Assessment Report

on

"Air Quality Predictor"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AI&ML)

By

Name : Ayush Chaudhary

Roll Number : 202401100400063

Section: A

Under the supervision of

"BIKKI KUMAR SIR"

KIET Group of Institutions, Ghaziabad

---

1. Introduction

This project aims to classify air quality levels based on environmental features such as PM2.5, NO2, and Temperature. The data is analyzed to assign air quality levels into six categories: "Good," "Moderate," "Unhealthy for Sensitive Groups," "Unhealthy," "Very Unhealthy," and "Hazardous."

The classification is achieved using a rule-based approach that calculates a composite air quality score. The results are visualized using charts to better understand the data and predictions.

---

2. Problem Statement

Understanding and monitoring air quality is critical for public health, urban planning, and environmental protection. Poor air quality contributes to respiratory illnesses, cardiovascular diseases, and reduced life expectancy. The challenge lies in effectively classifying air quality levels based on environmental data such as PM2.5, NO2, and temperature, and presenting actionable insights.

This project aims to develop a rule-based classification system that assigns air quality levels into six predefined categories. By leveraging simple yet interpretable calculations and visualizations, the project provides a user-friendly method for understanding the impact of key environmental factors on air quality.

---

3. Methodology

Feature Extraction and Score Calculation

A composite air quality score was calculated using the formula:

Classification

The air quality score was classified into six levels based on predefined thresholds:

AQI Level Classification

Good	<=20
Moderate	20-40
Unhealthy for Sensitive Groups	40-60
Unhealthy	60-80
Very Unhealthy	80-100
Hazardous	>100

Visualization

Seaborn was used to create visualizations for:

1. The distribution of air quality levels.
2. Relationships between PM2.5, NO2, and Temperature, colored by AQI levels.

4.Code

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
# Read the dataset
df = pd.read_csv('/content/air_quality.csv')

# Display the first few rows
df.head()
# Drop rows with missing values
df = df.dropna()

# Display basic information
print(df.info())
# Extract individual feature arrays
pm25 = df['pm25'].values
no2 = df['no2'].values
temp = df['temperature'].values
```

```python
# Calculate score using a simple weighted formula
score = (pm25 / 5) + (no2 / 10) + (temp / 10)

# Define conditions and AQI levels
conditions = [
    score <= 20,
    (score > 20) & (score <= 40),
    (score > 40) & (score <= 60),
    (score > 60) & (score <= 80),
    (score > 80) & (score <= 100),
    score > 100
]

aqi_levels = [
    'Good',
    'Moderate',
    'Unhealthy for Sensitive Groups',
    'Unhealthy',
    'Very Unhealthy',
    'Hazardous'
]

# Apply AQI classification
df['Predicted_AQI'] = np.select(conditions, aqi_levels, default='Unknown')

# Display the first 10 rows
df[['pm25', 'no2', 'temperature', 'Predicted_AQI']].head(10)
# Count the AQI levels
aqi_counts = df['Predicted_AQI'].value_counts()

# Plot the distribution of AQI levels
plt.figure(figsize=(8, 6))
sns.barplot(x=aqi_counts.index, y=aqi_counts.values)
plt.title("Air Quality Index Distribution", fontsize=16)
plt.xlabel("Air Quality Levels", fontsize=12)
plt.ylabel("Frequency", fontsize=12)
plt.xticks(rotation=45)
plt.show()
```

---

5. Result

	pm25	no2	temperature	Predicted_AQI
0	157.744434	5.376279	31.109108	Moderate
1	101.270316	130.903661	19.298140	Moderate
2	197.204350	17.254966	37.652832	Unhealthy for Sensitive Groups
3	81.580404	91.605322	39.682532	Moderate
4	152.419877	148.007264	12.175063	Unhealthy for Sensitive Groups
5	84.667995	82.805644	18.273000	Moderate
6	167.541202	138.986055	37.425730	Unhealthy for Sensitive Groups
7	118.125479	39.236897	30.255879	Moderate
8	22.067248	115.193534	6.683313	Good
9	16.996155	82.033534	32.353007	Good

---

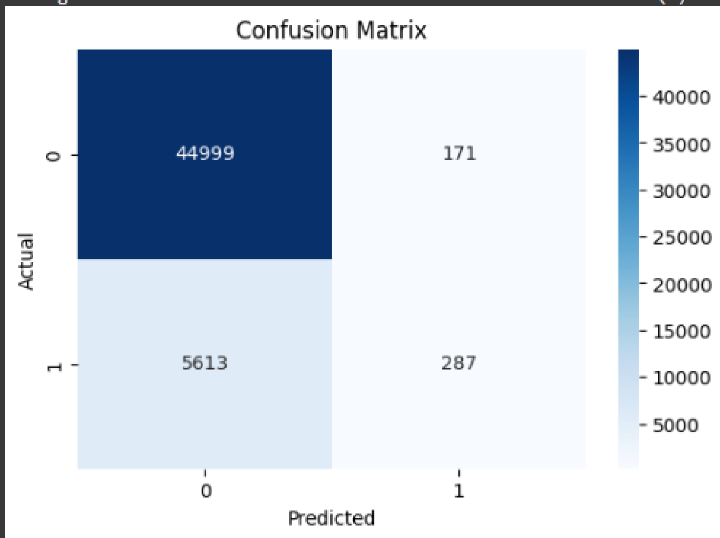
10. References

- numpy documentation
- pandas documentation
- Seaborn visualization library

Choose Files 1. Predict L... Default.csv

- **1. Predict Loan Default.csv**(text/csv) - 24834870 bytes, last modified: 4/18/2025 - 100% done

```
Saving 1. Predict Loan Default.csv to 1. Predict Loan Default (1).csv
```




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

           0       0.89      1.00      0.94     45170
           1       0.63      0.05      0.09      5900

    accuracy                           0.89     51070
   macro avg       0.76      0.52      0.51     51070
weighted avg       0.86      0.89      0.84     51070

✅ Accuracy: 0.89
✅ Precision: 0.63
✅ Recall: 0.05
```

```python
# 📌 Step 1: Import necessary libraries


from google.colab import files
uploaded = files.upload()

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report, accu

# ✅ Load the uploaded file (use the exact file name)
df = pd.read_csv('1. Predict Loan Default.csv')

# Drop 'LoanID' column (if exists)
if 'LoanID' in df.columns:
    df = df.drop(columns=['LoanID'])

# Drop missing values
df = df.dropna()

# Encode categorical columns
label_encoders = {}
for col in df.select_dtypes(include='object').columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

# Split features and target
X = df.drop('Default', axis=1)
y = df['Default']
```

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

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Train model
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

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

# Evaluation Metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print("Classification Report:\n", classification_report(y_test, y_pred))
print(f"✅ Accuracy: {accuracy:.2f}")
print(f"✅ Precision: {precision:.2f}")
print(f"✅ Recall: {recall:.2f}")
```