**report = classification\_report(y\_test\_cls, y\_pred\_cls, target\_names=le.classes\_)**

**# Print results**

**print("Accuracy:", accuracy)**

**print("Precision:", precision)**

**print("Recall:", recall)**

**print("\nClassification Report:\n", report)**

**cm = confusion\_matrix(y\_test\_cls, y\_pred\_cls)**

**disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=le.classes\_)**

**disp.plot(cmap=plt.cm.Blues)**

**plt.title('Confusion Matrix')**

**plt.show()**

**df = pd.merge(df, df\_stations[['StationId', 'City']], on='StationId', how='left')**

**city\_aqi = df.groupby('City')['AQI'].mean().sort\_values(ascending=False)**

**plt.figure(figsize=(12, 6))**

**sns.barplot(x=city\_aqi.index, y=city\_aqi.values, palette='viridis')**

**plt.xticks(rotation=90)**

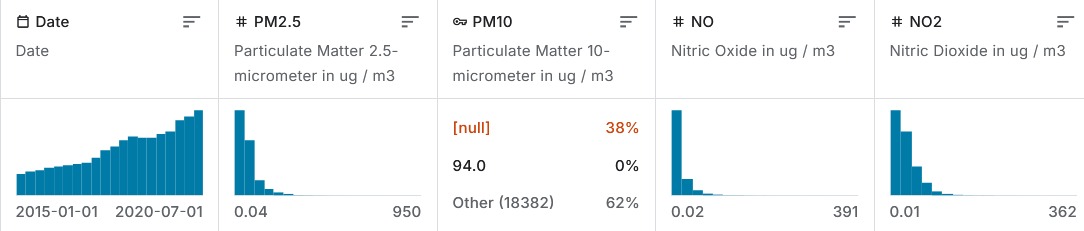
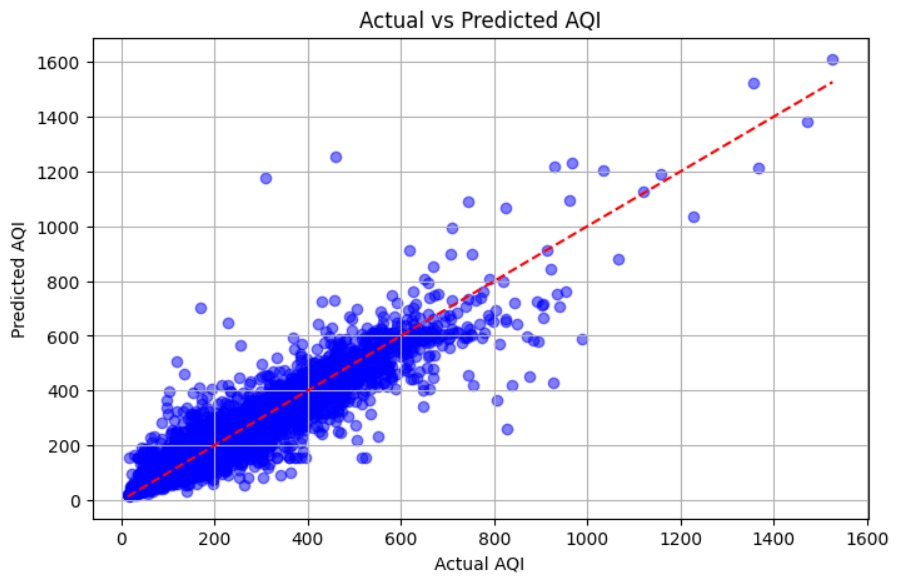
**plt.ylabel('Average AQI')**

**plt.title('Average AQI by City')**

**plt.tight\_layout()**

**plt.show()**

**OUTPUT :**

****

