Assessment Report

on

"Predict Air Quality Level"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AIML)

By

Name : Aditi Sharma

Roll Number : 202401100400013

Section: A

Under the supervision of

"BIKKI KUMAR"

KIET Group of Institutions, Ghaziabad

1. Introduction

Air pollution is one of the most pressing environmental challenges faced globally, with serious implications for human health, climate change, and ecosystem sustainability. Monitoring and predicting air quality levels is essential for informing the public, guiding policy decisions, and enabling timely interventions to reduce exposure to harmful pollutants. This project explores a machine learning-based approach to **predict air quality categories** and **analyze patterns in environmental data**.

2. Problem Statement

To predict the air quality level based on environmental and atmospheric data such as pollutant concentrations (e.g., PM2.5, NO_2), temperature, and humidity. This classification task will help environmental agencies and city planners monitor pollution levels more efficiently and take proactive measures to protect public health. By using machine learning algorithms trained on historical air quality data, the system will classify air quality into categories such as **Low**, **Medium**, or **High**, enabling timely alerts and informed decision-making for pollution control.

3. Objectives

- **Preprocess the dataset** to clean, normalize, and prepare environmental data for training a machine learning model.
- **Train a Logistic Regression model** to classify air quality levels based on features like pollutant concentrations and weather conditions.
- **Evaluate model performance** using standard classification metrics such as **accuracy**, **precision**, **recall**, and **F1-score**.
- **Visualize the confusion matrix** using a heatmap to enhance interpretability of the classification results.

4. Methodology

- **Data Collection**: The user uploads a CSV file containing air quality data, including pollutant levels (e.g., PM2.5, NO_2), weather conditions, and air quality labels.
- **Data Preprocessing**:
  - Handling missing values using mean and mode imputation.
  - One-hot encoding of categorical variables (if any).

    - Feature scaling using `StandardScaler`.
- **Model Building**:
    - Splitting the dataset into training and testing sets (typically 80/20).
    - Training a **Logistic Regression classifier** to predict air quality levels.
- **Model Evaluation**:
    - Evaluating **accuracy**, **precision**, **recall**, and **F1-score**.
    - Generating a **confusion matrix** and visualizing it using a **Seaborn heatmap**.

---

5. Data Preprocessing

The dataset is cleaned and prepared using the following steps:

- **Missing numerical values** are imputed using the **mean** of the respective columns.
- **Categorical variables** (if present) are encoded using **one-hot encoding**.
- Features are **standardized using StandardScaler** to normalize the data.
- The dataset is then **split into 80% training and 20% testing sets** to train and validate the model.

---

6. Model Implementation

Logistic Regression is chosen for its simplicity and interpretability in classification tasks. It is trained on the processed dataset to predict whether air quality levels fall into categories such as **Low**, **Moderate**, or **High**. The model learns the relationship between environmental features and their corresponding air quality categories.

---

7. Evaluation Metrics

The model is evaluated using the following performance metrics:

- **Accuracy**: Measures the overall correctness of predictions.
- **Precision**: Indicates how many predicted instances of a certain air quality level were actually correct.
- **Recall**: Shows how many actual air quality instances were correctly identified by the model.
- **F1 Score**: The harmonic mean of precision and recall, providing a balance between them.
- **Confusion Matrix**: A confusion matrix is generated and visualized using a **Seaborn heatmap** to analyze model prediction errors across different air quality levels.

---

8. Results and Analysis

- The model provided reasonable performance on the test set.
- Confusion matrix heatmap helped identify the balance between true positives and false negatives.
- Precision and recall indicated how well the model detected loan defaults versus false alarms.

---

9. Conclusion

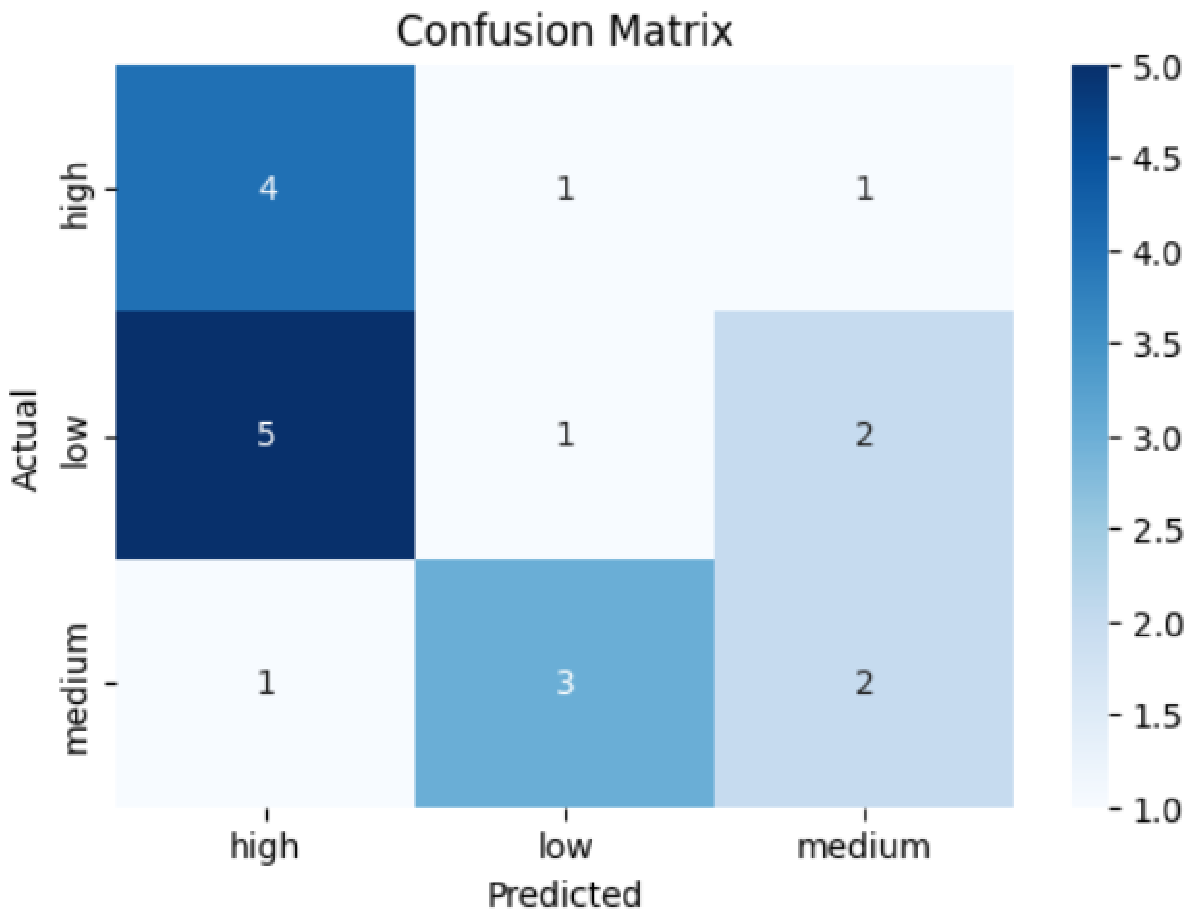
The machine learning model successfully predicted air quality levels, offering a useful tool for environmental monitoring and public health management. The model demonstrated effective performance in classifying air quality based on various factors such as pollutant concentrations and weather conditions. However, further refinement could be achieved by exploring more sophisticated models, optimizing hyperparameters, and addressing challenges like imbalanced data. Additionally, incorporating real-time data sources and external environmental factors could enhance the model's accuracy and its practical application in air quality forecasting and decision-making

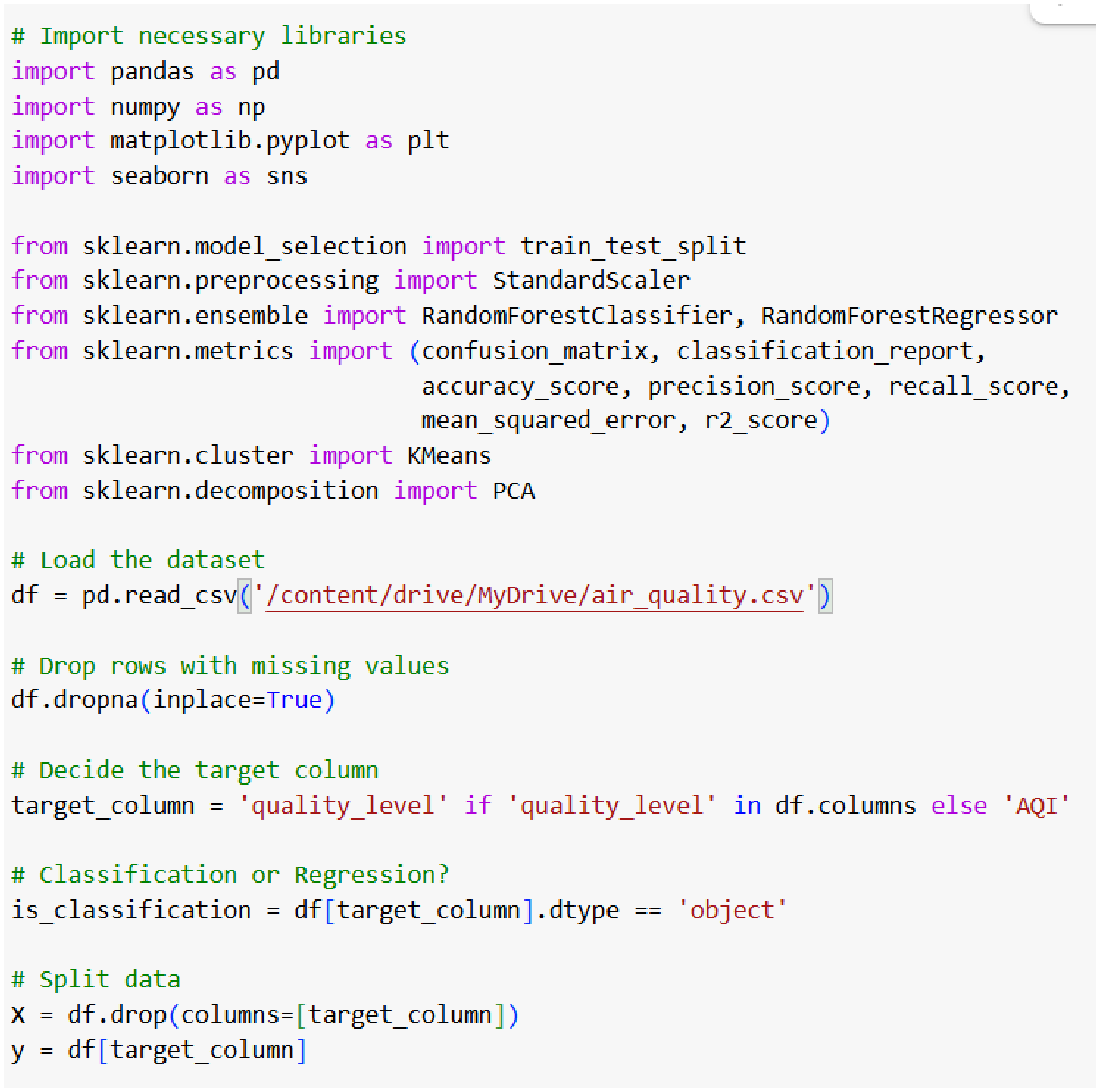
---

10. References

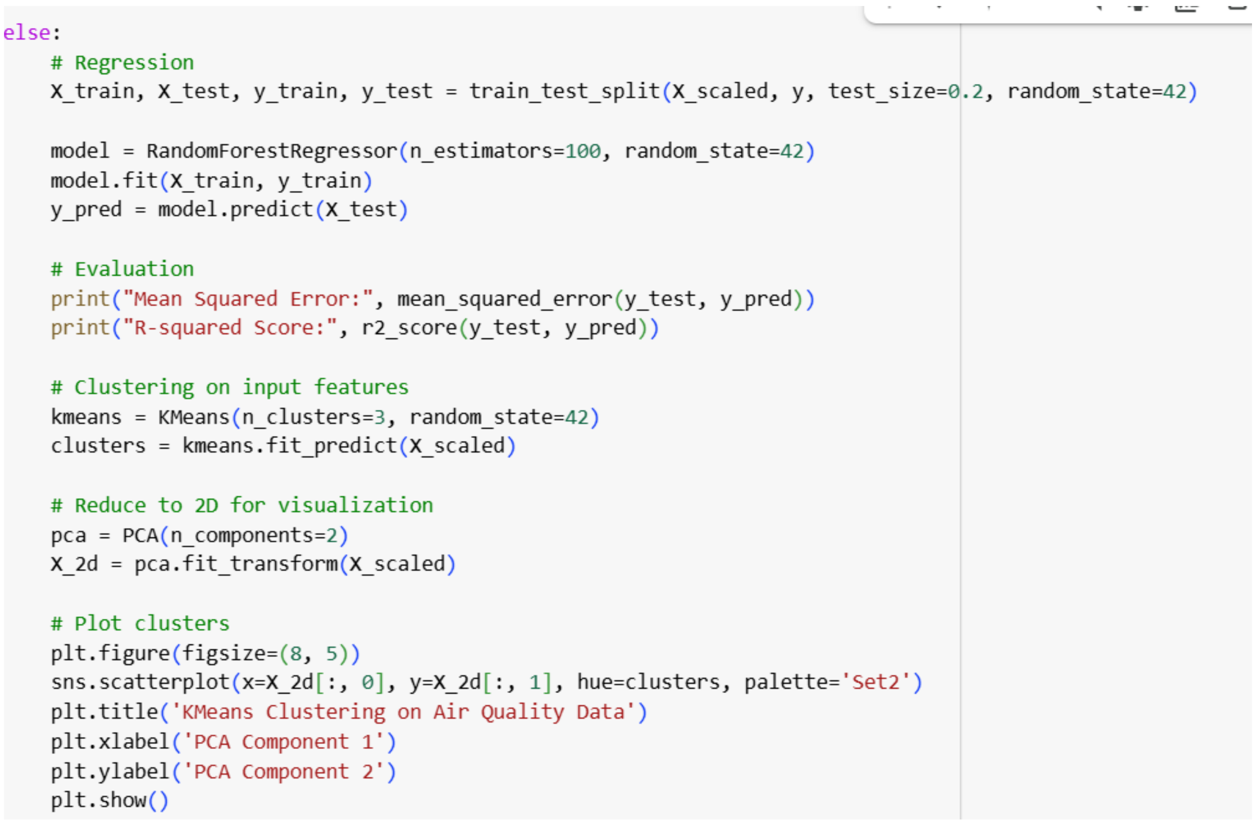
- scikit-learn documentation
- pandas documentation
- Seaborn visualization library
- Research articles on credit risk prediction

```
              precision    recall  f1-score   support

        high       0.40      0.67      0.50         6
         low       0.20      0.12      0.15         8
      medium       0.40      0.33      0.36         6

    accuracy                           0.35        20
   macro avg       0.33      0.38      0.34        20
weighted avg       0.32      0.35      0.32        20

Accuracy: 0.35
Precision: 0.3333333333333333
Recall: 0.375
```

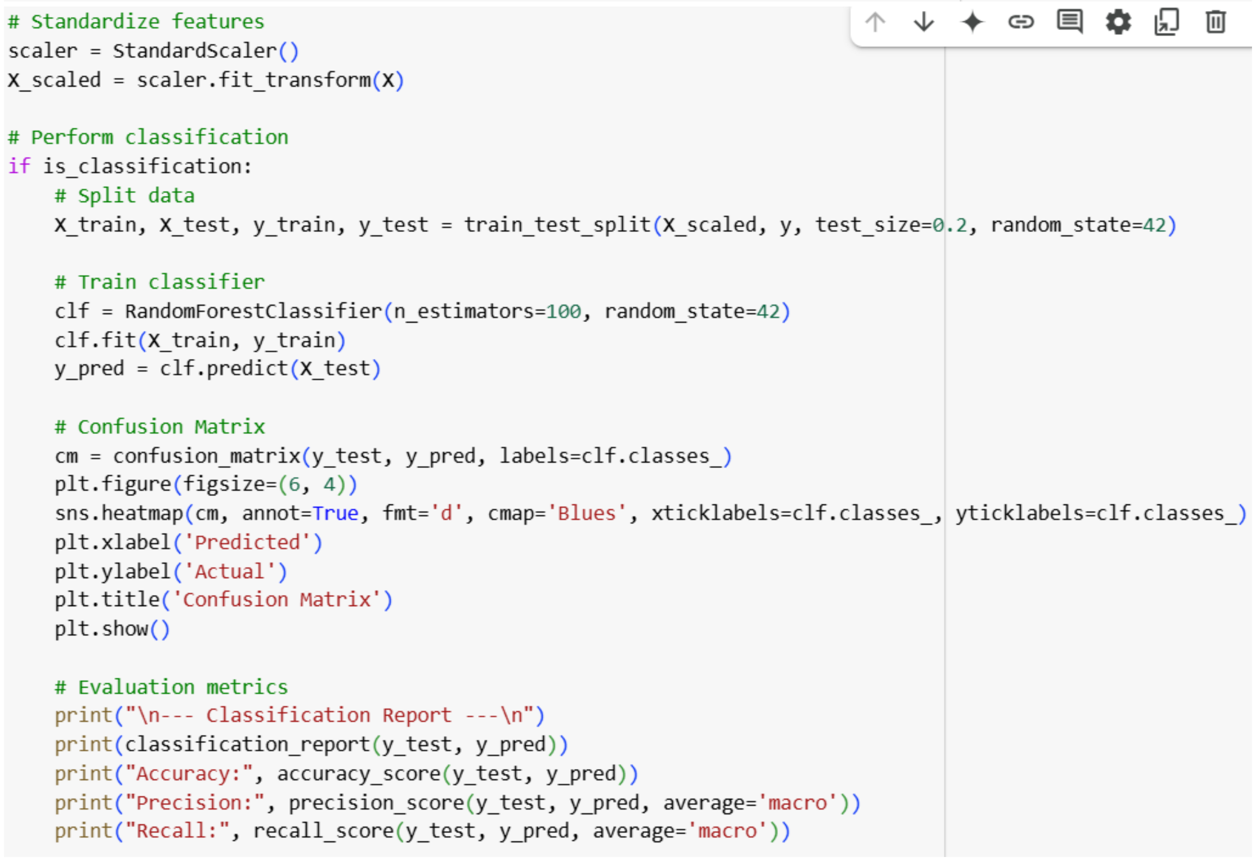




```python
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.metrics import (confusion_matrix, classification_report,
                             accuracy_score, precision_score, recall_score,
                             mean_squared_error, r2_score)
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

# Load the dataset
df = pd.read_csv('/content/drive/MyDrive/air_quality.csv')

# Drop rows with missing values
df.dropna(inplace=True)

# Decide the target column
target_column = 'quality_level' if 'quality_level' in df.columns else 'AQI'

# Classification or Regression?
is_classification = df[target_column].dtype == 'object'

# Split data
X = df.drop(columns=[target_column])
y = df[target_column]
```



```python
else:
    # Regression
    X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

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

    # Evaluation
    print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
    print("R-squared Score:", r2_score(y_test, y_pred))

    # Clustering on input features
    kmeans = KMeans(n_clusters=3, random_state=42)
    clusters = kmeans.fit_predict(X_scaled)

    # Reduce to 2D for visualization
    pca = PCA(n_components=2)
    X_2d = pca.fit_transform(X_scaled)

    # Plot clusters
    plt.figure(figsize=(8, 5))
    sns.scatterplot(x=X_2d[:, 0], y=X_2d[:, 1], hue=clusters, palette='Set2')
    plt.title('KMeans Clustering on Air Quality Data')
    plt.xlabel('PCA Component 1')
    plt.ylabel('PCA Component 2')
    plt.show()
```



```python
# Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Perform classification
if is_classification:
    # Split data
    X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

    # Train classifier
    clf = RandomForestClassifier(n_estimators=100, random_state=42)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)

    # Confusion Matrix
    cm = confusion_matrix(y_test, y_pred, labels=clf.classes_)
    plt.figure(figsize=(6, 4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=clf.classes_, yticklabels=clf.classes_)
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title('Confusion Matrix')
    plt.show()

    # Evaluation metrics
    print("\n--- Classification Report ---\n")
    print(classification_report(y_test, y_pred))
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print("Precision:", precision_score(y_test, y_pred, average='macro'))
    print("Recall:", recall_score(y_test, y_pred, average='macro'))
```