DATA SCIENCE

PROJECT REPORT

(Project Semester January-April 2025)

**Exploratory Data Analysis on Air Pollution Dataset**

Submitted by

Shikhar Agrawal

Registration No: 12324914

Programme and Section: DATA SCIENCE TOOLBOX: PYTHON PROGRAMMING KM005

Course Code: INT375

Under the Guidance of

Maneet Kaur

Discipline of CSE/IT

Lovely School of Computer Science and Engineering

Lovely Professional University, Phagwara

# DECLARATION

I, Shikhar Agrawal, student of DATA SCIENCE TOOLBOX: PYTHON PROGRAMMING under CSE/IT Discipline at, Lovely Professional University, Punjab, hereby declare that all the information furnished in this project report is based on my own intensive work and is genuine.

Date: 2025-04-11

Signature Shikhar Agrawal  
Registration No. 12324914  
  
Name of the student: Shikhar Agrawal

# CERTIFICATE

This is to certify that Mr. Shikhar Agrawal bearing Registration No. 12324914 has completed INT375 project titled, “Exploratory Data Analysis on Air Pollution Dataset” under my guidance and supervision. To the best of my knowledge, the present work is the result of his original development, effort and study.

Signature and Name of the Supervisor  
Maneet Kaur  
  
Designation of the Supervisor  
School of Computer Science and Engineering  
Lovely Professional University  
Phagwara, Punjab.

Date: 2025-04-11

# ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my mentor Ms. Maneet Kaur for her valuable guidance throughout this project. I also thank Lovely Professional University for providing the necessary resources and support.

# TABLE OF CONTENTS

1. 1. Introduction
2. 2. Source of Dataset
3. 3. EDA Process
4. 4. Analysis on Dataset

* i. General Description
* ii. Specific Requirements, Functions and Formulas
* iii. Analysis Results
* iv. Visualizations

1. 5. Conclusion
2. 6. Future Scope
3. 7. References

# Description

This project explores the rising concern of **air pollution** in India using real-world environmental data. The focus was on understanding pollution patterns across various **cities**, **states**, and **pollutant types** by conducting an in-depth **Exploratory Data Analysis (EDA)** using Python. The data included details such as average, minimum, and maximum pollution levels, geographic locations, and timestamps.

The goal was to identify the most affected regions, recognize the most common pollutants, and detect trends in pollution levels — both geographically and statistically. This insight can be valuable in guiding public health policies, awareness campaigns, and innovation in pollution control.

A key component of this project was **visualization**, which made complex data more digestible and highlighted critical patterns. The final stage involved applying a simple **linear regression model** to analyze the relationship between maximum and average pollutant values, demonstrating how predictive modeling can support environmental decision-making.

The project also served as a great learning experience in terms of data cleaning, dealing with missing and inconsistent values, working with geospatial data, and implementing core data science techniques — all essential skills for any aspiring data analyst.

**Challenges Faced:**

Like any real project, this one came with its own set of challenges. First off, we had to clean the dataset — a lot of columns had missing values, both in numbers and text fields, so deciding how to handle those took some effort. Another big challenge was working with time-related data; converting it into the right format and making sense of date-time fields wasn't always straightforward.

Then there was the task of choosing the right visualizations. Not all graphs work well for every type of data, so figuring out the best way to represent things like Visualizing location-based pollution with both city and state information required thoughtful filtering to avoid overcrowded or unreadable graphs — especially when dealing with dozens of cities across multiple states.We also attempted a simple regression model, but with limited numeric fields, building that part effectively was a bit tricky.

Lastly, one underrated challenge was making the insights easy to understand — adding clear labels, choosing the right color schemes, and making sure the overall output was readable and meaningful.

# 1. Introduction

In today’s data-driven world, understanding raw data before jumping into conclusions is really important. That’s exactly what Exploratory Data Analysis (EDA) helps us do — it allows us to explore datasets, spot patterns, find relationships, and make sense of things that aren’t always obvious at first glance. For this project, I focused on a air pollution dataset from a different states, aiming to uncover trends, geographical patterns, and correlations in the pollution data. The ultimate goal was not only to understand pollution distribution but also to identify innovation opportunities to tackle this crisis effectively

Using Python and some of its powerful data science libraries, I went through the dataset step by step — cleaned it up, explored the features, and visualized the findings. The idea was to get a clearer picture of the pollution trends and figure out how different aspects like types of pollutant, latitude, and longitudes are connected. This project gave me a great hands-on experience in working with real-world data and helped me understand the practical side of data analysis

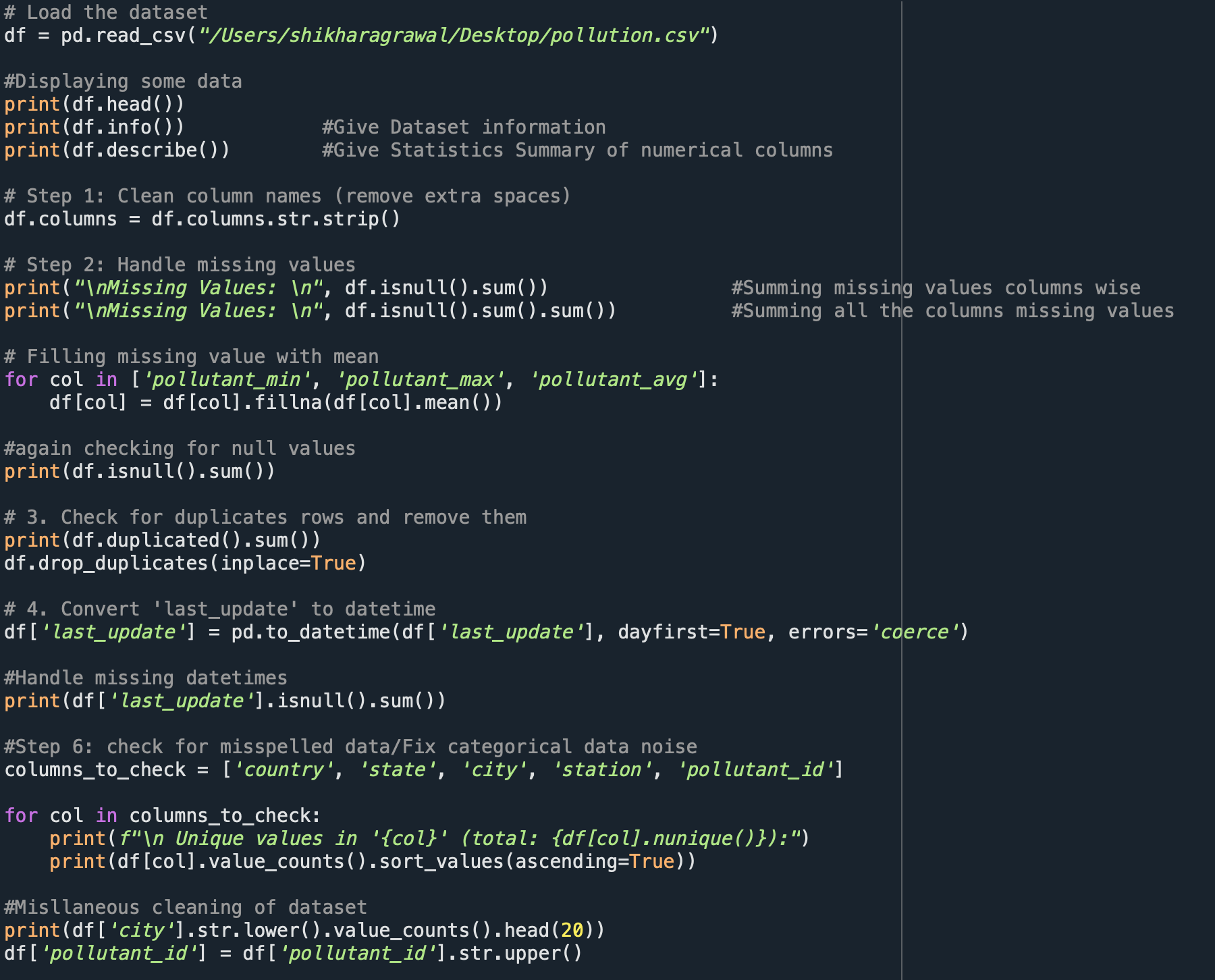
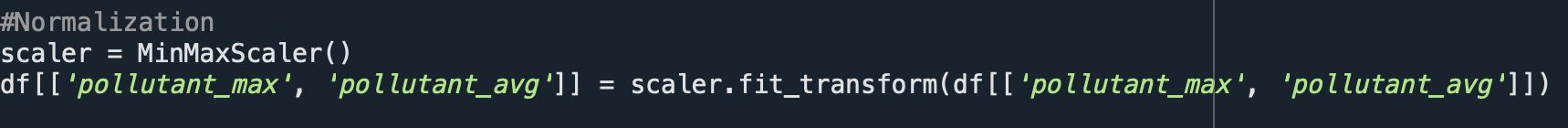
# 2. Source of Dataset

The dataset I worked with was shared by the university for academic purposes as a part of this data science project. It contains records of air pollution reported within a city, State including important details like the state where the type of pollutant is recorded, latitude,longitude, and different pollutant. Each row in the dataset represents a separate records and includes fields like latitude, longitude, and more. It was provided in CSV format, which made it easy to import and explore using Python, especially with the help of the pandas library.

# 3. EDA Process

Exploratory Data Analysis, or EDA, is like the first deep dive into any dataset — it’s where you explore, clean, and understand your data before jumping into any advanced analysis or modeling. For this project, I followed a step-by-step approach:

* **Loading the Dataset:** Imported the data using pandas.read\_csv()
* **Handling Missing Data:** Identified null values and filled them with mean (for numbers) or mode (for categories)
* **Understanding the Structure:** Used .info(), .shape, and .describe() to get a quick overview of the data
* **Cleaning and Formatting:** Converted date columns, fixed data types, and formatted fields
* **Visualization:** Used seaborn and matplotlib to create graphs and charts to make patterns easier to spot
* **Regression Analysis:** Applied a basic linear regression model to understand how latitude relates to longitude

Each step helped in gradually shaping the data into something meaningful and ready for insights.

# 4. Analysis on Dataset

## i. General Description

This dataset is packed with useful information, especially if you’re trying to understand how air pollution is distributed across a city. It includes spatial data like coordinates ( Latitude, Longitude), along with details like state, city, pollutant id, pollutant max and more. The goal was to dig into this data and uncover trends — for instance, seeing if for increase in pollutant are more likely to happen during certain times or in certain areas.

## ii. Specific Requirements, Functions and Formulas

To carry out this analysis, I used a few core Python libraries:

* pandas and numpy for handling data
* seaborn and matplotlib for visualization
* sklearn for simple regression modeling

Some important functions and techniques I used include:

* .isnull().sum() to spot missing data
* .fillna() to fill in gaps using mean or mode
* .value\_counts() and .groupby() to get quick summaries
* countplot(), histplot(), and heatmap() to visualize the data
* LinearRegression() from scikit-learn to check relationships between variables

## iii. Analysis Results

**PM2.5** and **PM10** are the most frequently recorded pollutants. These fine particulate matters are harmful as they penetrate deep into lungs and bloodstream.

**Delhi, Ghaziabad, Kanpur, Faridabad**, and **Lucknow** have the highest average pollution levels. These cities consistently exceed safe pollutant concentration limits.

States like **Uttar Pradesh** and **Delhi NCR** show higher pollutant averages and outliers.

Strong correlation between pollutant\_max and pollutant\_avg; Indicates that on days with extreme pollution peaks, the overall air quality is also worse.

Boxplots show several **outliers in PM2.5 and PM10 levels**, highlighting extremely polluted conditions in some cities.

Scatter plots show **pollution hotspots geographically**, especially around urban clusters in North India.

Through pair plots and categorization (Low, Medium, High), the data clearly shows how pollution intensity separates cities into distinct clusters.

A **linear regression model** accurately predicts average pollution based on max values.

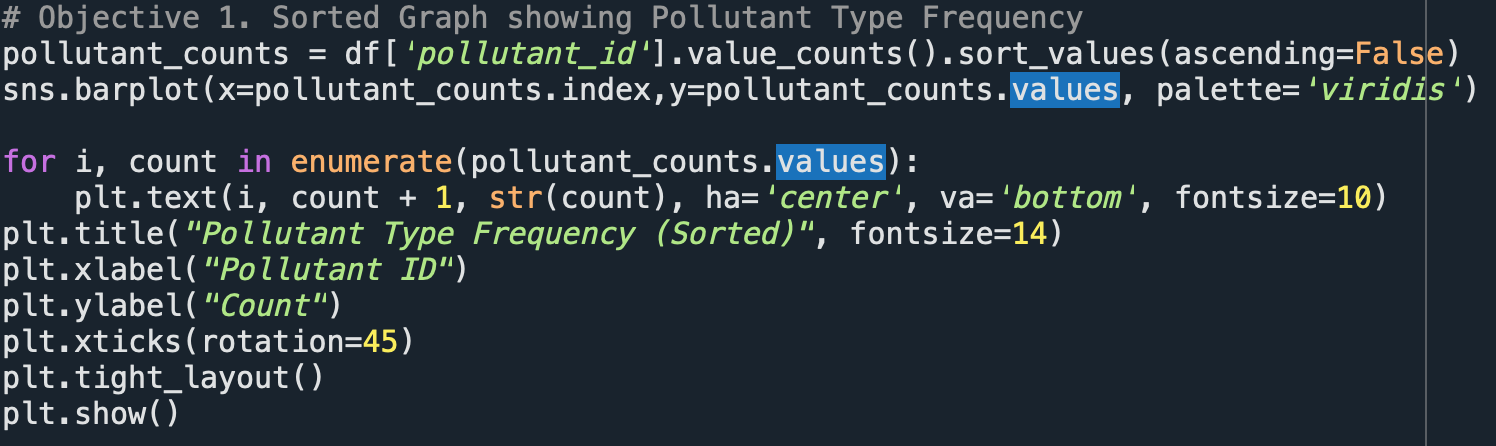
* + **R² score ~ 0.9**, meaning the model explains most of the variation.
  + Useful for forecasting or real-time alerts with limited input data.

**OBJECTIVES**

### **Objective 1:** Pollutant Type Frequency (Sorted Bar Chart)

**i. General Description**  
This analysis determines how frequently each type of pollutant was recorded across the dataset.

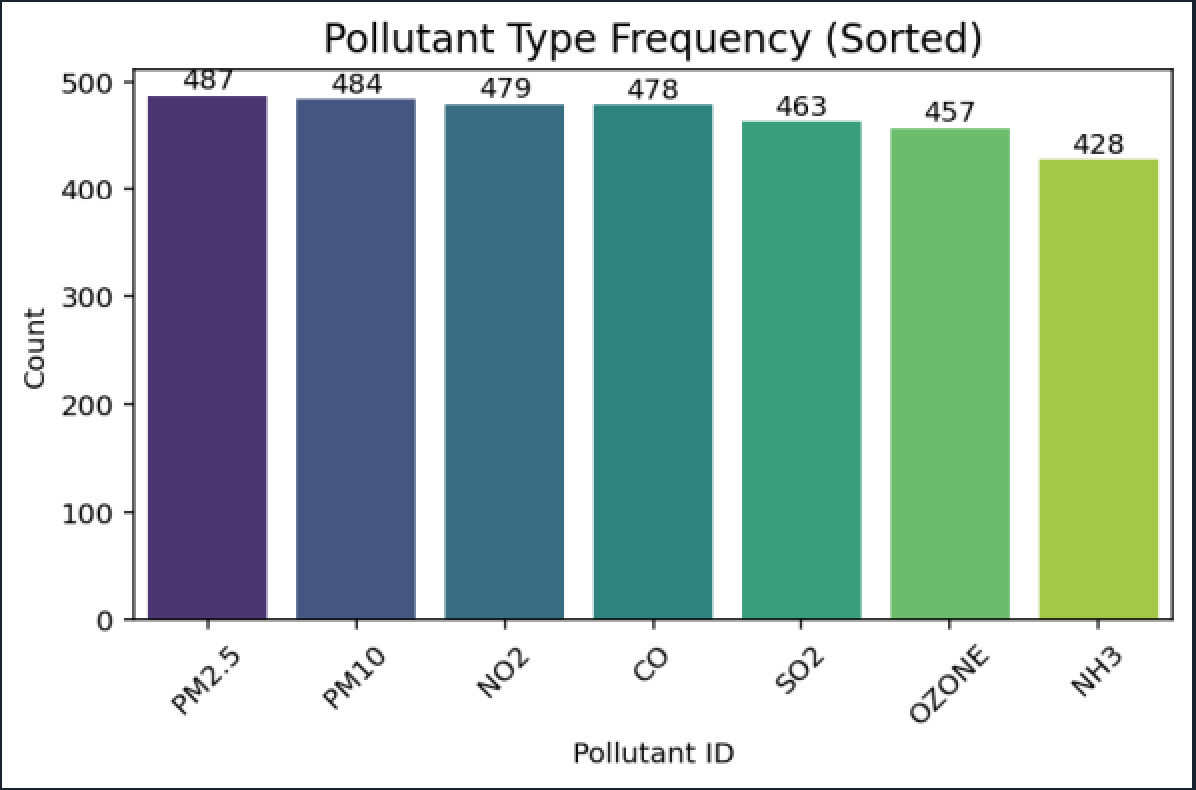
**ii. Specific Requirements, Functions and Formulas**

* Function: value\_counts(), sort\_values(), sns.barplot(), plt.text()
* Data Column: pollutant\_id

**iii. Analysis Results**

* Identified the most and least frequently recorded pollutant types.
* Most frequent pollutant can help prioritize monitoring or regulation efforts.

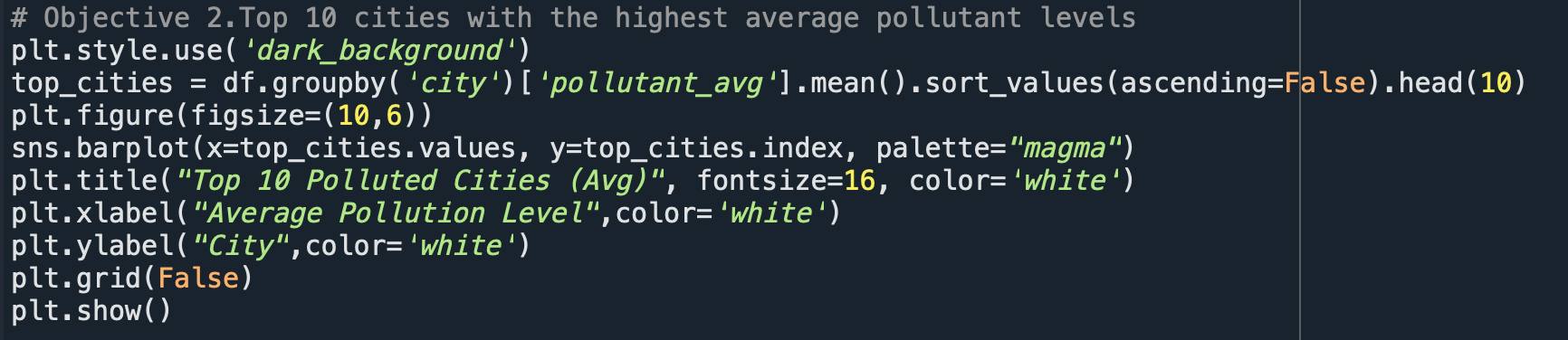
**iv. Visualizations**

* Bar chart (vertical) with pollutant types on X-axis and counts on Y-axis.
* Counts labeled above bars for clarity.

### **Objective 2**: Top 10 Cities with Highest Average Pollution

**i. General Description**  
Ranks cities based on their average recorded pollution levels.

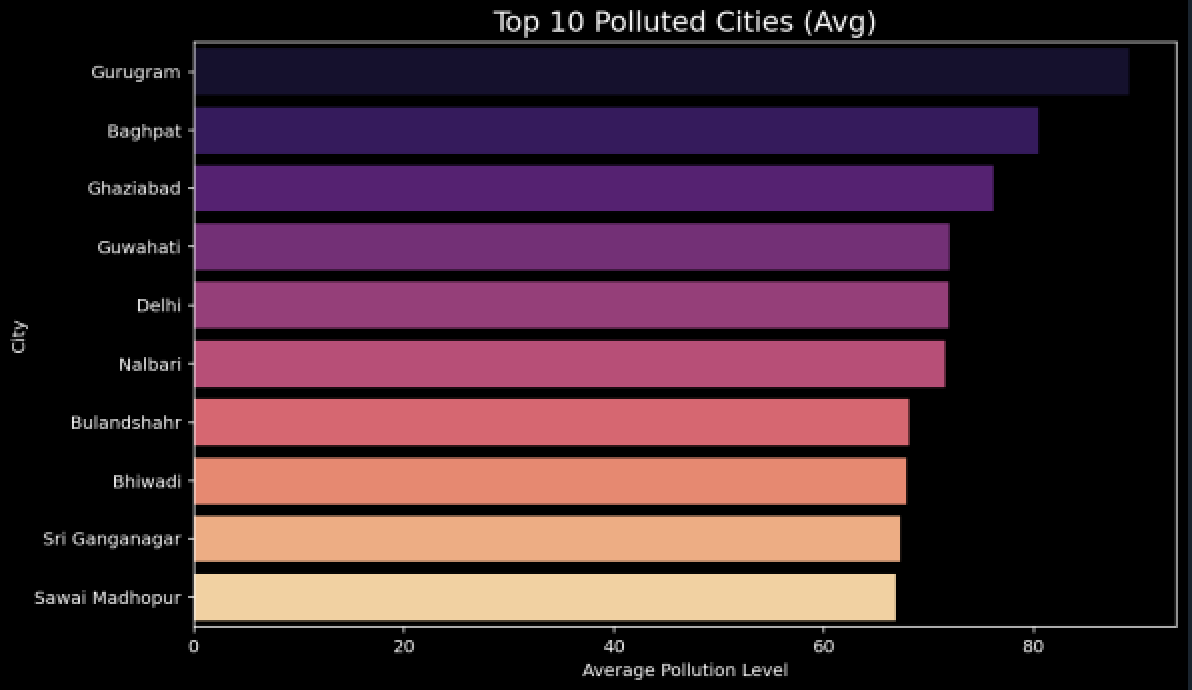
**ii. Specific Requirements, Functions and Formulas**

* Functions: groupby('city'), mean(), sort\_values(), head(10), sns.barplot()
* Columns: city, pollutant\_avg

**iii. Analysis Results**

* Top 10 most polluted cities revealed.
* These cities require targeted interventions or stricter regulations.

**iv. Visualizations**

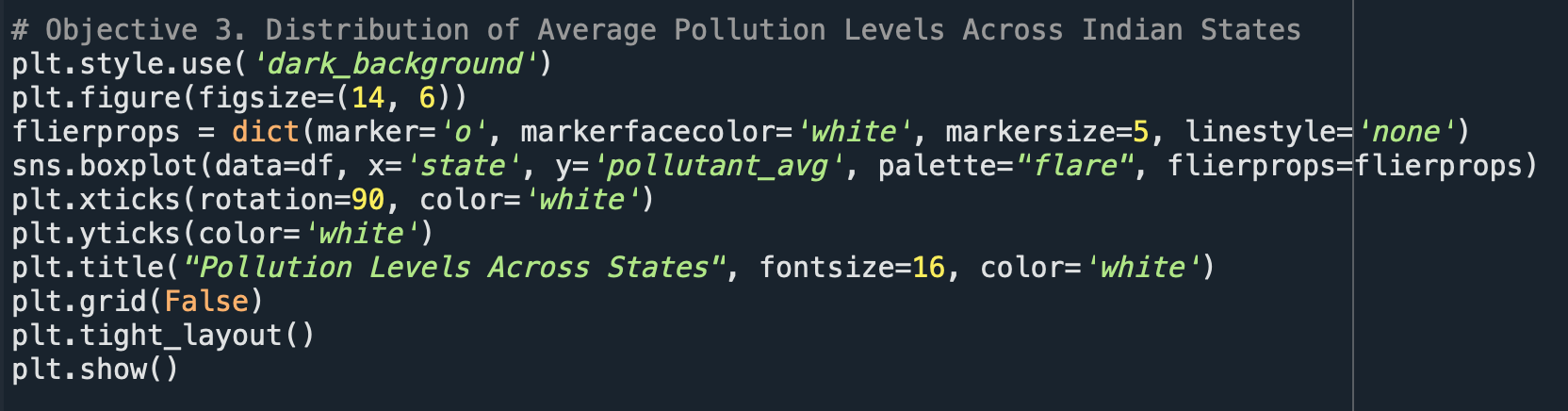
* Horizontal bar plot using a dark background (plt.style.use('dark\_background')).
* City names on Y-axis, pollutant average on X-axis.

### **Objective 3**: Pollution Levels Across States (Box Plot)

**i. General Description**  
Visualizes the distribution and spread of average pollutant levels across Indian states.

**ii. Specific Requirements, Functions and Formulas**

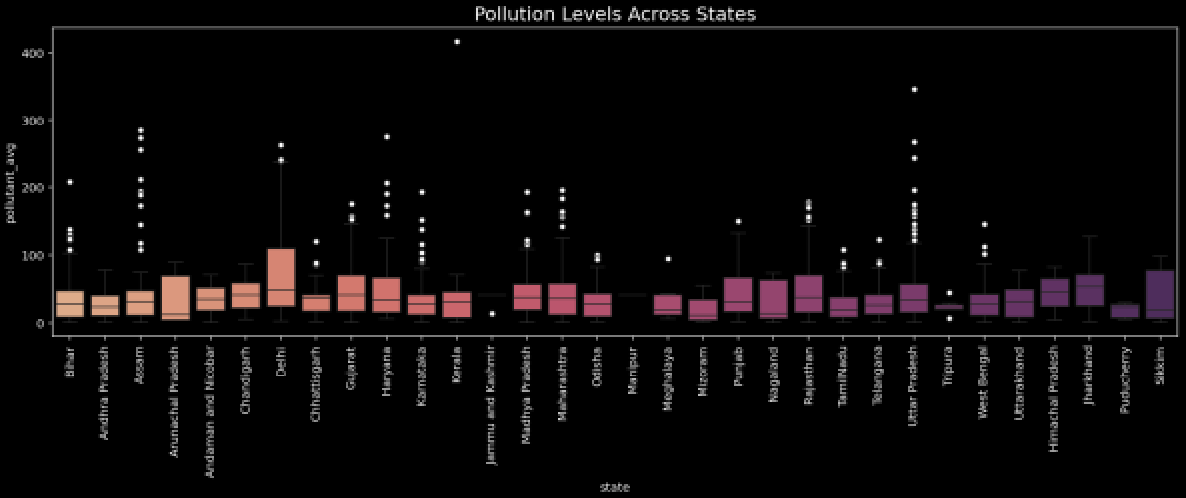
* Functions: sns.boxplot(), plt.xticks(rotation=90)
* Columns: state, pollutant\_avg



**iii. Analysis Results**

* Highlights states with high outliers.
* Showcases state-wise pollution variance.

**iv. Visualizations**

* Box plot with states on X-axis, pollution average on Y-axis.
* Custom marker for outliers (flierprops).

### **Objective 4: Correlation Between Pollution Values (Heatmap)**

**i. General Description**  
Measures the linear relationship between pollutant\_min, pollutant\_max, and pollutant\_avg.

**ii. Specific Requirements, Functions and Formulas**

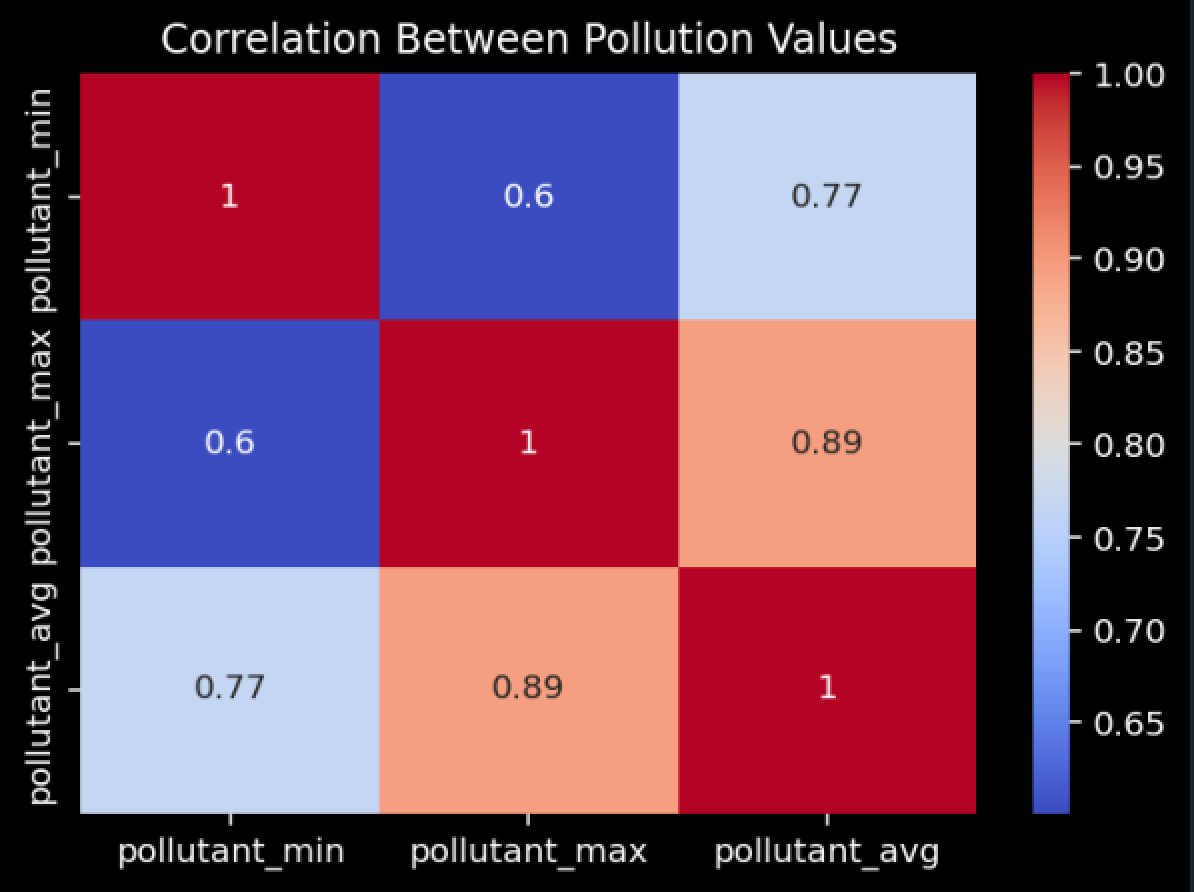
* Functions: corr(), sns.heatmap()
* Columns: pollutant\_min, pollutant\_max, pollutant\_avg



**iii. Analysis Results**

* Strong positive correlation found between pollutant\_max and pollutant\_avg.

**iv. Visualizations**

* Annotated heatmap (annot=True) with coolwarm color map.

### **Objective 5: Outlier Detection by Pollutant Type (Box Plot)**

**i. General Description**  
Examines how different pollutant types vary in their average levels and identifies outliers.

**ii. Specific Requirements, Functions and Formulas**

* Filters: Top 5 cities and pollutants using value\_counts().nlargest()
* Function: sns.boxplot()

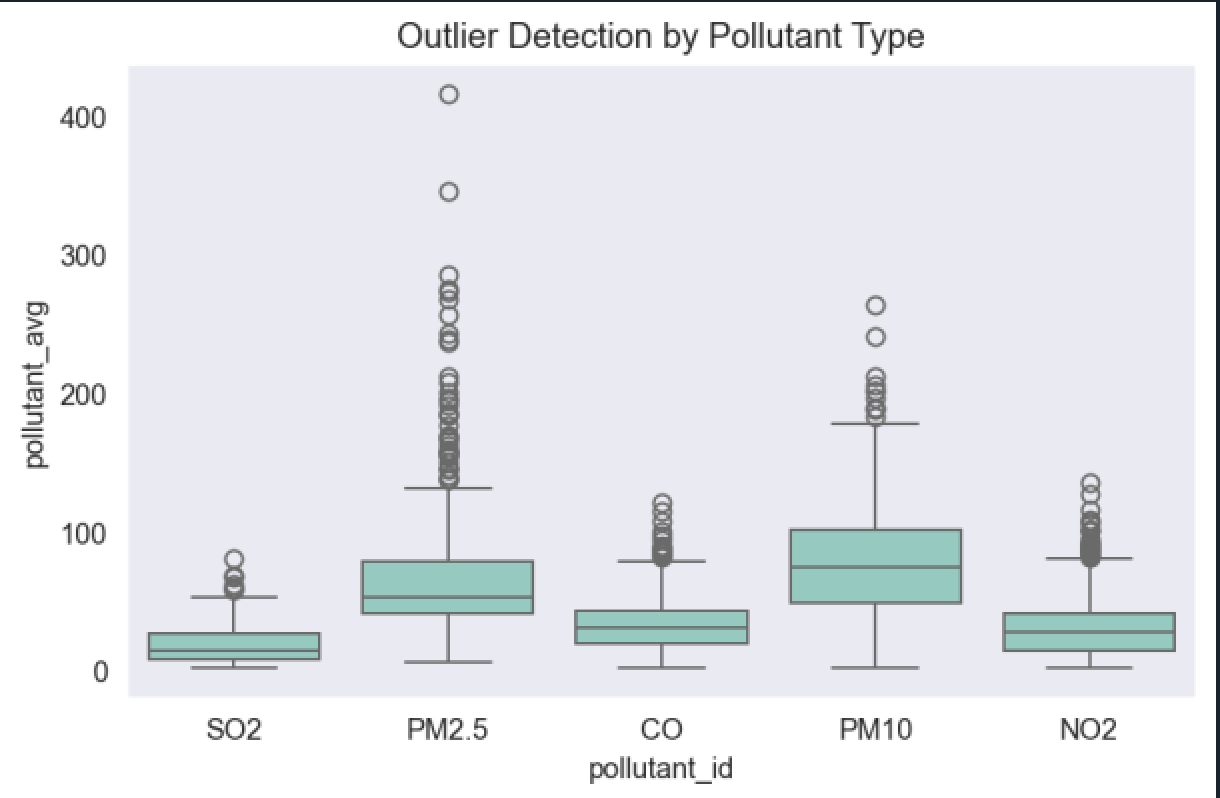
A screen shot of a computer code

AI-generated content may be incorrect.

**iii. Analysis Results**

* Identifies pollutants with consistently high or skewed values.
* Aids in targeted control strategies.

**iv. Visualizations**

* Box plots for each top pollutant type, showing variance and outliers.

### **Objective 6: Geographical Pollution Distribution (Scatter Plot)**

**i. General Description**  
Visualizes geographical pollution concentration using longitude and latitude.

**ii. Specific Requirements, Functions and Formulas**

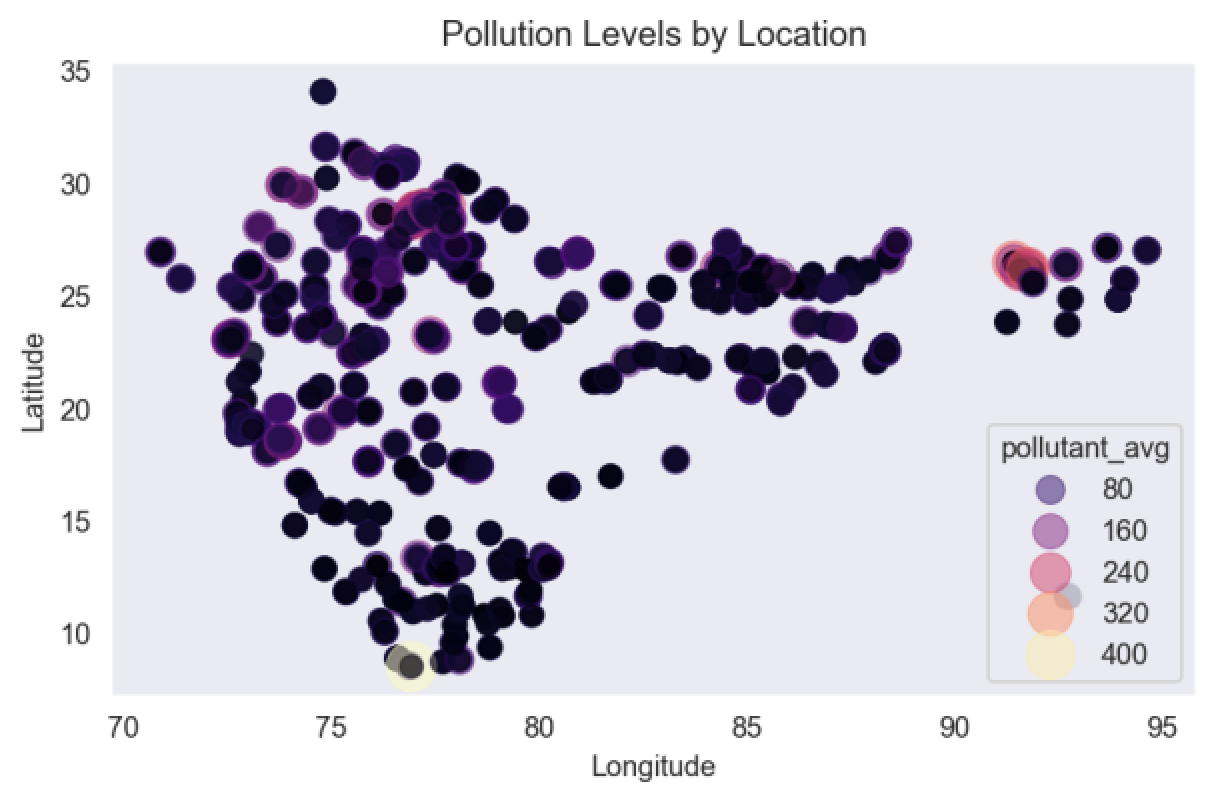
* Columns: longitude, latitude, pollutant\_avg
* A computer screen shot of a program code

  AI-generated content may be incorrect.Function: sns.scatterplot(), size, hue

**iii. Analysis Results**

* Pollution hotspots identified spatially.
* Potential for map-based alert systems.

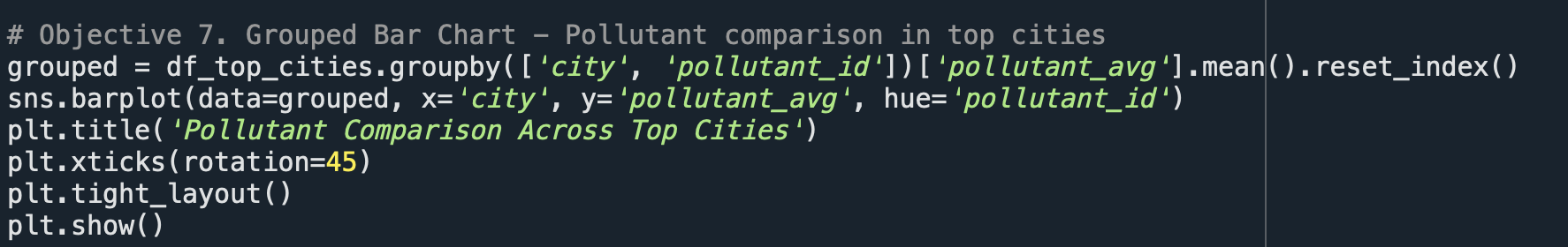
**iv. Visualizations**

* Colored and sized scatter plot.
* Larger and darker dots = higher pollution.

### **Objective 7: Pollutant Comparison Across Top Cities (Grouped Bar Chart)**

**i. General Description**  
Compares average pollutant levels across top cities for each pollutant type.

**ii. Specific Requirements, Functions and Formulas**

* Grouped by: city and pollutant\_id
* Functions: groupby(), reset\_index(), sns.barplot()

**iii. Analysis Results**

* Visual differentiation of pollutant dominance in different cities.

**iv. Visualizations**

* Grouped bar chart with different colors for each pollutant.

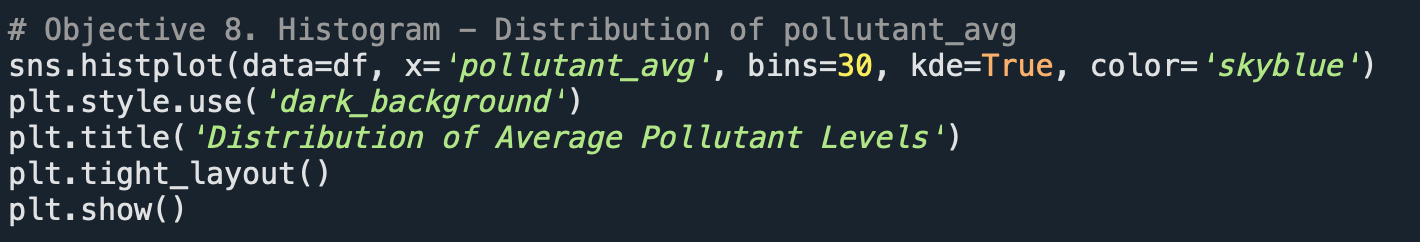
## 

### **Objective 8:** Distribution of pollutant\_avg (Histogram + KDE)

**i. General Description**  
Analyzes the frequency distribution of average pollutant levels.

**ii. Specific Requirements, Functions and Formulas**

* Function: sns.histplot() with kde=True

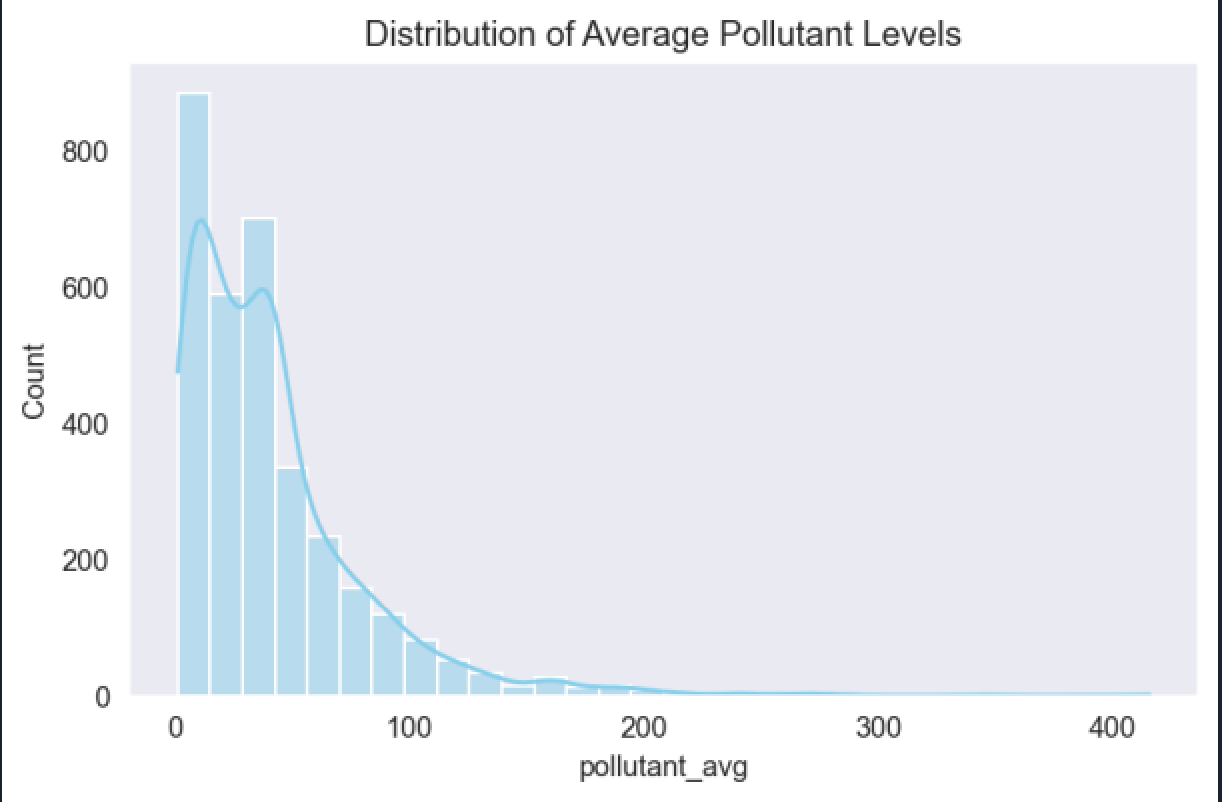


**iii. Analysis Results**

* Skewed distribution found.
* Most values concentrated in a certain range with long tail.

**iv. Visualizations**

* Histogram with KDE overlay for smoothed curve.

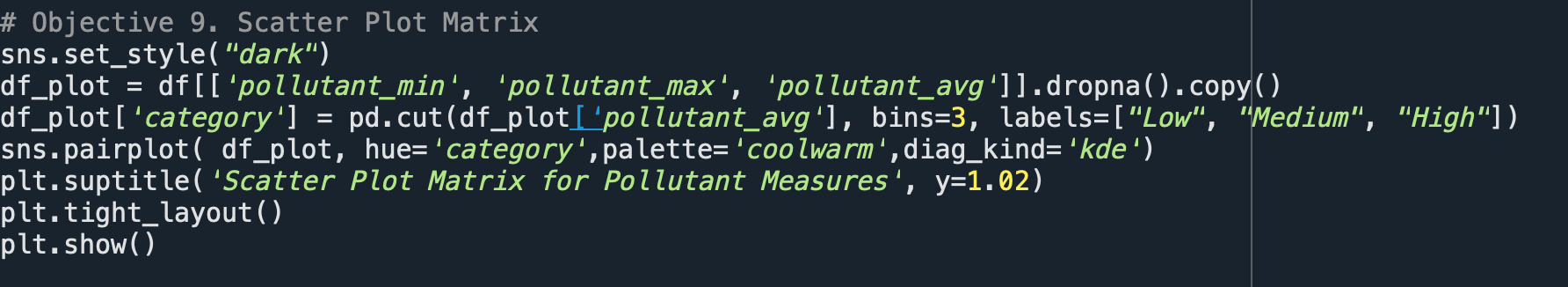


### **Objective 9: Scatter Plot Matrix for Pollution Category Classification**

**i. General Description**  
Explores pairwise relationships and distribution patterns for pollution measures.

**ii. Specific Requirements, Functions and Formulas**

* Functions: pd.cut() to categorize, sns.pairplot()

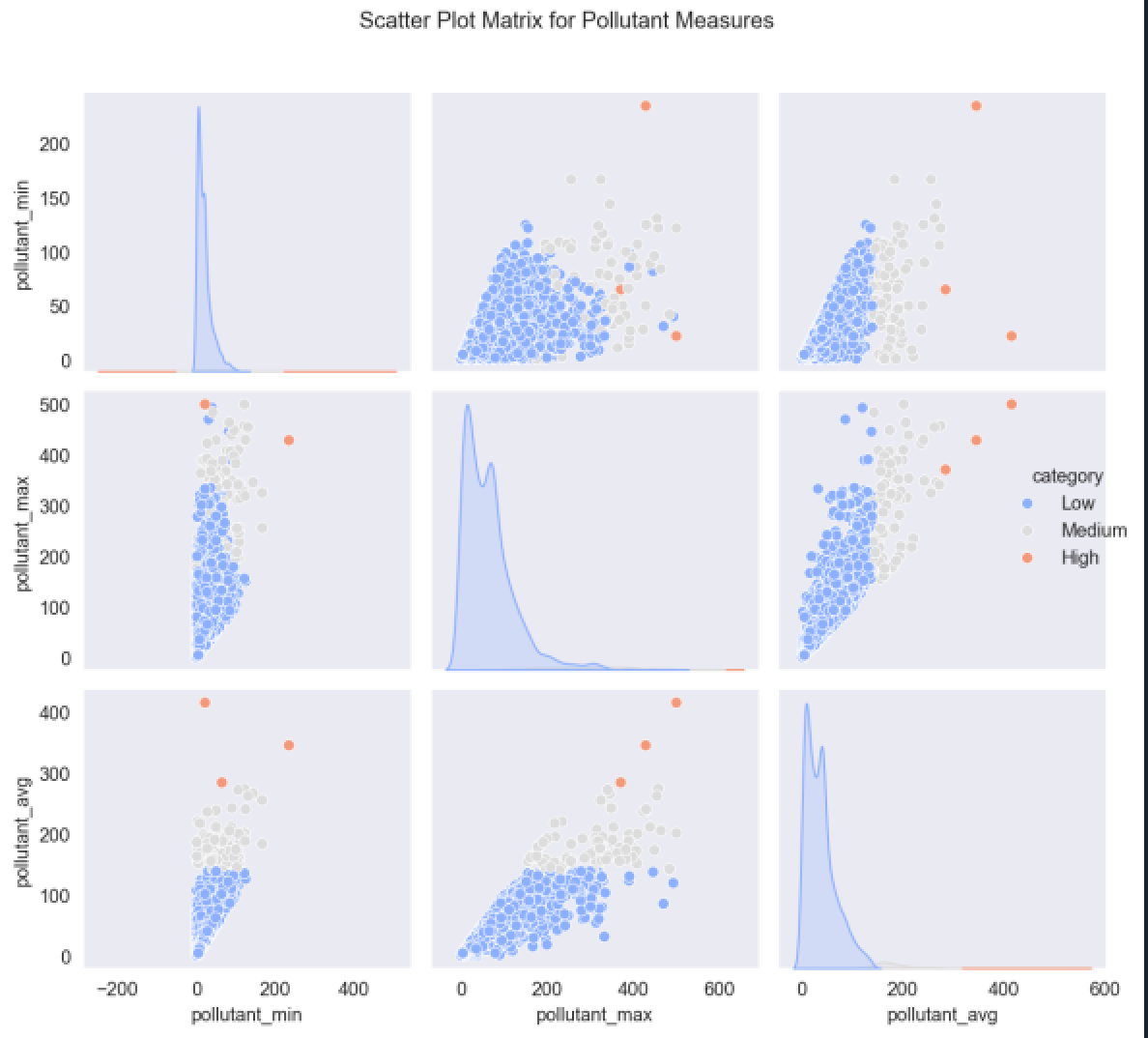


**iii. Analysis Results**

* Pollution level categories: Low, Medium, High.
* Helps visualize cluster behavior and inter-variable dependencies.

**iv. Visualizations**

* Pair plot (scatter matrix) with hue = category.

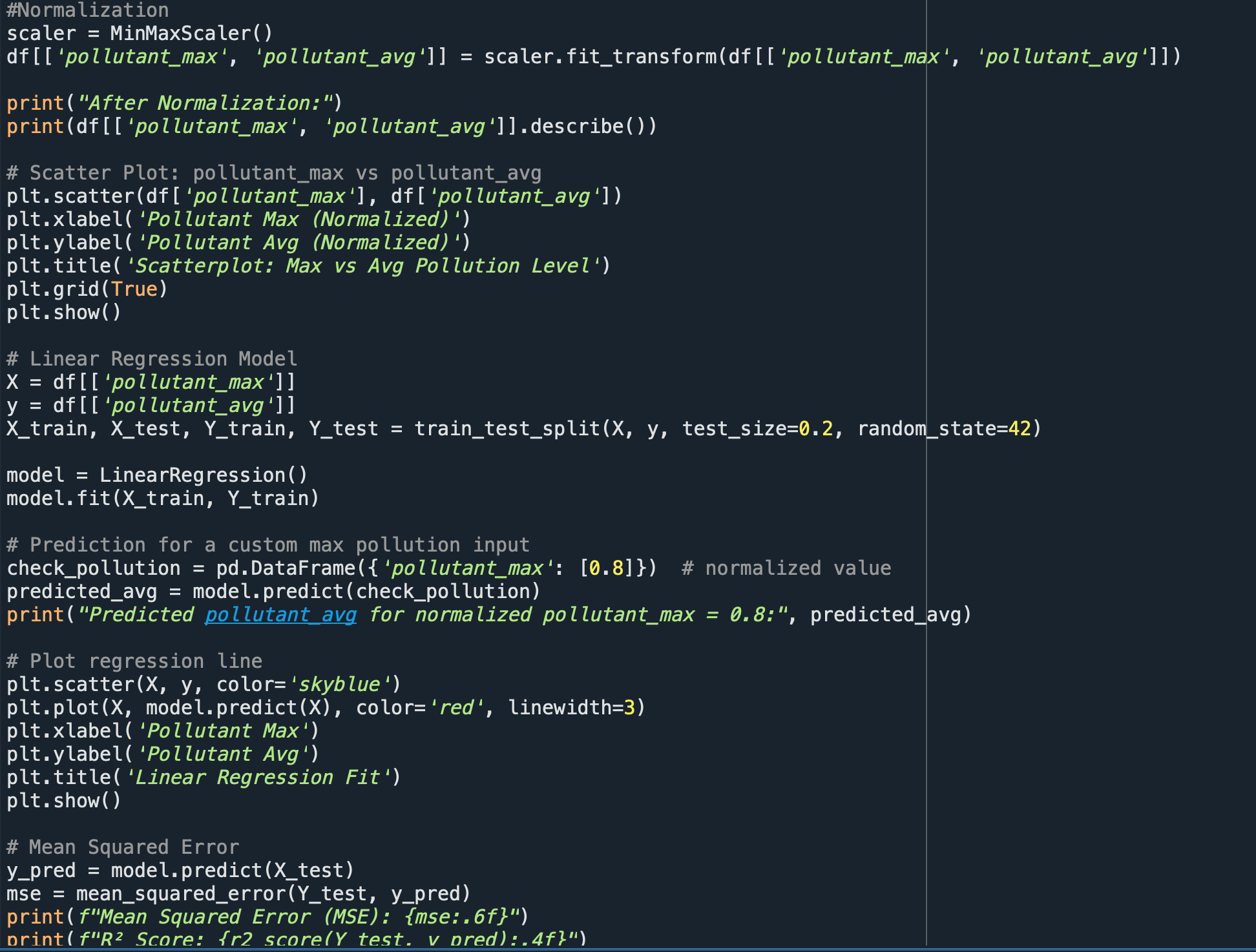


### **Objective 10:** Linear Regression Model (pollutant\_max ➝ pollutant\_avg)

**i. General Description**  
Builds a linear model to predict average pollution based on the maximum value.

**ii. Specific Requirements, Functions and Formulas**

* Libraries: LinearRegression, train\_test\_split, MinMaxScaler
* Metrics: mean\_squared\_error, r2\_score
* Prediction: model.predict()

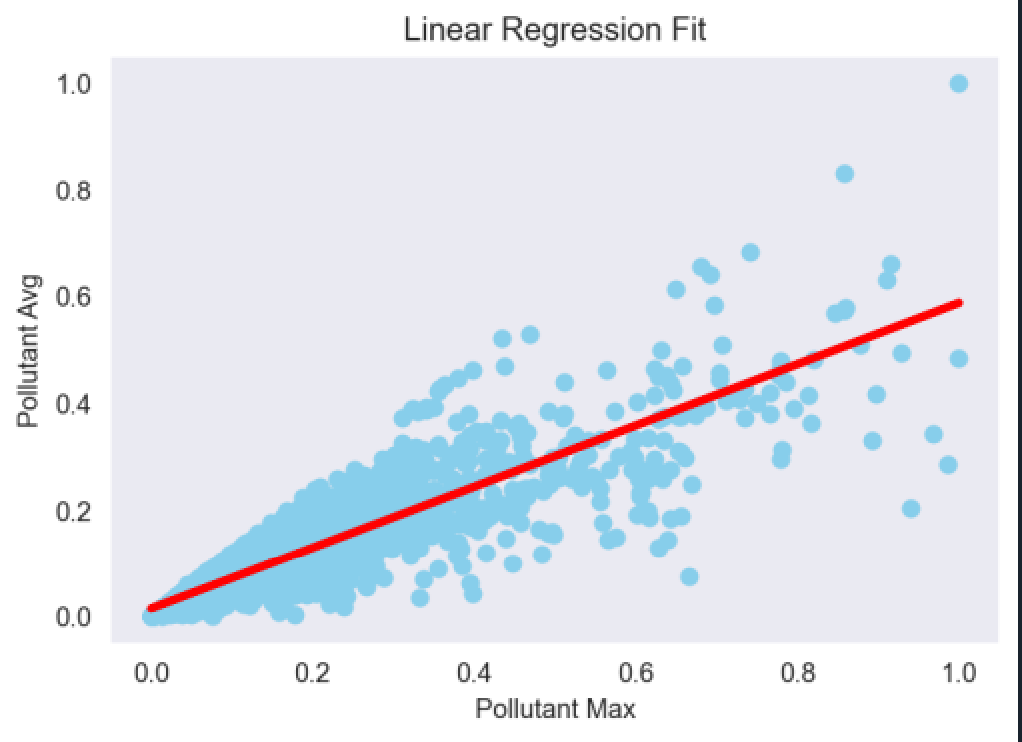


**iii. Analysis Results**

* Strong linear relationship.
* R² Score indicates model reliability.
* MSE provides error margin in prediction.

**iv. Visualizations**

* Regression line on scatter plot of pollutant\_max vs pollutant\_avg.



**WHOLE CODE**  
  
# Air Pollution and Innovation Opportunity Report

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load the dataset

df = pd.read\_csv("/Users/shikharagrawal/Desktop/pyton/pollution.csv")

#Displaying some data

print(df.head())

print(df.info()) #Give Dataset information

print(df.describe()) #Give Statistics Summary of numerical columns

# Step 1: Clean column names (remove extra spaces)

df.columns = df.columns.str.strip()

# Step 2: Handle missing values

print("\nMissing Values: \n", df.isnull().sum()) #Summing missing values columns wise

print("\nMissing Values: \n", df.isnull().sum().sum()) #Summing all the columns missing values

# Filling missing value with mean

for col in ['pollutant\_min', 'pollutant\_max', 'pollutant\_avg']:

df[col] = df[col].fillna(df[col].mean())

#again checking for null values

print(df.isnull().sum())

# 3. Check for duplicates rows and remove them

print(df.duplicated().sum())

df.drop\_duplicates(inplace=True)

# 4. Convert 'last\_update' to datetime

df['last\_update'] = pd.to\_datetime(df['last\_update'], dayfirst=True, errors='coerce')

#Handle missing datetimes

print(df['last\_update'].isnull().sum())

#Step 6: check for misspelled data/Fix categorical data noise

columns\_to\_check = ['country', 'state', 'city', 'station', 'pollutant\_id']

for col in columns\_to\_check:

print(f"\n Unique values in '{col}' (total: {df[col].nunique()}):")

print(df[col].value\_counts().sort\_values(ascending=True))

#Misllaneous cleaning of dataset

print(df['city'].str.lower().value\_counts().head(20))

df['pollutant\_id'] = df['pollutant\_id'].str.upper()

# Standardize state names for consistency

df['state'] = df['state'].str.replace('\_', ' ', regex=False)

#data cleaning completed

#Data Visualization

# Objective 1. Sorted Graph showing Pollutant Type Frequency

pollutant\_counts = df['pollutant\_id'].value\_counts().sort\_values(ascending=False)

sns.barplot(x=pollutant\_counts.index,y=pollutant\_counts.values, palette='viridis')

for i, count in enumerate(pollutant\_counts.values):

plt.text(i, count + 1, str(count), ha='center', va='bottom', fontsize=10)

plt.title("Pollutant Type Frequency (Sorted)", fontsize=14)

plt.xlabel("Pollutant ID")

plt.ylabel("Count")

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

# Objective 2.Top 10 cities with the highest average pollutant levels

plt.style.use('dark\_background')

top\_cities = df.groupby('city')['pollutant\_avg'].mean().sort\_values(ascending=False).head(10)

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

sns.barplot(x=top\_cities.values, y=top\_cities.index, palette="magma")

plt.title("Top 10 Polluted Cities (Avg)", fontsize=16, color='white')

plt.xlabel("Average Pollution Level",color='white')

plt.ylabel("City",color='white')

plt.grid(False)

plt.show()

# Objective 3. Distribution of Average Pollution Levels Across Indian States

plt.style.use('dark\_background')

plt.figure(figsize=(14, 6))

flierprops = dict(marker='o', markerfacecolor='white', markersize=5, linestyle='none')

sns.boxplot(data=df, x='state', y='pollutant\_avg', palette="flare", flierprops=flierprops)

plt.xticks(rotation=90, color='white')

plt.yticks(color='white')

plt.title("Pollution Levels Across States", fontsize=16, color='white')

plt.grid(False)

plt.tight\_layout()

plt.show()

# Objective 4. Correlation Between Pollution Values

plt.figure(figsize=(6,4))

sns.heatmap(df[['pollutant\_min', 'pollutant\_max', 'pollutant\_avg']].corr(), annot=True, cmap='coolwarm')

plt.title("Correlation Between Pollution Values")

plt.show()

# Objective 5. Box Plot - Distribution of pollutant\_avg by pollutant type

sns.set\_style("dark")

top\_cities = df['city'].value\_counts().nlargest(5).index

df\_top\_cities = df[df['city'].isin(top\_cities)]

top\_pollutants = df['pollutant\_id'].value\_counts().nlargest(5).index

df\_top\_pollutants = df[df['pollutant\_id'].isin(top\_pollutants)]

sns.boxplot(data=df\_top\_pollutants, x='pollutant\_id', y='pollutant\_avg')

plt.title('Outlier Detection by Pollutant Type')

plt.tight\_layout()

plt.show()

# Objective 6. Geographical Scatter Plot

sns.set\_style("dark")

sns.scatterplot(data=df, x='longitude', y='latitude', hue='pollutant\_avg', palette='magma',

size='pollutant\_avg', sizes=(50, 300),alpha=0.5,edgecolor=None)

plt.title('Pollution Levels by Location')

plt.tight\_layout()

plt.xlabel("Longitude")

plt.ylabel("Latitude")

plt.show()

# Objective 7. Grouped Bar Chart - Pollutant comparison in top cities

grouped = df\_top\_cities.groupby(['city', 'pollutant\_id'])['pollutant\_avg'].mean().reset\_index()

sns.barplot(data=grouped, x='city', y='pollutant\_avg', hue='pollutant\_id')

plt.title('Pollutant Comparison Across Top Cities')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

# Objective 8. Histogram - Distribution of pollutant\_avg

sns.histplot(data=df, x='pollutant\_avg', bins=30, kde=True, color='skyblue')

plt.style.use('dark\_background')

plt.title('Distribution of Average Pollutant Levels')

plt.tight\_layout()

plt.show()

# Objective 9. Scatter Plot Matrix

sns.set\_style("dark")

df\_plot = df[['pollutant\_min', 'pollutant\_max', 'pollutant\_avg']].dropna().copy()

df\_plot['category'] = pd.cut(df\_plot['pollutant\_avg'], bins=3, labels=["Low", "Medium", "High"])

sns.pairplot( df\_plot, hue='category',palette='coolwarm',diag\_kind='kde')

plt.suptitle('Scatter Plot Matrix for Pollutant Measures', y=1.02)

plt.tight\_layout()

plt.show()

#Model Training

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

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import MinMaxScaler

from sklearn.metrics import mean\_squared\_error

from sklearn.metrics import r2\_score

#Normalization

scaler = MinMaxScaler()

df[['pollutant\_max', 'pollutant\_avg']] = scaler.fit\_transform(df[['pollutant\_max', 'pollutant\_avg']])

print("After Normalization:")

print(df[['pollutant\_max', 'pollutant\_avg']].describe())

# Scatter Plot: pollutant\_max vs pollutant\_avg

plt.scatter(df['pollutant\_max'], df['pollutant\_avg'])

plt.xlabel('Pollutant Max (Normalized)')

plt.ylabel('Pollutant Avg (Normalized)')

plt.title('Scatterplot: Max vs Avg Pollution Level')

plt.grid(True)

plt.show()

# Linear Regression Model

X = df[['pollutant\_max']]

y = df[['pollutant\_avg']]

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = LinearRegression()

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

# Prediction for a custom max pollution input

check\_pollution = pd.DataFrame({'pollutant\_max': [0.8]}) # normalized value

predicted\_avg = model.predict(check\_pollution)

print("Predicted pollutant\_avg for normalized pollutant\_max = 0.8:", predicted\_avg)

# Plot regression line

plt.scatter(X, y, color='skyblue')

plt.plot(X, model.predict(X), color='red', linewidth=3)

plt.xlabel('Pollutant Max')

plt.ylabel('Pollutant Avg')

plt.title('Linear Regression Fit')

plt.show()

# Mean Squared Error

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

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

print(f"Mean Squared Error (MSE): {mse:.6f}")

print(f"R² Score: {r2\_score(Y\_test, y\_pred):.4f}")

**LinkedIn POST LINK: -**

[CLICK HERE](https://www.linkedin.com/posts/shikharagrawal2408_datawithpurpose-airpollutionanalysis-sustainability-activity-7316779156220911617-u2Nn?utm_source=share&utm_medium=member_desktop&rcm=ACoAAEc3-jQBI7l3CuBPLkTkBxH-q-8MbAbUtz4)

# 5. Conclusion

This project showcased the real-world application of data science in analyzing environmental issues. The dataset revealed major pollution hotspots and trends across Indian cities. The visualizations enhanced data understanding, and the regression model offered a simple yet effective way to estimate pollution levels. This project also highlighted the importance of clean data and the power of visual storytelling in data science.

# 6. Future Scope

There’s a lot of potential to take this analysis further. Some ideas for future work include:

 Integrate **geospatial tools** like Folium for map-based pollution analysis

 Use **time-series modeling** for forecasting future pollution levels

 Apply **classification algorithms** to predict pollutant types

 Incorporate **real-time IoT data** from air quality sensors

 Explore **clustering algorithms** to identify pollution zones

# 7. References

[1] Wes McKinney, *Python for Data Analysis*, O’Reilly Media  
[2] pandas documentation: <https://pandas.pydata.org/>  
[3] seaborn documentation: <https://seaborn.pydata.org/>  
[4] scikit-learn documentation: <https://scikit-learn.org/stable/>  
[5] matplotlib documentation: <https://matplotlib.org/>