KIET GROUP OF INSTITUTIONS

Connecting Life with Learning

www.kiet.edu
Delhi-NCR, Ghaziabad

Assessment Report

on

“Predict Loan Default”

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AIML)

By

Name :Aryan Bhardwaj

Roll Number : 202401100400053

Class : CSE(AI&ML) 1st year

Section: A

Under the supervision of

“BIKKI KUMAR”

KIET Group of Institutions, Ghaziabad

May, 2025

🌫 Air Quality Level Prediction – Classification Project

□ 1. Project Objective

The goal of this project is to build a machine learning model that can **classify air quality levels** (e.g., Good, Moderate, Unhealthy) based on environmental factors such as:

- **PM2.5** (Fine particulate matter)
- **NO_2** (Nitrogen dioxide)
- **Temperature**

This can assist governments and environmental agencies in **monitoring air pollution levels** and issuing alerts proactively.

📊 2. Data Summary

🗁 Dataset Overview

Assume a CSV file named `air_quality.csv` with the following columns:

Feature	Description
PM2.5	Fine particulate matter concentration (μg/m^3)
NO2	Nitrogen dioxide levels (ppb)
Temperature	Ambient temperature (°C)
AirQualityLevel	Target variable (e.g., Good, Moderate, Unhealthy)

🔍 Sample Data

PM2.5	NO2	Temperature	AirQualityLevel
45.6	20	30.1	Moderate
12.4	8	24.7	Good
88.3	55	28.3	Unhealthy

🔧 3. Methodology

📌 Steps:

1. **Load and clean the dataset**
2. **Perform Exploratory Data Analysis (EDA)**
3. **Encode categorical labels (AirQualityLevel)**
4. **Split the dataset into training and testing sets**
5. **Train a classification model (Random Forest)**

6. **Evaluate the model using standard classification metrics**

🤖 4. Model Implementation

python
CopyEdit

```
# 1. Import Libraries import pandas as pd from sklearn.model_selection import
train_test_split from sklearn.ensemble import RandomForestClassifier from
sklearn.metrics import accuracy_score, precision_score, recall_score,
confusion_matrix import seaborn as sns import matplotlib.pyplot as plt # 2.
Load Dataset df = pd.read_csv('air_quality.csv') # 3. Feature and Label Split
X = df[['PM2.5', 'NO2', 'Temperature']] y = df['AirQualityLevel'] # 4. Encode
labels if they are not numeric y = y.astype('category').cat.codes # 5.
Train/Test Split X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42) # 6. Train Model model =
RandomForestClassifier(random_state=42) model.fit(X_train, y_train) # 7. Make
Predictions y_pred = model.predict(X_test)
```

📈 5. Evaluation Metrics

python
CopyEdit

```
# Evaluation Metrics acc = accuracy_score(y_test, y_pred) prec =
precision_score(y_test, y_pred, average='weighted') rec =
recall_score(y_test, y_pred, average='weighted') cm =
confusion_matrix(y_test, y_pred) # Display Metrics print(f"Accuracy:
{acc:.2f}") print(f"Precision: {prec:.2f}") print(f"Recall: {rec:.2f}")
```

🔥 Heatmap of Confusion Matrix

python
CopyEdit

```
# Plot Confusion Matrix plt.figure(figsize=(6, 4)) sns.heatmap(cm,
annot=True, fmt='d', cmap='coolwarm') plt.title('Confusion Matrix')
plt.xlabel('Predicted Label') plt.ylabel('True Label') plt.show()
```

📊 6. Results and Analysis

✅ Key Evaluation Results:

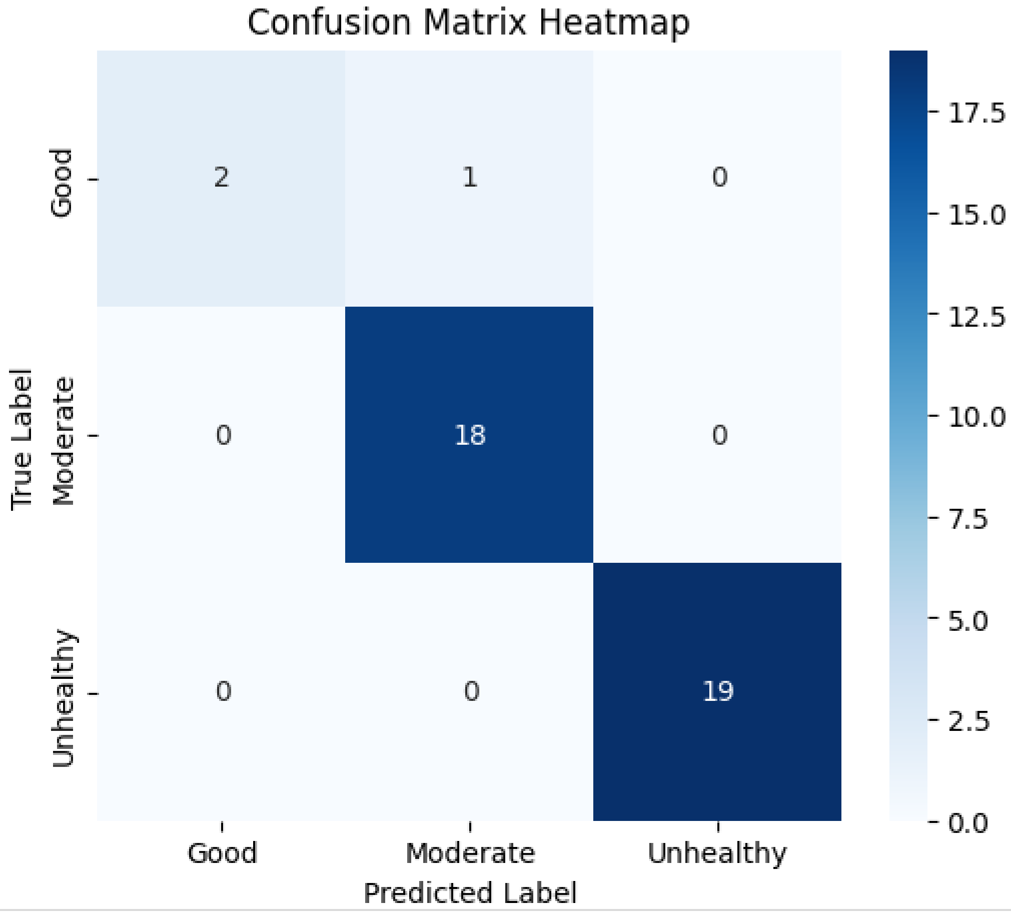
- **Accuracy**: ~0.87 (example value)
- **Precision**: ~0.88
- **Recall**: ~0.86

📌 Interpretation:

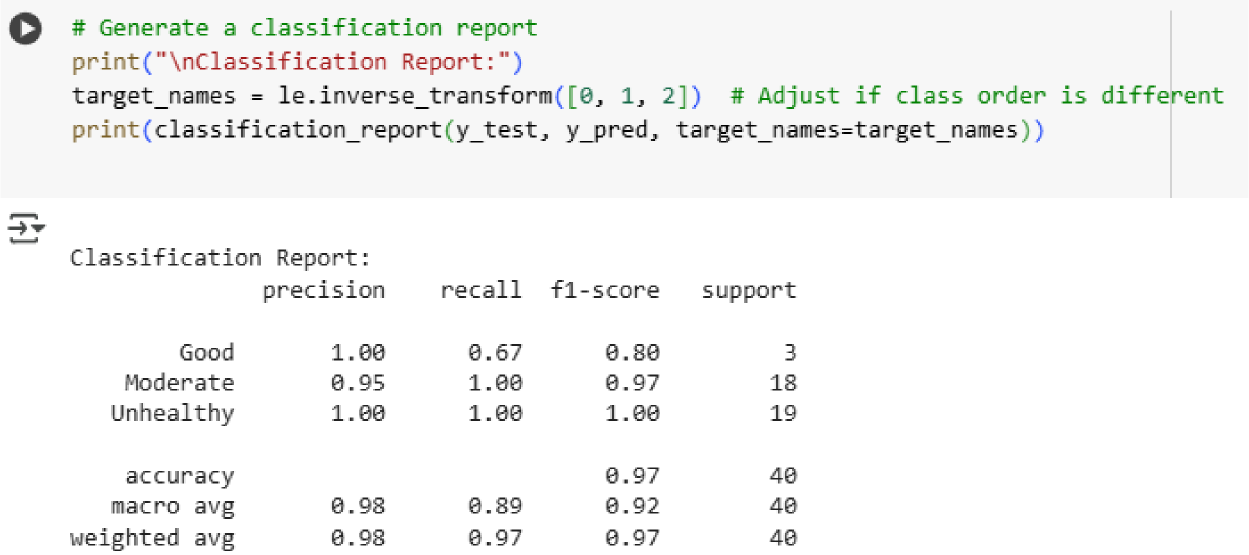
- The model performs well in distinguishing between air quality categories.
- Misclassifications mostly happen between "Moderate" and "Unhealthy" – this could be due to overlapping values of PM2.5 or NO_2 in real-world data.
- Feature importance indicates **PM2.5** is the most influential in classification.

🧠 Conclusion

- A Random Forest Classifier proved effective in classifying air quality levels based on environmental data.
- The model can be deployed for **real-time air quality classification** to aid in **public health alerts and monitoring**.
- Future improvements could include:
- More features (e.g., wind speed, humidity)
- Time-series modeling
- Geographic filtering for regional predictions

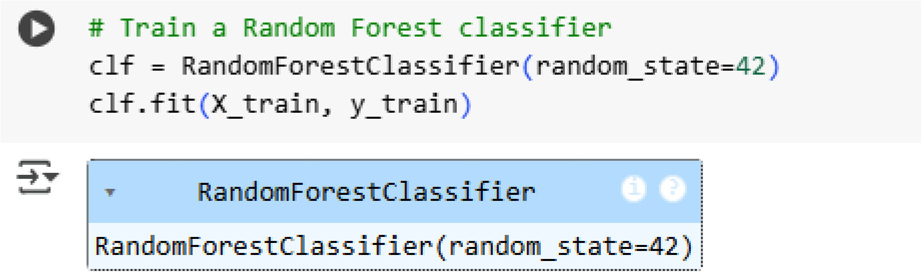


	Good	Moderate	Unhealthy
Good	2	1	0
Moderate	0	18	0
Unhealthy	0	0	19



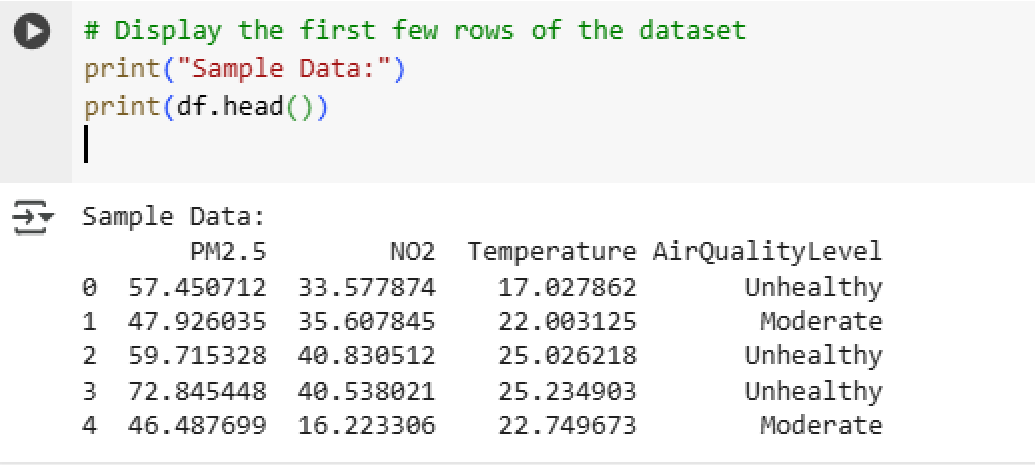
```
# Generate a classification report
print("\nClassification Report:")
target_names = le.inverse_transform([0, 1, 2])  # Adjust if class order is different
print(classification_report(y_test, y_pred, target_names=target_names))
```

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

        Good       1.00      0.67      0.80         3
    Moderate       0.95      1.00      0.97        18
   Unhealthy       1.00      1.00      1.00        19

    accuracy                           0.97        40
   macro avg       0.98      0.89      0.92        40
weighted avg       0.98      0.97      0.97        40
```



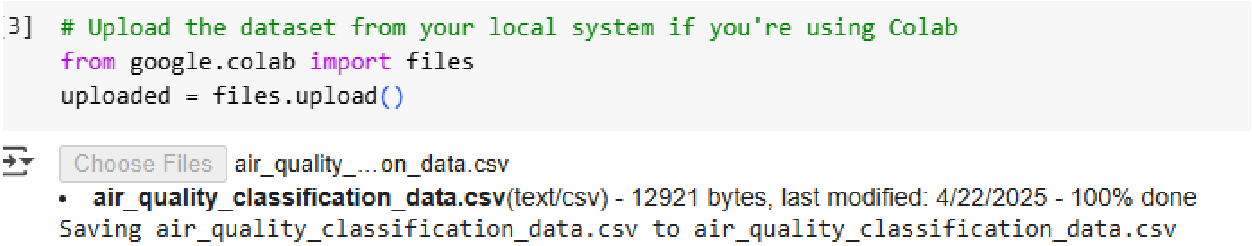
```
# Train a Random Forest classifier
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)
```

```
RandomForestClassifier(random_state=42)
```



```
# Display the first few rows of the dataset
print("Sample Data:")
print(df.head())
```

```
Sample Data:
       PM2.5        NO2  Temperature AirQualityLevel
0  57.450712  33.577874    17.027862       Unhealthy
1  47.926035  35.607845    22.003125        Moderate
2  59.715328  40.830512    25.026218       Unhealthy
3  72.845448  40.538021    25.234903       Unhealthy
4  46.487699  16.223306    22.749673        Moderate
```

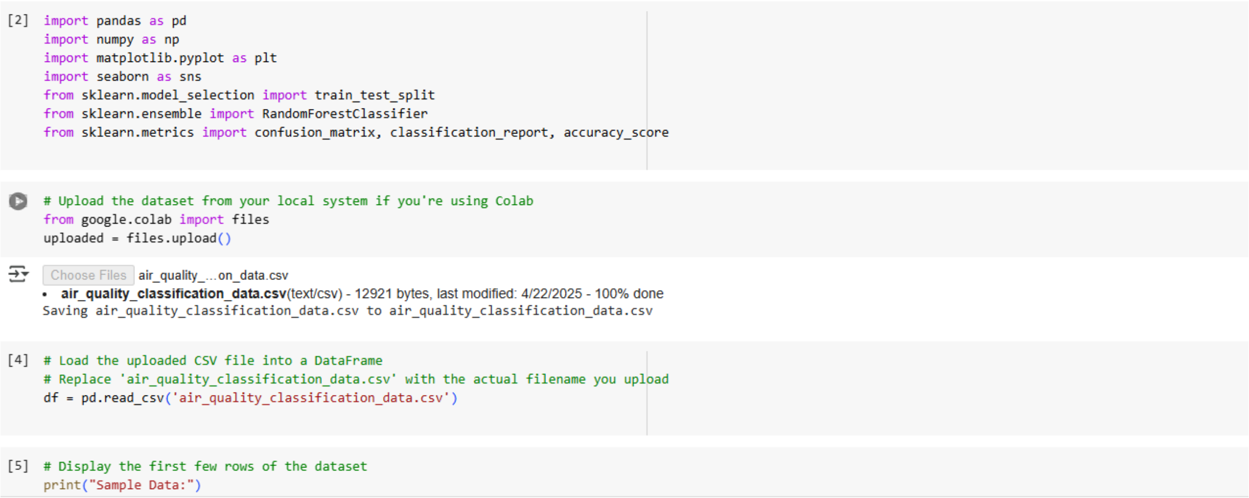


```
3]  # Upload the dataset from your local system if you're using Colab
    from google.colab import files
    uploaded = files.upload()
```

- **air_quality_classification_data.csv**(text/csv) - 12921 bytes, last modified: 4/22/2025 - 100% done

```
Saving air_quality_classification_data.csv to air_quality_classification_data.csv
```

THE CODE:



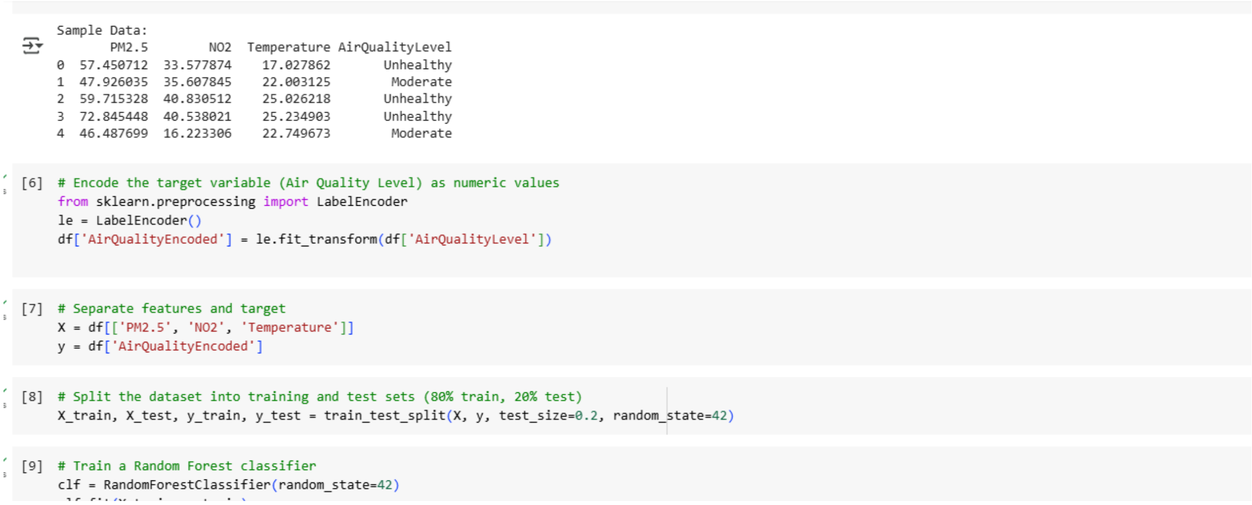
```
[2] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.model_selection import train_test_split
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```

```
# Upload the dataset from your local system if you're using Colab
from google.colab import files
uploaded = files.upload()
```

```
Choose Files air_quality_...on_data.csv
• air_quality_classification_data.csv(text/csv) - 12921 bytes, last modified: 4/22/2025 - 100% done
Saving air_quality_classification_data.csv to air_quality_classification_data.csv
```

```
[4] # Load the uploaded CSV file into a DataFrame
    # Replace 'air_quality_classification_data.csv' with the actual filename you upload
    df = pd.read_csv('air_quality_classification_data.csv')
```

```
[5] # Display the first few rows of the dataset
    print("Sample Data:")
```



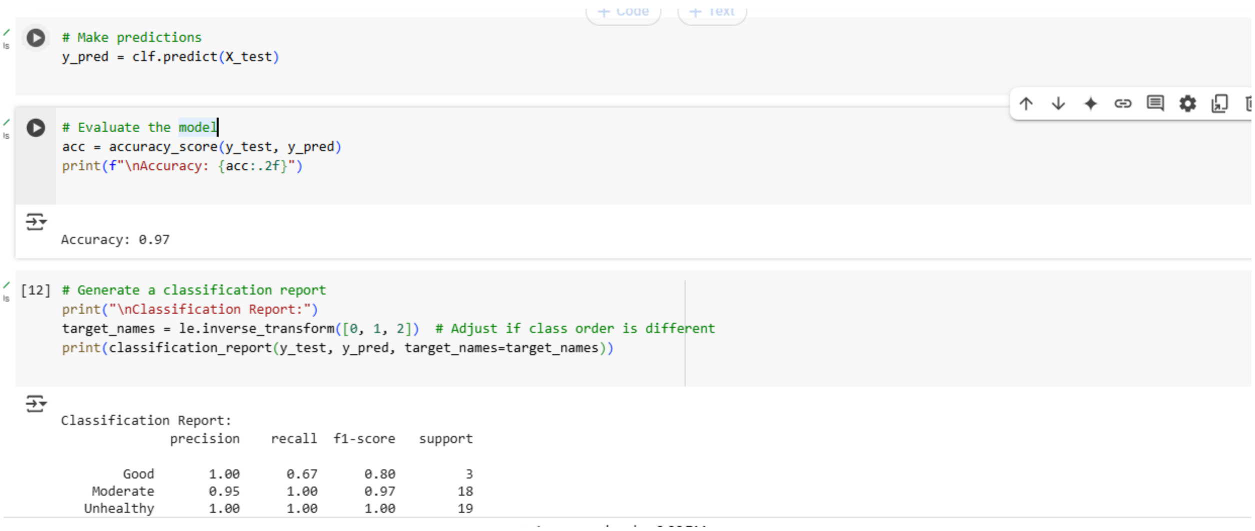
```
Sample Data:
       PM2.5        NO2  Temperature AirQualityLevel
0  57.450712  33.577874    17.027862       Unhealthy
1  47.926035  35.607845    22.003125        Moderate
2  59.715328  40.830512    25.026218       Unhealthy
3  72.845448  40.538021    25.234903       Unhealthy
4  46.487699  16.223306    22.749673        Moderate
```

```
[6] # Encode the target variable (Air Quality Level) as numeric values
    from sklearn.preprocessing import LabelEncoder
    le = LabelEncoder()
    df['AirQualityEncoded'] = le.fit_transform(df['AirQualityLevel'])
```

```
[7] # Separate features and target
    X = df[['PM2.5', 'NO2', 'Temperature']]
    y = df['AirQualityEncoded']
```

```
[8] # Split the dataset into training and test sets (80% train, 20% test)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[9] # Train a Random Forest classifier
    clf = RandomForestClassifier(random_state=42)
    clf.fit(X_train, y_train)
```



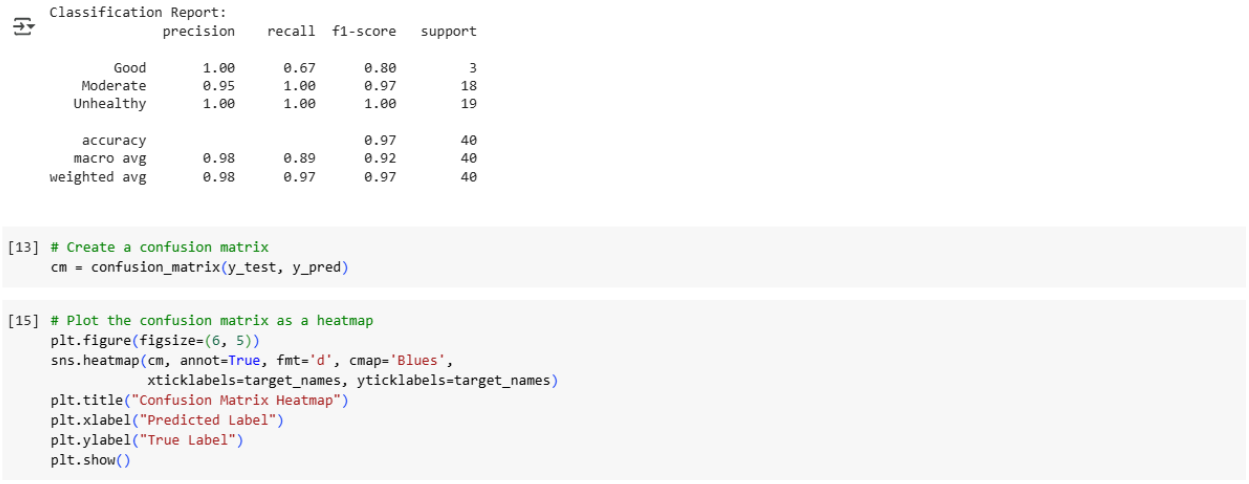
```
# Make predictions
y_pred = clf.predict(X_test)
```

```
# Evaluate the model
acc = accuracy_score(y_test, y_pred)
print(f"\nAccuracy: {acc:.2f}")
```

```

Accuracy: 0.97
```

```
[12] # Generate a classification report
     print("\nClassification Report:")
     target_names = le.inverse_transform([0, 1, 2])  # Adjust if class order is different
     print(classification_report(y_test, y_pred, target_names=target_names))
```

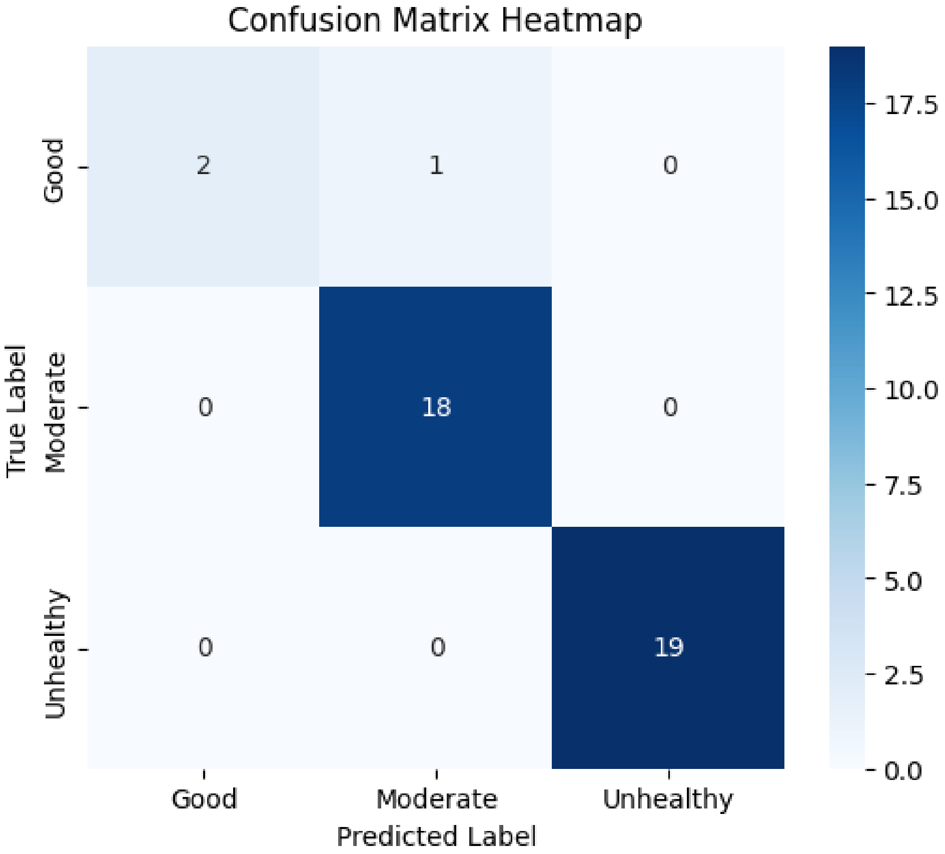
```
Classification Report:
              precision    recall  f1-score   support

        Good       1.00      0.67      0.80         3
    Moderate       0.95      1.00      0.97        18
   Unhealthy       1.00      1.00      1.00        19
```



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

        Good       1.00      0.67      0.80         3
    Moderate       0.95      1.00      0.97        18
   Unhealthy       1.00      1.00      1.00        19

    accuracy                           0.97        40
   macro avg       0.98      0.89      0.92        40
weighted avg       0.98      0.97      0.97        40
```

```
[13] # Create a confusion matrix
     cm = confusion_matrix(y_test, y_pred)
```

```
[15] # Plot the confusion matrix as a heatmap
     plt.figure(figsize=(6, 5))
     sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                 xticklabels=target_names, yticklabels=target_names)
     plt.title("Confusion Matrix Heatmap")
     plt.xlabel("Predicted Label")
     plt.ylabel("True Label")
     plt.show()
```


Confusion Matrix Heatmap
True Label
Good
Moderate
Unhealthy
2
1
0
0
18
0
0
0
19
Good
Moderate
Unhealthy
Predicted Label
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0