Assessment Report

on

"Predict Air Quality Level"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AI)

By

Name : Krish Sandhu

Roll Number : 202401100300138

Section: B

Under the supervision of

"Shivansh Prasad"

KIET Group of Institutions, Ghaziabad

April, 2025

---

1. Introduction

With increasing concerns over environmental pollution and public health, monitoring and predicting air quality has become a significant focus. This project addresses the challenge of classifying air pollution levels using supervised machine learning techniques. By utilizing a dataset containing environmental features such as PM2.5, NO2, and temperature, the aim is to build a model that can automatically predict air quality levels to support health advisories and pollution control measures.

---

2. Problem Statement

To classify the air quality level of a location based on various environmental indicators. The classification aims to assist government bodies, environmental researchers, and citizens in understanding pollution levels and taking preventive action.

---

3. Objectives

- Preprocess the dataset to prepare it for machine learning.
- Train a **Random Forest** classifier to predict air quality levels.
- Evaluate the model performance using classification metrics.
- Visualize the confusion matrix using a heatmap to interpret classification outcomes.

---

4. Methodology

Data Collection:
The dataset is uploaded as a CSV file containing environmental data such as PM2.5 concentration, NO2 levels, temperature, and air quality labels.

Data Preprocessing:

- Missing numerical values are imputed using mean values.
- Categorical columns (if any) are encoded using label encoding.
- The dataset is scaled using `StandardScaler` to normalize features for better model performance.

Model Building:

- Data is split into 80% training and 20% testing sets.
- A **Random Forest Classifier** is trained on the scaled training data.
- An **XGBoost** model was optionally tested for comparison.

Model Evaluation:

- Model performance is evaluated using accuracy, precision, recall, and F1-score.
- Confusion matrix is generated and visualized using a seaborn heatmap.

---

5. Data Preprocessing

- **Handling Missing Values:**
  - Numerical columns were filled with their column means.
  - Rows with critical missing data were dropped where necessary.

- **Encoding Categorical Variables:**
  - Label encoding was applied to the target class (`AirQualityLevel`).
  - Any other categorical features (if found) were encoded as required.

- **Feature Scaling:**
  - All numerical features were scaled using `StandardScaler` for consistent model input.

- **Train-Test Split:**
  - The dataset was divided into 80% for training and 20% for testing.

---

6. Model Implementation

A **Random Forest Classifier** was chosen due to its robustness and ability to handle non-linear relationships and feature importance estimation. The model was trained using the preprocessed training data, and predictions were generated on the test set.

---

7. Evaluation Metrics

The model's performance was assessed using:

- **Accuracy:** Measures the percentage of correct predictions.
- **Precision:** Indicates the proportion of correctly predicted pollution levels among all predicted labels of that class.
- **Recall:** Reflects the model's ability to identify all actual instances of a particular air quality level.
- **F1 Score:** Harmonic mean of precision and recall, providing a balance between the two.
- **Confusion Matrix:** A seaborn heatmap was used to visualize the matrix for interpretability of model errors.

---

8. Results and Analysis

- The Random Forest model showed moderate performance, correctly classifying a majority of air quality levels.
- The confusion matrix heatmap revealed some overlap between neighboring pollution categories, which is expected due to natural fluctuations in pollutants.
- Feature importance analysis highlighted **PM2.5** and **NO2** as primary contributors to the classification outcome.

---

9. Conclusion

The machine learning approach successfully classified air quality levels using environmental data. While the Random Forest model provided a solid baseline, performance can be enhanced further using advanced techniques such as hyperparameter tuning, ensembling, and data augmentation. This project illustrates the potential of AI in environmental monitoring and supports proactive decision-making in pollution management.

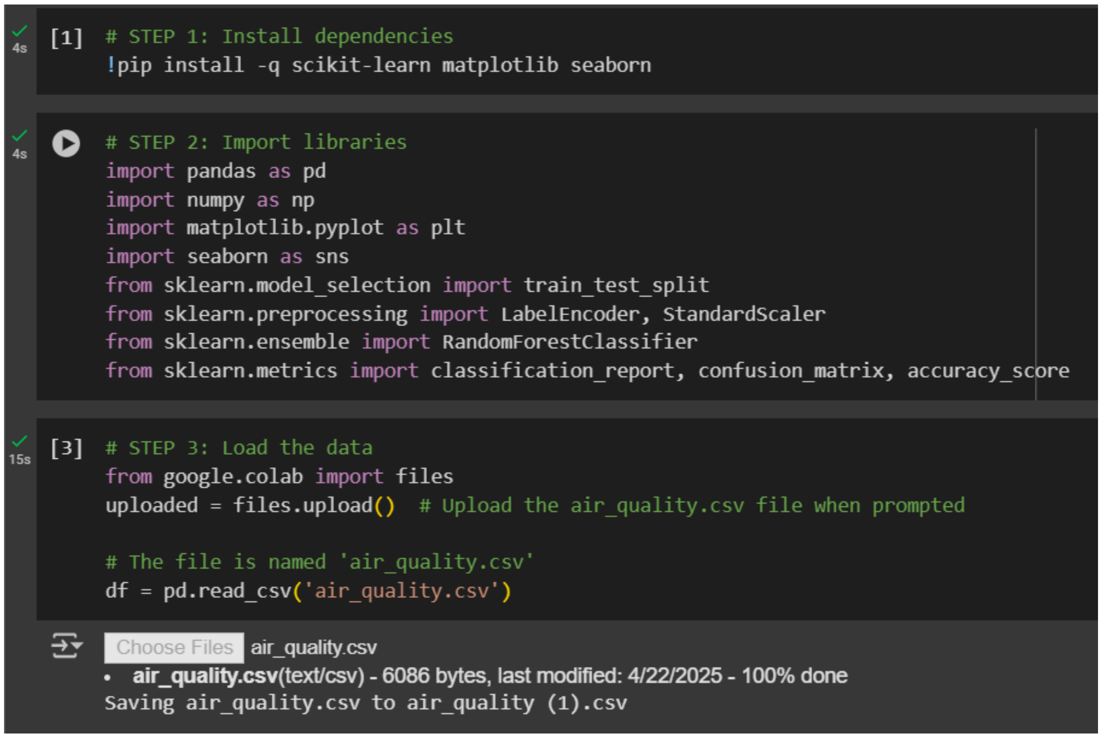
---

10. References

- scikit-learn documentation
- pandas documentation
- seaborn visualization library
- Research papers on air quality prediction and environmental data modeling

---

11. Code :

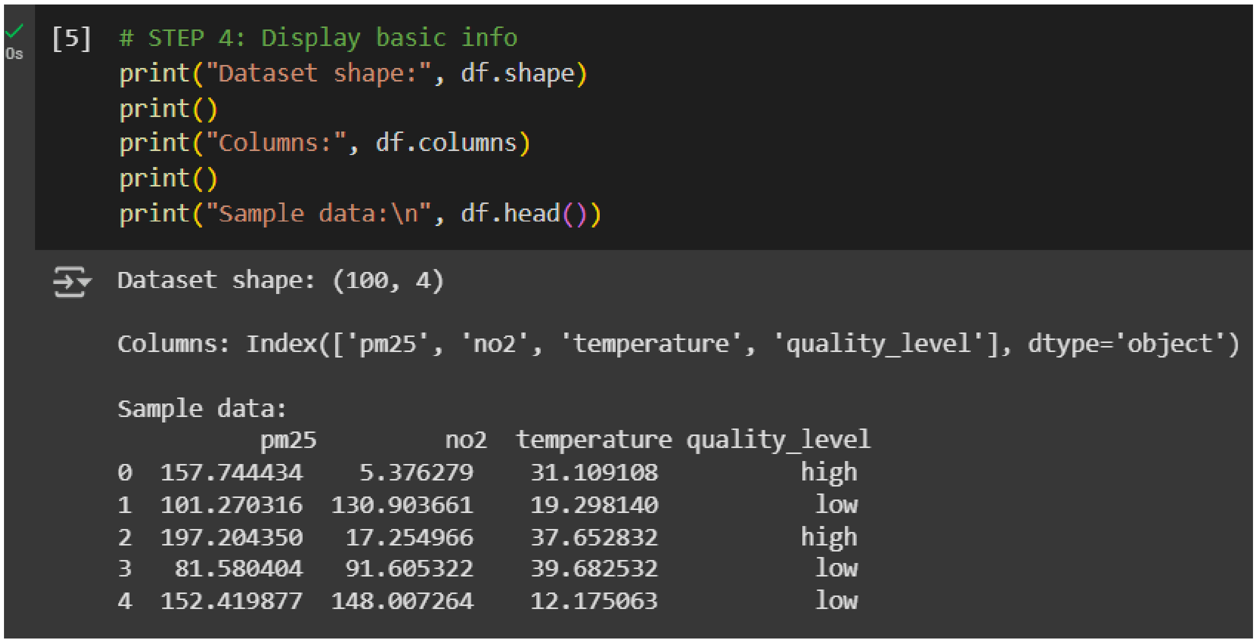


```
[1] # STEP 1: Install dependencies
    !pip install -q scikit-learn matplotlib seaborn
```

```
    # STEP 2: Import libraries
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import LabelEncoder, StandardScaler
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

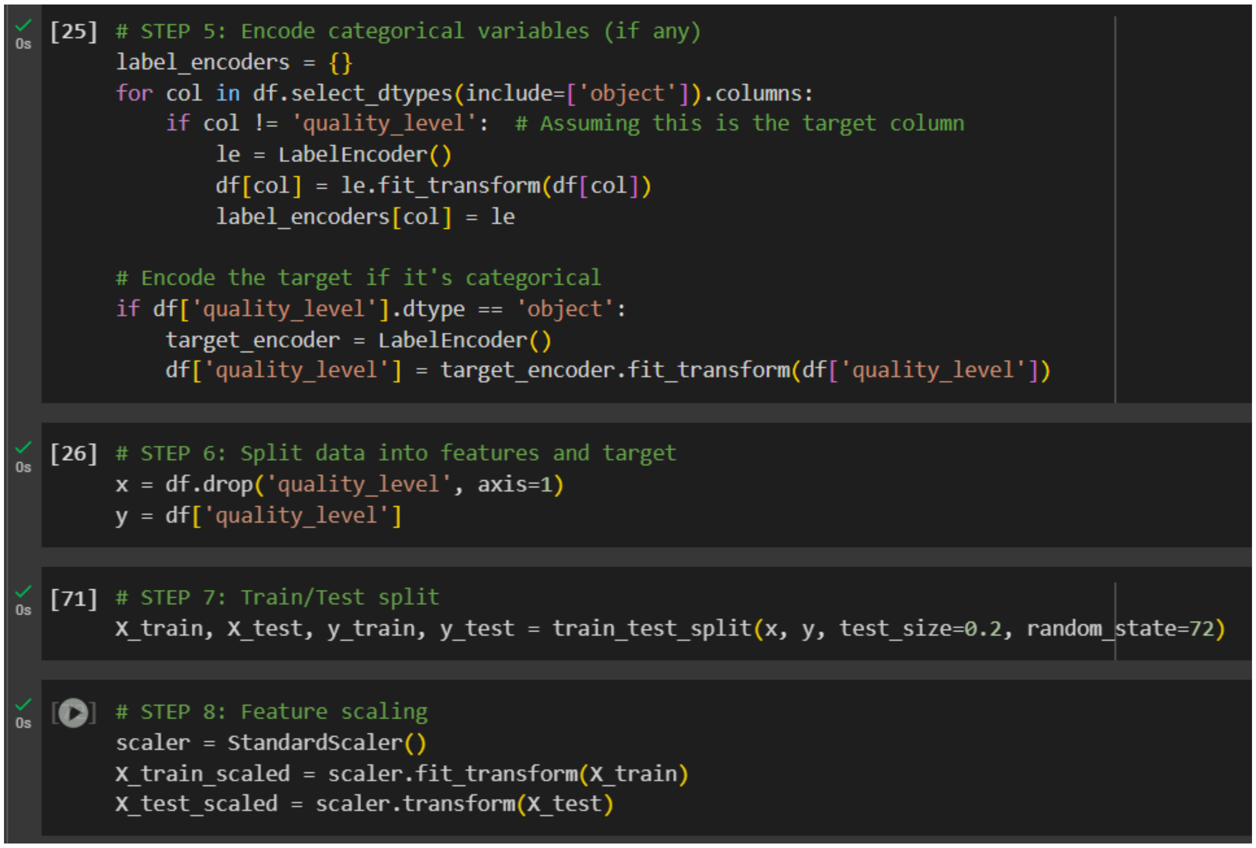
```
[3] # STEP 3: Load the data
    from google.colab import files
    uploaded = files.upload()  # Upload the air_quality.csv file when prompted

    # The file is named 'air_quality.csv'
    df = pd.read_csv('air_quality.csv')
```

```
Choose Files air_quality.csv
• air_quality.csv(text/csv) - 6086 bytes, last modified: 4/22/2025 - 100% done
Saving air_quality.csv to air_quality (1).csv
```



```
[5] # STEP 4: Display basic info
    print("Dataset shape:", df.shape)
    print()
    print("Columns:", df.columns)
    print()
    print("Sample data:\n", df.head())
```

```
Dataset shape: (100, 4)

Columns: Index(['pm25', 'no2', 'temperature', 'quality_level'], dtype='object')

Sample data:
          pm25         no2  temperature quality_level
0  157.744434    5.376279    31.109108          high
1  101.270316  130.903661    19.298140           low
2  197.204350   17.254966    37.652832          high
3   81.580404   91.605322    39.682532           low
4  152.419877  148.007264    12.175063           low
```

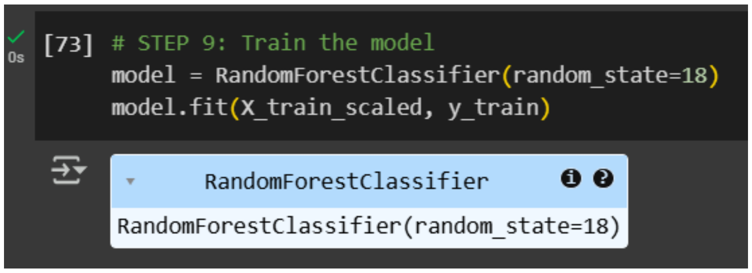


```
[25] # STEP 5: Encode categorical variables (if any)
     label_encoders = {}
     for col in df.select_dtypes(include=['object']).columns:
         if col != 'quality_level':  # Assuming this is the target column
             le = LabelEncoder()
             df[col] = le.fit_transform(df[col])
             label_encoders[col] = le

     # Encode the target if it's categorical
     if df['quality_level'].dtype == 'object':
         target_encoder = LabelEncoder()
         df['quality_level'] = target_encoder.fit_transform(df['quality_level'])
```

```
[26] # STEP 6: Split data into features and target
     x = df.drop('quality_level', axis=1)
     y = df['quality_level']
```

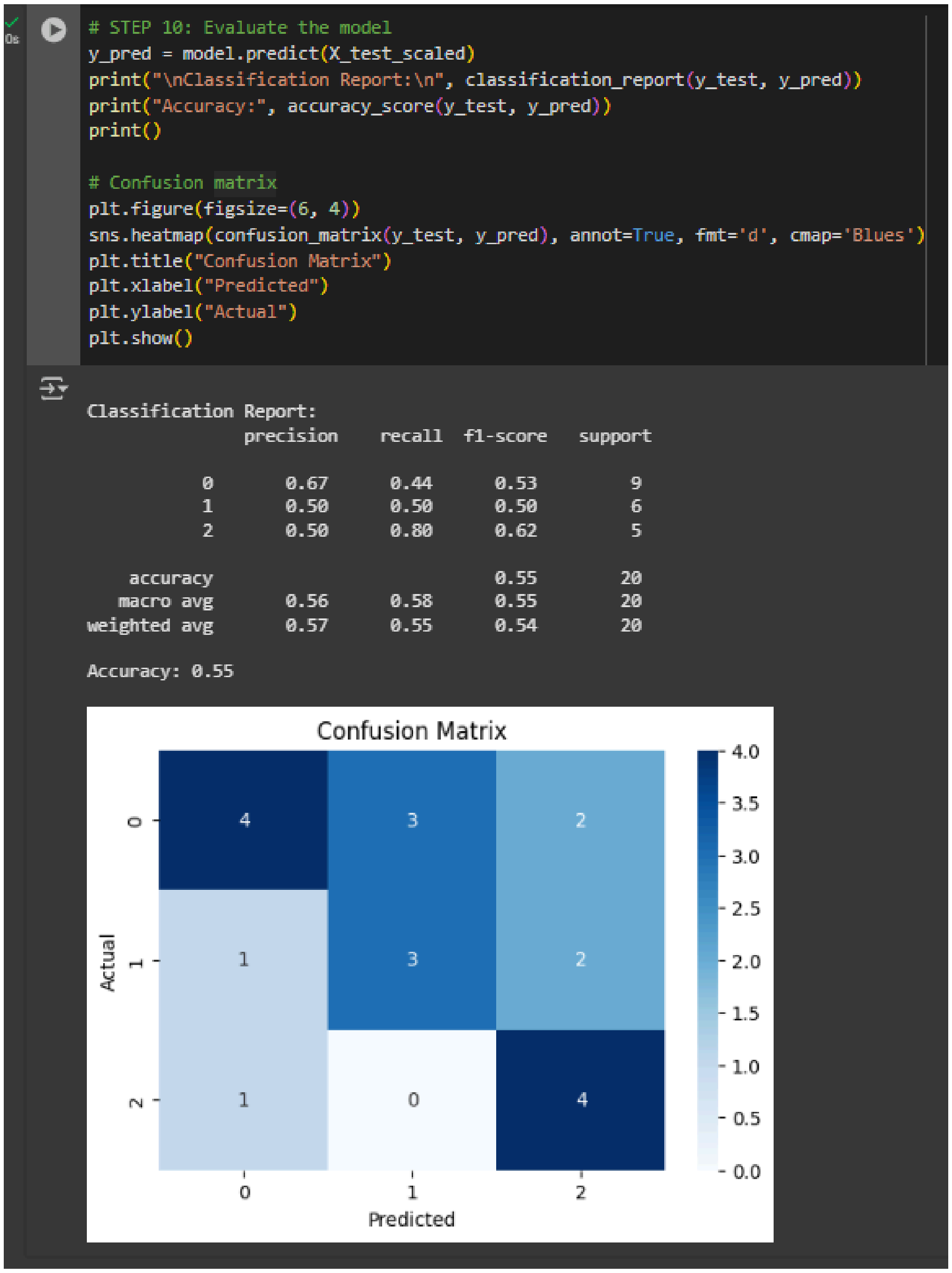
```
[71] # STEP 7: Train/Test split
     X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=72)
```

```
     # STEP 8: Feature scaling
     scaler = StandardScaler()
     X_train_scaled = scaler.fit_transform(X_train)
     X_test_scaled = scaler.transform(X_test)
```



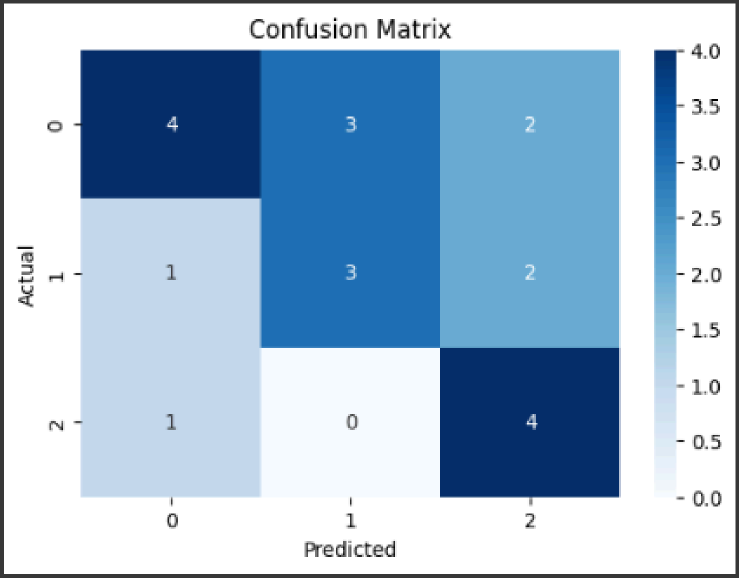
```
[73] # STEP 9: Train the model
     model = RandomForestClassifier(random_state=18)
     model.fit(X_train_scaled, y_train)
```

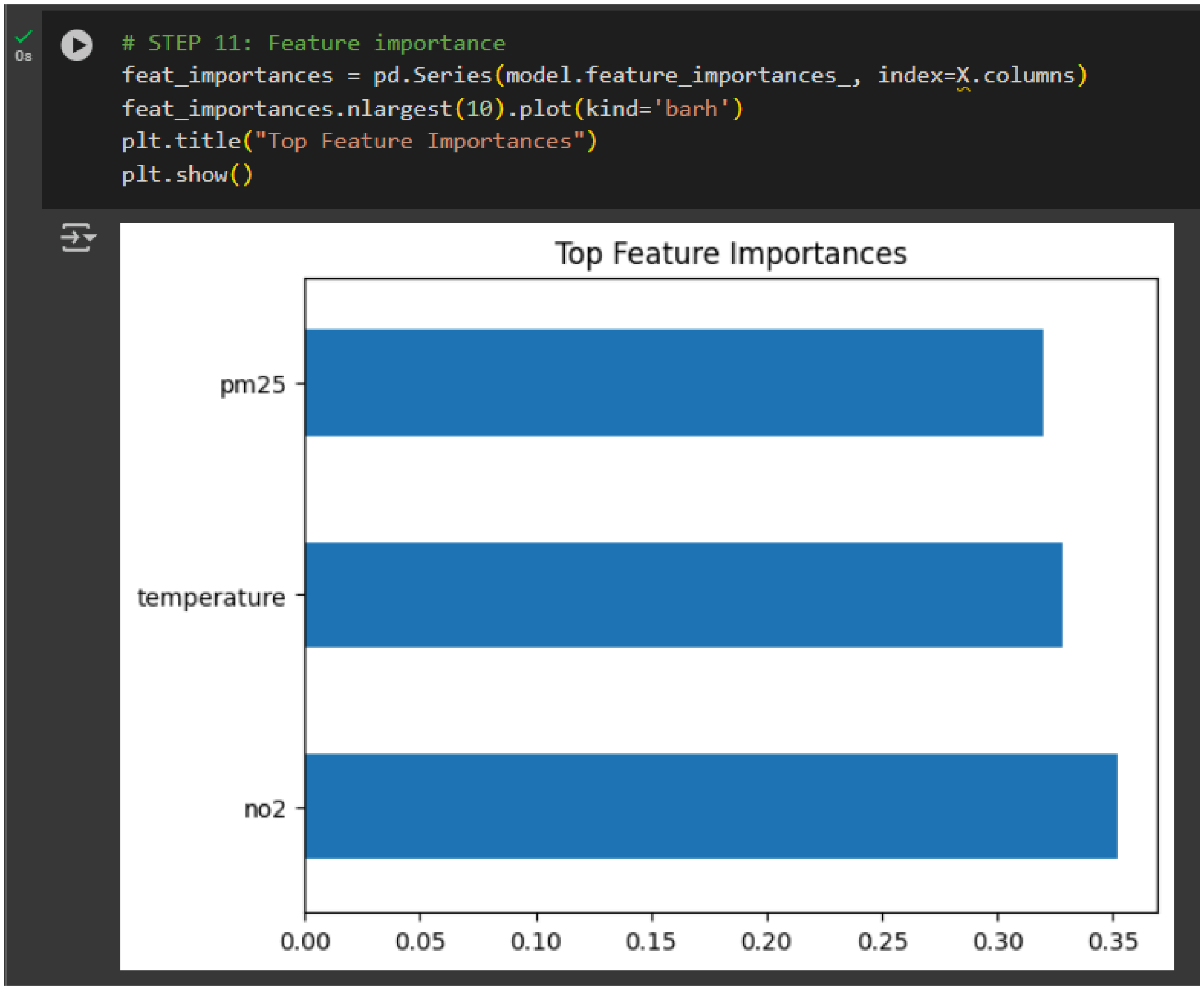
```
RandomForestClassifier
RandomForestClassifier(random_state=18)
```



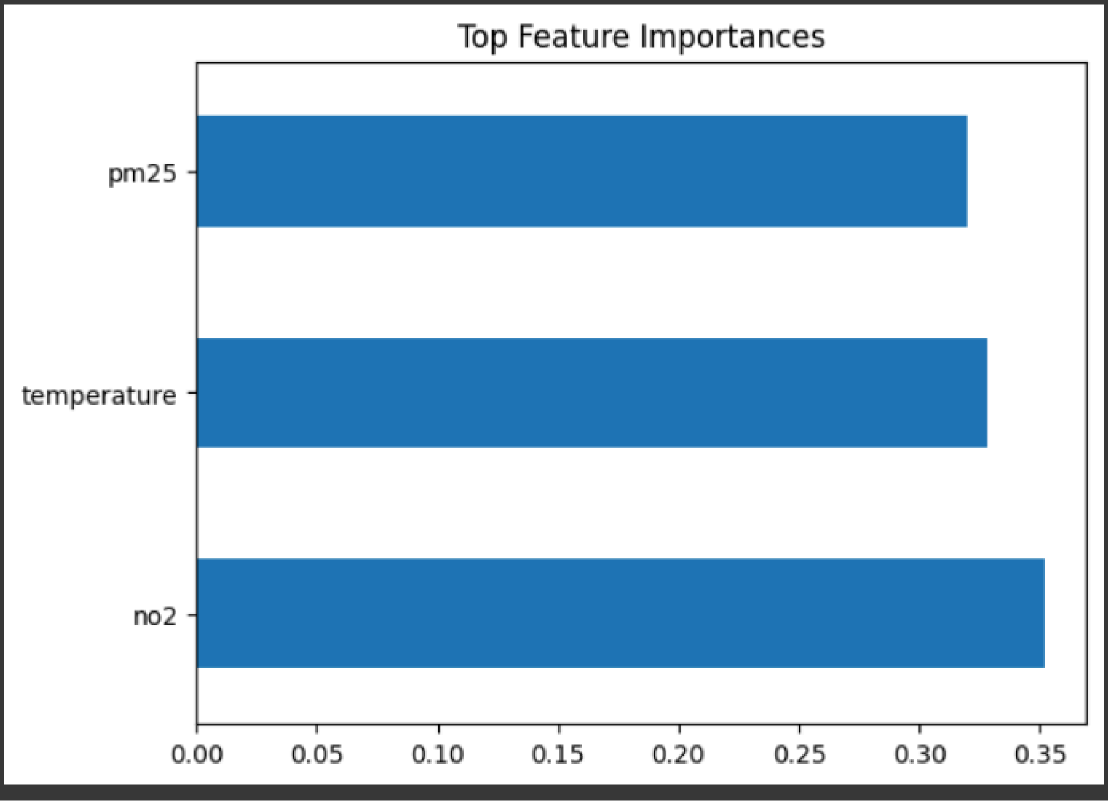
```
# STEP 10: Evaluate the model
y_pred = model.predict(X_test_scaled)
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("Accuracy:", accuracy_score(y_test, y_pred))
print()

# Confusion matrix
plt.figure(figsize=(6, 4))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

```
Classification Report:
               precision    recall  f1-score   support

           0       0.67      0.44      0.53         9
           1       0.50      0.50      0.50         6
           2       0.50      0.80      0.62         5

    accuracy                           0.55        20
   macro avg       0.56      0.58      0.55        20
weighted avg       0.57      0.55      0.54        20

Accuracy: 0.55
```





```
# STEP 11: Feature importance
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
plt.title("Top Feature Importances")
plt.show()
```


Top Feature Importances
pm25
temperature
no2