DATA PREPROCESSING IN PYTHON

Steps of preprocessing of data

Step-1: IMPORT NECCESSSARY LIBRARY

Step-2: READ DATASET

Step-3: SANITY CHECK OF DATA

Step-4: EXPOLATORY DATA ANALYSIS (EDA)

Step-5: MISSING VALUE TREATMENTS

Step-6: OUTLIERS TREATMENTS

Step-7: DUPLICATE VALUES AND GARBAGE VALUES TREATMENT

STEP-1 IMPORT NECCESSSARY LIBRARY

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

STEP-2 READ DATASET

```
df=pd.read_csv('D:\\Air Quality Data\\Air_Cleaned_Processed.csv')

df.head() #It will show top 5 data
```

```
        City        Date       PM2.5        PM10         NO2
SO2  \
0  Allahabad  2013-01-01   72.190988   28.038859   74.466703
90.101024
1     Nagpur  2013-01-02  156.162525  109.705592   79.357025
62.660190
2   Vadodara  2013-01-03   37.035248  320.996249   19.526083
30.341156
3     Nashik  2013-01-04  222.000056  217.004109   85.572290
23.413217
4   Srinagar  2013-01-05  259.690694   47.634851  135.613423
```

```
32.026520

          CO          O3  Temperature   Humidity         AQI  \
0  2.737498  22.832620    23.538722  91.462662  139.203717
1  8.298729  87.444488    31.233321  55.057146  327.985193
2  0.268302  14.381349    39.919461  86.670590  270.996249
3  1.372084  25.333474    39.880965  76.203614  378.511671
4  6.027527  95.918899    24.098538  75.658559  404.455336

                      AQI Category  PM2.5/PM10 Ratio
0  Unhealthy for Sensitive Groups          1.500000
1                       Hazardous          1.423469
2                  Very Unhealthy          0.115376
3                       Hazardous          1.023022
4                       Hazardous          1.500000

df.tail() #It will show bottom 5 data

            City        Date       PM2.5         PM10          NO2
SO2  \
3995       Kota  2023-12-10  216.579951   50.677063   74.476868
99.373330
3996     Ranchi  2023-12-11   11.870857  239.002070   39.937385
5.815086
3997     Howrah  2023-12-12  189.599794  312.727515   97.029882
18.905951
3998  Hyderabad  2023-12-13  173.093545   53.921186  130.987842
97.598887
3999     Jaipur  2023-12-14  275.059510  273.342239  118.910881
82.707042

            CO         O3  Temperature   Humidity         AQI AQI
Category  \
3995  0.139059  57.206447    40.822590  77.312612  500.000000
Hazardous
3996  6.515407  79.966366    15.951747  80.208933  192.692651
Unhealthy
3997  7.323106  99.748538    10.104865  80.187652  353.646353
Hazardous
3998  2.455181  44.449088    20.986299  50.480373  340.978767
Hazardous
3999  7.439409  79.277567    31.209837  28.386461  410.565829
Hazardous

      PM2.5/PM10 Ratio
3995          1.500000
3996          0.049668
3997          0.606278
3998          1.500000
3999          1.006282
```

STEP-3 SANITY CHECK OF DATA

```
#SHAPE
df.shape
```

```
(4000, 13)
```

```
#Info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   City              4000 non-null   object 
 1   Date              4000 non-null   object 
 2   PM2.5             4000 non-null   float64
 3   PM10              4000 non-null   float64
 4   NO2               4000 non-null   float64
 5   SO2               4000 non-null   float64
 6   CO                4000 non-null   float64
 7   O3                4000 non-null   float64
 8   Temperature       4000 non-null   float64
 9   Humidity          4000 non-null   float64
 10  AQI               4000 non-null   float64
 11  AQI Category      4000 non-null   object 
 12  PM2.5/PM10 Ratio  4000 non-null   float64
dtypes: float64(10), object(3)
memory usage: 406.4+ KB
```

```
#Finding Missing Value
df.isnull().sum()
```

```
City                0
Date                0
PM2.5               0
PM10                0
NO2                 0
SO2                 0
CO                  0
O3                  0
Temperature         0
Humidity            0
AQI                 0
AQI Category        0
PM2.5/PM10 Ratio    0
dtype: int64
```

```
#Finding percentage of missing values
df.isnull().sum()/df.shape[0]*100
```

```
City                0.0
Date                0.0
PM2.5               0.0
PM10                0.0
NO2                 0.0
SO2                 0.0
CO                  0.0
O3                  0.0
Temperature         0.0
Humidity            0.0
AQI                 0.0
AQI Category        0.0
PM2.5/PM10 Ratio    0.0
dtype: float64
```

```
#Finding Duplicates
df.duplicated().sum()
```

```
0
```

```
#FIXING CITY NAME ISSUE
df["City"] = df["City"].replace({"LucknowUntitled document":
"Lucknow"})

# Standardize AQI Category Formatting
df["AQI Category"] = df["AQI Category"].str.strip().str.title()
#.str.strip() removes leading/trailing spaces.
#.str.title() ensures consistent casing (e.g., "very unhealthy" →
"Very Unhealthy").

#Date formatting
df["Date"] = pd.to_datetime(df["Date"], errors="coerce")
#Converts Date to a datetime format.
#If there are invalid dates, they become NaT (Not a Time), making them
easy to identify

#Identifying Garbage Value
#Garbage values are always present in the form of object data type
for i in df.select_dtypes(include="object").columns:
    print(df[i].value_counts())
    print("***"*10)
```

```
City
Chennai          116
Bangalore        114
Allahabad        111
Jabalpur         110
Kanpur           109
Chandigarh       109
Dhanbad          109
Raipur           109
```

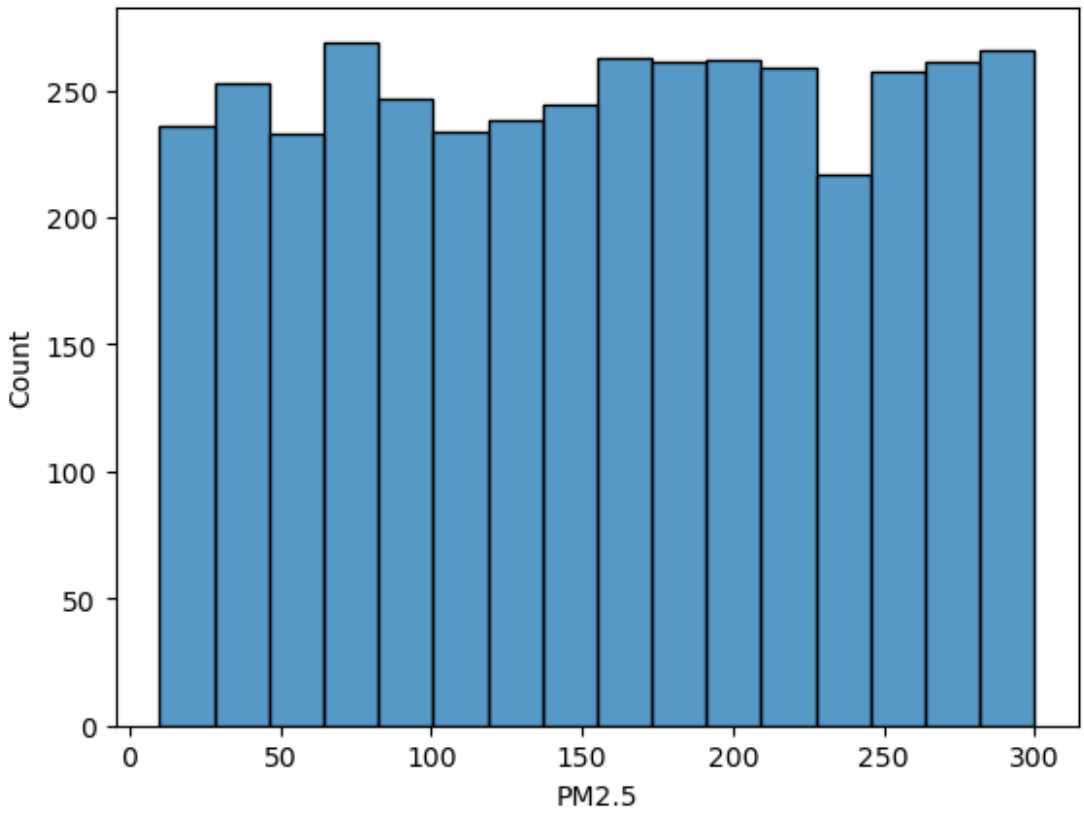
```
Aurangabad        108
Solapur           108
Nashik            105
Bhopal            104
Mumbai            104
Varanasi          104
Surat             103
Jaipur            103
Vadodara          103
Vijayawada        103
Ludhiana          102
Patna             102
Hyderabad         100
Nagpur            100
Lucknow            99
Amritsar           99
Pune               99
Ahmedabad          98
Srinagar           98
Delhi              98
Agra               97
Ranchi             94
Kota               94
Howrah             93
Visakhapatnam      91
Jodhpur            90
Navi Mumbai        88
Kolkata            87
Gwalior            87
Madurai            86
Coimbatore         83
Guwahati           83
Name: count, dtype: int64
******************************
AQI Category
Hazardous                          2827
Very Unhealthy                      581
Unhealthy                           431
Unhealthy For Sensitive Groups      153
Moderate                              8
Name: count, dtype: int64
******************************
```

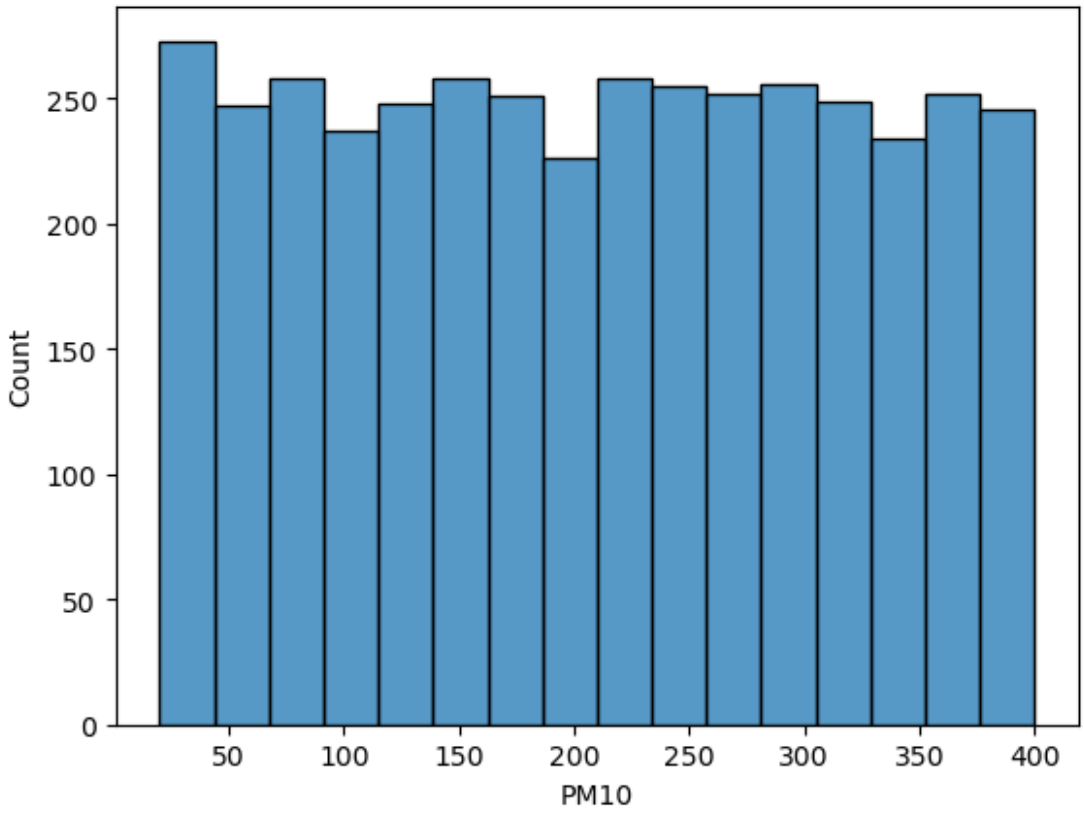
Step-4 EXPLORATORY DATA ANALYSIS (EDA)

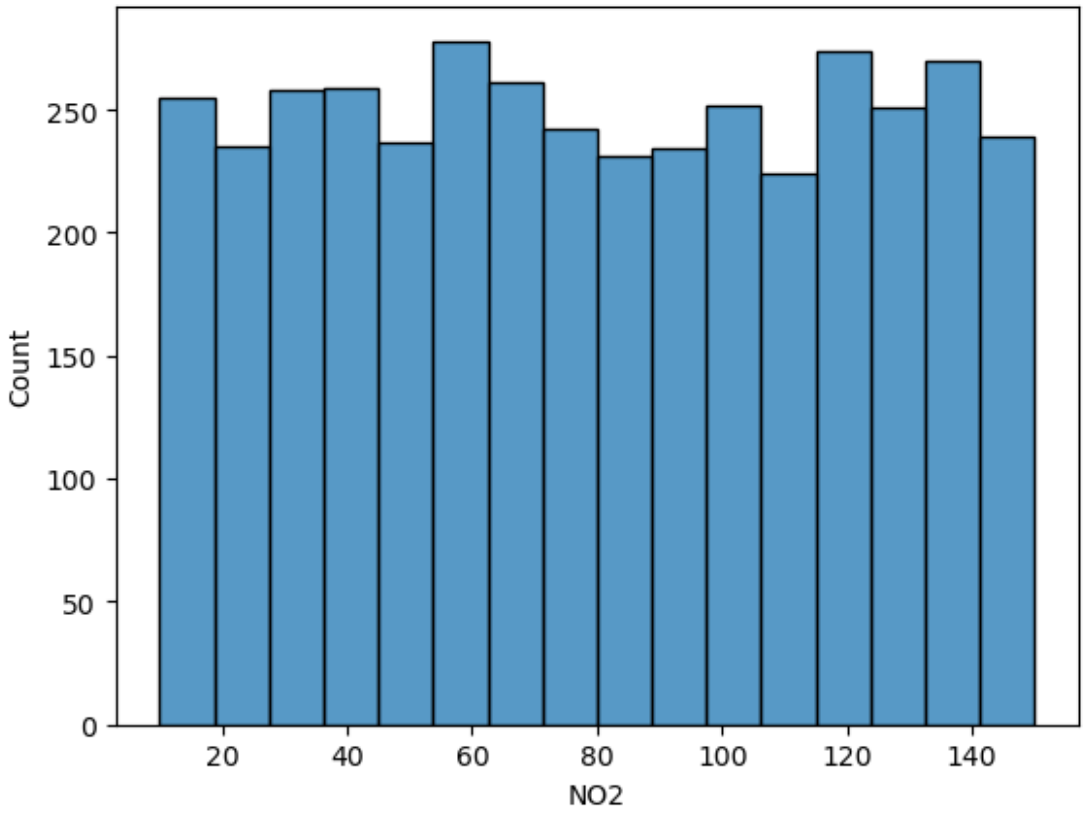
```
#Descriptive Statistics
# df.describe()
df.describe(include="object")
```

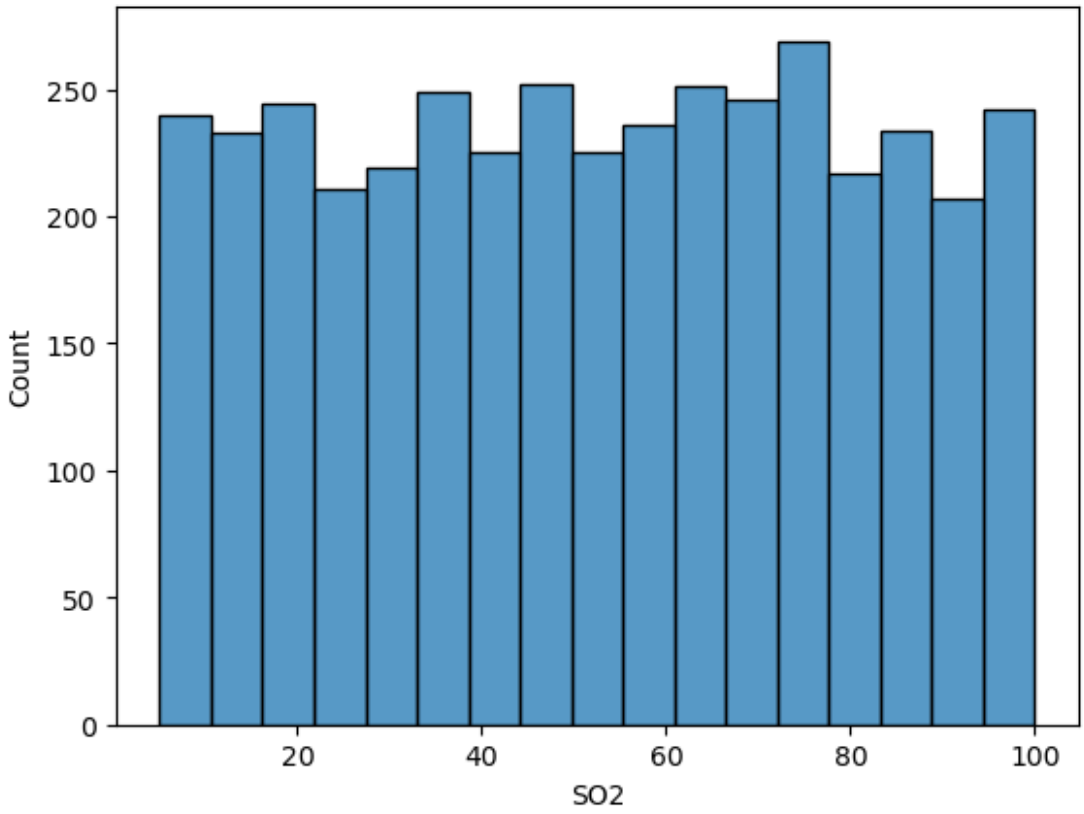
```
           City AQI Category
count      4000         4000
unique       40            5
top     Chennai    Hazardous
freq        116         2827
```

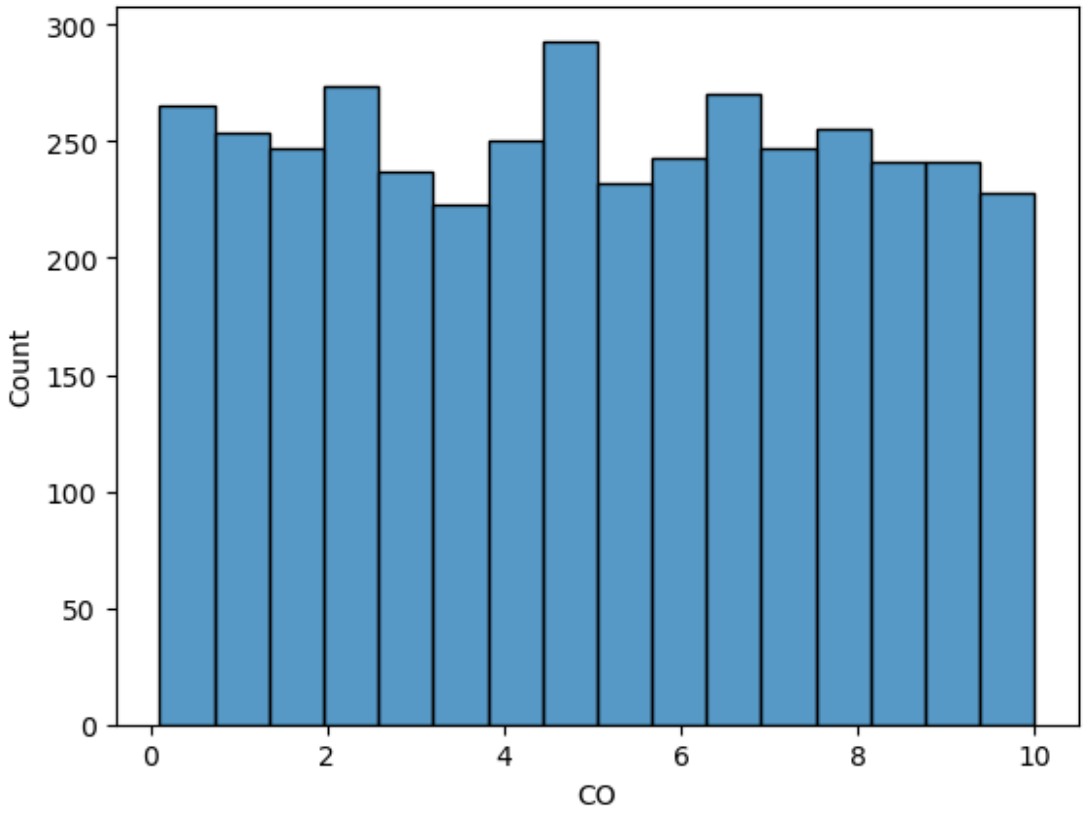
```
#Histogram to understand the distribution
import warnings
warnings.filterwarnings("ignore")
for i in df.select_dtypes(include="number").columns:
    sns.histplot(data=df,x=i)
    plt.show()
```

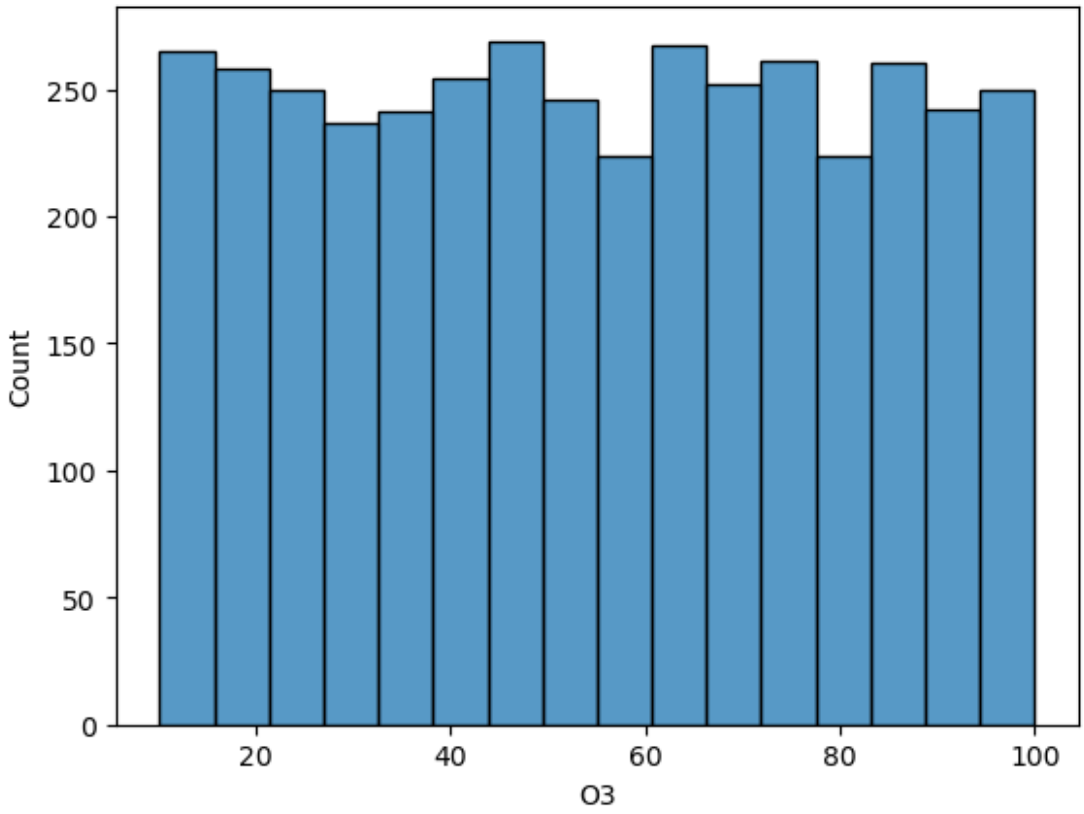


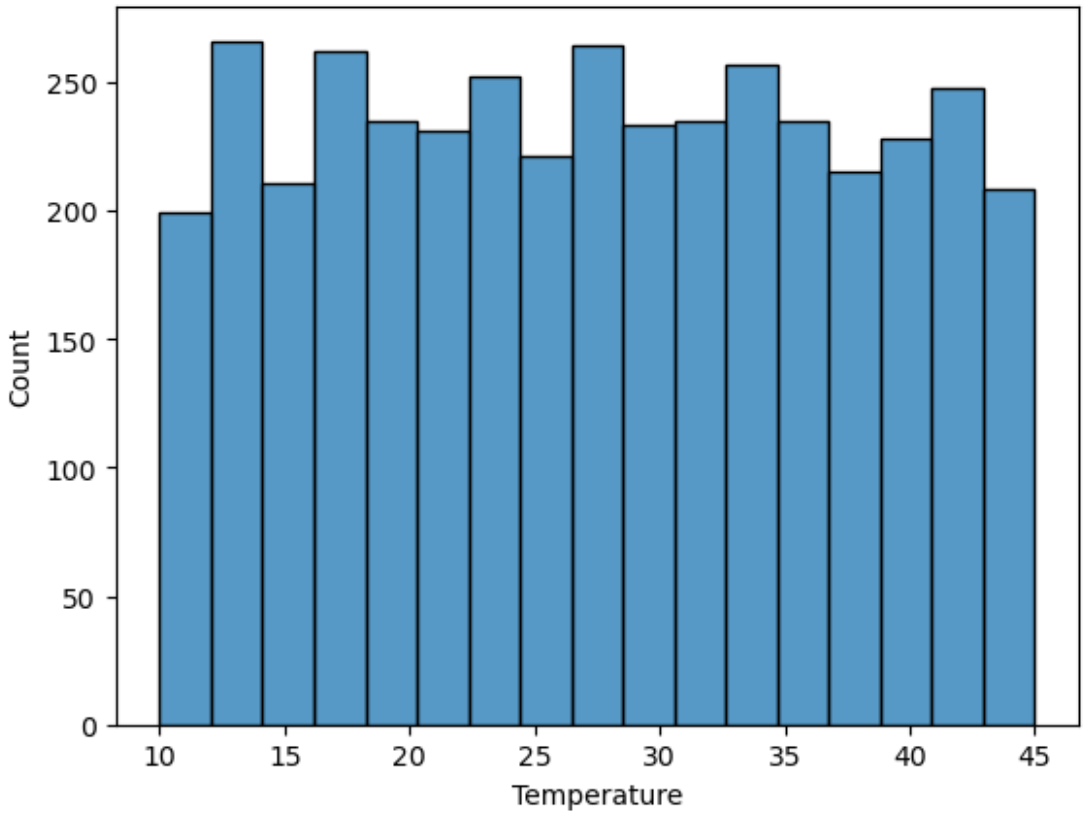

Count
0
50
100
150
200
250
50
100
150
200
250
300
350
400
PM10

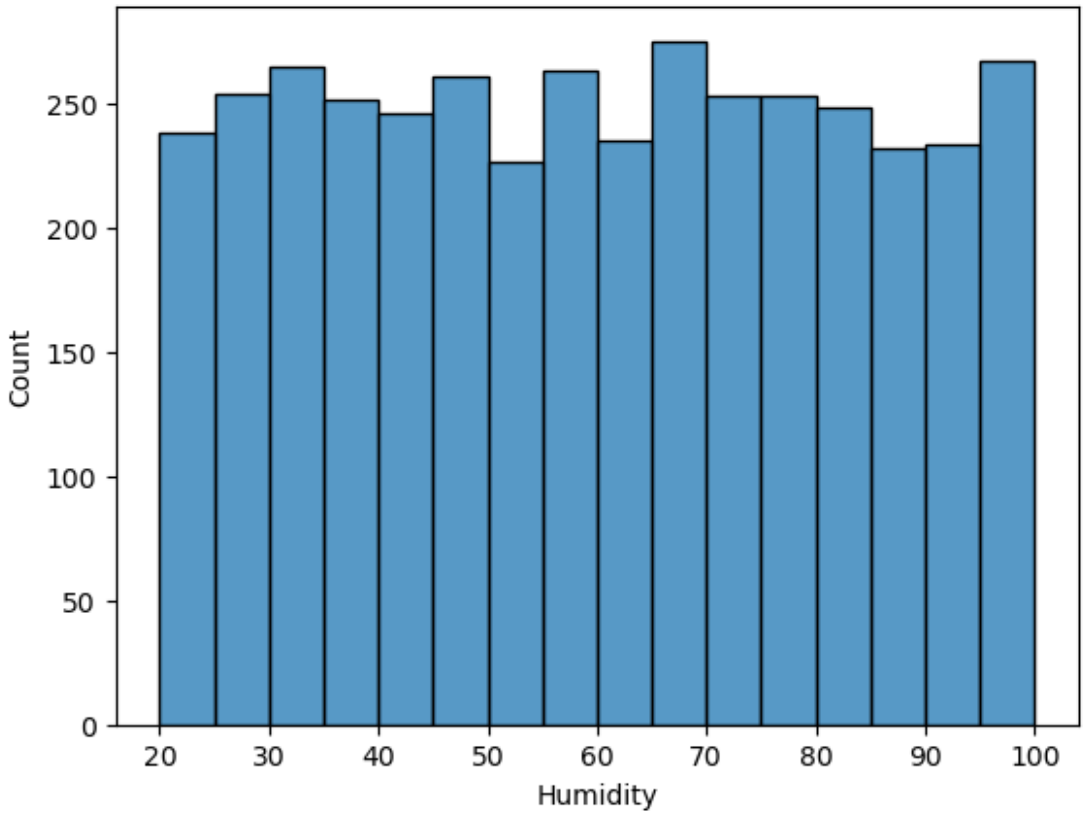

Count
0
50
100
150
200
250
20
40
60
80
100
120
140
NO2

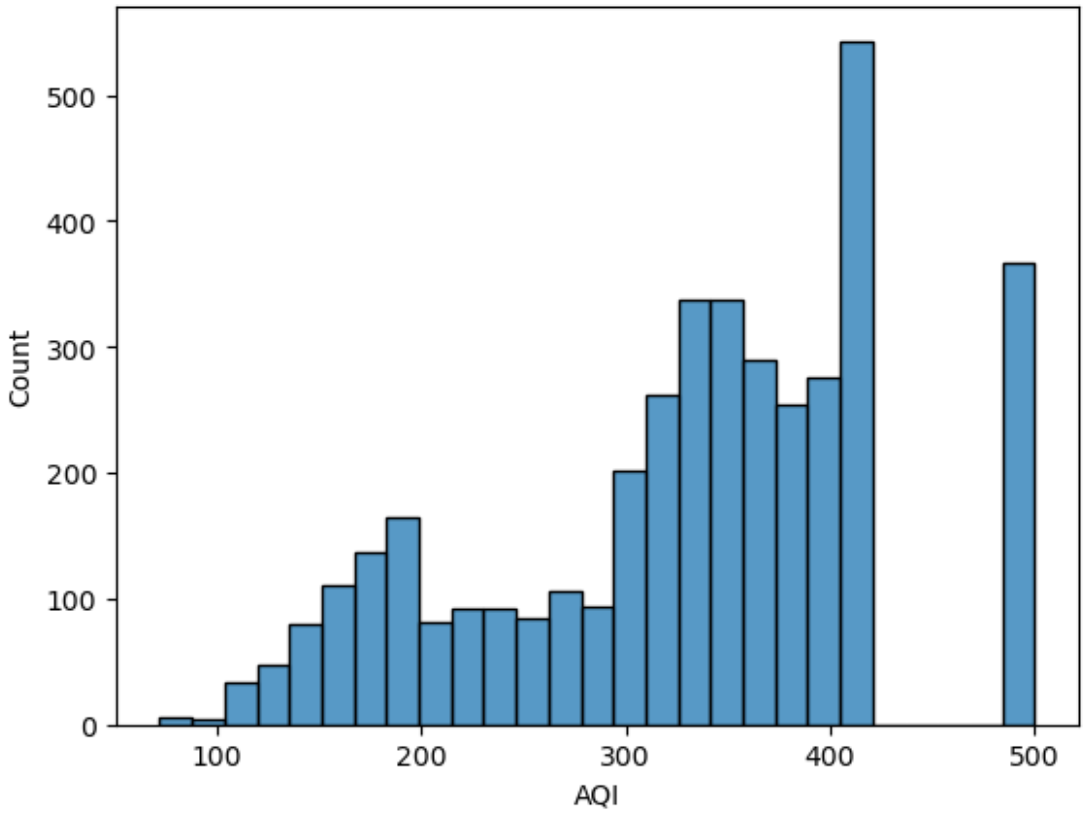

Count
250
200
150
100
50
0
20
40
60
80
100
SO2

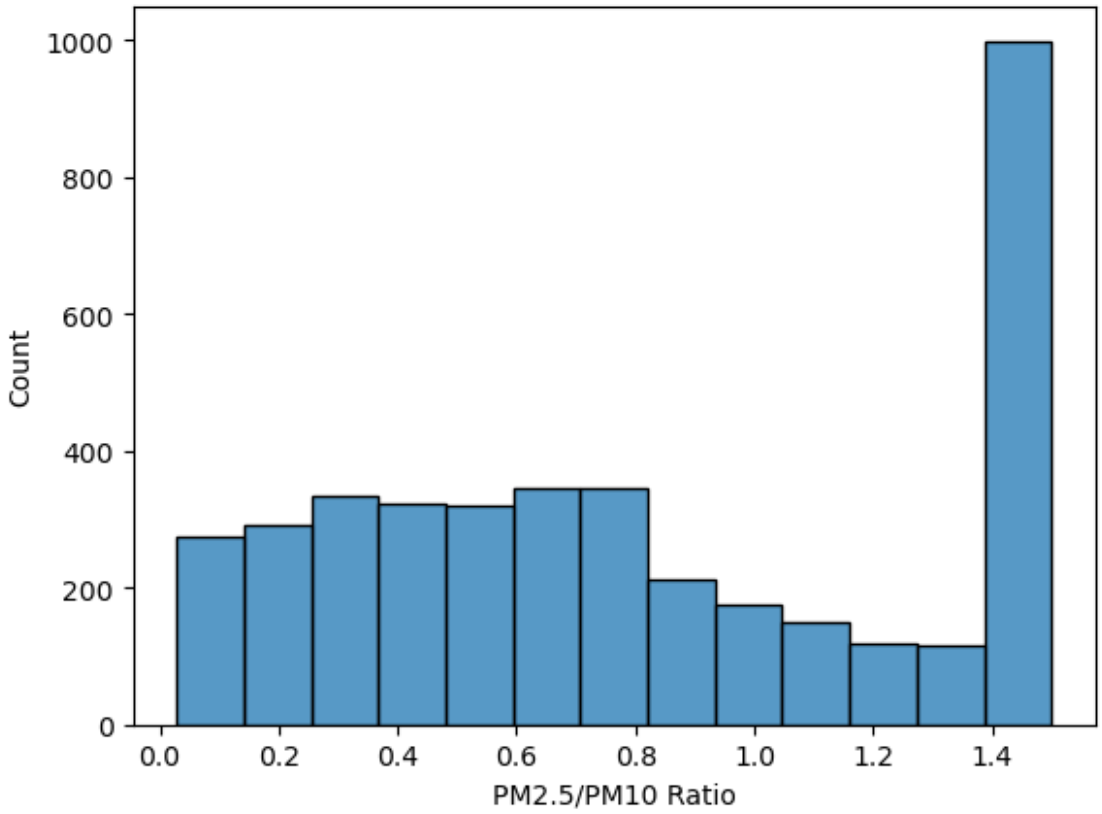

Count
300
250
200
150
100
50
0
0
2
4
6
8
10
CO


Count
250
200
150
100
50
0
20
40
60
80
100
O3

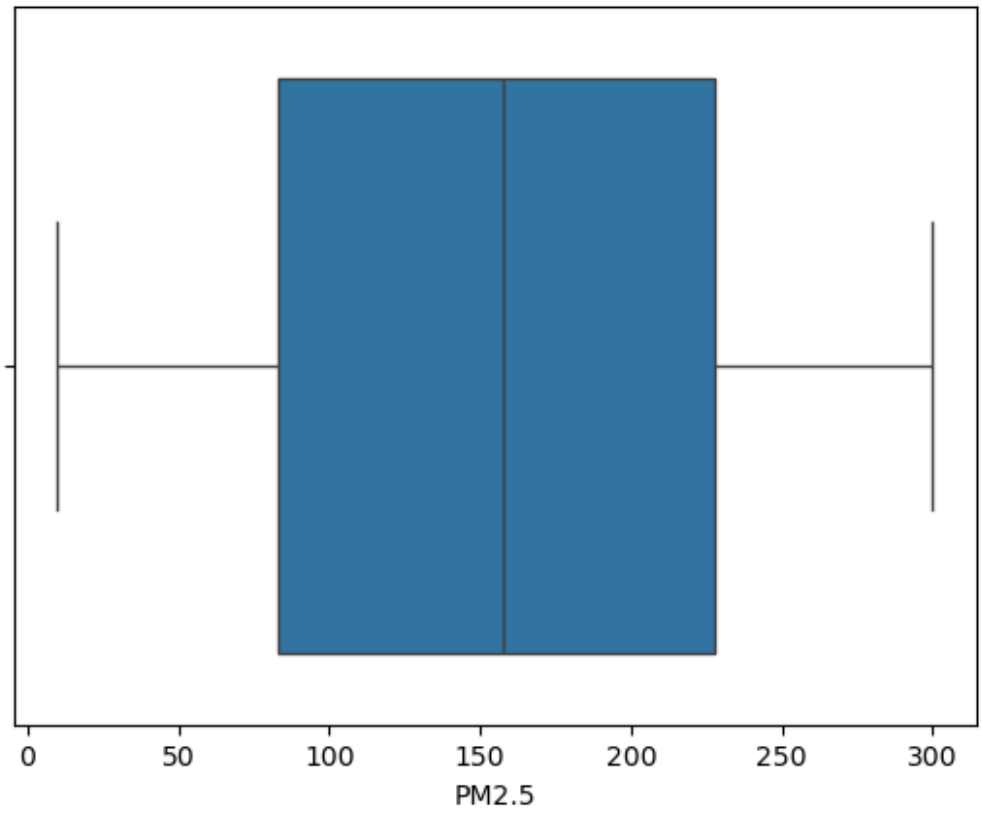

Count
250
200
150
100
50
0
10
15
20
25
30
35
40
45
Temperature

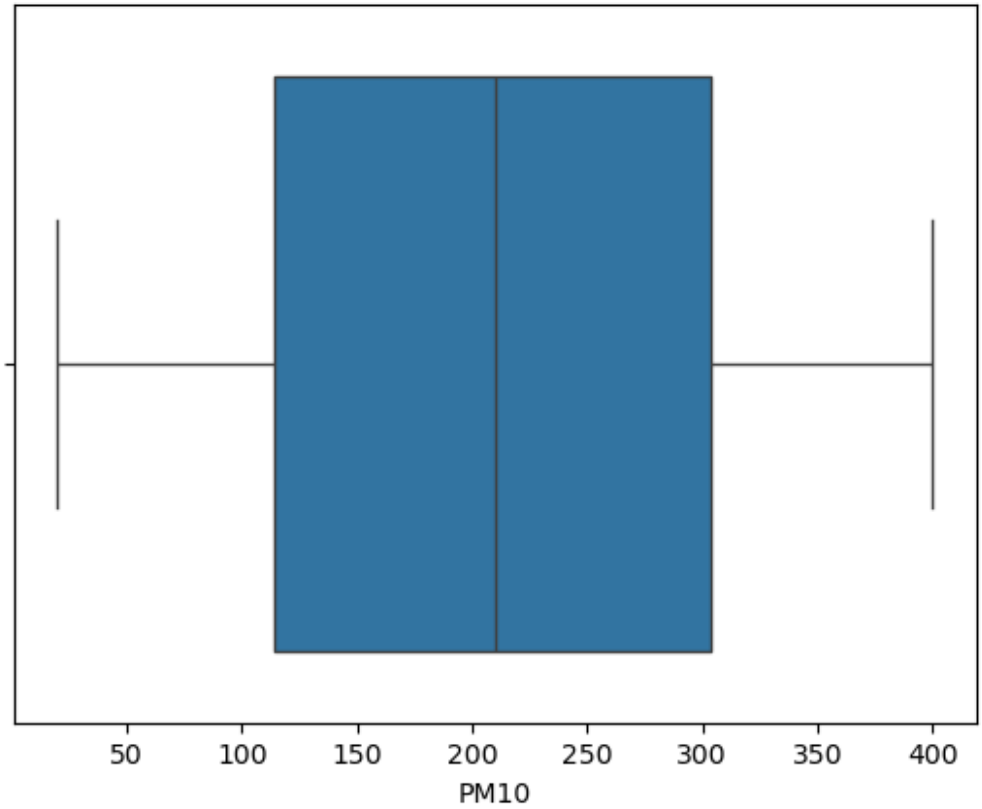

Count
0
50
100
150
200
250
20
30
40
50
60
70
80
90
100
Humidity

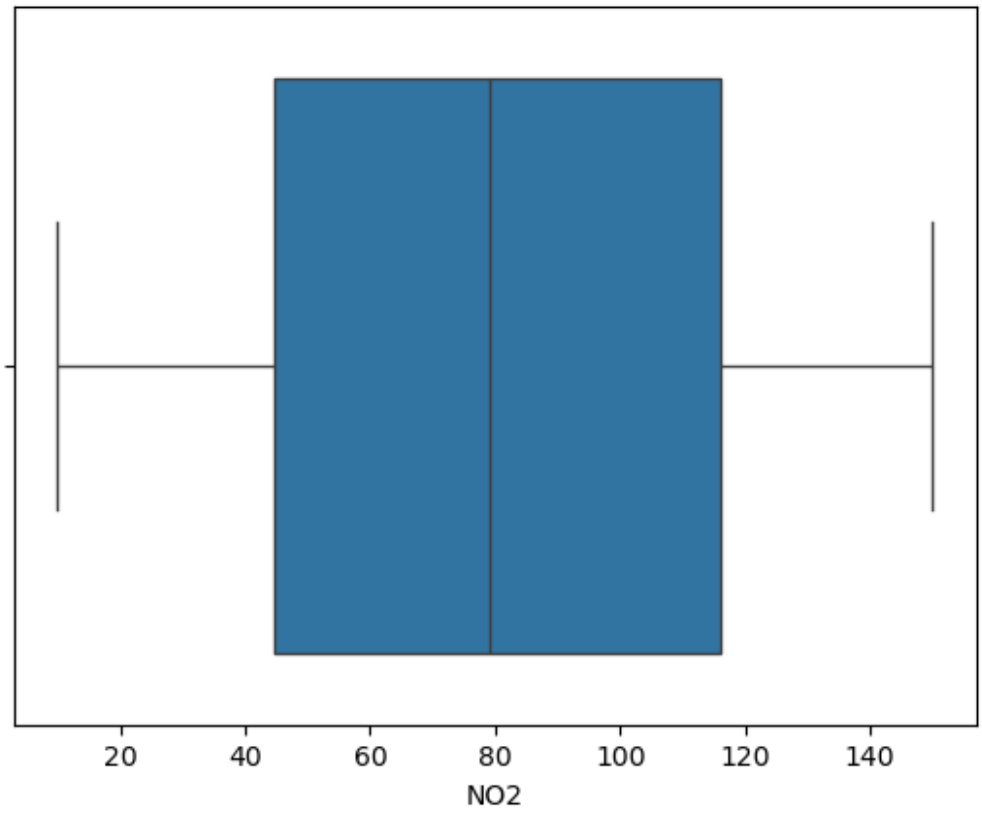

Count
0
100
200
300
400
500
100
200
300
400
500
AQI

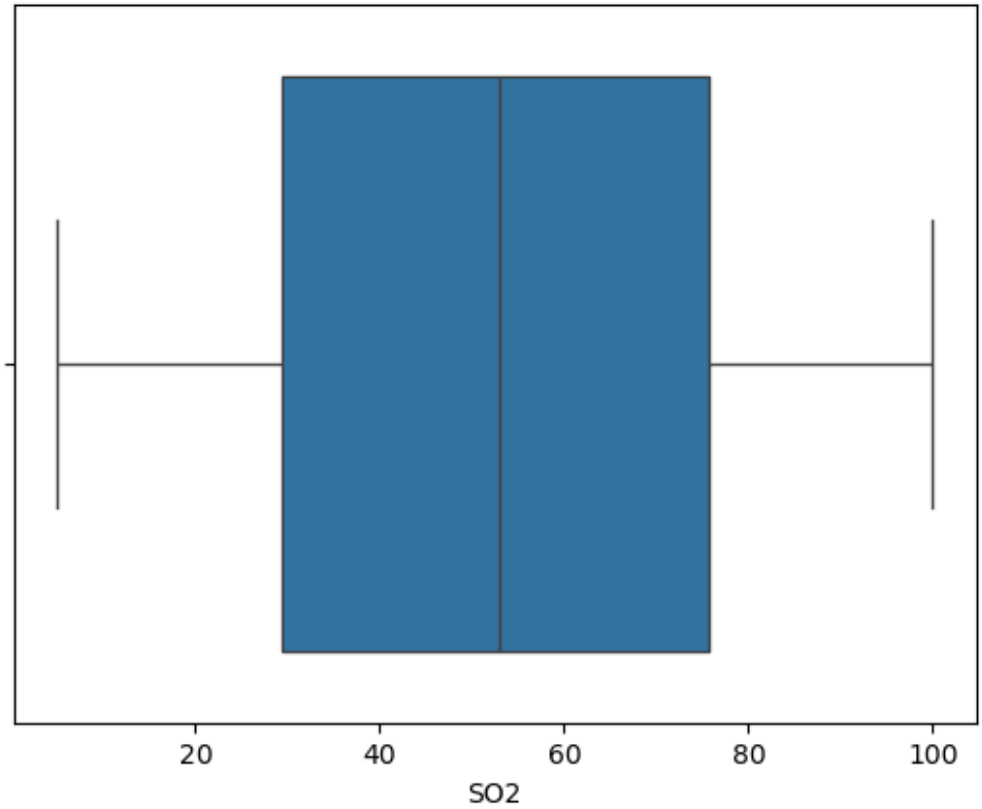


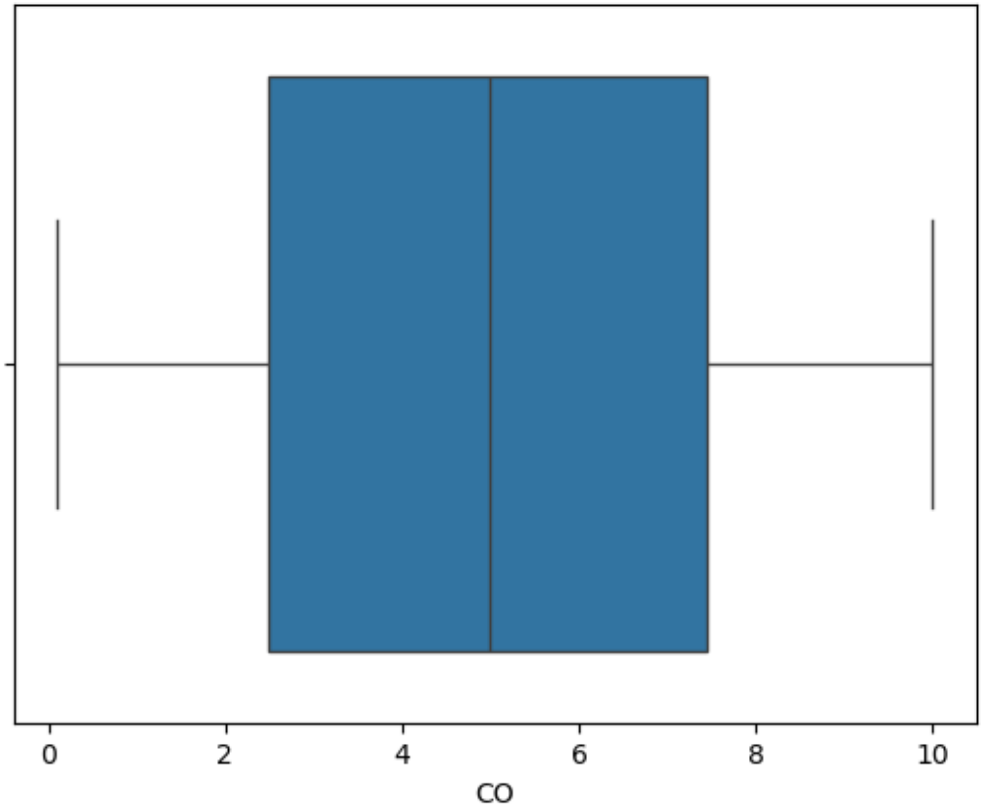

```
#Boxplot to indentify outliers
#There is no outliers
import warnings
warnings.filterwarnings("ignore")
for i in df.select_dtypes(include="number").columns:
    sns.boxplot(data=df,x=i)
    plt.show()
```

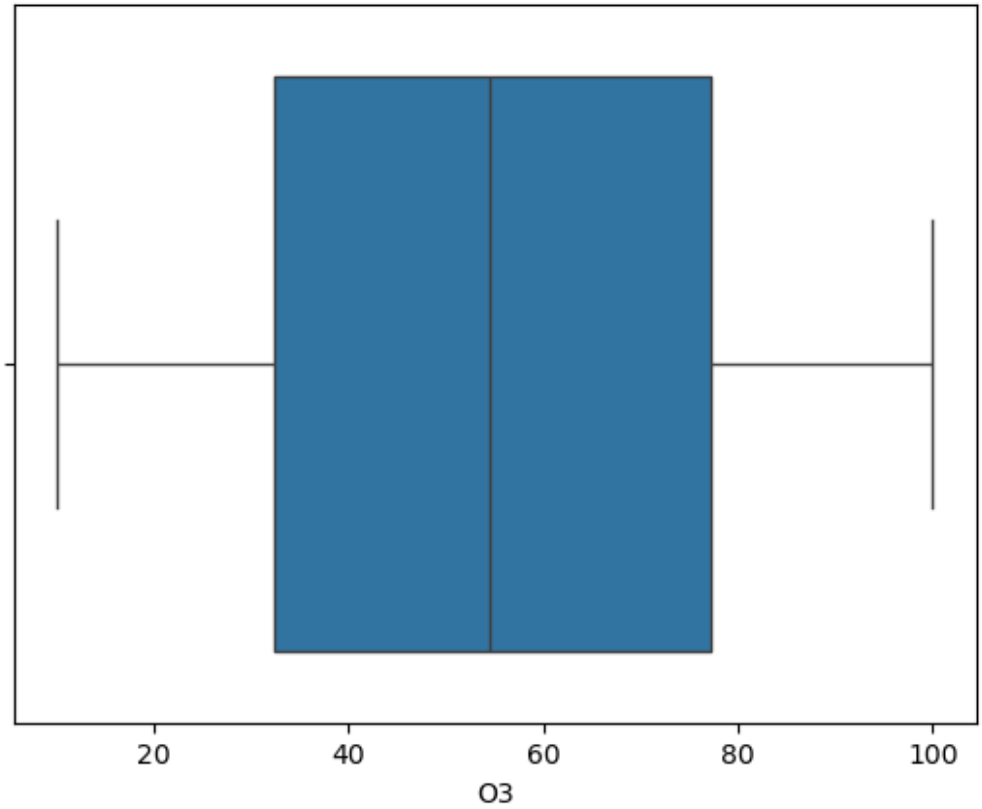

0
50
100
150
200
250
300
PM2.5

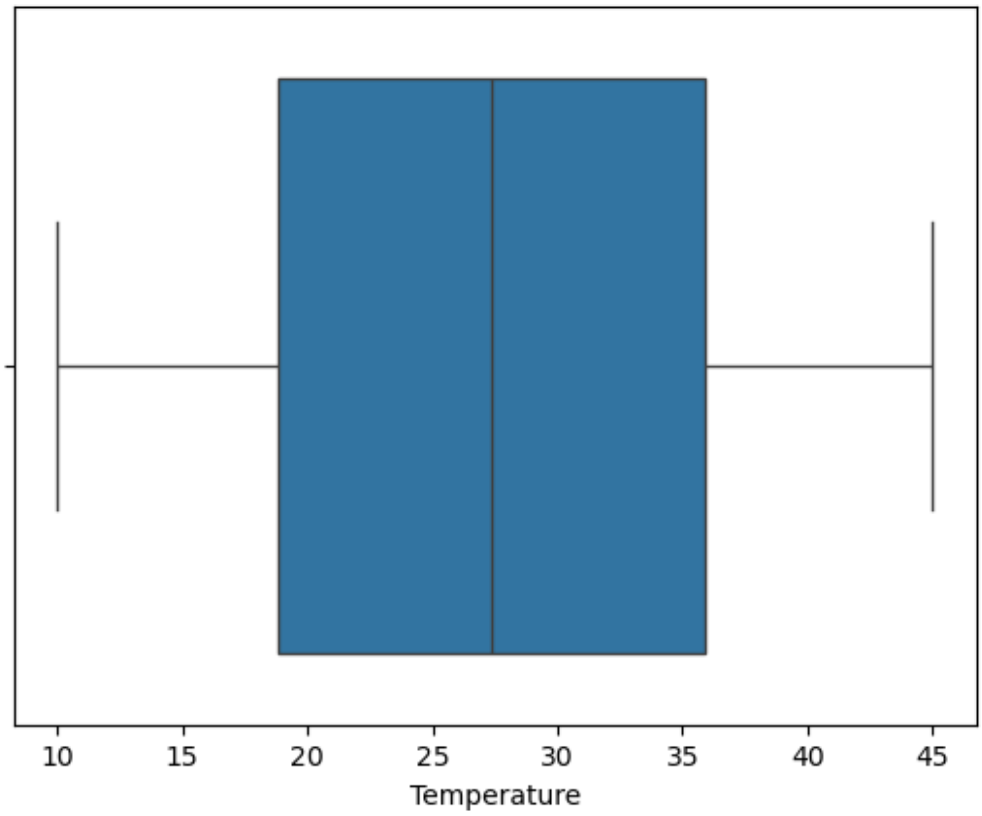

50
100
150
200
250
300
350
400
PM10

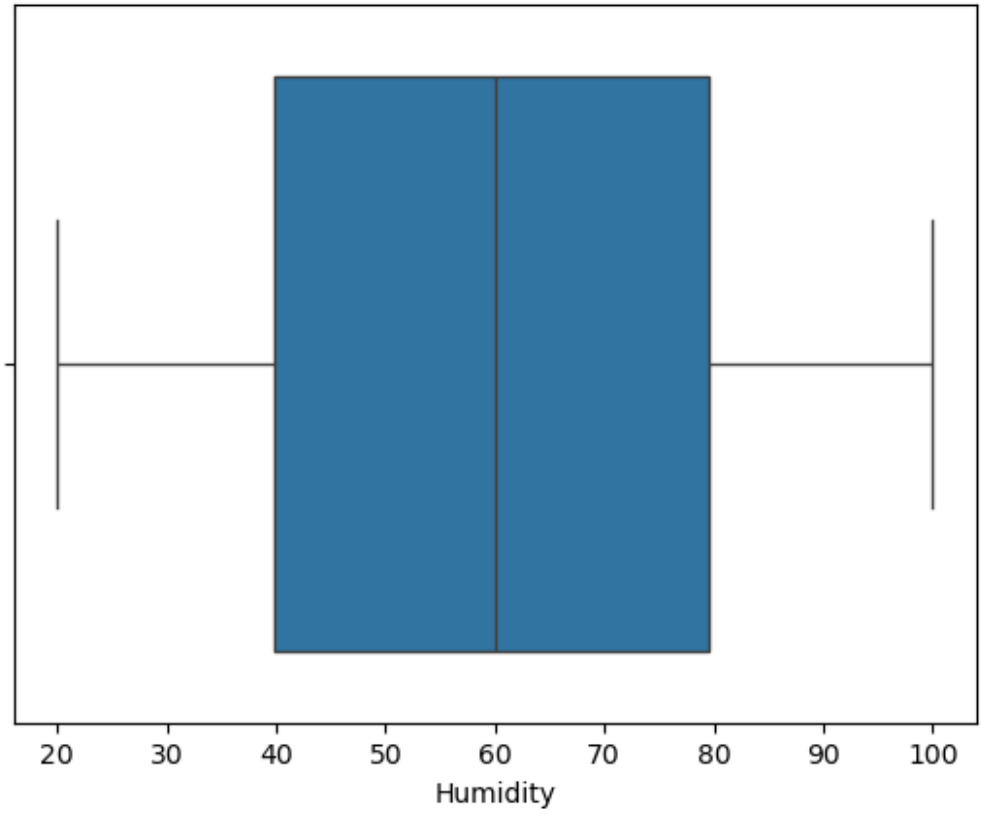

20
40
60
80
100
120
140
NO2

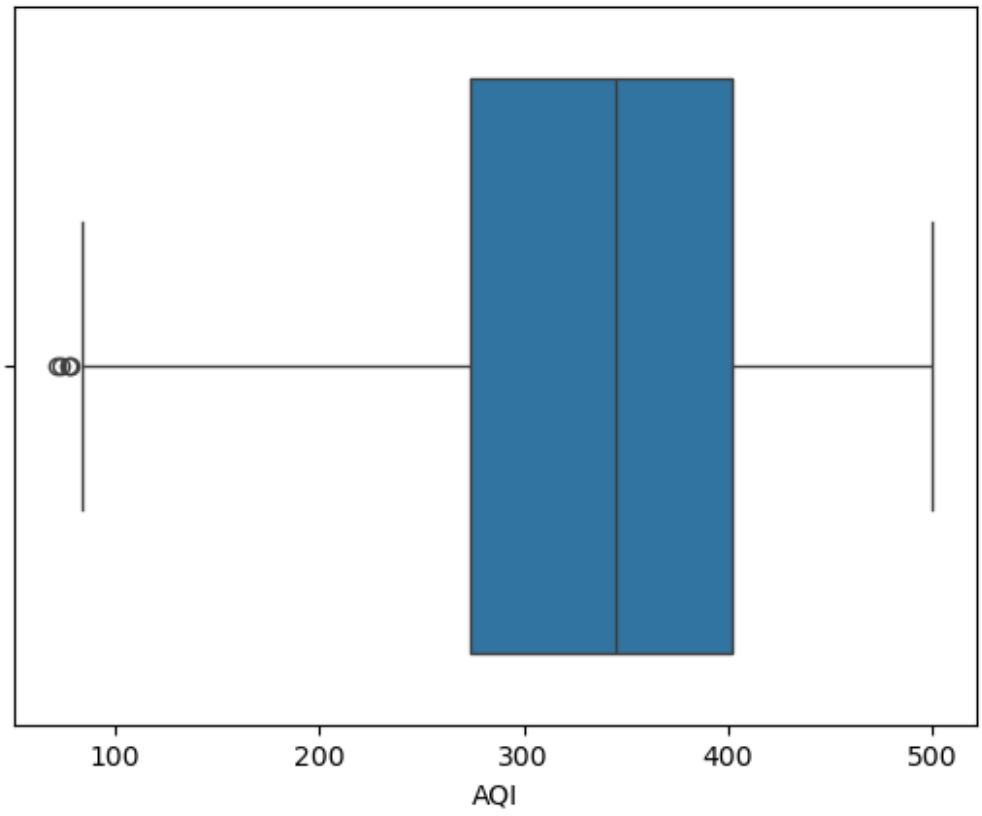

20
40
60
80
100
SO2

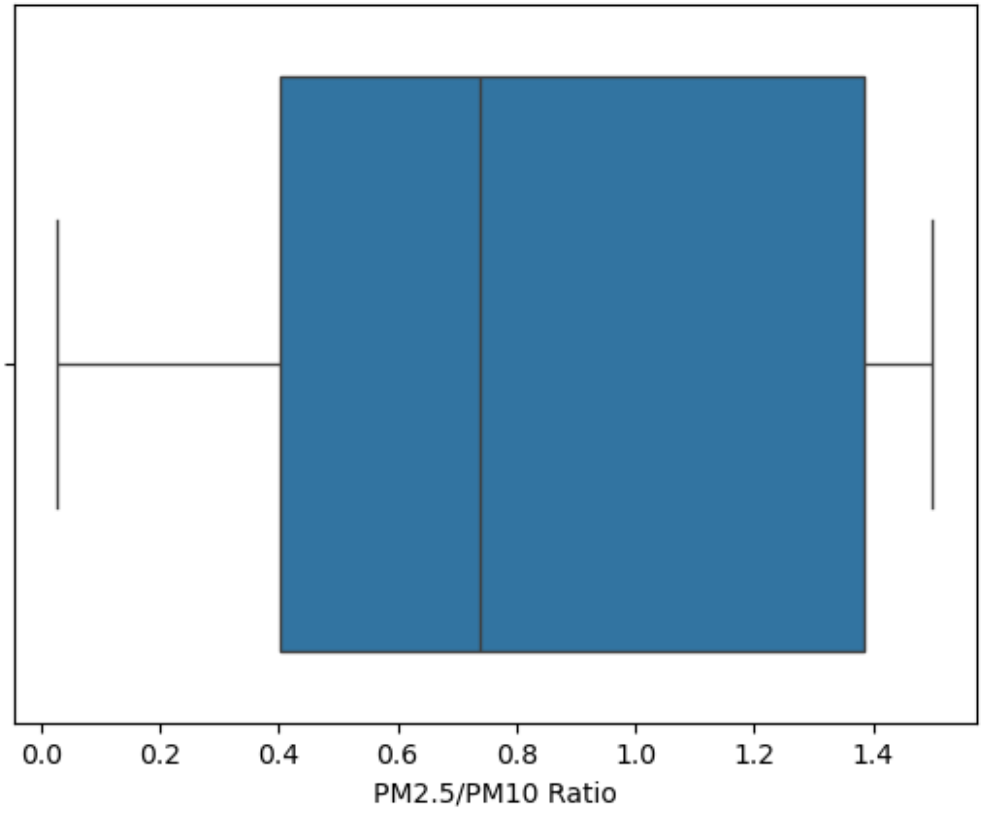

0
2
4
6
8
10
CO


20
40
60
80
100
O3

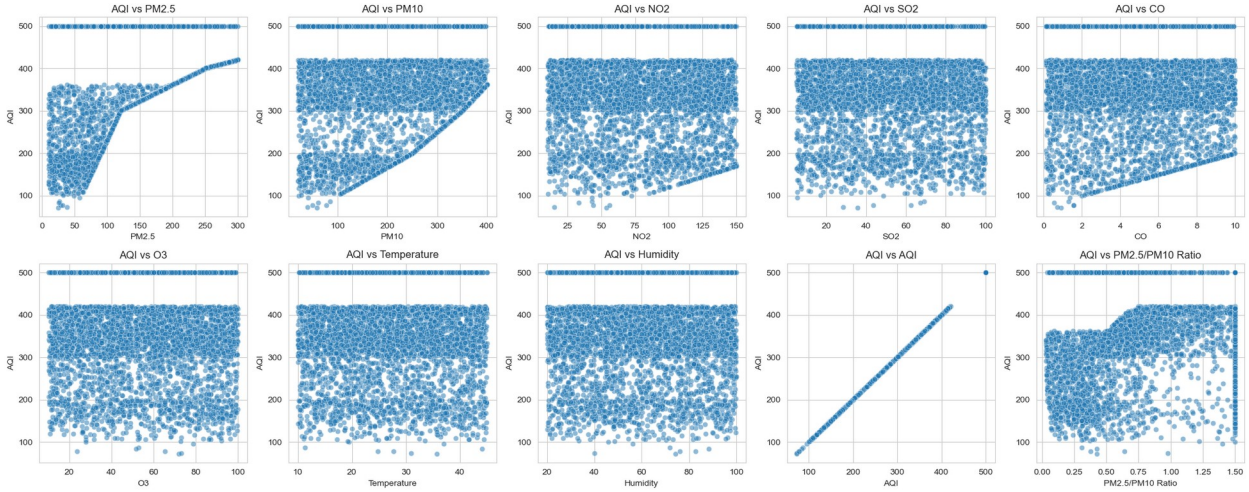

10
15
20
25
30
35
40
45
Temperature


20
30
40
50
60
70
80
90
100
Humidity

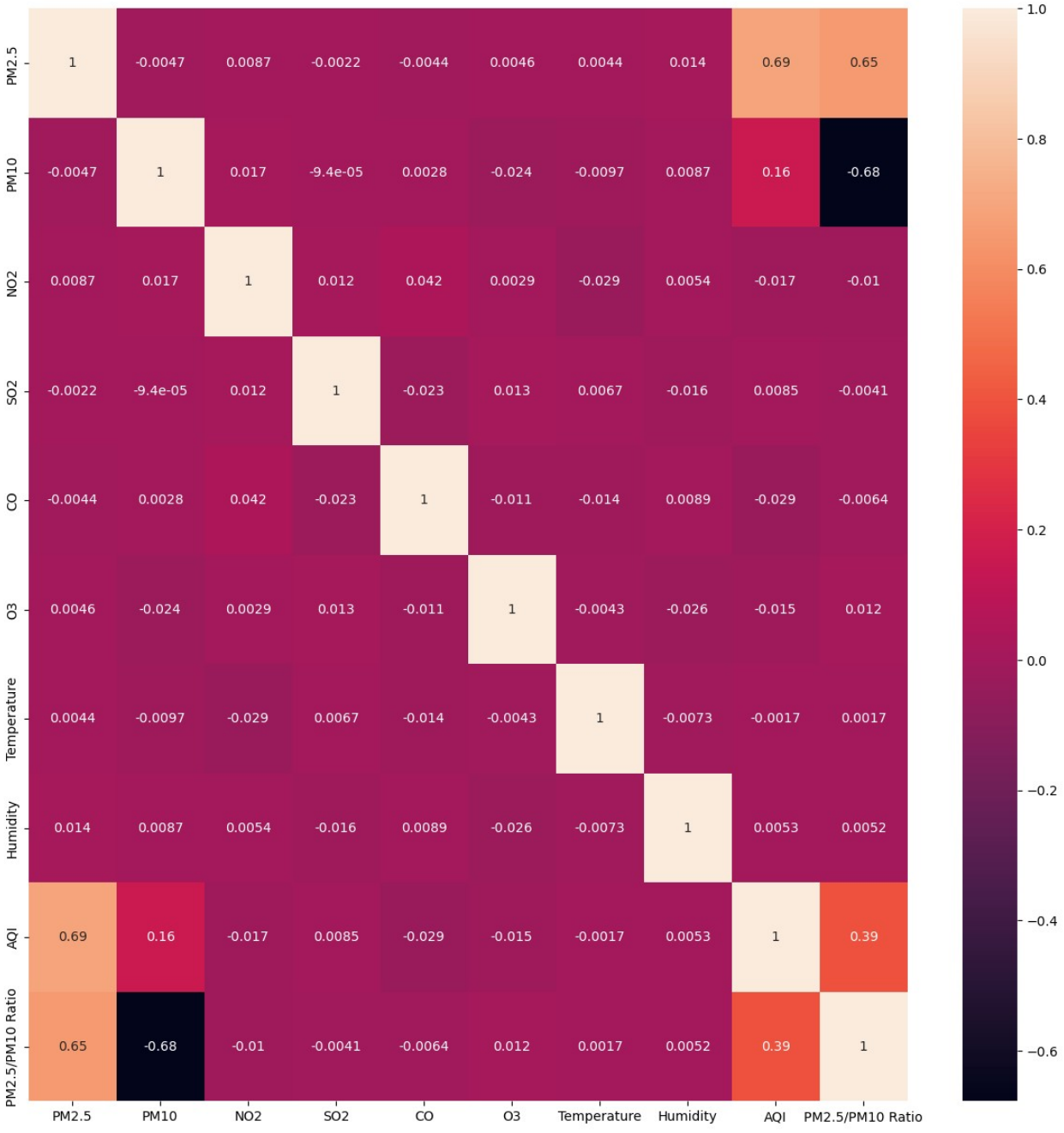

100
200
300
400
500
AQI


0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
PM2.5/PM10 Ratio

```
# #Scatter Plot to understand the relationship
# df.select_dtypes(include="number").columns It is used to get all the
columns

sns.set_style("whitegrid")  # Set a clean theme

features = ['PM2.5', 'PM10', 'NO2', 'SO2', 'CO', 'O3', 'Temperature',
            'Humidity', 'AQI', 'PM2.5/PM10 Ratio']

fig, axes = plt.subplots(nrows=2, ncols=5, figsize=(20, 8))  # Create
subplots
axes = axes.flatten()  # Flatten the 2D array for easy iteration

for i, feature in enumerate(features):
    sns.scatterplot(data=df, x=feature, y='AQI', alpha=0.5,
ax=axes[i])
    axes[i].set_title(f'AQI vs {feature}')
    axes[i].set_xlabel(feature)
    axes[i].set_ylabel('AQI')

plt.tight_layout()  # Adjust layout to avoid overlap
plt.show()
```




```
#correlation with heatmap to interpret the relation and
multicolliniarity
s=df.select_dtypes(include="number").corr()

plt.figure(figsize=(15,15))
sns.heatmap(s,annot=True)

<Axes: >
```


PM2.5 1 -0.0047 0.0087 -0.0022 -0.0044 0.0046 0.0044 0.014 0.69 0.65
PM10 -0.0047 1 0.017 -9.4e-05 0.0028 -0.024 -0.0097 0.0087 0.16 -0.68
NO2 0.0087 0.017 1 0.012 0.042 0.0029 -0.029 0.0054 -0.017 -0.01
SO2 -0.0022 -9.4e-05 0.012 1 -0.023 0.013 0.0067 -0.016 0.0085 -0.0041
CO -0.0044 0.0028 0.042 -0.023 1 -0.011 -0.014 0.0089 -0.029 -0.0064
O3 0.0046 -0.024 0.0029 0.013 -0.011 1 -0.0043 -0.026 -0.015 0.012
Temperature 0.0044 -0.0097 -0.029 0.0067 -0.014 -0.0043 1 -0.0073 -0.0017 0.0017
Humidity 0.014 0.0087 0.0054 -0.016 0.0089 -0.026 -0.0073 1 0.0053 0.0052
AQI 0.69 0.16 -0.017 0.0085 -0.029 -0.015 -0.0017 0.0053 1 0.39
PM2.5/PM10 Ratio 0.65 -0.68 -0.01 -0.0041 -0.0064 0.012 0.0017 0.0052 0.39 1
PM2.5 PM10 NO2 SO2 CO O3 Temperature Humidity AQI PM2.5/PM10 Ratio
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4 −0.6

STEP-5 MISSING VALUE TREATMENTS

Choose the method of imputing missing value

Like mean,meadian,mode or KNNIputer

There is no missing value in this dataset

STEP-6 OUTLIER TREATMENT

There is no outlier in this dataset

STEP-7 DUPLICATES AND GARBAGE VALUE TREATMENT

There is no duolicate value in this data but there are garbage values present in this dataset

To remove duplicates we will use:

df.drop_duplicates() function

To remove Garbage value , we will use the mean,median,mode function to change the value