AIR_QUALITY_BEIRUT_06_10_2023

October 7, 2023

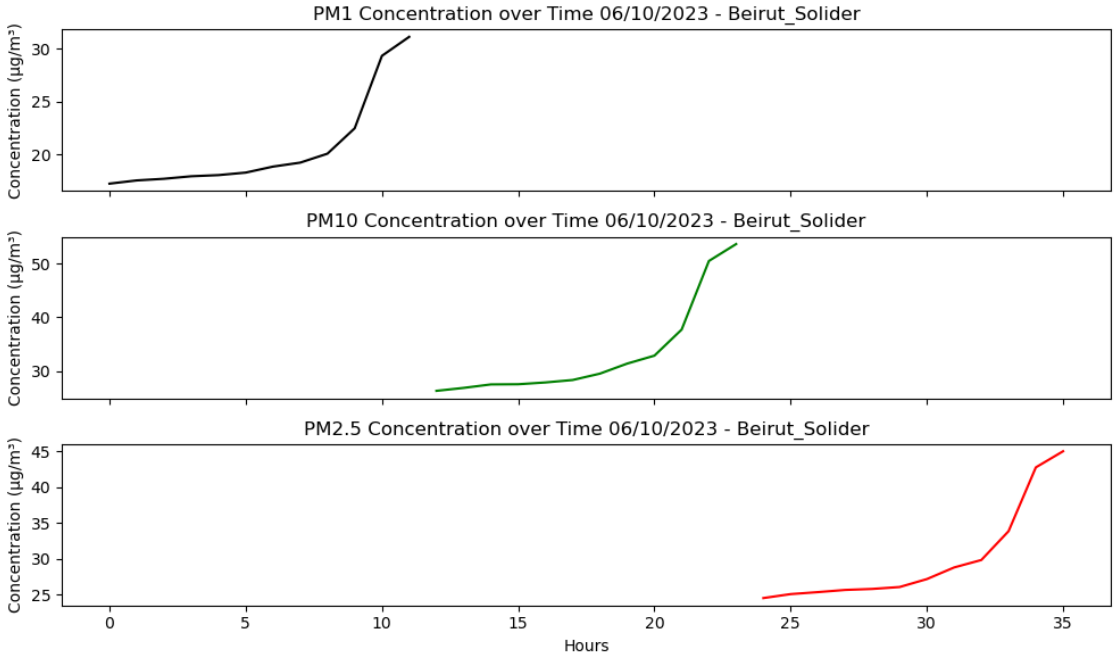
[1]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
```

[2]:
```
# Read the CSV Dataframe
df = pd.read_csv("/home/zorinosgh/Desktop/CLIMATE_CHANGE_MODELS/
 ↪air_pollution_beirut_06_10_2023.csv")

# Get the unique values of the parameters column
parameters = df['parameter'].unique()

# Print the unique values of the parameters column
print(parameters)

# Get the data type of the values column
values_dtype = df['value'].dtype

# Print the data type of the values column
print(values_dtype)

# Get the date and time of the first recording
first_recording_datetime = df['dateLocal'].iloc[0]

# Print the date and time of the first recording
print(first_recording_datetime)
```

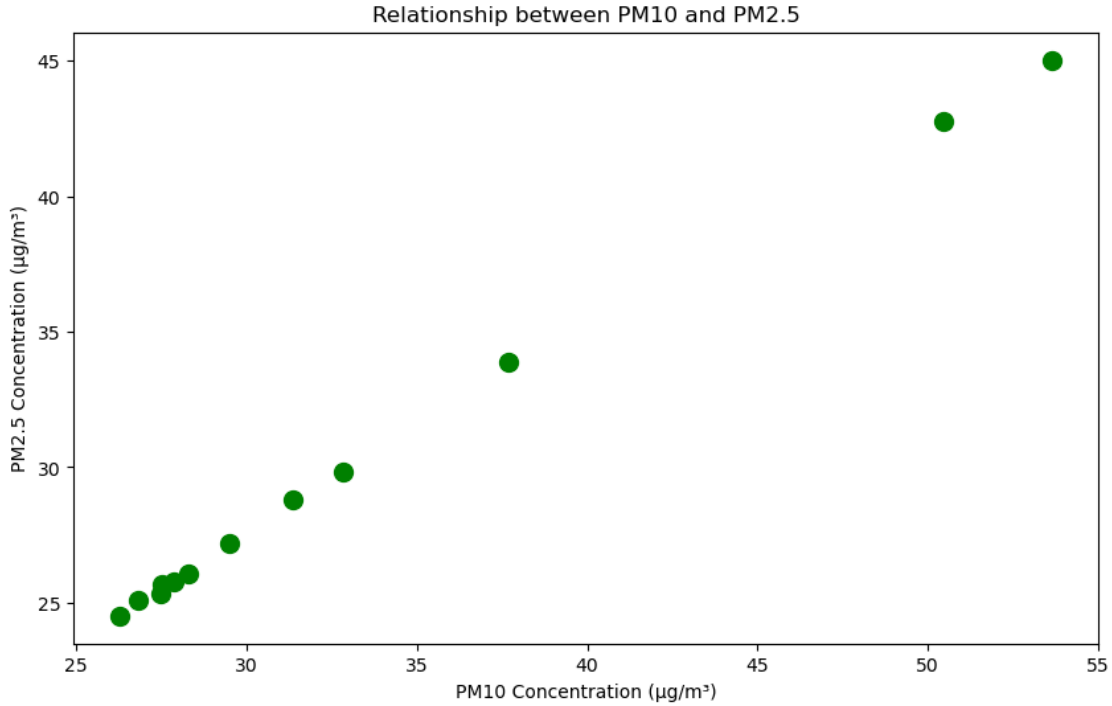
```
['pm1' 'pm10' 'pm25']
float64
2023-10-06T15:00:00+03:00
```

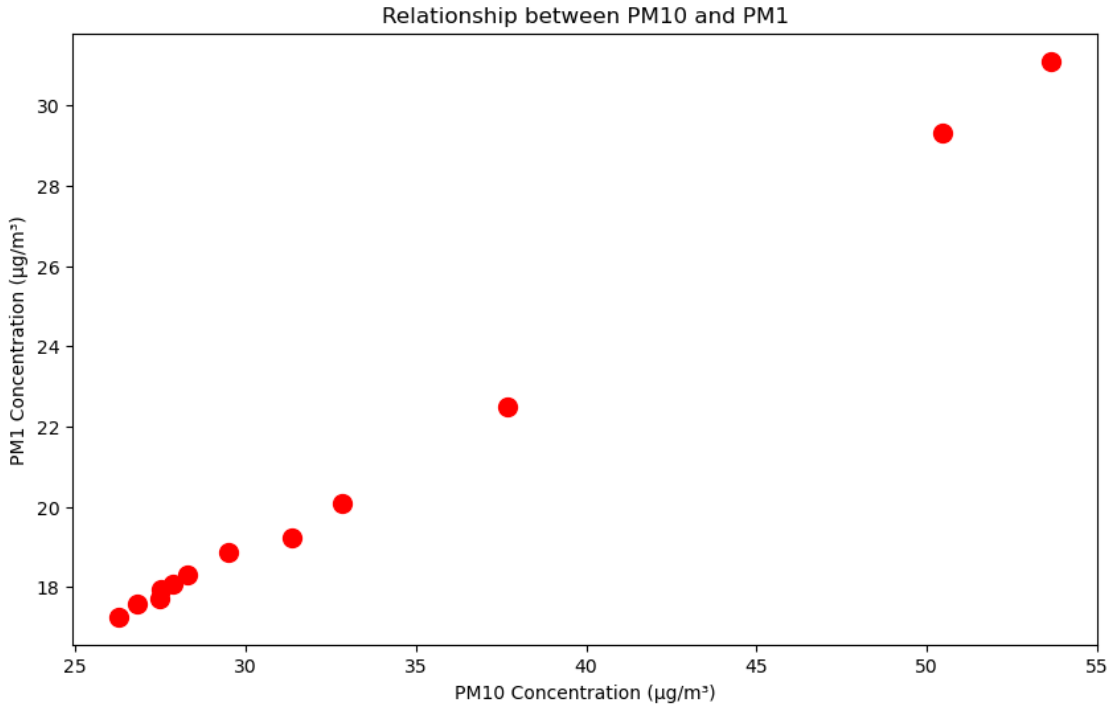
[3]:
```
# Create a figure with 3 subplots
fig, axes = plt.subplots(3, sharex=True, figsize=(10, 6))

# Plot the PM1 data on the first subplot
```

```
axes[0].plot(df['value'][df['parameter'] == 'pm1'], color='black')
axes[0].set_title('PM1 Concentration over Time 06/10/2023 - Beirut_Solider')

# Plot the PM10 data on the third subplot
axes[1].plot(df['value'][df['parameter'] == 'pm10'], color='green')
axes[1].set_title('PM10 Concentration over Time 06/10/2023 - Beirut_Solider')

# Plot the PM2.5 data on the second subplot
axes[2].plot(df['value'][df['parameter'] == 'pm25'], color='red')
axes[2].set_title('PM2.5 Concentration over Time 06/10/2023 - Beirut_Solider')

# Set the x-axis label for all subplots
axes[2].set_xlabel('Hours')

# Set the y-axis label for each subplot
for ax in axes:
    ax.set_ylabel('Concentration (µg/m³)')

# Tighten the layout of the subplots
fig.tight_layout()

# Show the plot
plt.show()
```



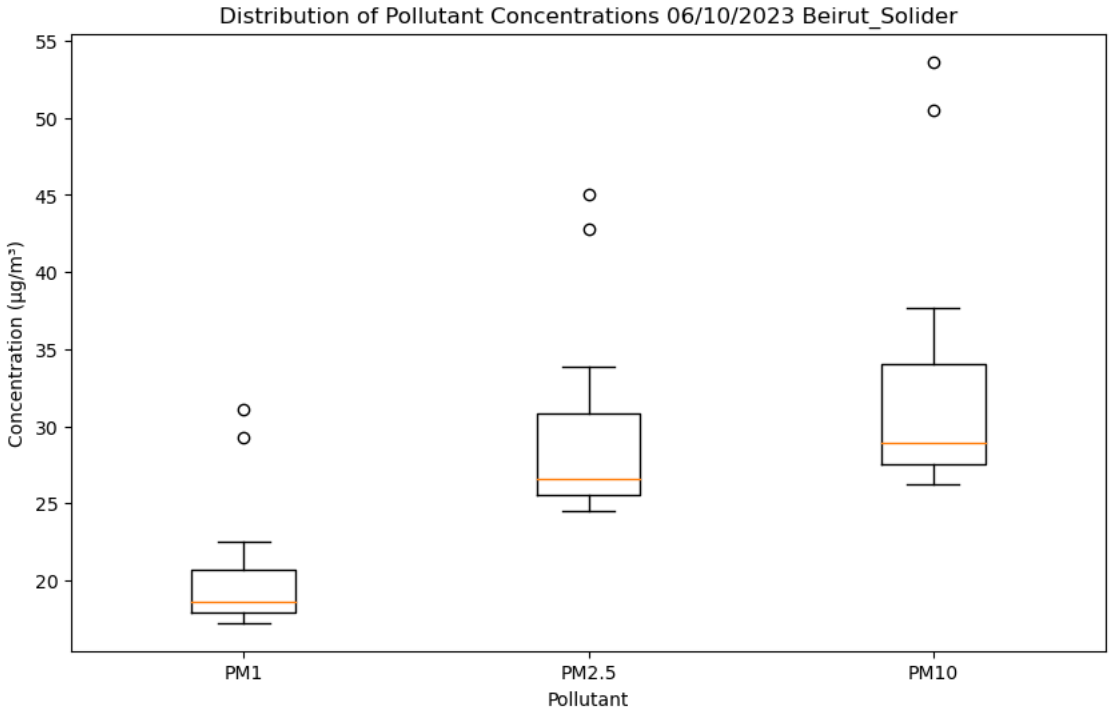
[4]:
```
# Create a scatter plot of PM10 and PM2.5
plt.figure(figsize=(10, 6))
plt.scatter(df['value'][df['parameter'] == 'pm10'], df['value'][df['parameter']␣
 ↪== 'pm25'], marker='o', s=100, color = 'green')
plt.xlabel('PM10 Concentration (µg/m³)')
plt.ylabel('PM2.5 Concentration (µg/m³)')
plt.title('Relationship between PM10 and PM2.5')
plt.show()

# Create a scatter plot of PM10 and PM1
plt.figure(figsize=(10, 6))
plt.scatter(df['value'][df['parameter'] == 'pm10'], df['value'][df['parameter']␣
 ↪== 'pm1'], marker='o', s=100, color ='red')
plt.xlabel('PM10 Concentration (µg/m³)')
plt.ylabel('PM1 Concentration (µg/m³)')
plt.title('Relationship between PM10 and PM1')
plt.show()
```

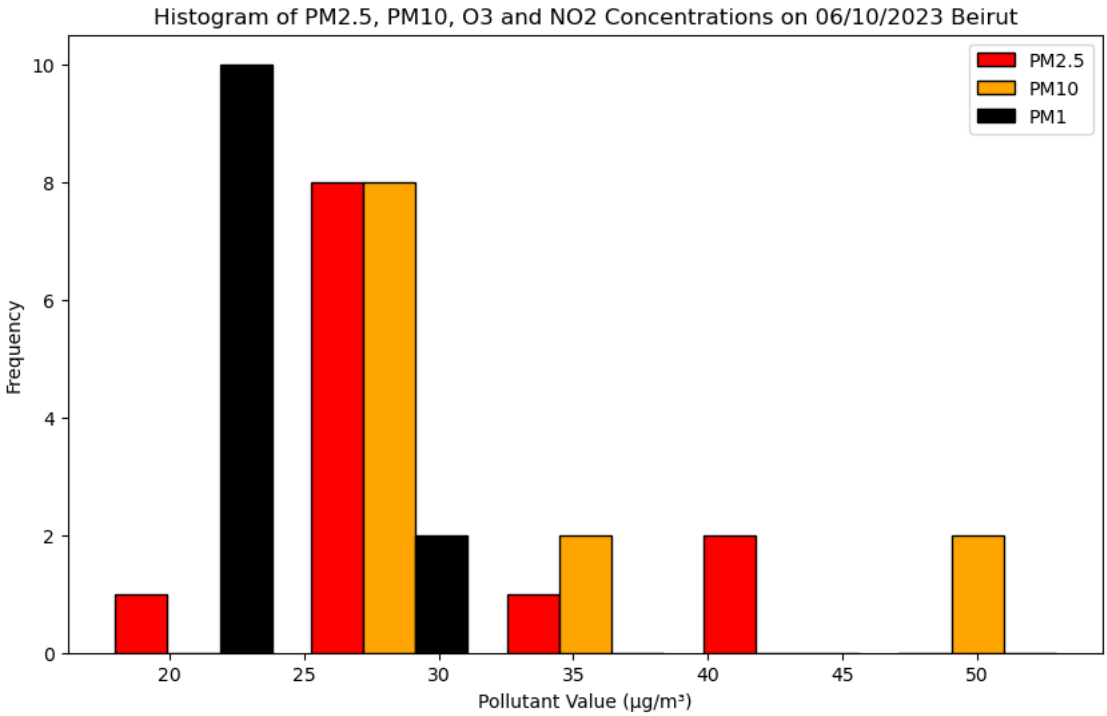




[5]:
```
# Create a box plot of all 4 pollutants
plt.figure(figsize=(10, 6))
plt.boxplot([df['value'][df['parameter'] == 'pm1'], df['value'][df['parameter']␣
 ↪== 'pm25'], df['value'][df['parameter'] == 'pm10']], labels=['PM1', 'PM2.5',␣
 ↪'PM10'])
plt.xlabel('Pollutant')
plt.ylabel('Concentration (µg/m³)')
plt.title('Distribution of Pollutant Concentrations 06/10/2023 Beirut_Solider')
plt.show()

# Get the values of the three pollutants
pm1 = df['value'][df['parameter'] == 'pm1']
pm25 = df['value'][df['parameter'] == 'pm25']
pm10 = df['value'][df['parameter'] == 'pm10']

# Create a list of the 3 pollutants
pollutants = ['PM2.5', 'PM10', 'PM1']

# Create a list of the colors to use for the histogram
colors = ['red', 'orange', 'black']

# Create the histogram
plt.figure(figsize=(10, 6))
plt.hist([pm25, pm10, pm1], bins=5, edgecolor='black', color=colors)
```

```
# Create a legend
plt.legend(pollutants, loc='upper right')

# Set the labels for the x- and y-axes
plt.xlabel('Pollutant Value (µg/m³)')
plt.ylabel('Frequency')

# Set the title of the plot
plt.title('Histogram of PM2.5, PM10, O3 and NO2 Concentrations on 06/10/2023␣
 ↪Beirut')

# Show the plot
plt.show()
```





[6]:
```
# Calculate the average concentration of each pollutant
pm1_avg = df['value'][df['parameter'] == 'pm1'].mean()
pm10_avg = df['value'][df['parameter'] == 'pm10'].mean()
pm25_avg = df['value'][df['parameter'] == 'pm25'].mean()

# Calculate the standard deviation of the concentration of each pollutant
pm1_std  = df['value'][df['parameter'] == 'pm1'].std()
pm10_std = df['value'][df['parameter'] == 'pm10'].std()
pm25_std = df['value'][df['parameter'] == 'pm25'].std()

# Create a list of pollutant names and their average and standard deviation
pollutant_info = [['PM1', pm1_avg, pm1_std], ['PM2.5', pm25_avg, pm25_std],␣
 ↪['PM10', pm10_avg, pm10_std]]

# Print the pollutant information in a table
print('Pollutant | Average Concentration (µg/m³) | Standard Deviation (µg/m³)␣
 ↪|')
print('------- | ------- | ------- |')
for pollutant in pollutant_info:
    print(f'{pollutant[0]} | {pollutant[1]} | {pollutant[2]} |')
```

```
Pollutant | Average Concentration (µg/m³) | Standard Deviation (µg/m³) |
------- | ------- | ------- |
PM1 | 20.664014652493115 | 4.690623603631506 |
```

PM2.5 | 29.993888105668287 | 7.010802409127218 |
PM10 | 33.315974328566284 | 9.337508586648235 |

[7]:
```
# Define the WHO air quality standards for the three pollutants
who_standards = {
    'pm1': 15,
    'pm25': 5,
    'pm10': 20
}

# Check the average concentrations of PM1, PM2.5, and PM10 against the WHO air␣
 ↪quality standards
def check_pollutant_concentrations(pollutant, concentration):
    who_standard = who_standards[pollutant]
    if concentration > who_standard:
        print('Warning: The average concentration of {} exceeds the WHO air␣
 ↪quality standard of {} µg/m³.'.format(pollutant, who_standard))

# Check the average concentrations of all three pollutants
check_pollutant_concentrations('pm1', pm1_avg)
check_pollutant_concentrations('pm25', pm25_avg)
check_pollutant_concentrations('pm10', pm10_avg)
```

Warning: The average concentration of pm1 exceeds the WHO air quality standard of 15 µg/m³.
Warning: The average concentration of pm25 exceeds the WHO air quality standard of 5 µg/m³.
Warning: The average concentration of pm10 exceeds the WHO air quality standard of 20 µg/m³.

[8]:
```
who_pm10 = 20
who_pm1 = 15
who_pm25 = 5

WHO_Diff_PM1 = pm1_avg - who_pm1
WHO_Diff_PM10 = pm10_avg - who_pm10
WHO_Diff_PM25 = pm25_avg - who_pm25

print("PM1 exceeds WHO standards by:", WHO_Diff_PM1, "µg/m³")
print("PM10 exceeds WHO standards by:", WHO_Diff_PM10, "µg/m³")
print("PM2.5 exceeds WHO standards by:", WHO_Diff_PM25, "µg/m³")

def check_air_quality(pm1_avg, pm25_avg, pm10_avg):
    """
    Checks the air quality based on the average concentrations of PM1, PM2.5,␣
 ↪and PM10.
```

```
    Args:
        pm1_avg: The average concentration of PM1 in µg/m³.
        pm25_avg: The average concentration of PM2.5 in µg/m³.
        pm10_avg: The average concentration of PM10 in µg/m³.

    Returns:
        A string indicating the air quality, either "Okay" or "Not Okay".
    """

    who_pm1 = 15
    who_pm25 = 5
    who_pm10 = 20

    if pm1_avg > who_pm1 or pm25_avg > who_pm25 or pm10_avg > who_pm10:
        return "BAD!!!"
    else:
        return "GOOD"

# Check the air quality
air_quality = check_air_quality(pm1_avg, pm25_avg, pm10_avg)

# Print the air quality
print("The air quality in Beirut on 06/10/2023 is:", air_quality)
```

```
PM1 exceeds WHO standards by: 5.664014652493115 µg/m³
PM10 exceeds WHO standards by: 13.315974328566284 µg/m³
PM2.5 exceeds WHO standards by: 24.993888105668287 µg/m³
The air quality in Beirut on 06/10/2023 is: BAD!!!
```