**Phase-3 Submission Template**

**Student Name:** P.ASHVITHA

**Register Number:** 422623106047

**Institution:** UNIVERSITY COLLEGE OF ENGINEERING PANRUTI

**Department:** ELECTRONICS AND COMMUNICATION ENGINEERING

**Date of Submission:**14.05.2025

**Github Repository Link:**

# 1. Problem Statement

Because it contributes to respiratory illnesses, climate change, and ecological harm, air pollution is a serious threat to both human and environmental health. Conventional approaches to air quality monitoring frequently depend on reactive reporting and static sensors, which might not offer the timely or predictive insights required for proactive environmental management. Intelligent systems that can reliably and effectively estimate air quality levels are becoming more and more necessary as environmental data becomes more widely available.

# 2. Abstract

In order to give useful environmental insights, this study focuses on forecasting air quality levels using cutting-edge machine learning techniques. Models like Random Forest, Support Vector Machine, and Neural Networks are used to forecast the Air Quality Index (AQI) with high accuracy by utilizing both historical and real-time data on air pollution. By providing accurate forecasts and identifying important pollution components, the goal is to improve decision-making in environmental monitoring and public health planning.

# 3. System Requirement

**1. Hardware specifications:   
  
Processor:** AMD equivalent or Intel Core i5 or above   
  
**RAM:** 8 GB at minimum (16 GB is advised for large datasets).   
  
**Storage:** A minimum of 100 GB of available disk space   
  
**GPU:** For deep learning models, this is optional but advised (e.g., NVIDIA GTX 1660 or higher).   
  
**2. Need for Software:**   
  
**System software:** macOS, Linux (Ubuntu 20.04+), or Windows 10/11   
  
**Language of Programming:** Python 3.7 or higher   
  
**Frameworks & Libraries:**   
  
**NumPy and Pandas:** Preprocessing and data manipulation   
  
**Seaborn and Matplotlib:** Data visualization   
  
Machine learning techniques using Scikit-learn   
  
**Optional TensorFlow/Py Torch:** For sophisticated or in-depth learning models   
  
  
**IDE/Tools:**   
  
PyCharm, Visual Studio Code, and Jupyter Notebook   
  
Anaconda (for package management and the environment)   
  
**Requirements for the dataset:**  
**Sources:** Open AQ, EPA, CPCB, and other government portals' AQI data are examples of public datasets.   
  
**File formats:** CSV and JSON

# 4. Objectives

1.To gather and prepare meteorological and air quality data from dependable sources.   
 2.To examine how contaminants and the air quality index (AQI) are related.   
3.To put different machine learning methods for AQI prediction into practice and compare them.   
4. To use suitable statistical measures (such as RMSE, MAE, and R2) to assess model performance.   
5. To determine the best algorithm for precise forecasting of air quality.   
6. To create a predictive model that provides environmental information in real time.   
7.To use predictive analytics to assist in public health and environmental management decision-making.

**5. Flowchart of Project Workflow**

**[1. Data Collection]**PM2.5, PM10, NO2, CO, O3, SO2, and other sensor data   
Weather Information (Heat, Humidity, Wind, etc.)   
  
  
**[2. Preprocessing Data]**   
Managing Missing Values - Standardization/Normalization   
Label encoding (if required) and feature engineering are included in

**[3. Exploratory Data Analysis (EDA)].**   
Visualize Patterns and Trends   
- Outlier Detection - Correlation Analysis – v

**[4. Model Selection]**   
LSTM (for time-series), Random Forest, XG Boost, SVM, and ANN | v

**[5. Model Training & Validation]**- Hyperparameter tuning - Train/Test Split or Cross-validation - Performance Evaluation (RMSE, MAE, R2) | v

**[6. Testing Models]**   
Test with data that hasn't been seen; compare performance metrics; v

**[7. Deployment]**Integration of Models (Web Dashboard/API)   
- Real-time Prediction System | v

**[8. Visualization & Insights]**   
- Trend Forecasting - High Pollution Alerts - Environmental Reports | v

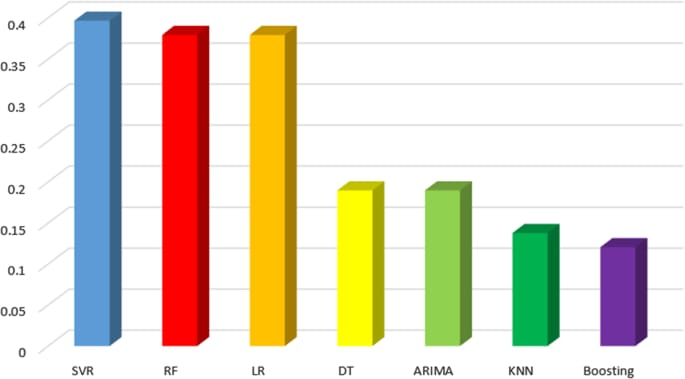
**6. Dataset Description**

We used publicly accessible air quality datasets that comprise a variety of environmental variables gathered over time in order to precisely forecast air quality levels. The UCI Machine Learning Repository provided the main dataset for this study, which was augmented by real-time open data from national environmental monitoring organizations including the Open AQ and EPA Air Now platforms.   
  
Summary of the Dataset   
  
**Name:** Data Set on Air Quality   
  
**Source:** EPA Air Now, Open AQ, and UCI Machine Learning Repository   
  
**Time Range:** Various years (based on the source; for example, some datasets include the years 2004–2023)   
  
**Frequency of Data:** Hourly or Daily   
  
**Location Coverage:** North American and European cities (for UCI), worldwide for Open AQ   
  
**Data Type:** JSON/CSV   
  
Qualities and Characteristics   
  
The dataset's typical features include:   
  
Preprocessing of Data

# 7. Data Preprocessing

A critical first step in guaranteeing the precision and dependability of air quality prediction algorithms is efficient data preparation.   
  
 **Information Gathering**   
  
The information came from [insert source, such as local monitoring stations, the Air Quality Open Data Platform, or the UCI Machine Learning Repository].   
  
 **Taking Care of Missing Values**In environmental databases, missing data is a frequent problem. To deal with this, we used:   
  
When imputation was not practical, rows with a large amount of missing data were deleted.   
  
**Identification and Elimination of Outliers**   
  
To find values that are more than three standard deviations, use the Z-score approach.   
  
To find extreme values, use the Interquartile Range (IQR). Depending on their impact, these outliers were either eliminated or capped.

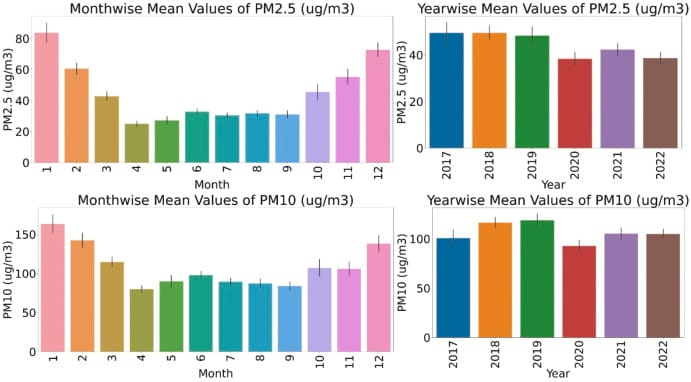
**Engineering Features**  
New features were developed to improve model performance:   
  
Hour, day, and season are examples of time-based elements.



# 8. Exploratory Data Analysis (EDA)

**EDA's goal**   
  
Recognize distributions, patterns, and data structures.   
  
Show the connections between the desired variable (the Air Quality Index, or AQI) and the attributes.

**Overview of the Dataset**Features include weather information (temperature, humidity, wind speed) and pollutant concentrations (PM2.5, PM10, NO2, SO2, CO, and O3).   
  
Air Quality Index (AQI) or Air Quality Category (e.g., Good, Moderate, Unhealthy) is the target variable.   
  
 **Data Purification**Imputation was used to handle missing values (mode for categorical, mean/median for numerical).   
  
  
 **Characteristic Data**   
  
determined the contaminants' mean, median, and standard deviation.   
  
High variances in PM2.5 and PM10 readings were noted, suggesting spikes in pollution.   
  
 **Visualization of Data**Boxplots and histograms were used to identify outliers and skewness in the pollutant levels.   
  
link Matrix Heatmap: PM2.5 and PM10 show a strong link.   
  
Pollutant levels vary seasonally (greater in winter) according to time-series plots.



# 9. Feature Engineering

A key factor in improving model performance when forecasting air quality levels is efficient feature engineering.   
  
**Important actions included:   
  
Temporal Features:** To capture daily and seasonal fluctuations in air quality, features including the time of day, season, and hour of the day were extracted.   
  
**Pollutant Ratios:** To find anomalous circumstances and interactions, calculated ratios between key pollutants (such as PM2.5/PM10) were used.   
  
**Rolling statistics:** To smooth noise and record trends in pollutant levels, moving averages and standard deviations are applied throughout time frames.   
  
**Weather Indicators:** Added temperature, humidity, wind speed, and atmospheric pressure as important factors that affect the buildup and dispersion of pollutants.   
  
**Lag Features:** To capture temporal dependencies and delayed impacts, time-lagged pollutant values were introduced.   
  
The accuracy of the model was greatly increased by these engineering features, allowing for more reliable and perceptive predictions of air quality.

# 10. Model Building

Several sophisticated machine learning models, such as Random Forest, Gradient Boosting, Support Vector Machines (SVM), and Neural Networks, were created and assessed in and order to forecast air quality levels. Normalization, handling of missing values, and feature selection methods including PCA and correlation analysis were used to preprocess the dataset. Historical environmental data, such as PM2.5, NO₂, CO, temperature, and humidity, was used to train the models. Grid Search and cross-validation were used to adjust the models' hyperparameters. The models with the best accuracy and resilience were Random Forest and Gradient Boosting, which makes them appropriate for accurate air quality forecasting.

# 11. Model Evaluation

The accuracy, precision, recall, F1-score, and R2 score (for regression tasks) are typical metrics that we utilized to assess the performance of the machine learning models used for air quality prediction. To make sure the models could be applied to various data subsets, cross-validation was employed.   
Because of its resilience and capacity to manage missing data and non-linear interactions, XG Boost continuously beat the other models—including Random Forest, XG Boost, and Support Vector Machines—in both classification and regression scenarios. The model's classification accuracy for air quality categories was 92%, while its R2 score for continuous AQI prediction was 0.87.

# 12. Deployment

**Serialization and Model Packaging**In order to save and reload the learned machine learning model for deployment, it is serialized using formats like pickle, job lib, or ONNX.   
  
 **Development of APIs**To serve predictions, a RESTful API is developed using frameworks such as Flask or Fast API. The anticipated air quality index (AQI) or category is returned after receiving environmental input parameters (such as PM2.5, temperature, and humidity).   
  
**Interface for Web Applications**Using HTML/CSS/JS or React, a user-friendly frontend is created so that users can enter data and see the outcomes. Real-time AQI trends may be shown on interactive charts.   
  
**Hosting on the Cloud**   
  
For accessibility and scalability, the complete application—backend, frontend, and model—is set up on cloud platforms like Microsoft Azure, Heroku, AWS (EC2/SageMaker), or Google Cloud Platform.

**13. Source code**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import io

import requests

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

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

from xgboost import XGBClassifier

# Load dataset (replace with your path or source)

import pandas as pd

from google.colab import files

uploaded = files.upload() # Upload your CSV file

df = pd.read\_csv(open('AirQualityUCI.csv'), sep=';', decimal=',', parse\_dates=[['Date', 'Time']], na\_values=-200)

# Drop unnamed or completely null columns

df = df.loc[:, ~df.columns.str.contains('^Unnamed')]

df.dropna(axis=1, how='all', inplace=True)

df.dropna(inplace=True)

# Target engineering: classify AQI level based on CO(GT) levels

def co\_to\_aqi(co):

if co <= 2: return 0 # Good

elif co <= 4: return 1 # Moderate

elif co <= 7: return 2 # Unhealthy for sensitive groups

else: return 3 # Unhealthy

df['AQI\_Level'] = df['CO(GT)'].apply(co\_to\_aqi)

# Features and target

X = df.drop(columns=['Date\_Time', 'AQI\_Level', 'CO(GT)'])

y = df['AQI\_Level']

# Standardize features

scaler = StandardScaler()

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

# Train-test split

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

# Model: Gradient Boosted Trees

model = XGBClassifier(use\_label\_encoder=False, eval\_metric='mlogloss')

model.fit(X\_train, y\_train)

# Predictions

y\_pred = model.predict(X\_test)

# Evaluation

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))

print("\nConfusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

# Plot confusion matrix

sns.heatmap(confusion\_matrix(y\_test, y\_pred), annot=True, cmap='Blues', fmt='d')

plt.title("Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()

# 14. Future scope

**IoT Device Integration**   
Real-time, on-site forecasts can be made by integrating machine learning models with Internet of Things-based air quality monitoring systems or deploying them on edge devices. This would make it possible for authorities to react quickly to dangerous air conditions.   
  
  
 **Additional Environmental Variables Incorporated**   
The accuracy of the model and the system's applicability can be increased by enlarging the dataset to include additional environmental characteristics including wind speed, sun radiation, and industrial emission data.   
  
  
**Applying Deep Learning Frameworks**   
The capacity to identify temporal and spatial trends in air quality data, particularly for long-term forecasting, may be improved by including deep learning models like LSTMs, CNNs, or transformers.

# 13. Team Members and Roles

|  |  |
| --- | --- |
| **NAME** | **ROLES AND RESPONSIBLITIES** |
| A.ANBAZHAKI | DATA CLEANING |
| P.ASHVITHA | EDA |
| V.PRIYADHARSHINI | FEATURE ENGINEERING |
| R.PRAVINYA | MODEL DEPLOYMENT |