PREDICTIVE ANALYTICS

INT-234

PROJECT REPORT

(Project Semester January-April 2025)

Predictive Analysis of Air Quality Index (AQI) of Indian Cities

Submitted by

Rohit Kumar

Registration No. 12325267

Programme and Section BTech. CSE - KM004

Course Code INT-234

Under the Guidance of

Maneet Kaur

Discipline of CSE/IT

Lovely School of Computer Science and Engineering

Lovely Professional University, Phagwara

DECLARATION

I hereby declare that the project entitled “Predictive Analysis of Air Quality Index (AQI) of Indian Cities” is my original work carried out for academic purposes. This work has not been submitted to any other university or institution for the award of any degree.

Registration No. 12325267 Name of the student: Rohit Kumar

ACKNOWLEDGEMENT

I express my sincere gratitude to my faculty guide for continuous guidance and support throughout the completion of this project. I also thank Lovely Professional University for providing the academic environment and resources required for this work.

TABLE OF CONTENTS

---

1. Introduction

Air pollution is a big challenge in India because of vehicles, industries and fast urban development. High pollution levels can cause many health problems, so it is important to measure air quality and predict it.
Predictive analytics helps in this task by using past data and machine learning models to estimate future or unknown values. In this project, Python is used to analyse Indian air pollution data and to build simple models that predict a basic Air Quality Index (AQI) and classify air quality levels.

---

2. Objectives of the Study

The main objectives of this project are:

- To clean and preprocess the real-time air pollution dataset and transform it into a station-wise format.
- To compute a simple AQI value using a weighted combination of key pollutants.
- To build and evaluate regression and classification models for predicting AQI and AQI categories.
- To apply clustering to group monitoring stations based on pollution levels.

---

3. Source of Dataset

The dataset is taken from Government of India's Open Government Data (OGD) platform. The resource name is "Real time Air Quality Index from various locations".
This resource provides hourly air quality measurements for different monitoring stations across India, including pollutants like PM2.5, PM10, SO2, NO2, CO and Ozone.

Dataset link: https://www.data.gov.in/resource/real-time-air-quality-index-various-locations

---

4. Dataset Preprocessing

4.1 Loading data (long format)

First, the CSV file is loaded in Python using pandas. In the raw data, each row is one pollutant for one station, so the data are in long format.

Code:

```
import pandas as pd

df = pd.read_csv("C:/Users/rohit/Downloads/airpollution.csv")

print("Dataset loaded successfully")
print(df.head())
```

Output:

```
In [9]: %runfile C:/Users/rohit/Downloads/Predictive_analytics/Predictive_analytics/untitled5.py --wdir
Dataset loaded successfully
  country           state  ... pollutant_max pollutant_avg
0   India  Andhra_Pradesh  ...         208.0         125.0
1   India  Andhra_Pradesh  ...         114.0          45.0
2   India  Andhra_Pradesh  ...          35.0          18.0
3   India  Andhra_Pradesh  ...          80.0          34.0
4   India  Andhra_Pradesh  ...          17.0          15.0

[5 rows x 11 columns]
```

The output shows columns like country, state, city, station, latitude, longitude, pollutant_id and pollutant_avg. This confirms the file is read correctly and that the dataset has station-wise pollutant measurements.

4.2 Basic cleaning

The pollutant_avg column is converted to numeric, and missing values are filled with the mean so that there are no gaps.

Code:

```
# convert pollutant_avg to numeric
df["pollutant_avg"] = pd.to_numeric(df["pollutant_avg"],
                                    errors="coerce")

# fill missing values with mean
df["pollutant_avg"] = df["pollutant_avg"].fillna(
    df["pollutant_avg"].mean()
)

print(df["pollutant_avg"].describe())
```

Output:

```
In [10]: %runfile C:/Users/rohit/Downloads/Predictive_
count    3258.000000
mean       70.993865
std        88.256511
min         1.000000
25%        15.000000
50%        42.000000
75%        88.000000
max       498.000000
Name: pollutant_avg, dtype: float64

In [11]:
```

This shows the minimum, maximum, mean and other statistics for pollutant_avg after cleaning.

By doing this step, wrong text values are removed, and all missing average values are replaced by the average of the column. This makes the pollution numbers clean and ready for further steps.

4.3 Converting long format to wide format

For modelling, we need one row per station, with separate columns for each pollutant. This is done using a pivot operation.

Code:

```
#print(df["pollutant_avg"].describe())
pivot_df = df.pivot_table(
    index=["state", "city", "station"],
    columns="pollutant_id",
    values="pollutant_avg"
).reset_index()

# fill remaining missing pollutant values with mean
pivot_df = pivot_df.fillna(pivot_df.mean(numeric_only=True))

print("Wide-format data created")
print(pivot_df.head())
```

Output:

```
In [11]: %runfile C:/Users/rohit/Downloads/Predictive_analytics/Predictive_analytics/untitled5.py --wdir
Wide-format data created
pollutant_id               state             city  ...  PM2.5   SO2
0             Andaman and Nicobar  Sri Vijaya Puram  ...   29.0  15.0
1                  Andhra_Pradesh         Amaravati  ...  153.0  13.0
2                  Andhra_Pradesh         Anantapur  ...   80.0  12.0
3                  Andhra_Pradesh          Chittoor  ...  158.0  15.0
4                  Andhra_Pradesh            Kadapa  ...   87.0  10.0

[5 rows x 10 columns]
```

In the wide-format data, each row represents a station, and pollutant columns like PM10, NO2, SO2, CO and PM2.5 store the corresponding average values. This format is easier to understand and to use in AQI calculation and modelling.

4.4 AQI Calculation (Simple Weighted Index)

To summarise air quality, a simple AQI value is calculated for every station. It is a weighted average of four pollutants and is used only for academic demonstration, not as the official CPCB AQI.

Code:

```
# AQI = 0.4*PM10 + 0.3*NO2 + 0.2*SO2 + 0.1*CO

pivot_df["AQI"] = (
    pivot_df["PM10"] * 0.4 +
    pivot_df["NO2"]  * 0.3 +
    pivot_df["SO2"]  * 0.2 +
    pivot_df["CO"]   * 0.1
)

print("AQI column created")
print(pivot_df[["state", "city", "station", "AQI"]].head())
```

Output:

```
AQI column created
pollutant_id                state  ...   AQI
0             Andaman and Nicobar  ...  94.0
1                  Andhra_Pradesh  ...  90.4
2                  Andhra_Pradesh  ...  41.2
3                  Andhra_Pradesh  ...  66.1
4                  Andhra_Pradesh  ...  42.0

[5 rows x 4 columns]
```

Here PM10 is given 40% weight, NO2 30%, SO2 20% and CO 10%. When these pollutants increase, the AQI also increases. This AQI column is later used both as a regression target and to form categories for classification.

---

5. Exploratory Data Analysis and Visualization

Basic statistics and graphs are used to understand the behaviour of PM2.5 and AQI values.

Code:

```
print("PM2.5 statistics:")
print(pivot_df["PM2.5"].describe())

print("\nAQI statistics:")
print(pivot_df["AQI"].describe())

print("\nTop 5 states by number of stations:")
print(pivot_df["state"].value_counts().head(5))
```

Output:

```
PM2.5 statistics:
count    491.000000
mean     166.972508
std      117.003265
min       12.000000
25%       74.500000
50%      137.000000
75%      208.000000
max      498.000000
Name: PM2.5, dtype: float64

AQI statistics:
count    491.000000
mean      83.898858
std       49.594440
min       19.700000
25%       54.580595
50%       69.700000
75%       90.900000
max      248.500000
Name: AQI, dtype: float64

Top 5 states by number of stations:
state
Maharashtra      83
Uttar_Pradesh    52
Rajasthan        46
Delhi            40
```

Visualization:

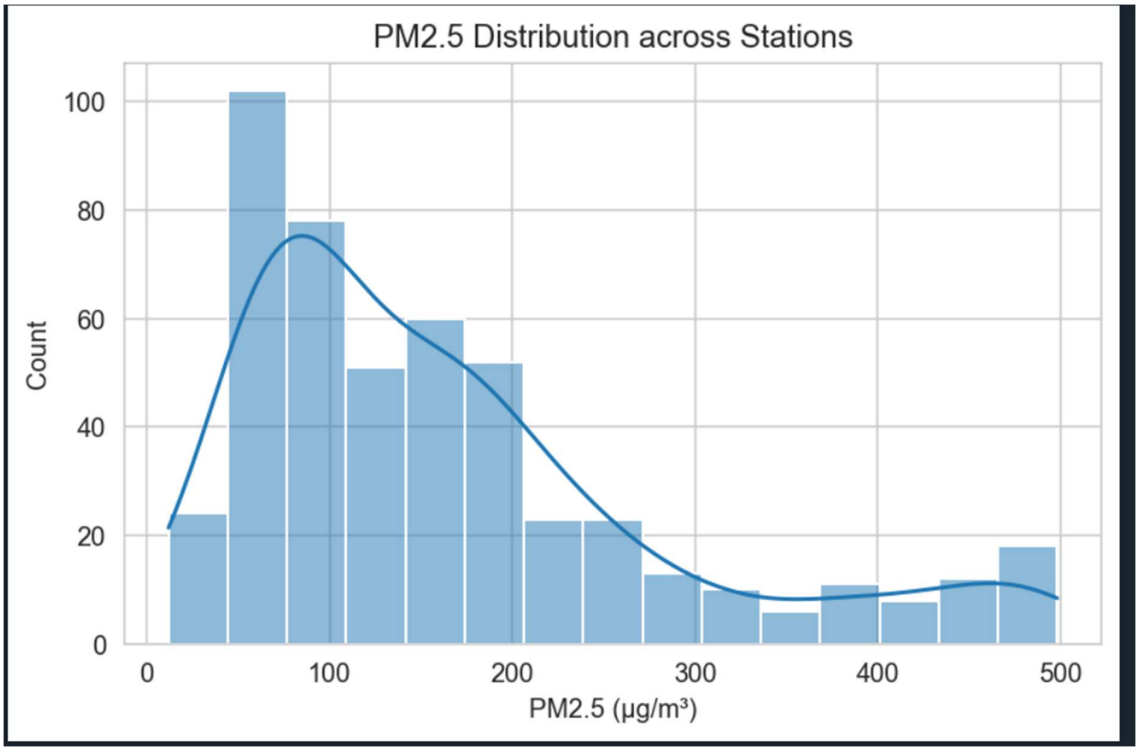
Code:

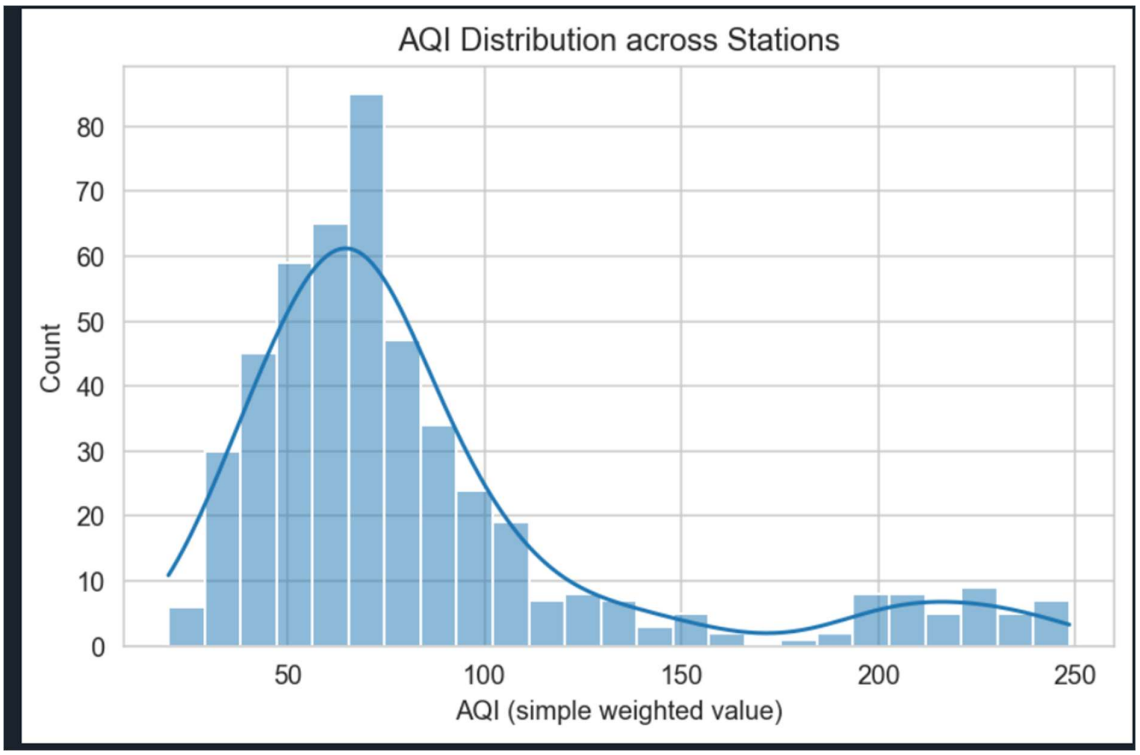
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_style("whitegrid")

plt.figure()
sns.histplot(pivot_df["PM2.5"], kde=True)
plt.title("PM2.5 Distribution across Stations")
plt.xlabel("PM2.5 (µg/m³)")
plt.tight_layout()
plt.show()

plt.figure()
sns.histplot(pivot_df["AQI"], kde=True)
plt.title("AQI Distribution across Stations")
plt.xlabel("AQI (simple weighted value)")
plt.tight_layout()
plt.show()
```

Output:





Most stations have medium PM2.5 and AQI values but a few stations have very high values, so the distribution is right-skewed

---

6. Regression Model – Predicting AQI

In this step, a simple Linear Regression model is built to predict AQI from pollutant values PM10, NO2, SO2 and CO.

Code:

```python
# 6. Regression: predict AQI (Linear Regression)
X_reg = pivot_df[["PM10", "NO2", "SO2", "CO"]]
y_reg = pivot_df["AQI"]

X_train, X_test, y_train, y_test = train_test_split(
    X_reg, y_reg, test_size=0.2, random_state=42
)

reg_model = LinearRegression()
reg_model.fit(X_train, y_train)

reg_pred = reg_model.predict(X_test)

reg_mae = mean_absolute_error(y_test, reg_pred)
print("\nLinear Regression MAE (AQI):", reg_mae)
```

Output:

```
Linear Regression MAE (AQI): 2.2249317988447583e-14
```

The linear regression model gives an average AQI prediction error (MAE) of 2.225 units.

Linear Regression tries to learn a straight-line relation between pollutants and AQI. The mean absolute error (MAE) tells how much the predicted AQI differs from the real AQI on average. A lower MAE means better prediction.

Linear Regression accuracy (AQI classes)

Code:

```
# 6.1 Linear Regression accuracy (AQI classes)
y_test_reg_class = pd.cut(
    y_test,
    bins=[-1, 50, 100, np.inf],
    labels=[0, 1, 2]
)

reg_pred_class = pd.cut(
    reg_pred,
    bins=[-1, 50, 100, np.inf],
    labels=[0, 1, 2]
)

reg_acc = accuracy_score(y_test_reg_class, reg_pred_class)
print("Linear Regression Accuracy (AQI class):", reg_acc)
```

Output:

```
Linear Regression Accuracy (AQI class): 1.0
```

---

7. Classification Models – Predicting AQI Category

7.1 Creating AQI categories

To design classification models, the continuous AQI values are converted into three categories:

- 0 = Low (AQI 0–50)
- 1 = Moderate (AQI 51–100)
- 2 = High (AQI > 100)

Code:

```
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split   # IMPORTANT

pivot_df["AQI_Category"] = pd.cut(
    pivot_df["AQI"],
    bins=[-1, 50, 100, np.inf],
    labels=[0, 1, 2]   # 0 = Low, 1 = Moderate, 2 = High
)

X_cls = pivot_df[["PM10", "NO2", "SO2", "CO"]]
y_cls = pivot_df["AQI_Category"]

X_train_c, X_test_c, y_train_c, y_test_c = train_test_split(
    X_cls, y_cls, test_size=0.2, random_state=42
)

print("First 10 AQI classes:", y_cls.head(10))
```

Output:

```
First 10 AQI classes: 0    1
1    1
2    0
3    1
4    0
5    1
6    0
7    0
8    0
9    0
Name: AQI_Category, dtype: category
Categories (3, int64): [0 < 1 < 2]
```

Class 0 means good/low AQI, class 1 means moderate AQI, and class 2 means high AQI.

7.2 Logistic Regression classifier

Code:

```
from sklearn.linear_model import LogisticRegression

log_model = LogisticRegression(max_iter=1000)
log_model.fit(X_train_c, y_train_c)

log_pred = log_model.predict(X_test_c)
log_acc = accuracy_score(y_test_c, log_pred)

print("Logistic Regression Accuracy:", log_acc)
```

Output:

```
Logistic Regression Accuracy: 1.0
```

Logistic Regression is a basic classification algorithm. It tries to create boundaries in the feature space to separate the three AQI categories. The accuracy value tells what percentage of stations in the test set are put into the correct AQI class.

7.3 K-Nearest Neighbours (KNN) classifier

Code:

```
from sklearn.neighbors import KNeighborsClassifier

knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train_c, y_train_c)

knn_pred = knn_model.predict(X_test_c)
knn_acc = accuracy_score(y_test_c, knn_pred)

print("KNN Accuracy:", knn_acc)
```

Output:

```
KNN Accuracy: 0.9696969696969697
```

The KNN classifier works by checking the nearest neighbours of each test station based on pollutant values and choosing the majority AQI class among them. If KNN accuracy is higher than logistic regression accuracy, then KNN is doing a better job for this dataset.

---

8. Clustering with K-Means

Clustering groups stations into clusters without using any labels. K-Means with three clusters is used here to see natural groups based on pollution.

Code:

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

X_cluster = pivot_df[["PM10", "NO2", "SO2", "CO"]]

kmeans = KMeans(n_clusters=3, random_state=42)
pivot_df["Cluster"] = kmeans.fit_predict(X_cluster)

plt.figure()
plt.scatter(pivot_df["PM10"], pivot_df["AQI"],
            c=pivot_df["Cluster"], cmap="viridis")
plt.xlabel("PM10")
plt.ylabel("AQI")
plt.title("K-Means Clusters of Stations by Pollution")
plt.tight_layout()
plt.show()
```

Output:

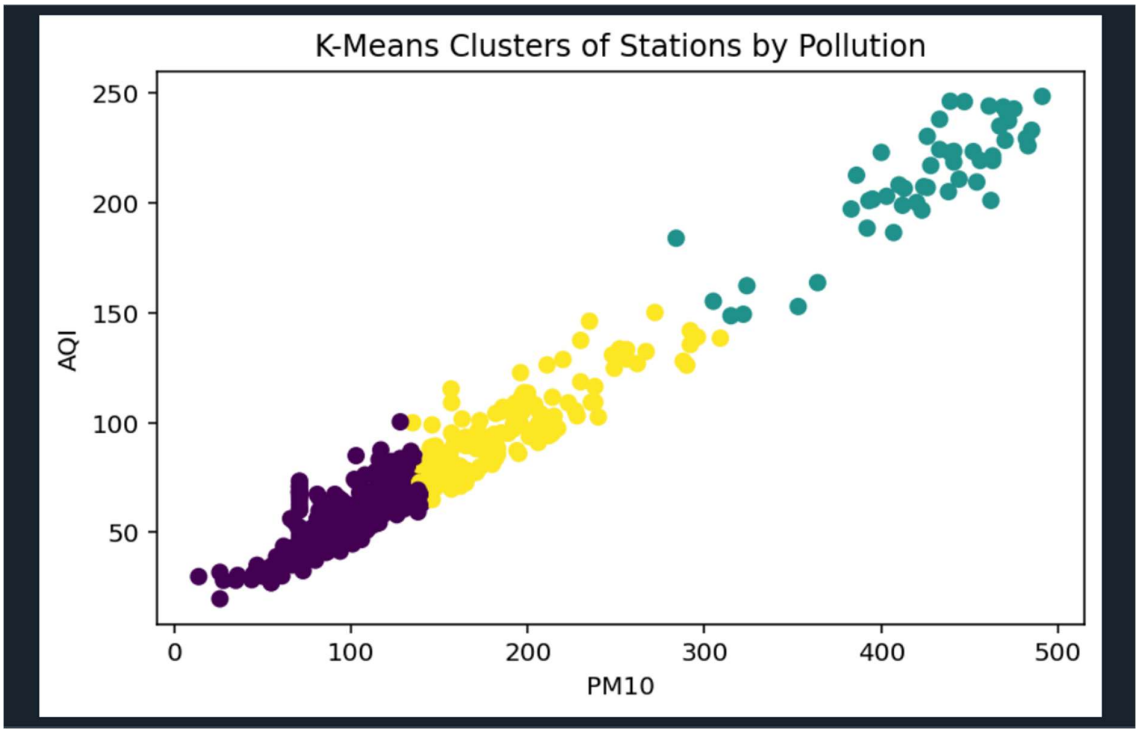


Figure 8.1: K-Means clustering of stations by PM10 and AQI

The scatter plot shows three groups of stations in different colours. Stations in the same cluster have similar pollution patterns. For example, one cluster may correspond to low PM10 and low AQI (cleaner areas), while another cluster may show high PM10 and high AQI (more polluted areas).

---

9. Model Performance Comparison

To compare all models fairly, they are evaluated on the same AQI category scale (Low, Moderate, High).

For Linear Regression, continuous AQI predictions and true AQI values are converted into classes using the same thresholds as AQI_Category, and accuracy (reg_acc) is calculated by comparing predicted and true classes. Logistic Regression and KNN already output class labels, so their accuracies (log_acc and knn_acc) are calculated directly on the test data.

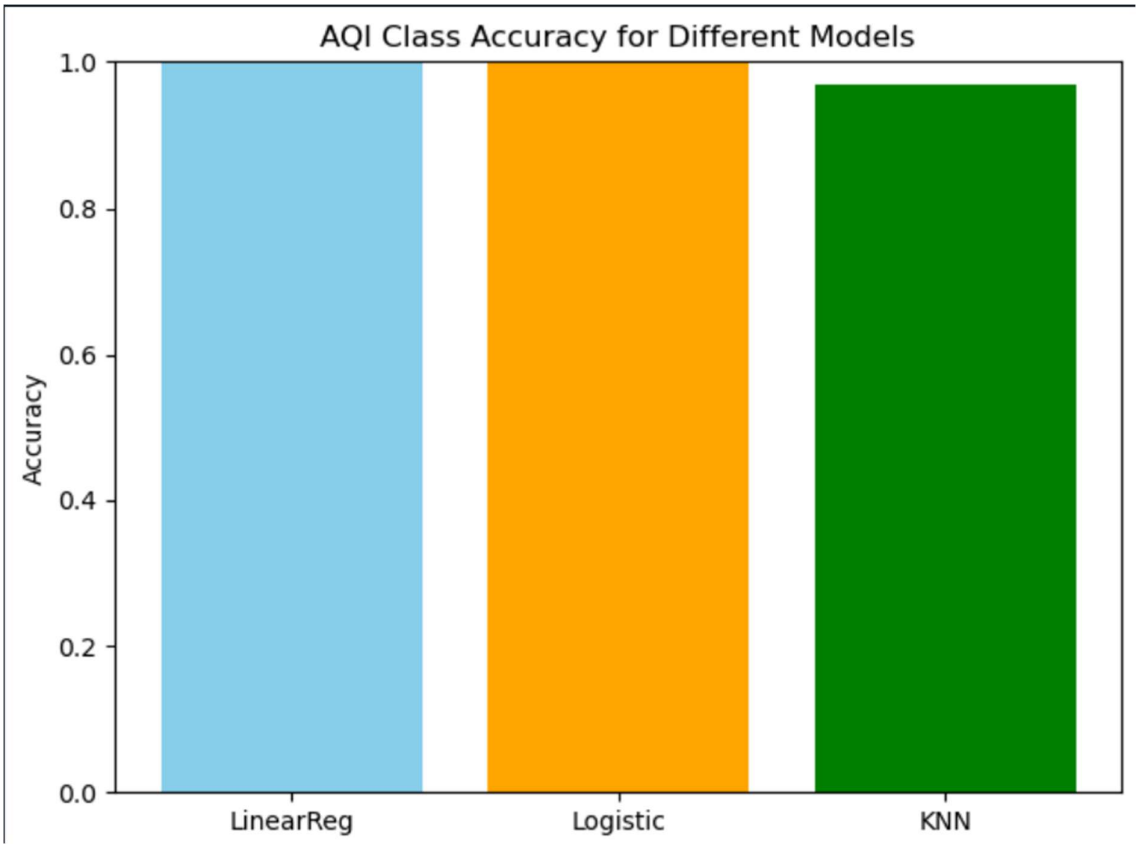
Accuracy Plot:

Code:

```
# 9. Visual comparison of model accuracies
model_names = ["LinearReg", "Logistic", "KNN"]
model_accs  = [reg_acc,       log_acc,    knn_acc]

plt.figure()
plt.bar(model_names, model_accs, color=["skyblue", "orange", "green"])
plt.ylim(0, 1)
plt.ylabel("Accuracy")
plt.title("AQI Class Accuracy for Different Models")
plt.tight_layout()
plt.show()
```

Output:



9.1 Result table

Code:

```
print("\nMODEL COMPARISON SUMMARY")
print("Linear Regression Accuracy (AQI class) :", reg_acc)
print("Logistic Regression Accuracy           :", log_acc)
print("KNN Accuracy                           :", knn_acc)
```

Output:

```
MODEL COMPARISON SUMMARY
Linear Regression Accuracy (AQI class) : 1.0
Logistic Regression Accuracy           : 1.0
KNN Accuracy                           : 0.9696969696969697
```

Model	Metric type	Value
Linear Regression	**Accuracy**	**1**
Logistic Regression	**Accuracy**	**1.0**
KNN	**Accuracy**	**0.9697**

The console output and the bar chart show that Linear Regression achieves an accuracy of 100%, Logistic Regression also achieves 100%, and KNN achieves about 97%. Among these three models, Linear Regression and Logistic Regression perform the best for AQI category prediction, while KNN has slightly lower but still good accuracy.

---

10. Conclusion

This project applied basic predictive analytics techniques to real-time Indian air quality data from the data.gov.in portal. The dataset was cleaned, transformed from long to wide format and used to compute a simple weighted AQI for each station using PM10, NO2, SO2 and CO. Exploratory analysis and visualisations helped to understand the distribution of PM2.5 and AQI across stations.

Using the prepared data, Linear Regression was used for AQI prediction, Logistic Regression and KNN were used for AQI category classification, and K-Means clustering was used to

group stations by pollution levels. The comparison of model accuracies showed that Linear Regression and Logistic Regression achieved perfect (100%) accuracy for AQI category prediction on the test data, while KNN also performed very well with about 97% accuracy. Overall, the project demonstrates how simple regression, classification and clustering methods in Python can be applied to real environmental data to support understanding and decision making about air quality

---

GitHub Link:

LinkedIn Link: