# Air Quality Prediction using Sample Dataset

# Step 1: Import Required 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 xgboost import XGBClassifier

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

# Step 2: Load the Sample Dataset

df = pd.read\_csv("/content/drive/MyDrive/Colab Notebooks/large\_air\_quality\_dataset-1.csv")

# Step 3: Separate Features and Target

X = df.drop('Air\_Quality\_Level', axis=1)

y = df['Air\_Quality\_Level']

# Step 4: Encode Target Labels

le = LabelEncoder()

y\_encoded = le.fit\_transform(y)

# Step 5: Normalize the Features

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Step 6: Train-Test Split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y\_encoded, test\_size=0.3, random\_state=42)

# Step 7: Train Models

# Random Forest

rf\_model = RandomForestClassifier(n\_estimators=100, random\_state=42)

rf\_model.fit(X\_train, y\_train)

rf\_pred = rf\_model.predict(X\_test)

# XGBoost

xgb\_model = XGBClassifier(use\_label\_encoder=False, eval\_metric='mlogloss')

xgb\_model.fit(X\_train, y\_train)

xgb\_pred = xgb\_model.predict(X\_test)

# Step 8: Evaluate Models

print("Random Forest Accuracy:", accuracy\_score(y\_test, rf\_pred))

print("Random Forest Classification Report:\n", classification\_report(y\_test, rf\_pred, target\_names=le.classes\_))

print("\nXGBoost Accuracy:", accuracy\_score(y\_test, xgb\_pred))

print("XGBoost Classification Report:\n", classification\_report(y\_test, xgb\_pred, target\_names=le.classes\_))

# Step 9: Visualize Confusion Matrix - Random Forest

plt.figure(figsize=(8, 6))

sns.heatmap(confusion\_matrix(y\_test, rf\_pred), annot=True, fmt='d',

            xticklabels=le.classes\_, yticklabels=le.classes\_, cmap='Blues')

plt.title("Confusion Matrix - Random Forest")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()

# Step 10: Visualize Feature Importance - Random Forest

importances = rf\_model.feature\_importances\_

plt.figure(figsize=(10, 6))

sns.barplot(x=importances, y=X.columns)

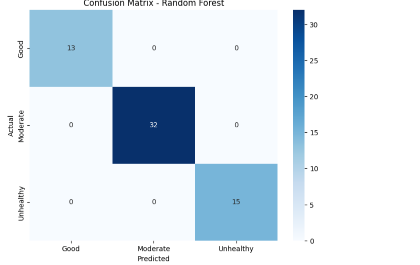
plt.title("Feature Importance - Random Forest")

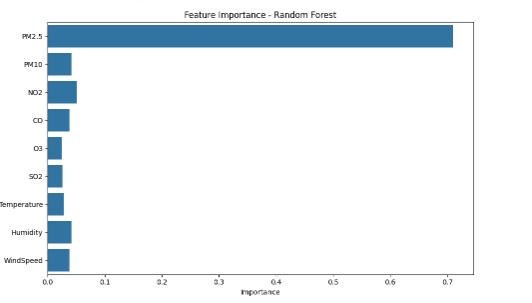
plt.xlabel("Importance")

plt.ylabel("Feature")

plt.tight\_layout()

plt.show()





!pip install Gradio

# Air Quality Prediction and Gradio Deployment (All-in-One)

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import gradio as gr

import joblib

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

# Step 1: Load the Dataset

df = pd.read\_csv("/content/drive/MyDrive/Colab Notebooks/large\_air\_quality\_dataset-1.csv")  # Ensure the CSV is in the same directory

# Step 2: Prepare Features and Target

X = df.drop('Air\_Quality\_Level', axis=1)

y = df['Air\_Quality\_Level']

# Step 3: Encode target labels

le = LabelEncoder()

y\_encoded = le.fit\_transform(y)

# Step 4: Normalize features

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Step 5: Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y\_encoded, test\_size=0.3, random\_state=42)

# Step 6: Train Random Forest Model

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

# Step 7: Evaluation

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print("Classification Report:\n", classification\_report(y\_test, y\_pred, target\_names=le.classes\_))

# Step 8: Save Model Artifacts

joblib.dump(model, "rf\_model.pkl")

joblib.dump(scaler, "scaler.pkl")

joblib.dump(le, "label\_encoder.pkl")

# Step 9: Gradio Interface for Deployment

def predict\_air\_quality(pm25, pm10, no2, co, o3, so2, temp, humidity, windspeed):

    model = joblib.load("rf\_model.pkl")

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

    le = joblib.load("label\_encoder.pkl")

    features = np.array([[pm25, pm10, no2, co, o3, so2, temp, humidity, windspeed]])

    scaled = scaler.transform(features)

    prediction = model.predict(scaled)

    label = le.inverse\_transform(prediction)[0]

    return label

# Define Gradio UI

inputs = [

    gr.Number(label="PM2.5"),

    gr.Number(label="PM10"),

    gr.Number(label="NO2"),

    gr.Number(label="CO"),

    gr.Number(label="O3"),

    gr.Number(label="SO2"),

    gr.Number(label="Temperature (°C)"),

    gr.Number(label="Humidity (%)"),

    gr.Number(label="Wind Speed (m/s)")

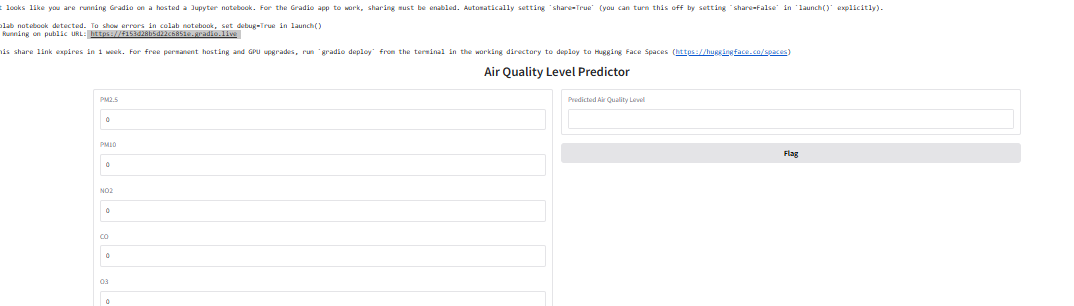
]

output = gr.Textbox(label="Predicted Air Quality Level")

# Launch Gradio App

app = gr.Interface(fn=predict\_air\_quality, inputs=inputs, outputs=output, title="Air Quality Level Predictor")

app.launch()

0