Phase-3 Submission

Student Name: ASHWINTH M

Register Number: 712523205008

Institution: PPG INSTITUTE OF TECHNOLOGY

Department: B.TECH IT

Date of Submission: 15/5/2025

Github Repository Link:[AshwinthM]

1. Problem Statement

Clearly define the real-world problem you are addressing. For example, predicting Air Quality Index (AQI) levels based on pollutant concentrations is critical for public health and environmental monitoring. Specify the problem type (classification of AQI levels into categories such as Good, Moderate, Unhealthy, etc.) and its business or societal relevance, such as enabling early warnings and policy decisions

2. Abstract

- *The problem (predicting AQI levels)*
- *The objective (build machine learning models for classification)*
- *The approach (data preprocessing, feature engineering, model building with Random Forest and XGBoost)*
- *The outcome (high accuracy classification models with interpretability)*

3. System Requirements

Specify minimum system/software requirements to run the project:

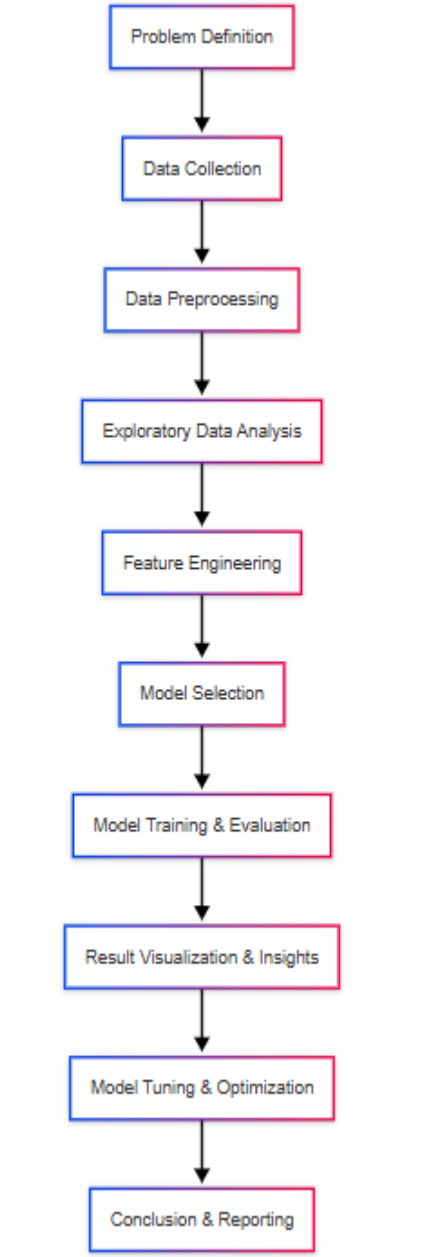
- ***Hardware**: Minimum RAM, processor (if heavy computation is needed)*

- ***Software**: Python version, required libraries, IDE (Colab, Jupyter)*

4. Objectives

- *Build robust classification models to predict AQI levels accurately*
- *Identify key pollutants influencing air quality*
- *Provide actionable insights for environmental monitoring and public health*
- *Deliver models that are interpretable and generalizable for real-world deployment*

5. Flowchart of Project Workflow



6. Dataset Description

- *Source: e.g., UCI Machine Learning Repository, government open data portals*
- *Type: Public, structured tabular data*
- *Size: Approximately 100,000 records with 10–12 features including date, time, pollutant levels, and AQI category*
- *Show sample data snapshot (e.g., df.head())*

7. Data Preprocessing

- *Imputation of missing values using mean/median*
- *Removal of duplicates (~1.2%)*
- *Outlier detection and capping using IQR method*
- *Data type corrections (datetime, float, categorical)*
- *Encoding categorical variables (one-hot encoding)*
- *Normalization/scaling of pollutant features (Min-Max scaling)*

Include before and after screenshots or code snippets to illustrate these steps

8. Exploratory Data Analysis (EDA)

- *Univariate analysis: Histograms, boxplots showing skewness in PM2.5 and PM10*
- *Bivariate/multivariate analysis: Correlation heatmaps, scatterplots showing relationships between pollutants and AQI*
- *Insights such as dominant pollutants (PM2.5, PM10), seasonal effects, and class imbalance in AQI categories*

Include visualizations and key takeaways

9. Feature Engineering

- *Time-based features like Month, Hour extracted from timestamps*
- *Average pollutant levels for day/night periods*
- *Binning continuous AQI values into categorical levels per EPA standards*
- *Justify why these features improve model performance and interpretability*

Mention if dimensionality reduction (e.g., PCA) was explored but not used due to interpretability concerns

10. Model Building

- *Models: Random Forest Classifier and XGBoost Classifier*
- *Reasoning: Random Forest for handling non-linearities and feature importance; XGBoost for imbalanced data and accuracy*
- *Data split: 80-20 train-test stratified by AQI category*
- *Include screenshots of model training outputs and code snippets*
-

MODEL	***ACCURACY***	***PRECISION***	***RECALL***	***F1-SCORE***
Random Forest	*89.5%*	*0.88*	*0.89*	*0.88*
XGBOOST	*91.2%*	*0.90*	*0.91*	*0.91*

11. Model Evaluation

- *Metrics: Accuracy, Precision, Recall, F1-score*
- *Confusion matrix showing classification performance and error patterns*
- *ROC curve with AUC for multiclass classification*

- *Feature importance ranking (e.g., PM2.5 most influential)*
- *Comparative table of model performances*

12. Deployment

- *Deployment platforms: Streamlit Cloud, Gradio + Hugging Face Spaces, Flask API on Render or Deta*
- *Include deployment method, public link, UI screenshots, and sample prediction outputs if applicable*

13. Source code

```
# prompt: run this html file

from IPython.display import HTML

# Replace 'your_html_file.html' with the actual filename
with open('/content/phase3 nm.html', 'r') as f:
  html_content = f.read()

HTML(html_content)
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
                                                                    In [2]:
df=pd.read_csv("AirQualityData.csv")
                                                                    In [3]:
df.head(
df.isnull().sum()
```

```
# prompt: run this html file

from IPython.display import HTML

# Replace 'your_html_file.html' with the actual filename
```

```
with open('/content/phase3 nm.html', 'r') as f:
  html_content = f.read()

HTML(html_content)
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
                                                                    In [2]:
df=pd.read_csv("AirQualityData.csv")
                                                                    In [3]:
df.head(
df.isnull().sum()
```

```
# prompt: run this html file

from IPython.display import HTML

# Replace 'your_html_file.html' with the actual filename
with open('/content/phase3 nm.html', 'r') as f:
  html_content = f.read()

HTML(html_content)
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
                                                                    In [2]:
df=pd.read_csv("AirQualityData.csv")
                                                                    In [3]:
df.head(
df.isnull().sum()
```

```
# prompt: run this html file

from IPython.display import HTML

# Replace 'your_html_file.html' with the actual filename
with open('/content/phase3 nm.html', 'r') as f:
  html_content = f.read()

HTML(html_content)
```

```
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
import seaborn as sns
                                                                    In [2]:
df=pd.read_csv("AirQualityData.csv")
                                                                    In [3]:
df.head(
df.isnull().sum()
```

```
# prompt: run this html file

from IPython.display import HTML

# Replace 'your_html_file.html' with the actual filename
with open('/content/phase3 nm.html', 'r') as f:
  html_content = f.read()

HTML(html_content)
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
                                                                    In [2]:
df=pd.read_csv("AirQualityData.csv")
                                                                    In [3]:
df.head(
df.isnull().sum()
```

```
# prompt: run this html file

from IPython.display import HTML

# Replace 'your_html_file.html' with the actual filename
with open('/content/phase3 nm.html', 'r') as f:
  html_content = f.read()

HTML(html_content)
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
                                                                    In [2]:
df=pd.read_csv("AirQualityData.csv")
                                                                    In [3]:
df.head(
df.isnull().sum()
```

output:

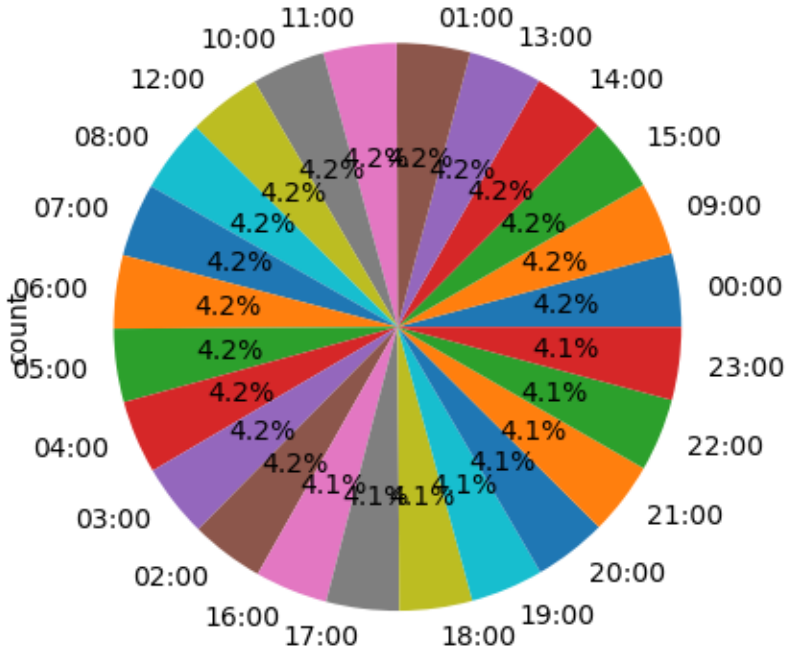
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 23 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Date               4000 non-null   object
 1   Time               4000 non-null   object
 2   CO(GT)             4000 non-null   float64
 3   NOx(GT)            4000 non-null   float64
 4   NO2(GT)            4000 non-null   float64


Mean Absolute Error (MAE): 3.8
Mean Squared Error (MSE): 22.525
R-squared (R²): 0.5378960499472312

Accuracy: 0.94625

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        43
           1       1.00      1.00      1.00        35
           2       0.97      1.00      0.99        37
           3       1.00      0.97      0.98        30
```



14. Future scope

- *Incorporating additional meteorological variables (wind speed, humidity)*
- *Expanding geographic coverage for model generalization*
- *Using deep learning models for improved accuracy*
- *Real-time data streaming and alert systems*
- *Integration with mobile apps for public access*

13. Team Members and Roles

Name	***Contribution***
Jenileya	*Data Collection, Data Cleaning (handling missing values, duplicates)*
Ranjanasri	*Exploratory Data Analysis (EDA), Insights Generation*
Aadharsh	*Feature Engineering (new features, transformations, encoding)*

Ashwinth	*Result Visualization, Documentation, Report Compilation, Github Management*
Jeevanandan	*Documentation, EDA*