**Step-by-Step Guide**

**1. Install Necessary Libraries**

First, ensure that you have the necessary libraries installed. You can install them using pip:

bash

pip install pandas numpy scikit-learn matplotlib seaborn

**2. Python Code for Predicting Air Quality Levels**

Here's the Python code to load the data, preprocess it, and train a predictive model (using Random Forest Regressor).

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 RandomForestRegressor

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

# Load the dataset

url = 'https://www.kaggle.com/datasets/niharika41298/air-quality-prediction/download'

data = pd.read\_csv(url)

# Display first few rows of the dataset

print(data.head())

# Check for missing values

print(data.isnull().sum())

# Fill missing values (for simplicity, using the mean of each column)

data.fillna(data.mean(), inplace=True)

# Features and target variable

features = ['PM10', 'NO2', 'CO', 'O3', 'temperature', 'humidity', 'windspeed']

target = 'PM2.5'

# Split the data into features (X) and target (y)

X = data[features]

y = data[target]

# Split the data into training and testing 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)

# Initialize the Random Forest Regressor model

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

# Train the model on the training data

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

# Make predictions on the test data

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

# Evaluate the model's performance

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = np.sqrt(mse)

r2 = r2\_score(y\_test, y\_pred)

# Print evaluation metrics

print(f"Mean Absolute Error (MAE): {mae}")

print(f"Mean Squared Error (MSE): {mse}")

print(f"Root Mean Squared Error (RMSE): {rmse}")

print(f"R2 Score: {r2}")

# Plotting true vs predicted PM2.5 levels

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

plt.scatter(y\_test, y\_pred, color='blue')

plt.plot([0, 100], [0, 100], color='red', linestyle='--') # Line of perfect prediction

plt.title('True vs Predicted PM2.5 Levels')

plt.xlabel('True PM2.5')

plt.ylabel('Predicted PM2.5')

plt.show()

**Explanation of the Code:**

1. **Data Loading and Preprocessing:**
   * Load the dataset using pd.read\_csv(). In this case, you’ll replace the url variable with the actual dataset's path from Kaggle or any local dataset you have.
   * Check for missing values using data.isnull().sum() and fill them with the mean values of each column.
2. **Feature Selection:**
   * Select relevant features (like PM10, NO2, CO, temperature, humidity, etc.) that might affect the PM2.5 levels.
3. **Train-Test Split:**
   * Split the data into 80% training data and 20% testing data using train\_test\_split().
4. **Random Forest Model:**
   * Initialize a RandomForestRegressor model and fit it on the training data.
5. **Model Evaluation:**
   * Make predictions on the test set and evaluate the model using **Mean Absolute Error (MAE)**, **Mean Squared Error (MSE)**, **Root Mean Squared Error (RMSE)**, and **R2 Score**.
6. **Visualization:**
   * Plot a scatter plot to compare true vs. predicted values of PM2.5.

**Example Output (After Running the Code):**

The output of the program will include:

1. **Evaluation Metrics:**  
   Example output:

Mean Absolute Error (MAE): 2.134

Mean Squared Error (MSE): 9.234

Root Mean Squared Error (RMSE): 3.037

R2 Score: 0.87

* + These metrics provide insight into how well the model predicts the air quality levels:
    - **MAE** measures the average magnitude of errors in the predictions.
    - **MSE** and **RMSE** give you the average squared error and its square root, respectively.
    - **R2 Score** indicates how well the model explains the variance in the target variable. A score of 1 means perfect predictions.

1. **True vs Predicted Plot:**  
   A scatter plot showing the true PM2.5 values against the predicted values. Ideally, the points will lie along the diagonal line, which indicates that the model's predictions are close to the true values.

**Conclusion:**

This program demonstrates how to predict air quality levels (specifically PM2.5) using a Random Forest model. You can replace the dataset with any dataset containing air quality data. The evaluation metrics and visualization will give you a clear understanding of the model's performance.

For further improvement:

* You can try other algorithms like **XGBoost** or **LSTM** (for time series).
* Experiment with **feature engineering** to add more predictive features.
* Fine-tune the hyperparameters using techniques like **Grid Search**.

Let me know if you'd like more details or assistance with any part of the code!