**Pollution Analysis and Visualization Report**

**Prepared By:** SHAILESH K R

**Mentor:** M MUVENDIRAN

**Project title:** Analysis of Air Pollution Data

**INTRODUCTION**

Air pollution is a critical environmental issue affecting human health and ecological balance. This project explores air quality data, focusing on parameters like PM2.5 concentrations, temperature trends, and the influence of meteorological factors on pollution levels. During the internship period, advanced data visualization techniques were employed to analyze these factors using Python libraries such as pandas, seaborn, and matplotlib.

**PROJECT DESCRIPTION**

The analysis showcases advanced techniques in data handling and visualization, providing a foundation for further exploration and potential machine learning applications. This project was completed as part of my internship program under the guidance of M. Muvendiran.

**CODE EXPLANATION**

The code systematically processes and analyzes air pollution data, focusing on PM2.5 concentrations and meteorological factors. It includes data loading, cleaning, and visualization using Python libraries like pandas, matplotlib, and seaborn. Key features include handling missing values, exploring correlations, and generating insightful visualizations such as heatmaps, scatter plots, and trend lines.

**CODE OVERVIEW**

1. import pandas as pd

data = pd.read\_csv('pollution.csv')

**Explanation:** The code uses the pandas library to load the dataset "pollution.csv" into a DataFrame. The read\_csv() function reads the CSV file and stores it in the variable data, allowing for further data manipulation and analysis. This step is essential for accessing and working with the dataset in Python.

2) print(data.head())

print(data.info())

print(data.describe())

**Explanation:** The code provides a quick overview of the dataset using three `pandas` functions: `head()` to display the first few rows, `info()` to show data types and missing values, and `describe()` to generate descriptive statistics for numerical columns, helping to understand the structure and distribution of the data.

3) data.dropna()

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

print("Missing values:\n", data.isnull().sum())

data['pm2.5'] = data['pm2.5'].fillna(data['pm2.5'].median())

print ("Updated missing values:", data.isnull().sum())

**Explanation:** The code handles missing values in the dataset by first dropping any rows with `dropna()`, then checking for remaining null values using `isnull().sum()`. It then fills missing values in the 'pm2.5' column with the median value of that column using `fillna()`, and prints the updated count of missing values to confirm the changes.

4) import matplotlib.pyplot as plt

import seaborn as sns

**Explanation:** The code imports the `matplotlib.pyplot` and `seaborn` libraries for data visualization. `seaborn.set\_style('whitegrid')` sets the visual style of the plots to a clean, white background with gridlines, making the graphs easier to interpret and aesthetically pleasing.

5) plt.figure(figsize=(8, 5))

plt.plot(data['TEMP'], label='Temperature', color='blue')

plt.xlabel('Index')

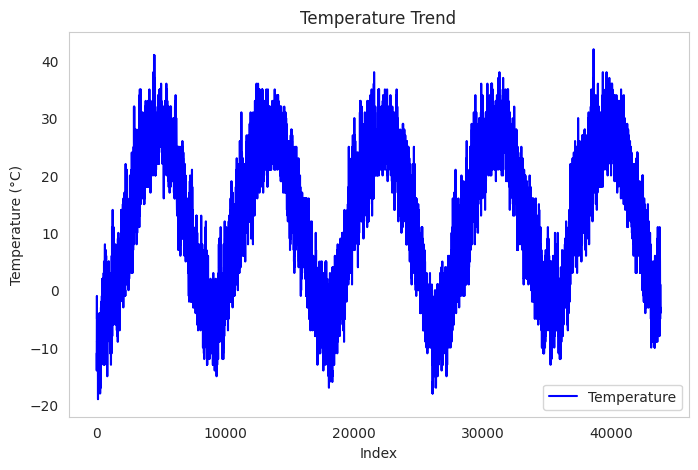
plt.ylabel('Temperature (°C)')

plt.title('Temperature Trend')

plt.legend()

plt.grid()

plt.show()



**Explanation:** The code generates a line plot to visualize the temperature trend over the dataset. It sets the figure size to 8x5 inches, plots the 'TEMP' column in blue, and labels the x-axis as 'Index' and the y-axis as 'Temperature (°C)'. The plot is titled 'Temperature Trend', includes a legend, and displays a grid for better readability, followed by showing the plot using `plt.show()`.

6) plt.figure(figsize=(8, 5))

plt.scatter(data['TEMP'], data['pm2.5'], color='green', alpha=0.6)

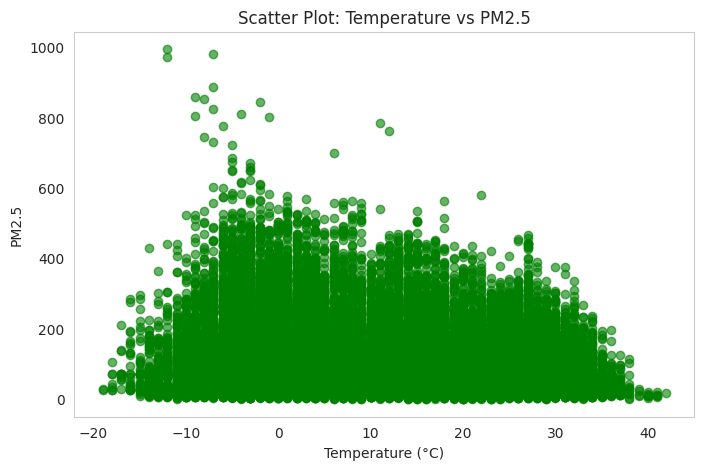
plt.xlabel('Temperature (°C)')

plt.ylabel('PM2.5')

plt.title('Scatter Plot: Temperature vs PM2.5')

plt.grid()

plt.show()



**Explanation:** The code creates a scatter plot to examine the relationship between temperature ('TEMP') and PM2.5 levels. It sets the figure size to 8x5 inches, uses green for the plot points with an alpha transparency of 0.6, and labels the x-axis as 'Temperature (°C)' and the y-axis as 'PM2.5'. The plot is titled 'Scatter Plot: Temperature vs PM2.5' and includes a grid for better visualization, followed by displaying the plot with `plt.show()`.

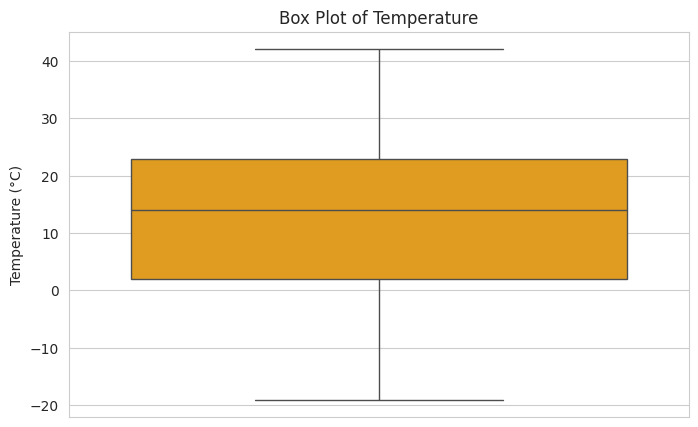
7) plt.figure(figsize=(8, 5))

sns.boxplot(y=data['TEMP'], color='orange')

plt.title('Box Plot of Temperature')

plt.y label('Temperature (°C)')

plt.show()



**Explanation:** The code generates a box plot to visualize the distribution of temperature values ('TEMP'). It sets the figure size to 8x5 inches, uses orange for the plot color, and labels the y-axis as 'Temperature (°C)'. The plot is titled 'Box Plot of Temperature' and is displayed with `plt.show()`, helping to identify the spread, median, and any outliers in the temperature data.

8) plt.figure(figsize=(8, 5))

plt.hist(data['pm2.5'], bins=20, color='skyblue', edgecolor='black')

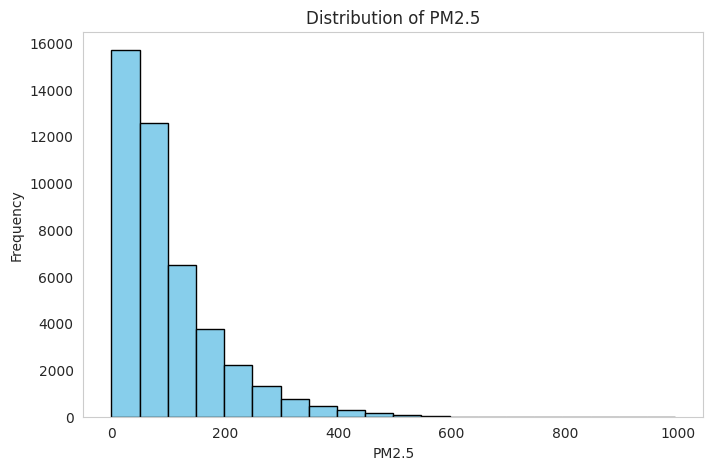
plt.title('Distribution of PM2.5')

plt.xlabel('PM2.5')

plt.ylabel('Frequency')

plt.grid()

plt.show()



**Explanation:** The code creates a histogram to visualize the distribution of PM2.5 values in the dataset. It sets the figure size to 8x5 inches, uses skyblue for the bars with black edges, and divides the data into 20 bins. The x-axis is labeled 'PM2.5', the y-axis is labeled 'Frequency', and the plot is titled 'Distribution of PM2.5'. The histogram is displayed with a grid for better readability using `plt.show()`.

9) cbwd\_counts = data['cbwd'].value\_counts()

plt.figure(figsize=(8, 5))

cbwd\_counts.plot(kind='bar', color='teal', edgecolor='black')

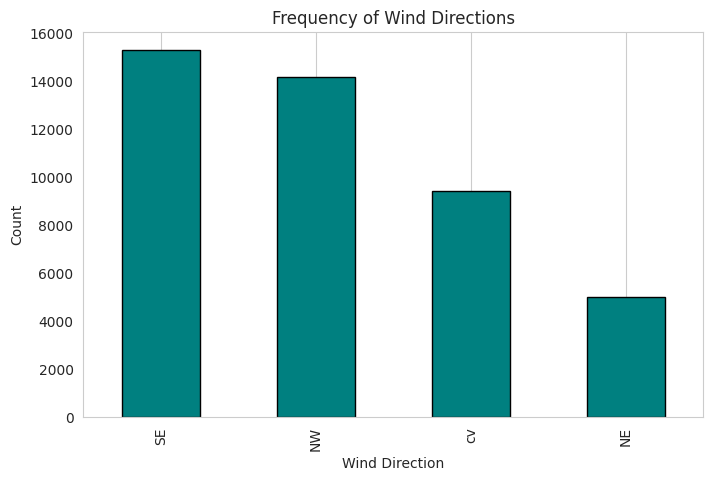
plt.title('Frequency of Wind Directions')

plt.xlabel('Wind Direction')

plt.ylabel('Count')

plt.grid(axis='y')

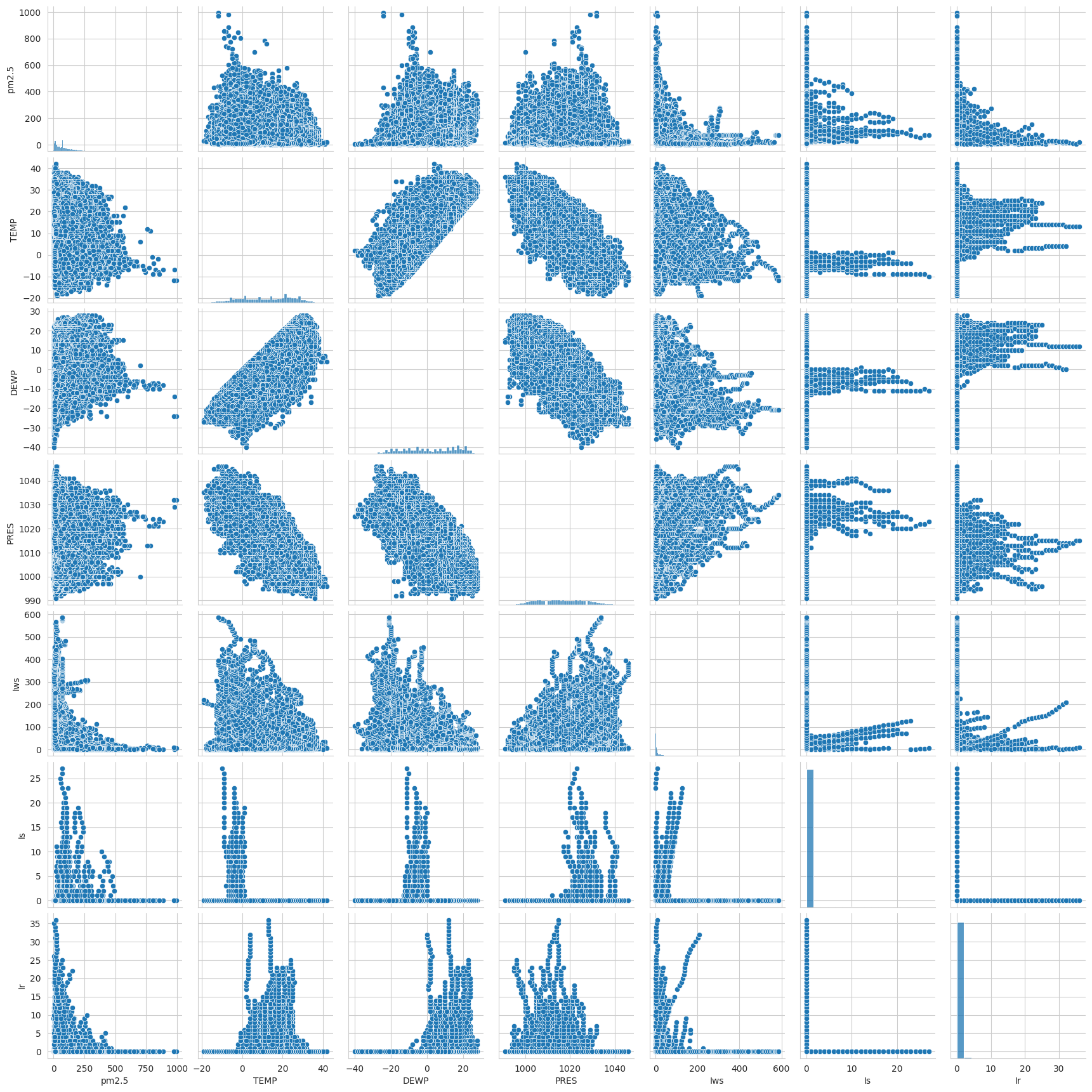
plt.show()



**Explanation:** The code counts the frequency of each unique wind direction in the 'cbwd' column using `value\_counts()` and stores the result in `cbwd\_counts`. It then creates a bar plot with the wind direction counts, using teal for the bars and black edges. The plot is set to a size of 8x5 inches, with labeled x-axis as 'Wind Direction' and y-axis as 'Count'. The plot is titled 'Frequency of Wind Directions' and includes a grid along the y-axis for better visualization, followed by displaying the plot using `plt.show()`.

10) sns.pairplot(data[['pm2.5', 'TEMP', 'DEWP', 'PRES', 'Iws', 'Is', 'Ir']])

plt.show()



**Explanation:** The code generates a pair plot using the `seaborn` library to visualize the relationships between multiple variables: 'pm2.5', 'TEMP', 'DEWP', 'PRES', 'Iws', 'Is', and 'Ir'. This plot creates scatter plots for each pair of variables along with histograms on the diagonal to show the distribution of individual variables. The `plt.show()` function displays the plot, allowing for a detailed exploration of correlations and patterns between the selected features.

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

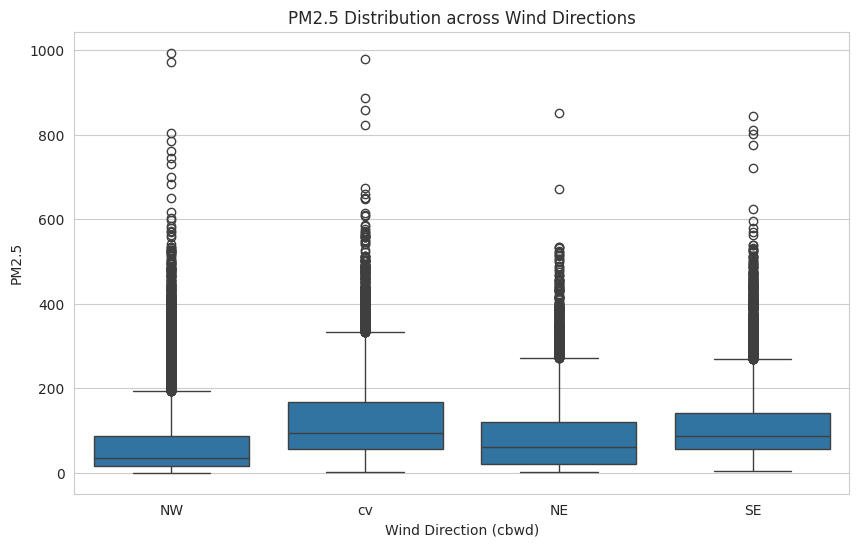
sns.boxplot(x='cbwd', y='pm2.5', data=data)

plt.xlabel('Wind Direction (cbwd)')

plt.ylabel('PM2.5')

plt.title('PM2.5 Distribution across Wind Directions')

plt.show()



**Explanation:** The code generates a box plot to visualize the distribution of PM2.5 levels across different wind directions ('cbwd'). The figure size is set to 10x6 inches, with the x-axis labeled as 'Wind Direction (cbwd)' and the y-axis as 'PM2.5'. The plot is titled 'PM2.5 Distribution across Wind Directions' and helps identify variations, medians, and outliers of PM2.5 concentrations for each wind direction category. The plot is displayed using `plt.show()`.

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

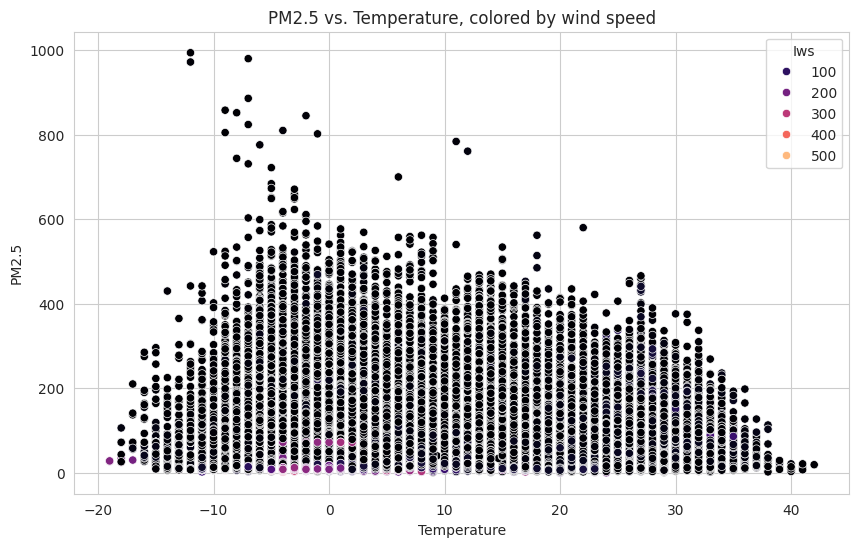
sns.scatterplot(x='TEMP', y='pm2.5', hue='Iws', data=data,

palette='magma')

plt.xlabel('Temperature') plt.ylabel('PM2.5')

plt.title('PM2.5 vs. Temperature, colored by wind speed')

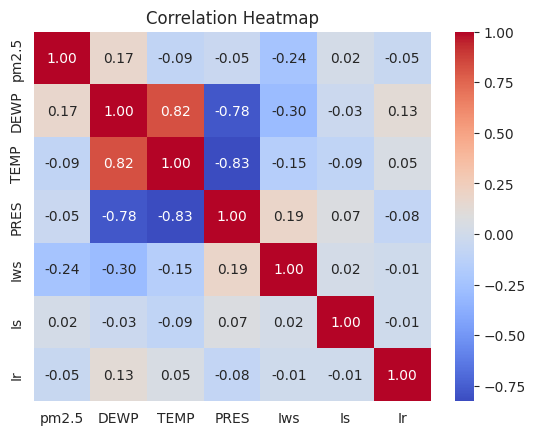
plt.show()



**Explanation:** The code creates a scatter plot to examine the relationship between temperature ('TEMP') and PM2.5 concentrations, with points colored by wind speed ('Iws') using the 'magma' color palette. The figure size is set to 10x6 inches, with labeled x-axis as 'Temperature' and y-axis as 'PM2.5'. The plot is titled 'PM2.5 vs. Temperature.

13) sns.heatmap(data[['pm2.5', 'DEWP', 'TEMP', 'PRES', 'Iws', 'Is', 'Ir']].corr(), annot=True, cmap='coolwarm', fmt=".2f")

plt.title('Correlation Heatmap') plt.show()



**Explanation:** The code creates a scatter plot to examine the relationship between temperature ('TEMP') and PM2.5 concentrations, with points colored by wind speed ('Iws') using the 'magma' color palette. The figure size is set to 10x6 inches, with labeled x-axis as 'Temperature' and y-axis as 'PM2.5'. The plot is titled 'PM2.5 vs. Temperature, colored by wind speed', providing insights into how wind speed influences the relationship between temperature and PM2.5 levels. The plot is displayed using `plt.show()`.

**CONCLUSION**

This report presents a comprehensive analysis of the pollution dataset, highlighting key trends and relationships through data visualization. The insights generated can inform environmental policy or further machine learning projects for predictive analysis.

**Acknowledgments**: This task was completed during my internship under the guidance of Mentor M MUVENDIRAN.

**DATASET DETAILS**

The dataset used in this analysis, "pollution.csv," contains various features including PM2.5 concentrations, temperature, dew point, atmospheric pressure, wind speed, and wind directions. This dataset was preprocessed and visualized to uncover trends and relationships among the features.

**Link to Dataset:** <https://github.com/jbrownlee/Datasets/blob/master/pollution.csv>