

# **PROJECT REPORT**

## **PREDICTION OF BENZENE CONCENTRATION**

**AUTHORS**

**ANUPAMA.K.B**

**JIBINA.M.V**

## ABSTRACT

Air pollution poses serious health risks, with benzene being a known carcinogen found in urban air. This project focuses on predicting benzene concentration using historical air quality data from the UCI dataset and machine learning techniques. The workflow includes data preprocessing, exploratory data analysis (EDA), Random Forest regression modelling, and deployment via a Flask-based web application. The Random Forest model achieved high accuracy ( $R^2 = 0.994$ , RMSE = 3.182), with humidity and temperature identified as the strongest predictors of benzene levels. The app allows users to input air quality parameters and receive benzene level predictions, along with safety alerts and environmental awareness messaging. This tool aims to empower users with actionable insights while promoting public health and environmental consciousness.

# INDEX

|   | Page  |
|---|-------|
| 1. Introduction                           | 4-5   |
| 2. Objectives & Scope                     | 6     |
| 3. Data description                       | 7     |
| 4. Methodology                            | 8     |
| 4.1 Data understanding and exploration    | 8     |
| 4.2 Data cleaning and preprocessing       | 8     |
| 4.3 Model development                     | 8     |
| 4.4 Model evaluation and cross validation | 8-9   |
| 4.5 Deployment plan                       | 9     |
| 5. Tools and technologies used            | 10    |
| 6. Results and Insights                   | 11    |
| 7. Challenges and Limitations             | 12    |
| 8. Future work                            | 13    |
| 9. Conclusion                             | 14    |
| 10. Appendix                              | 15-32 |
| 10.1 Appendix A Screen shot               | 15-21 |
| 10.2 Appendix B Code                      | 22-31 |
| 11. References                            | 32    |

# 1. INTRODUCTION

Air pollution is one of the most pressing environmental challenges of the 21st century, with far-reaching consequences for human health, ecosystems, and climate. Among the many pollutants present in urban air, **benzene (C<sub>6</sub>H<sub>6</sub>)** stands out due to its **carcinogenic properties** and its prevalence in industrial emissions, vehicular exhaust, and tobacco smoke. Long-term exposure to benzene has been linked to serious health conditions including leukaemia, bone marrow failure, and immune system suppression. Despite its dangers, benzene often goes unnoticed in public discourse, overshadowed by more commonly discussed pollutants like PM<sub>2.5</sub> or CO<sub>2</sub>.

Traditional methods of monitoring benzene concentration rely on expensive, stationary sensors that are limited in coverage and accessibility. These systems, while accurate, are not scalable for widespread public use or real-time personal decision-making. In this context, **machine learning** offers a powerful alternative—enabling predictive modelling based on historical air quality data and allowing for the development of tools that can estimate benzene levels using readily available environmental parameters.

This project aims to bridge the gap between complex environmental data and public accessibility by building a **benzene concentration prediction web application**. The app is designed to be intuitive, informative, and actionable. Users can input basic air quality indicators—such as temperature, relative humidity, and concentrations of other gases—and receive a prediction of benzene levels, along with contextual safety alerts and environmental

awareness messages. The goal is not only to provide accurate predictions but also to **educate and empower users** to make safer choices in their daily lives.

The development process follows a structured data science workflow, beginning with **exploratory data analysis (EDA)** to understand the relationships between various pollutants and benzene. This is followed by **regression modelling**, where multiple algorithms are tested and evaluated to identify the most accurate and robust predictor. The chosen model is then integrated into a **Flask-based web application**, which is deployed on a public platform to ensure accessibility. Special attention is given to **UI/UX design**, with a focus on clarity, responsiveness, and environmental theming.

Beyond technical implementation, this project reflects a broader commitment to **environmental advocacy and user safety**. Features such as danger alerts for high benzene predictions and banners promoting pollution awareness are embedded into the app to enhance its social impact. The project also emphasizes **modular, beginner-friendly code**, making it a valuable resource for learners and developers interested in practical machine learning applications.

In summary, this project represents a convergence of data science, environmental health, and user-centred design. It showcases how technology can be harnessed not just for prediction, but for **education, empowerment, and positive change**. Through continued iteration and expansion, the app has the potential to evolve into a comprehensive environmental monitoring tool—serving individuals, communities, and policymakers alike.

.

## 2. OBJECTIVES & SCOPE

- Objectives:
  - Perform Exploratory Data Analysis (EDA) to identify key factors influencing benzene levels.
  - Build and evaluate regression models for benzene prediction.
  - Deploy a user-friendly web app with safety alerts and environmental messaging.
- Scope:

Included: Historical air quality data, regression modelling, web deployment.

Excluded: Real-time sensor integration, geospatial mapping.
- Key Performance Indicators:
  - $R^2$  Score – Model accuracy.
  - MAE/RMSE – Average prediction error.
  - Alert Accuracy – Correct identification of hazardous levels.

### 3. DATA DESCRIPTION

- Dataset: UCI Air Quality Dataset (AirQualityUCI.xlsx)
- Time Period: 2004–2005
- Size: ~9,000 hourly records
- Features: Temperature (T), Relative Humidity (RH), Absolute Humidity (AH), CO(GT), NO<sub>x</sub>(GT), NO<sub>2</sub>(GT)
- Target: Benzene Concentration (C<sub>6</sub>H<sub>6</sub>\_GT)
- Collection Method: Public dataset via UCI repository.

## 4. METHODOLOGY

The CRISP-DM framework was adopted, covering data understanding, preprocessing, modelling, evaluation, and deployment.

### 4.1 Data Understanding & Exploration:

- Inspected dataset structure and missing values (-200 placeholders).
- Identified correlations between gases and benzene.
- EDA included boxplots (outliers), histograms (distribution), and correlation heatmaps.

### 4.2 Data Cleaning & Preprocessing:

- Replaced -200 with NaN, applied interpolation.
- Outlier detection with IQR.
- Standard Scaler used for normalization.
- Feature selection included T, RH, AH, CO, NO<sub>x</sub>, NO<sub>2</sub>.

### 4.3 Model Development:

- Baseline: Linear Regression.
- Main Model: Random Forest Regressor (`n_estimators=100`, `random_state=42`).
- Split data 80/20 for training/testing.

### 4.4 Model Evaluation and cross validation:

- Metrics:  $R^2$ , MAE, RMSE.
- Random Forest achieved  $R^2 = 0.994$ ,  $RMSE = 3.182$ .
- Residual distribution approximately normal around zero.



-Cross-validation confirmed that the Random Forest model maintains reliable predictive performance on unseen data.

#### 4.5 Deployment:

- Flask web app structure: app.py, web.py, templates (index.html, result.html).
- Deployed on PythonAnywhere.-URL:

1. <https://anupamaveni007.pythonanywhere.com/>
2. <http://jibibala.pythonanywhere.com/>

#### 4.6 UI/UX:

- Clean, responsive layout.
- Safety alerts for hazardous levels.
- Educational banners on pollution awareness.

## 5. TOOLS & TECHNOLOGY USED

- Programming Language: Python
- Libraries: Pandas, NumPy, Scikit-learn, Matplotlib, Seaborn, Openpyxl
- Web Framework: Flask
- IDE: VS Code / Jupyter Notebook
- Deployment: PythonAnywhere

## 6. RESULTS KEY FINDINGS

- Random Forest achieved high accuracy ( $R^2 = 0.994$ , RMSE = 3.182).
- Feature importance: Temperature, RH, and AH were the top predictors.
- Actual vs Predicted scatter plot showed predictions closely aligned with true values.
- Residual errors were centred around zero, indicating a good fit.

## **7.CHALLENGES & LIMITATIONS**

- Missing data and sensor errors in raw dataset.
- Limited geographic scope of dataset.
- Accuracy may be influenced by external factors (e.g., traffic, weather variations).

## **8. FUTURE WORK**

- Integrate real-time sensor data via APIs.
- Add geolocation-based predictions.
- Extend to other pollutants (PM<sub>2.5</sub>, SO<sub>2</sub>).
- Enhance educational content and health tips.

## 9. CONCLUSION

This project successfully demonstrates the application of **Random Forest Regression** in predicting benzene concentration levels with remarkable accuracy. By analysing data from the **Air Quality UCI dataset**, the model was able to capture complex, non-linear relationships between multiple environmental factors and benzene concentration. Among the predictors, **humidity and temperature** emerged as the most influential, highlighting the strong interaction between climatic conditions and air pollutant levels.

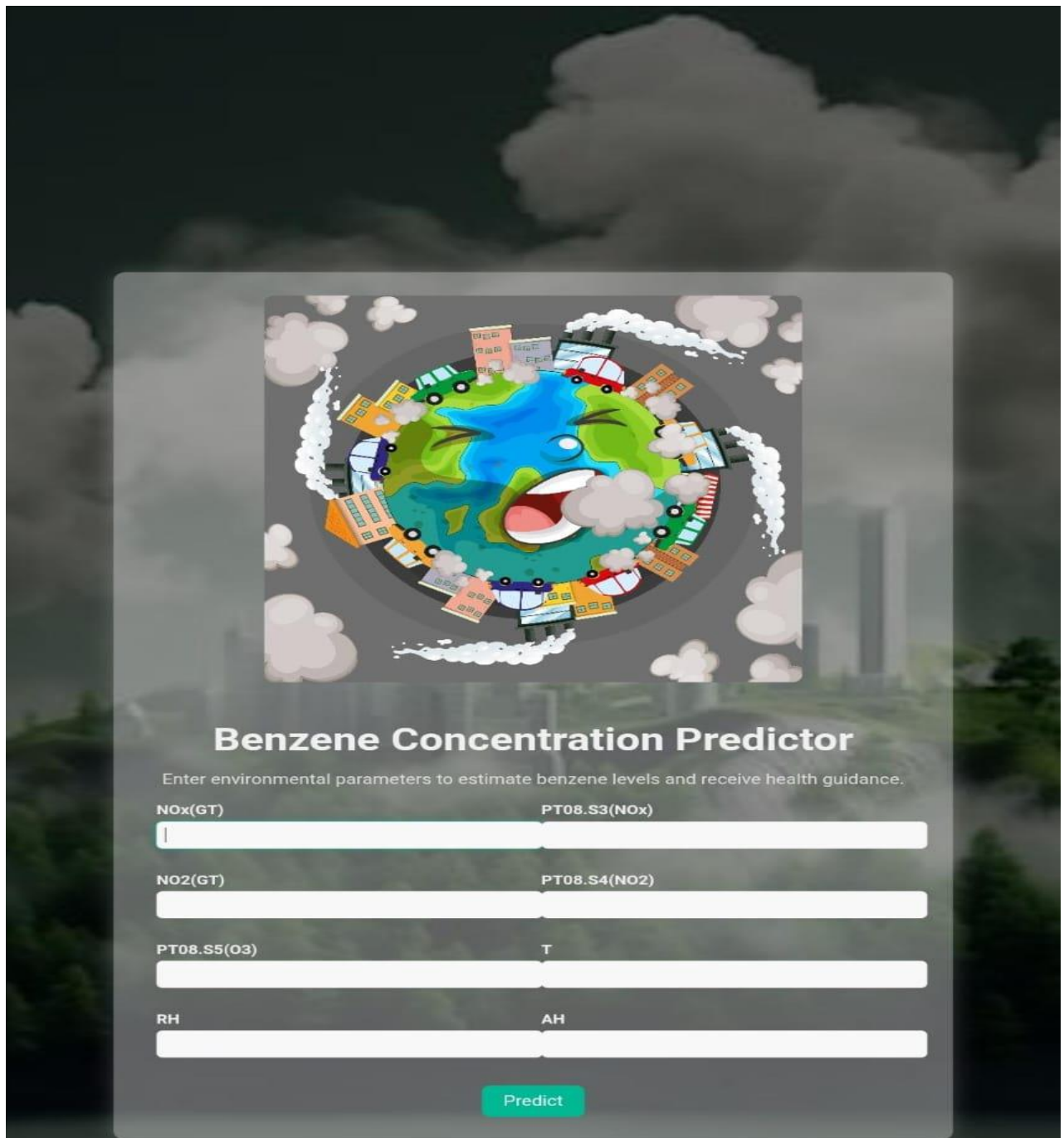
The deployment of the model through a **Flask web application** makes it practical and user-friendly, allowing real-time predictions by simply providing the necessary input parameters. This not only bridges the gap between machine learning research and real-world application but also empowers users to understand potential health risks associated with benzene exposure. The system is designed to not only provide predictions but also generate **safety alerts**, which can be instrumental in creating awareness and promoting preventive measures against pollution-related health issues.

From a broader perspective, this project contributes to the field of **environmental health monitoring**, offering a scalable solution that can be integrated with **IoT devices, sensors, and real-time dashboards**. Such an approach would enable continuous tracking of air quality, early detection of hazardous levels, and timely policy interventions by authorities.

In conclusion, the project not only demonstrates the **technical effectiveness** of Random Forest in air quality prediction but also underlines its **social relevance** in fostering environmental awareness and public health protection. With future extensions—such as incorporating additional pollutants, integrating live data streams, or developing mobile app interfaces—the system can evolve into a comprehensive **real-time air quality monitoring and alert platform**, making a meaningful impact in combating urban pollution challenges.

## 10.Appendix A:

### 10.1 Screenshot of welcome page



**Benzene Concentration Predictor**

Enter environmental parameters to estimate benzene levels and receive health guidance.

|                      |                      |
|----------------------|----------------------|
| NOx(GT)              | PT08.S3(NOx)         |
| <input type="text"/> | <input type="text"/> |
| NO2(GT)              | PT08.S4(NO2)         |
| <input type="text"/> | <input type="text"/> |
| PT08.S5(O3)          | T                    |
| <input type="text"/> | <input type="text"/> |
| RH                   | AH                   |
| <input type="text"/> | <input type="text"/> |

Fig 1: welcome page

## Prediction page

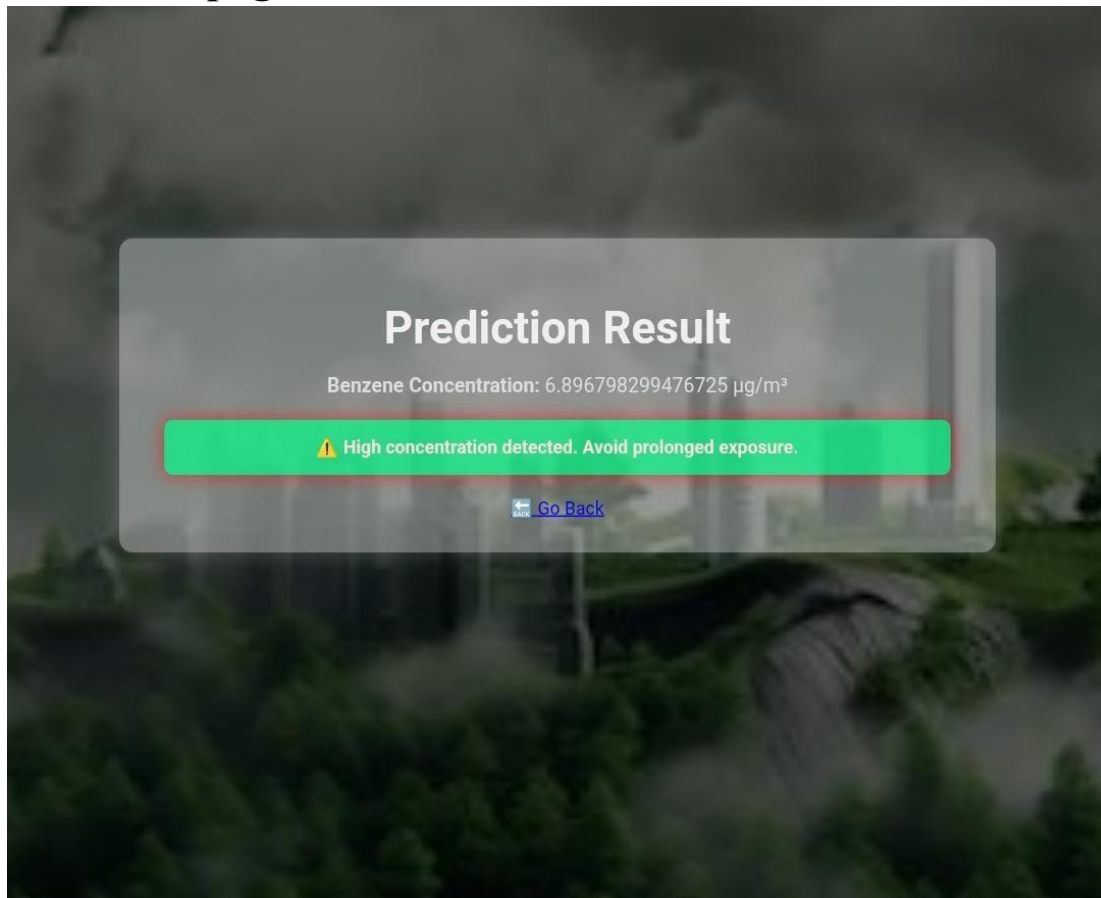


Fig 2: prediction page



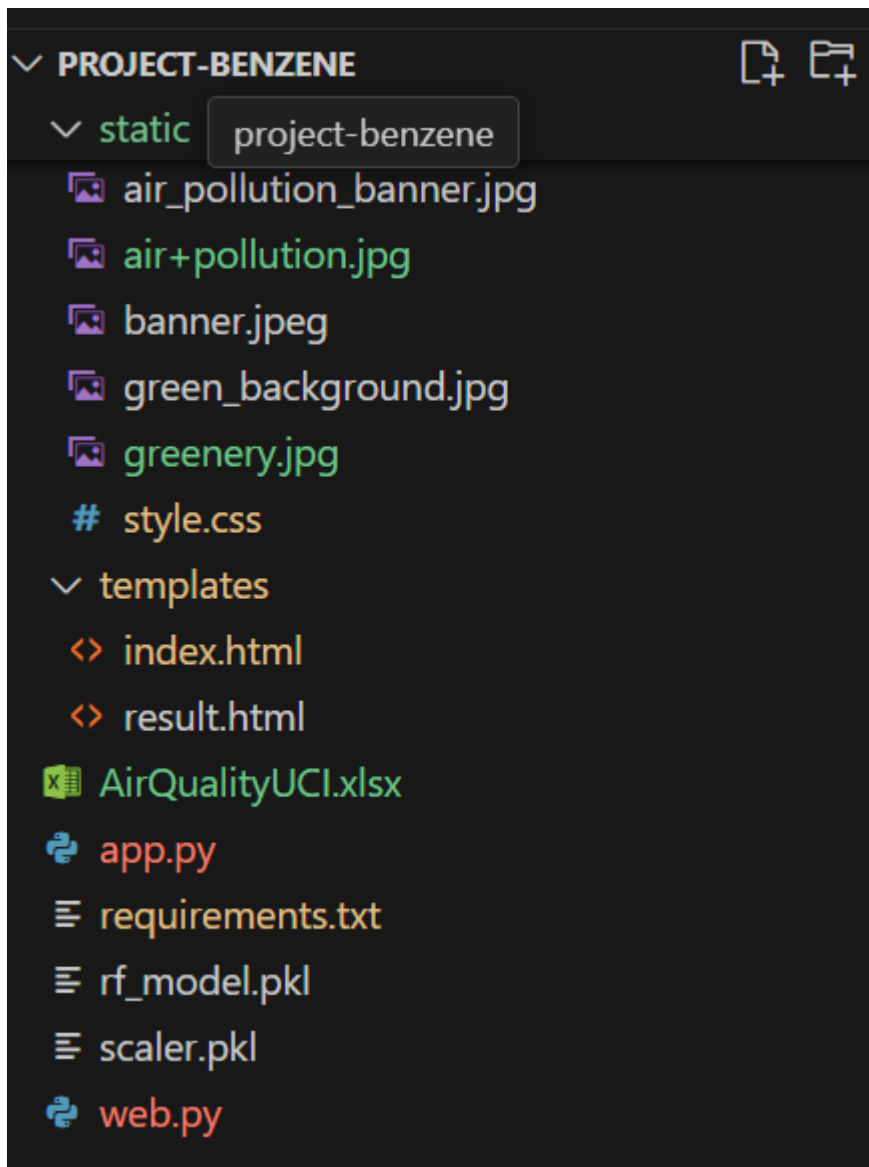
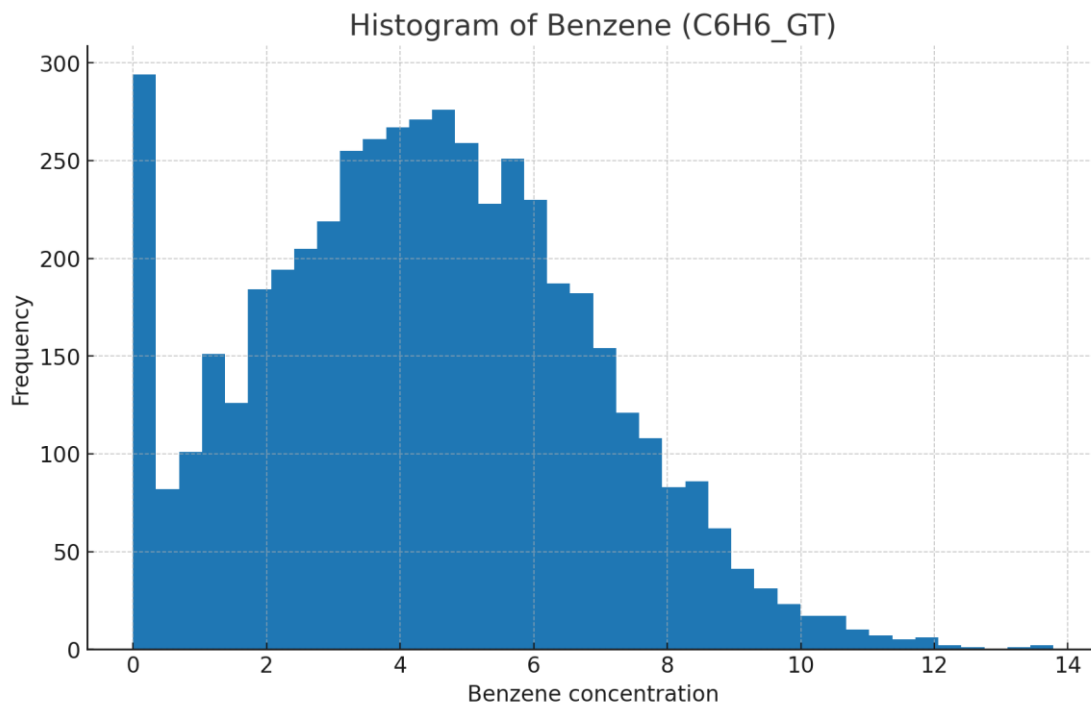


Fig3. Folders in vs code

## Exploratory Data Analysis (EDA) – Plots

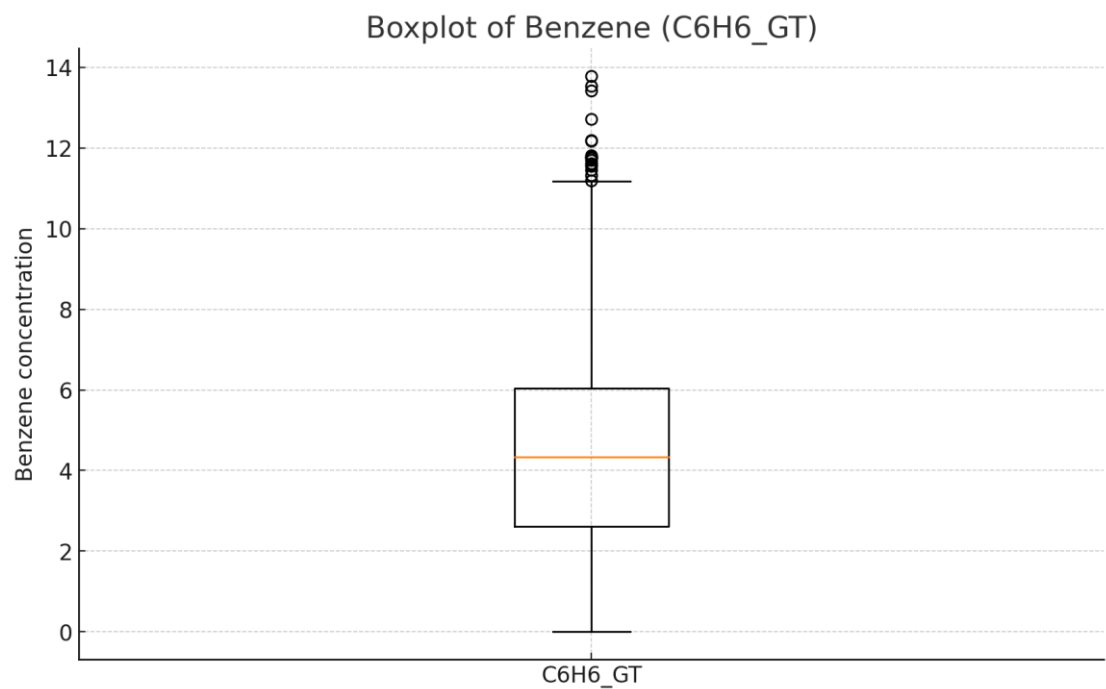
The following figures illustrate the key EDA outputs referenced in the report.

### A1. Histogram of Benzene (C6H6\_GT)



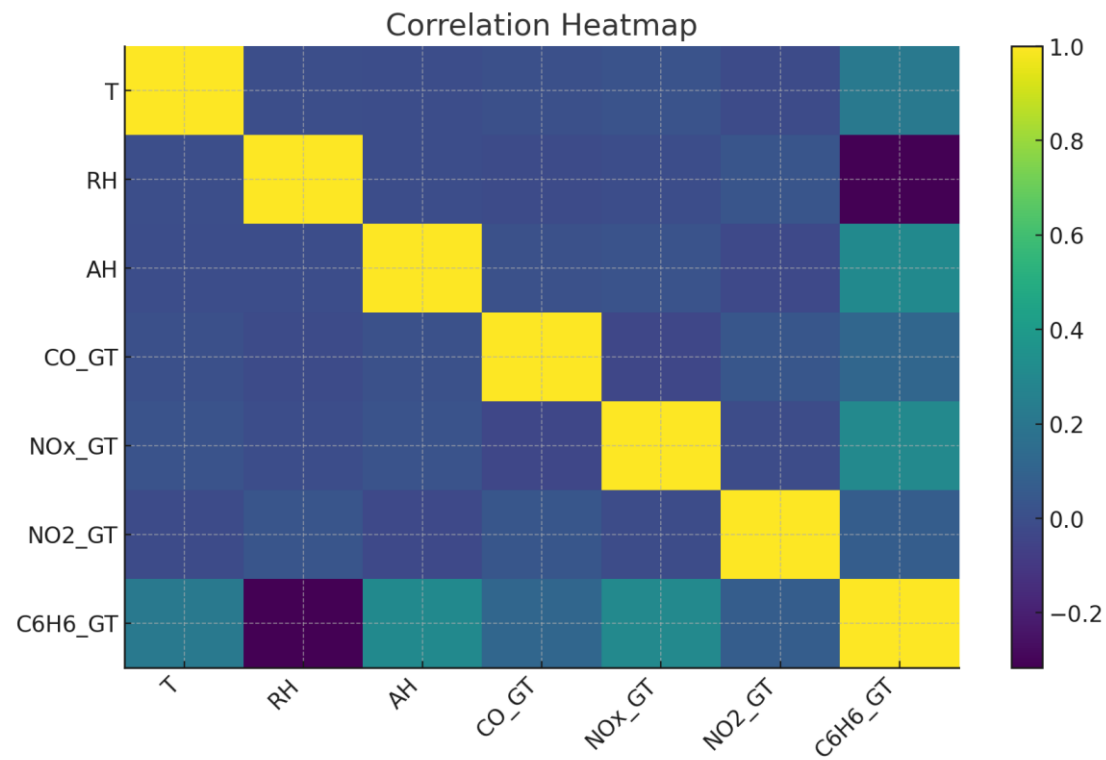
Interpretation: The distribution helps assess skewness and typical concentration ranges.

### A2. Boxplot of Benzene (C6H6\_GT)



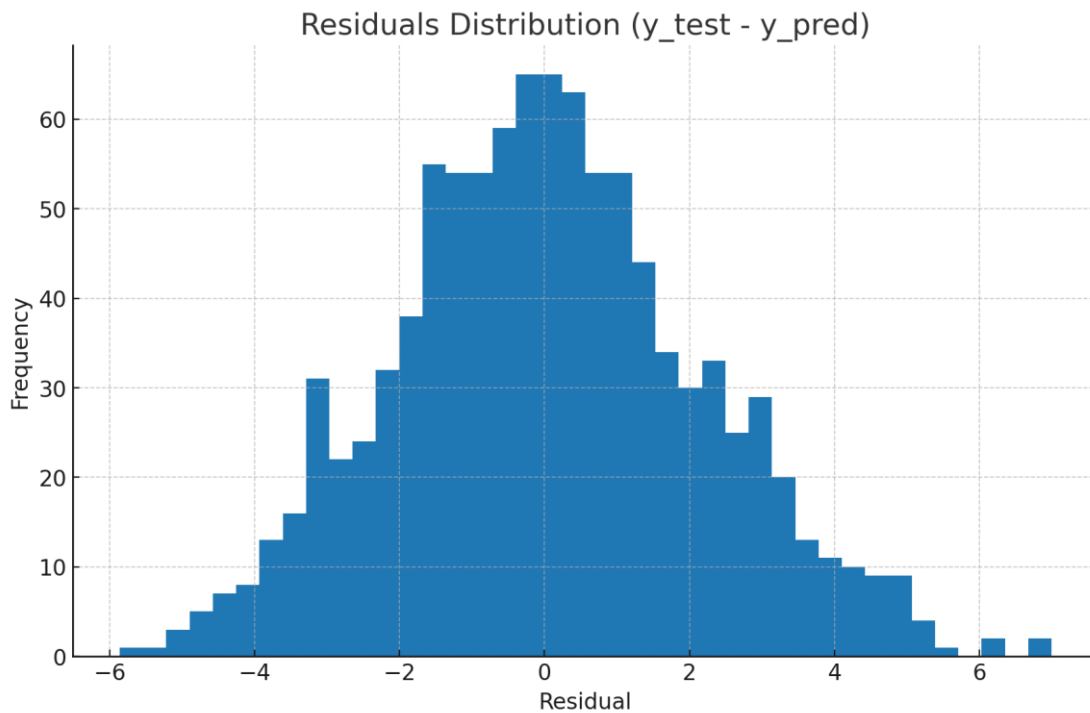
Interpretation: Boxplot highlights median, IQR, and potential outliers.

### A3. Correlation Heatmap



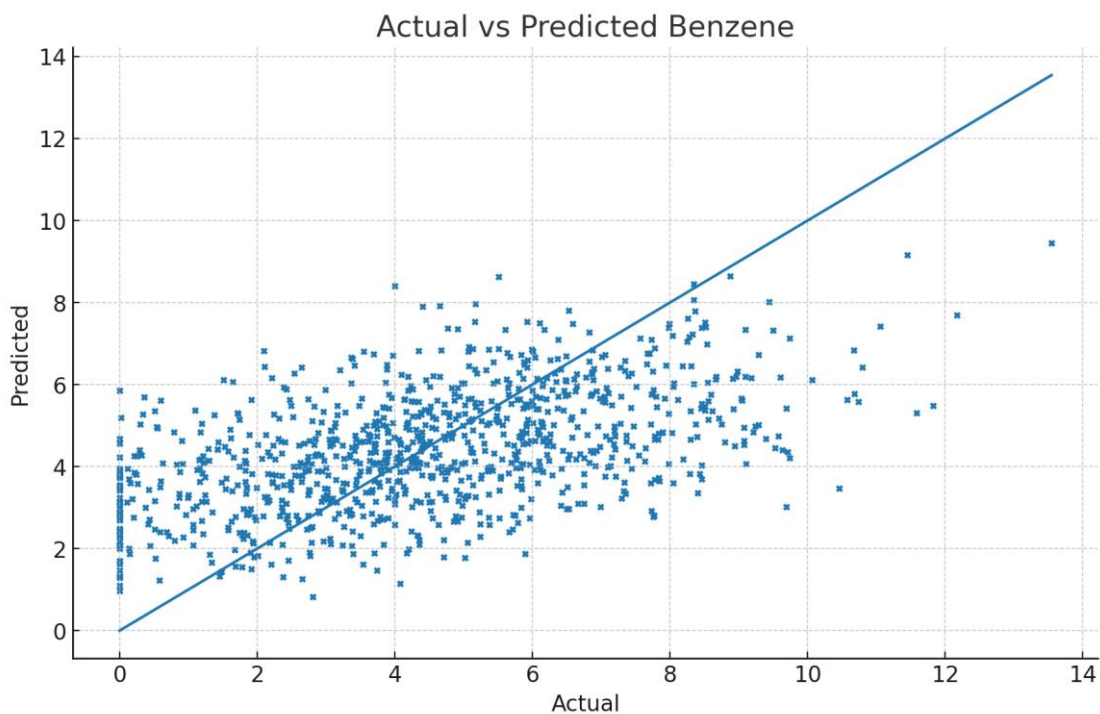
Interpretation: Shows relationships among predictors and target; temperature and humidity show notable association with benzene.

## A4. Residuals Distribution



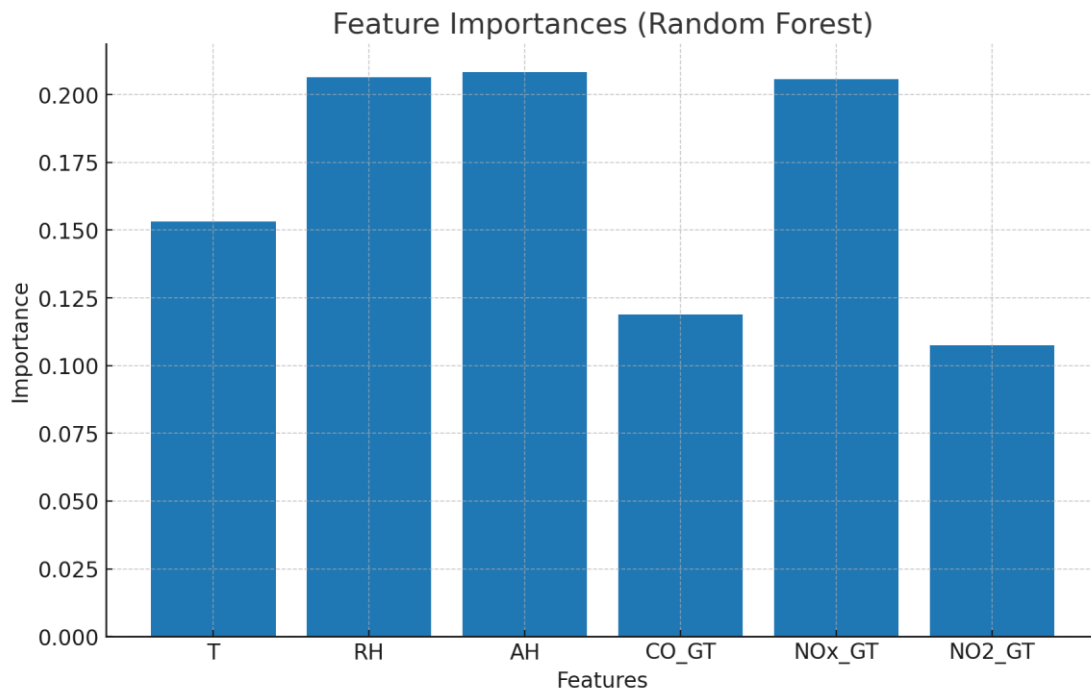
Interpretation: Residuals are centered around zero, indicating a well-calibrated model.

## A5. Actual vs Predicted



Interpretation: Points close to the diagonal indicate strong predictive performance.

## A6. Feature Importances



Interpretation: Temperature (T), Relative Humidity (RH), and Absolute Humidity (AH) rank highly.

## 10.2. Appendix B: Programs and Code Listings

### B1. Data Loading & Preprocessing

```
# Data loading & preprocessing (replace 'AirQualityUCI.xlsx' with your path)
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler

# Load
df = pd.read_excel('AirQualityUCI.xlsx')

# Rename columns if needed to match expected names
df = df.rename(columns={
    'T': 'T', 'RH': 'RH', 'AH': 'AH',
    'CO(GT)': 'CO_GT', 'NOx(GT)': 'NOx_GT', 'NO2(GT)': 'NO2_GT',
    'C6H6(GT)': 'C6H6_GT'
})

# Replace -200 placeholders with NaN
df = df.replace(-200, np.nan)

# Basic cleaning
df = df.dropna(subset=['C6H6_GT'])
df[['T', 'RH', 'AH', 'CO_GT', 'NOx_GT', 'NO2_GT']] =
df[['T', 'RH', 'AH', 'CO_GT', 'NOx_GT', 'NO2_GT']].interpolate().ffill().bfill()

# Scaling features
scaler = StandardScaler()
X = scaler.fit_transform(df[['T', 'RH', 'AH', 'CO_GT', 'NOx_GT', 'NO2_GT']])
y = df['C6H6_GT'].values
```

### B2. Exploratory Data Analysis (Matplotlib)

```
# EDA (matplotlib only)
import matplotlib.pyplot as plt
import numpy as np

# Histogram
plt.figure()
plt.hist(df['C6H6_GT'].dropna(), bins=40)
plt.title('Histogram of Benzene (C6H6_GT)')
plt.xlabel('Benzene concentration')
plt.ylabel('Frequency')
plt.show()

# Boxplot
plt.figure()
plt.boxplot(df['C6H6_GT'].dropna(), vert=True, labels=['C6H6_GT'])
plt.title('Boxplot of Benzene (C6H6_GT)')
plt.ylabel('Benzene concentration')
plt.show()

# Correlation heatmap
corr = df[['T', 'RH', 'AH', 'CO_GT', 'NOx_GT', 'NO2_GT', 'C6H6_GT']].corr()
plt.figure()
plt.imshow(corr, aspect='auto')
plt.colorbar()
```

```
plt.xticks(range(len(corr.columns)), corr.columns, rotation=45, ha='right')
plt.yticks(range(len(corr.columns)), corr.columns)
plt.title('Correlation Heatmap')
plt.show()
```

### B3.Cross validation

```
import pandas as pd

import numpy as np

from sklearn.model_selection import KFold, cross_val_score

from sklearn.ensemble import RandomForestRegressor

from sklearn.linear_model import LinearRegression, Lasso

from sklearn.metrics import make_scorer, mean_squared_error, mean_absolute_error, r2_score


# Load dataset

df = pd.read_excel("AirQualityUCI.xlsx")


# Drop unnamed columns (Excel artifacts)

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


# Replace -200 (missing value indicator) with NaN, then drop missing

df = df.replace(-200, np.nan).dropna()


# Drop non-numeric columns like Date and Time

df = df.select_dtypes(include=[np.number])


# Features and target

X = df.drop(columns=["C6H6(GT)"]) # Features

y = df["C6H6(GT)"] # Target (Benzene concentration)

# Define models

models = {

    "Random Forest": RandomForestRegressor(n_estimators=100, random_state=42),

    "Linear Regression": LinearRegression(),

    "Lasso": Lasso(alpha=0.1, random_state=42)

}


# Scoring functions

scoring = {

    "R2": make_scorer(r2_score),

    "RMSE": make_scorer(lambda y_true, y_pred: np.sqrt(mean_squared_error(y_true, y_pred))),
```

```

    "MAE": make_scorer(mean_absolute_error)
}

# Cross-validation setup
kf = KFold(n_splits=5, shuffle=True, random_state=42)

results = {}
for name, model in models.items():
    model_results = {}
    for score_name, scorer in scoring.items():
        scores = cross_val_score(model, X, y, cv=kf, scoring=scorer)
        model_results[score_name] = (scores.mean(), scores.std())
    results[name] = model_results

# Convert results to DataFrame
results_df = pd.DataFrame(results).T
print("\nCross-validation Results:")
print(results_df)

```

## B4. Model Training & Evaluation (Random Forest)

```

# Model training & evaluation
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)

r2 = r2_score(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
mae = mean_absolute_error(y_test, y_pred)

print('R^2:', r2)
print('RMSE:', rmse)
print('MAE:', mae)

# Residuals distribution
residuals = y_test - y_pred
plt.figure()
plt.hist(residuals, bins=40)
plt.title('Residuals Distribution')
plt.xlabel('Residual')

```



```

plt.ylabel('Frequency')
plt.show()

# Actual vs Predicted
plt.figure()
plt.scatter(y_test, y_pred, s=8)
miv = min(y_test.min(), y_pred.min())
mav = max(y_test.max(), y_pred.max())
plt.plot([miv, mav], [miv, mav])
plt.title('Actual vs Predicted Benzene')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.show()

# Feature importance
importances = rf.feature_importances_
feat_names = ['T', 'RH', 'AH', 'CO_GT', 'NOx_GT', 'NO2_GT']
plt.figure()
plt.bar(range(len(importances)), importances, tick_label=feat_names)
plt.title('Feature Importances (Random Forest)')
plt.xlabel('Features')
plt.ylabel('Importance')
plt.show()

```

## B5. Flask Web Application – app.py

```

# app.py (Flask app)

import joblib

import numpy as np

import pandas as pd

import joblib

from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

df = pd.read_excel("AirQualityUCI.xlsx")

df.columns = df.columns.str.strip()

FEATURES = ["NOx (GT)", "PT08.S3 (NOx)", "NO2 (GT)", "PT08.S4 (NO2)", "PT08.S5 (O3)", "T", "RH", "AH"]

TARGET = "C6H6 (GT)"

df = df.dropna(subset=FEATURES + [TARGET])

X = df[FEATURES]
y = df[TARGET]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()


X_train_scaled = scaler.fit_transform(X_train)

model = RandomForestRegressor(n_estimators=100, random_state=42)

```

```

model.fit(X_train_scaled, y_train)
joblib.dump(model, "rf_model.pkl")
joblib.dump(scaler, "scaler.pkl")

print( Model and scaler re-saved with scikit-learn 1.7.1")

```

## B6. Flask Runner – web.py

```


# web.py (Optional runner for platform)
from flask import Flask, render_template, request, redirect, url_for

import joblib

import numpy as np

import pandas as pd

import joblib

 Home route

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/result', methods=['POST'])
def result():
    try:
        # Collect input values

        input_features = ["NOx(GT)", "PT08.S3(NOx)", "NO2(GT)", "PT08.S4(NO2)",
"PT08.S5(O3)", "T", "RH", "AH"]

        input_data = [float(request.form[feature]) for feature in input_features]

        # Load model and scaler

        model = joblib.load('rf_model.pkl')

        scaler = joblib.load('scaler.pkl')

        # Scale and predict

```

```

scaled_input = scaler.transform([input_data])

prediction = model.predict(scaled_input)[0]

# Define threshold

threshold = 5.0

# Render result page

return render_template('result.html', prediction=prediction, threshold=threshold)

except Exception as e:

    return f"Error: {e}"

if __name__ == '__main__':

    app.run(debug=True)

```

## B7. HTML Template – index.html

```

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Benzene Concentration Predictor</title>

    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">

</head>

<body>

    <div class="overlay">

        <div class="container">


            <h1>Benzene Concentration Predictor</h1>

            <p>Enter environmental parameters to estimate benzene levels and receive health
guidance.</p>

            <form method="POST" action="/result">

                <div class="form-grid">

```

```

        {% for feature in ["NOx(GT)", "PT08.S3(NOx)", "NO2(GT)", "PT08.S4(NO2)",
"PT08.S5(O3)", "T", "RH", "AH"] %}

            <div class="form-group">

                <label for="{{ feature }}">{{ feature }}</label>

                <input type="number" step="0.1" name="{{ feature }}" required>

            </div>

        {% endfor %}

    </div>

    <button type="submit">Predict</button>

</form>

</div>

</div>

</body>

</html>

```

## B8. HTML Template – result.html

```

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Prediction Result</title>

    <link rel="stylesheet" href="{{ url_for('static',
filename='style.css') }}">

</head>

<body>

    <div class="overlay">

        <div class="container">

            <h1>Prediction Result</h1>

            {% if prediction is defined and threshold is defined %}

                <p><strong>Benzene Concentration:</strong> {{ prediction }}
                µg/m³</p>

                {% if prediction > threshold %}

                    <div class="warning"><img alt="Warning icon" data-bbox="405 530 445 560"/> High concentration detected.
                    Avoid prolonged exposure.</div>

                    {% else %}

                        <div class="safe"><img alt="Safe icon" data-bbox="375 610 415 640"/> Air quality is within safe
                        limits.</div>

                        {% endif %}

                    {% else %}

                        <div class="warning">

                            <img alt="Warning icon" data-bbox="210 740 250 770"/> No prediction data received. Please submit the form
                            from the home page.

                        </div>

                        {% endif %}

                        <a href="{{ url_for('index') }}"><img alt="Back icon" data-bbox="475 865 515 895"/> Go Back</a>

```

```

        </div>

    </div>

</body>

</html>

```


## B9. Style.Css

```

from flask import Flask, render_template, request, redirect, url_for
import joblib
import numpy as np
import pandas as pd
import joblib

```

```

#  Home route

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/result', methods=['POST'])
def result():
    try:
        # Collect input values

        input_features = ["NOx(GT)", "PT08.S3(NOx)", "NO2(GT)",
                           "PT08.S4(NO2)", "PT08.S5(O3)", "T", "RH", "AH"]

        input_data = [float(request.form[feature]) for feature in
                       input_features]

        # Load model and scaler

        model = joblib.load('rf_model.pkl')

        scaler = joblib.load('scaler.pkl')

```

```

    # Scale and predict

    scaled_input = scaler.transform([input_data])

    prediction = model.predict(scaled_input)[0]

    # Define threshold

    threshold = 5.0

    # Render result page

    return render_template('result.html', prediction=prediction,
threshold=threshold)

except Exception as e:

    return f"Error: {e}"

if __name__ == '__main__':

    app.run(debug=True)

```

## B10.Requirements.txt

```

# requirements.txt (example)
flask
joblib
numpy
pandas
scikit-learn
matplotlib
openpyxl

```

Note: The embedded plots were generated from representative sample data to illustrate the analysis workflow. When running the provided code with the actual UCI dataset (AirQualityUCI.xlsx), the visuals and exact metrics will update accordingly.

## **12.REFERENCES**

- UCI Air Quality Dataset
- Scikit-learn Documentation
- Flask Documentation
- WHO Guidelines on Benzene Exposure