PHASE-5

DATA ANALYST WITH COGNOS

PROJECT- Air Quality Analysis in Tamil Nadu using Machine Learning

610921104344

SIVAPERUMALRAJ G

Project:

Air Quality Analysis

Problem Statement:

The project aims to analyze and visualize air quality data from monitoring stations in

Tamil Nadu. The objective is to gain insights into air pollution trends, identify areas

with high pollution levels, and develop a predictive model to estimate RSPM/PM10 levels

based on SO2 and NO2 levels. This project involves defining objectives, designing the

analysis approach, selecting visualization techniques, and creating a predictive model

using Python and relevant libraries.

Approach:

Design Thinking:

Project Objectives: Define objectives such as analyzing air quality trends,

identifying pollution hotspots, and building a predictive model for

RSPM/PM10 levels.

Analysis Approach: Plan the steps to load, preprocess, analyze, and visualize

the air quality data.

Visualization Selection: Determine visualization techniques (e.g., line charts, heatmaps)

to effectively represent air quality trends and pollution levels.

Checking the requirements:

First we install the required software and install the modules required for the

project and then we set up the test environment by changing the path variables

and we launch the application.

Data collection and warehousing:

We collect various data in the form of an excel spreadsheet and convert it into

a CSV file and save it in the same folder where we are going to implement the algorithm.

We have several data on:

Stn Code : Contains pincode of the city

Sampling Date : contains date of sampling

State : contains the name of the state

City/Town/Village/Area : contains the name of the City/Town/Village/Area

Location of Monitoring Station : contains the place where the location of monitoring station is located.

Agency : contains the state/central pollution control board details

Type of Location : states whether the area is industrial/rural.

SO2 : Sulphur di oxide content

NO2 : Nitrogen di oxide content

RSPM/PM : Respirable Suspended Particulate Matter measured.

PM2.5 : It represents the value of particulate matter measured.

Approach for making design:

Data Mining:

Data mining or Knowledge Discovery (KD) is used to read and analyze large

datasets and then finding/extracting patterns from the data. It is used for predicting

the future trends or forecast patterns over a period. Data mining algorithms are usually

based on wellknown mathematical algorithms and techniques. There are two types of data

mining learning algorithms: 1) Supervised algorithms and 2) Unsupervised algorithms.

We are going to make optimal use of these to train our machine learning model for better

prediction. The dataset is provided in the Government website.

Dataset link : https://tn.data.gov.in/resource/location-wise-daily-ambient-air-quality-tamil-nadu-year-2014

Unsupervised learning algorithm:

The Unsupervised algorithm is the process in which the training dataset contains only the

input set and not the corresponding target vectors. The main criterion is to find groups

or patterns of similar examples within the dataset, called as clustering.

Supervised learning algorithm:

The Supervised algorithm is the process in which the training data comprises of both the

training and the corresponding output target vectors. In this project, a supervised

learning algorithm called Artificial Neural Network (ANN) has been used for training,

validation and testing the dataset. In addition, to the ANN, a Multiple Linear Regression (MLR)

model has been used for comparing the performance against the ANN. The below section introduces

the processes of Artificial Neural Network (ANN) and Multiple Linear Regression (MLR).

Test execution:

We use the pandas,scikit,matplotlib modules in python in order to implement the supervised

machine learning algorithm and to visualize it in a realistic manner.

Conclusion:

These steps are considered optimal for getting the desired output.

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn import tree

from sklearn.metrics import confusion_matrix

from sklearn.metrics import accuracy_score

from sklearn.metrics import precision_score

from sklearn.metrics import recall_score

import seaborn as sns

import scipy as sc

data=pd.read_csv("C:\DAC\DAC_PHASE1\DAC_Dataset.csv")

print(data.head())
```

```
   Stn Code Sampling Date       State City/Town/Village/Area  \
0        38      01-02-14  Tamil Nadu                Chennai
1        38      01-07-14  Tamil Nadu                Chennai
2        38      21-01-14  Tamil Nadu                Chennai
3        38      23-01-14  Tamil Nadu                Chennai
4        38      28-01-14  Tamil Nadu                Chennai

                     Location of Monitoring Station  \
0  Kathivakkam, Municipal Kalyana Mandapam, Chennai
1  Kathivakkam, Municipal Kalyana Mandapam, Chennai
2  Kathivakkam, Municipal Kalyana Mandapam, Chennai
3  Kathivakkam, Municipal Kalyana Mandapam, Chennai
4  Kathivakkam, Municipal Kalyana Mandapam, Chennai

                                    Agency Type of Location   SO2
NO2  \
0  Tamilnadu State Pollution Control Board  Industrial Area  11.0
17.0
1  Tamilnadu State Pollution Control Board  Industrial Area  13.0
17.0
2  Tamilnadu State Pollution Control Board  Industrial Area  12.0
18.0
3  Tamilnadu State Pollution Control Board  Industrial Area  15.0
16.0
4  Tamilnadu State Pollution Control Board  Industrial Area  13.0
14.0
```

```
   RSPM/PM10  PM 2.5
0       55.0     NaN
1       45.0     NaN
2       50.0     NaN
3       46.0     NaN
4       42.0     NaN
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2879 entries, 0 to 2878
Data columns (total 11 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   Stn Code                        2879 non-null   int64  
 1   Sampling Date                   2879 non-null   object 
 2   State                           2879 non-null   object 
 3   City/Town/Village/Area          2879 non-null   object 
 4   Location of Monitoring Station  2879 non-null   object 
 5   Agency                          2879 non-null   object 
 6   Type of Location                2879 non-null   object 
 7   SO2                             2868 non-null   float64
 8   NO2                             2866 non-null   float64
 9   RSPM/PM10                       2875 non-null   float64
 10  PM 2.5                          0 non-null      float64
dtypes: float64(4), int64(1), object(6)
memory usage: 247.5+ KB
```

```
s=(data['SO2'])
```

```
np.mean(s)
```

```
11.503138075313808
```

```
s.std()
```

```
5.051702402147344
```

```
s.var()
```

```
25.519697159861238
```

```
s.skew()
```

```
0.5627115328978132
```

```
s.kurtosis()
```

```
2.2658770230801273
```

```
data.describe()
```

```
          Stn Code           SO2           NO2    RSPM/PM10  PM 2.5
count  2879.000000  2868.000000  2866.000000  2875.000000     0.0
mean    475.750261    11.503138    22.136776    62.494261     NaN
std     277.675577     5.051702     7.128694    31.368745     NaN
min      38.000000     2.000000     5.000000    12.000000     NaN
25%     238.000000     8.000000    17.000000    41.000000     NaN
50%     366.000000    12.000000    22.000000    55.000000     NaN
75%     764.000000    15.000000    25.000000    78.000000     NaN
max     773.000000    49.000000    71.000000   269.000000     NaN
```

```
print(data.columns)
```

```
Index(['Stn Code', 'Sampling Date', 'State', 'City/Town/Village/Area',
       'Location of Monitoring Station', 'Agency', 'Type of Location',
'SO2',
       'NO2', 'RSPM/PM10', 'PM 2.5'],
      dtype='object')
```

```
feature=data[['SO2','NO2']]
```

```
x=np.asarray(feature)
```

```
y=np.asarray(data['Stn Code'])
```

```
clf=tree.DecisionTreeClassifier()
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,rando
m_state=1)
```

```
clf.fit(x_train,y_train)
```

```
DecisionTreeClassifier()
```

```
y_predict2=clf.predict(x_test)
```

```
cm=confusion_matrix(y_test,y_predict2)
```

```
print(cm)
```

```
[[13  3  4  0  0  0  0  0  0  2  0  0  0  0  2  0  0  0  0  0  0  0  0
0
   0  0  0  0  0  0]
 [ 7  1  2  1  0  0  0  0  1  1  0  0  0  0  2  0  0  0  0  0  1  1  0
0
   0  0  0  0  4  0]
 [13  3  6  0  0  0  0  0  1  2  0  0  0  0  3  0  0  0  0  0  0  1  1
0
   0  1  0  1  4  0]
 [ 1  1  0  8  2  3  1  3  1  0  0  1  1  2  0  2  0  0  0  2  2  0  0
0
   0  0  0  0  0  0]
 [ 0  0  0  4  5  4  2  2  0  0  0  0  0  5  0  4  0  1  1  2  0  0  0
```

```
0
   0  0  0  0  0  0]
 [ 0  0  1  6  0  2  2  4  0  0  0  1  0  0  1  1  2  0  0  1  0  0  0
0
   0  0  0  0  0  0]
 [ 0  0  0  1  0  1 17  0  0  0  0  0  0  6  0  2  0  0  0  0  0  0  0
0
   0  0  0  0  0  0]
 [ 0  0  0  0  0  0 19  3  0  0  0  0  0  2  0  7  0  0  0  1  1  0  0
0
   0  0  0  0  0  0]
 [ 5  1  0  0  0  0  0  0  7  7  0  0  0  0  0  0  2  0  0  0  0  1  2
0
   0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  8 11  0  1  0  0  0  0  2  0  0  0  0  2  2
1
   0  0  0  0  0  0]
 [ 1  1  0  0  0  0  0  0  1  1  0  1  0  3  0  0  2  0  1  4 10  1  1
1
   0  0  0  0  0  0]
 [ 1  1  0  1  0  1  0  0  5  0  2  1  1  1  0  0  5  1  0  0  2  2  5
1
   0  0  0  2  0  0]
 [ 0  1  2  1  0  3  0  0  4  4  0  6  2  0  0  0  2  1  1  0  0  0  1
3
   0  1  1  0  0  0]
 [ 0  0  0  0  0  0  0  1  0  0  8  0  0 20  0  0  0  0  1  5  5  0  0
0
   0  0  0  0  0  0]
 [ 1  1  1  0  0  0  0  0  0  0  0  0  0  0 20  0  1  0  0  1  0  0  0
0
   0  0  0  0  0  0]
 [ 0  0  0  1  0  0  7  0  0  0  0  0  0  0  0 17  0  0  0  0  0  0  0
0
   0  0  0  0  0  0]
 [ 1  0  0  0  0  0  0  0  3  0  0  0  1  0  0  0  6  1  4  1  4  0  2
0
   0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  1  0  0  1  0  0  1 19  8  1  2  0  0
0
   0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  4  1 11 10  5  0  1
0
   0  0  0  0  0  0]
 [ 1  0  0  0  0  0  0  0  0  0  3  0  0  9  0  0  1  1  2 13  1  0  0
0
   0  0  0  0  0  0]
 [ 0  0  0  1  0  1  0  0  0  0  5  0  1 13  0  0  4  0  0  2  9  0  0
0
```

```
   0  0  0  0  0  0]
 [ 1  2  6  0  0  0  0  0 16  5  1  1  1  0  0  0  3  0  0  1  0  3  2
0
   0  0  1  0  3  0]
 [ 1  1  2  0  0  0  0  0  5  3  1  0  0  0  0  0  6  0  1  0  1  4  3
0
   0  0  1  0  4  1]
 [ 0  0  0  1  0  0  0  0  1  2  1  1  1  0  0  0  1  0  0  0  0  0  1
15
   3  0  0  0  0  2]
 [ 0  0  0  0  0  0  0  0  1  1  1  0  2  0  0  0  0  0  0  0  0  0  1
19
   7  0  1  0  0  1]
 [ 1  0  2  0  0  0  0  0  3  0  3  0  0  0  0  0  0  0  0  0  0  0  0
0
   1  3  7  0  1  7]
 [ 4  0  2  0  0  0  0  0  0  1  0  0  3  0  1  0  0  0  0  0  0  0  0
0
   0  1  9  0  0  2]
 [ 4  0  0  0  0  0  0  0  0  0  0  0  0  0  7  0  0  0  0  0  0  2  0
0
   0  0  1  0  9  0]
 [ 3  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  1  0  0  0  0  2  0
0
   0  0  0  0  5  0]
 [ 3  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
1
   3  3  2  0  0  5]]
```

```
accuracy=accuracy_score(y_test,y_predict2)
```

```
print('Accuracy(Linear kernel):',"%.2f"%(accuracy*100))
```

```
Accuracy(Linear kernel): 27.89
```

```
precision=precision_score(y_test,y_predict2,average='weighted')
```

```
recall=recall_score(y_test,y_predict2,average='weighted')
```

```
print('Precision:',"%.2f"%(precision*100))
```

```
Precision: 28.23
```

```
print('Recall:',"%.2f"%(recall*100))
```

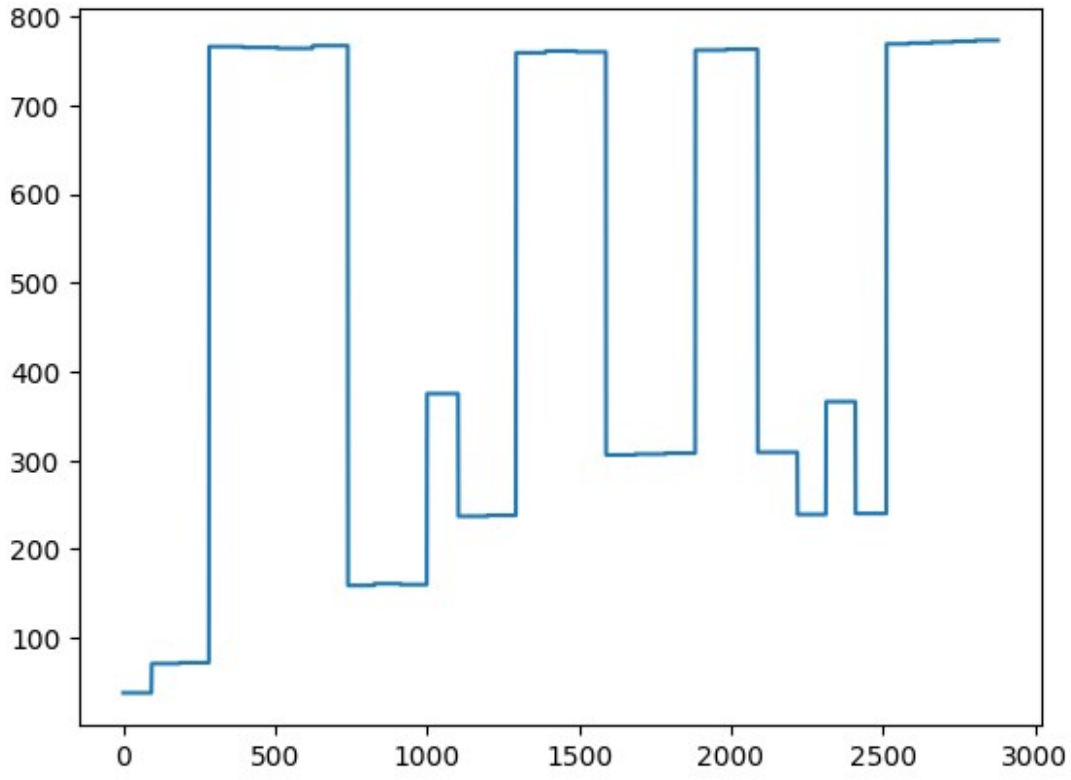
```
Recall: 27.89
```

```
x=(data['Stn Code'])
```

```
y=(data['NO2'])
```

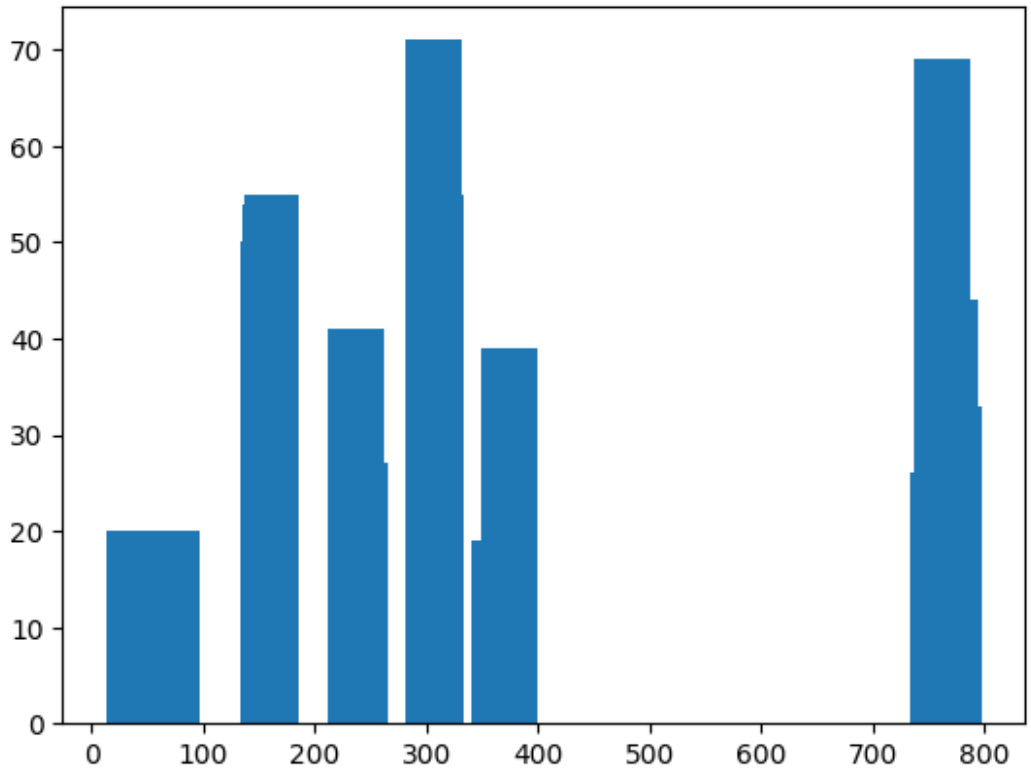
```
plt.plot(x)
```

```
[<matplotlib.lines.Line2D at 0x2418c19d220>]
```

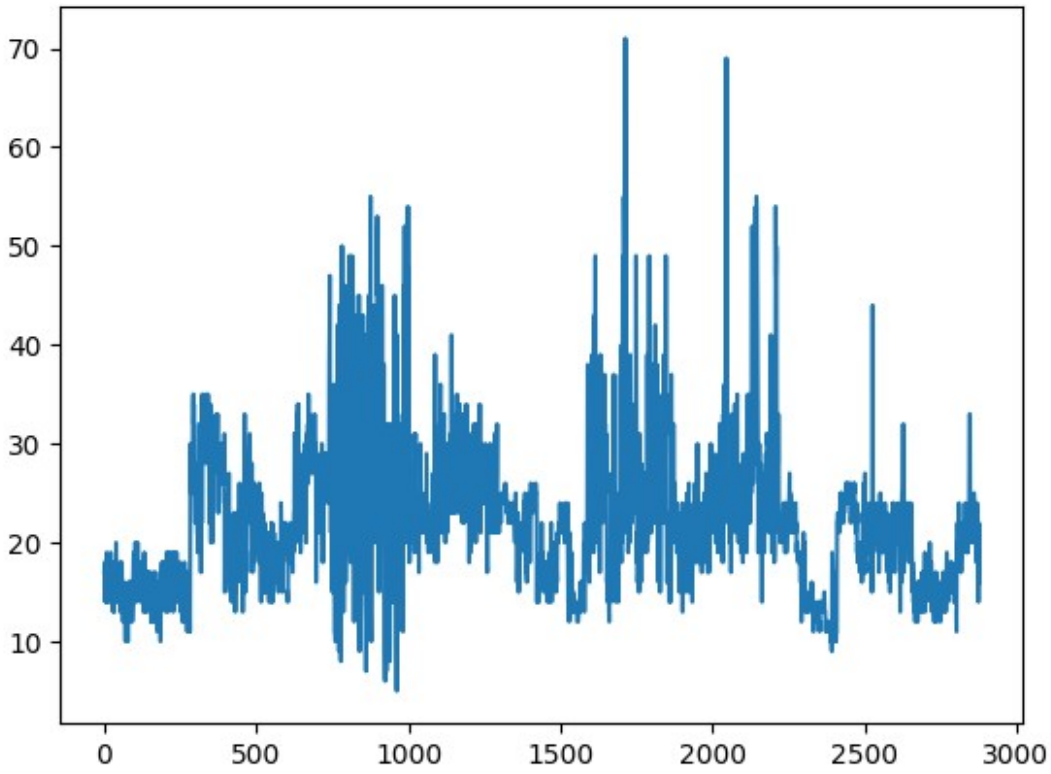



```
plt.bar(x,y,width=50)
```

```
<BarContainer object of 2879 artists>
```

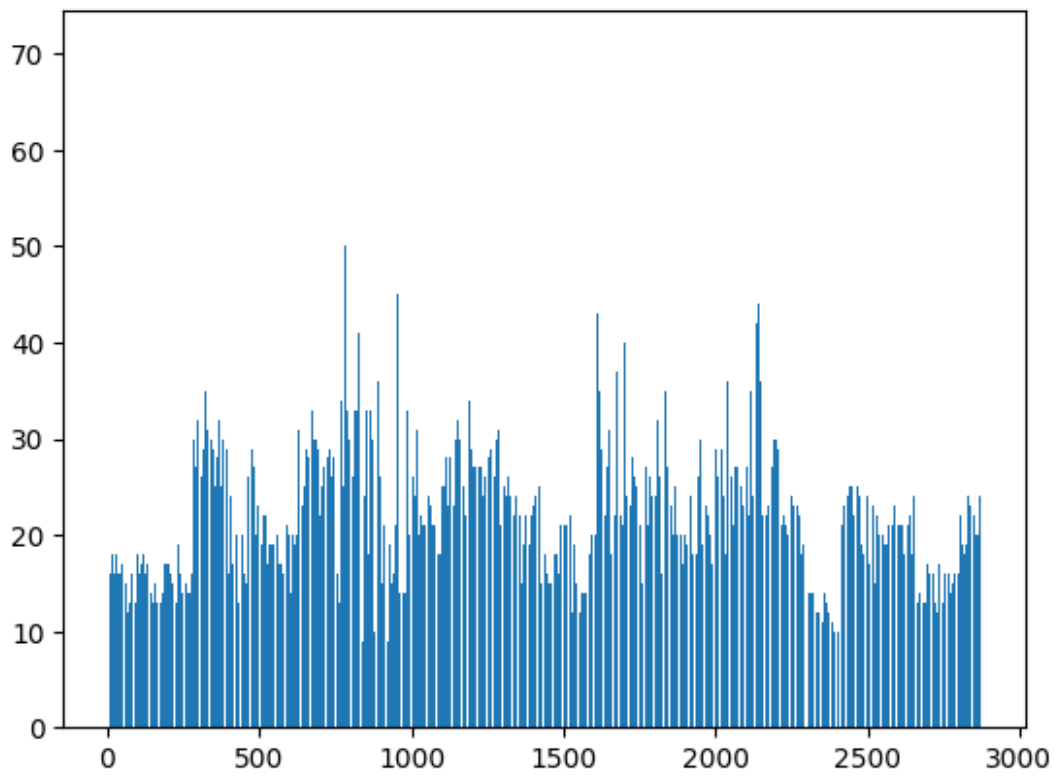



```
xpos=np.arange(len(x))

plt.plot(xpos,y)

[<matplotlib.lines.Line2D at 0x2419099be80>]
```




```
plt.bar(xpos,y)
```

```
<BarContainer object of 2879 artists>
```




```
heat=data[['Stn Code','NO2','SO2']]

heat.head()
```

```
   Stn Code   NO2   SO2
0        38  17.0  11.0
1        38  17.0  13.0
2        38  18.0  12.0
3        38  16.0  15.0
4        38  14.0  13.0
```

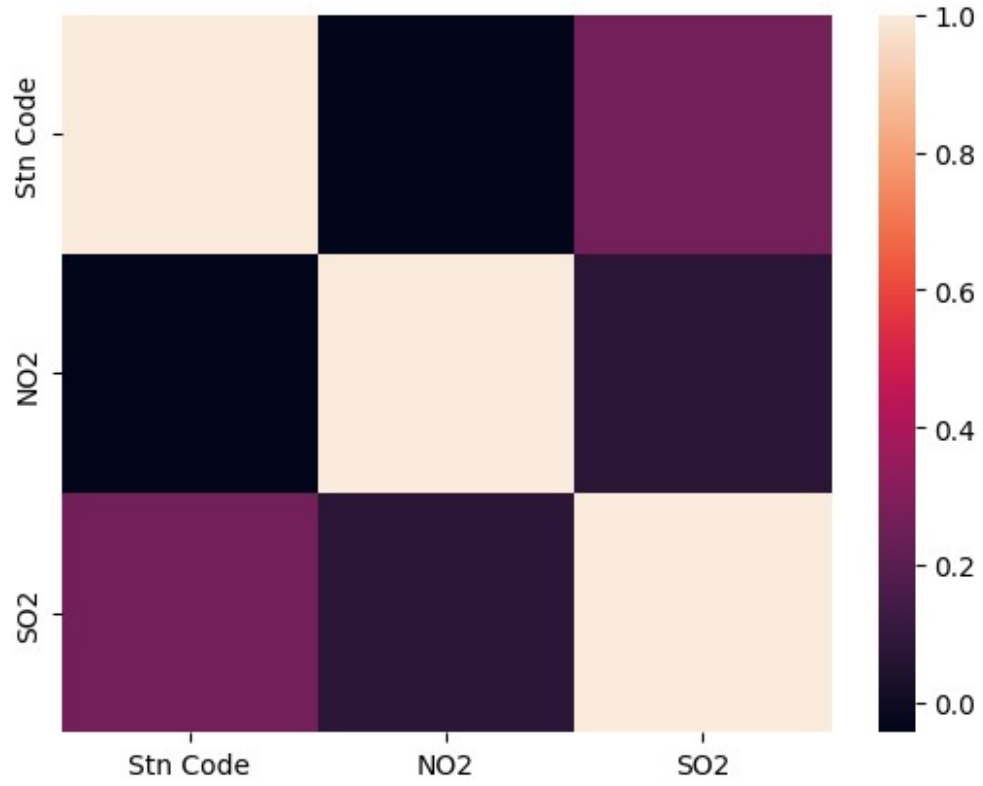
```
df_corr=heat.corr()

figure=plt.figure(figsize=(20,25))
```

```
<Figure size 2000x2500 with 0 Axes>
```

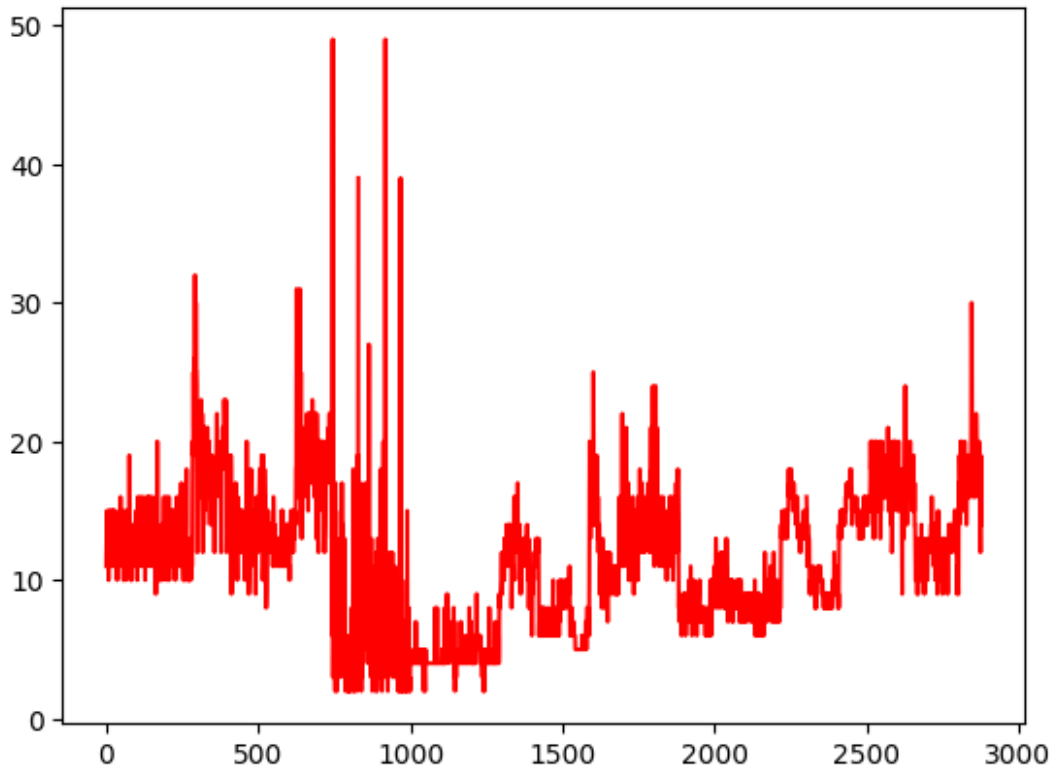
```
sns.heatmap(df_corr,annot=True,fmt='fig')
```

```
---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
Cell In[49], line 1
----> 1 sns.heatmap(df_corr,annot=True,fmt='fig')

File ~\AppData\Local\Programs\Python\Python38\lib\site-packages\
```

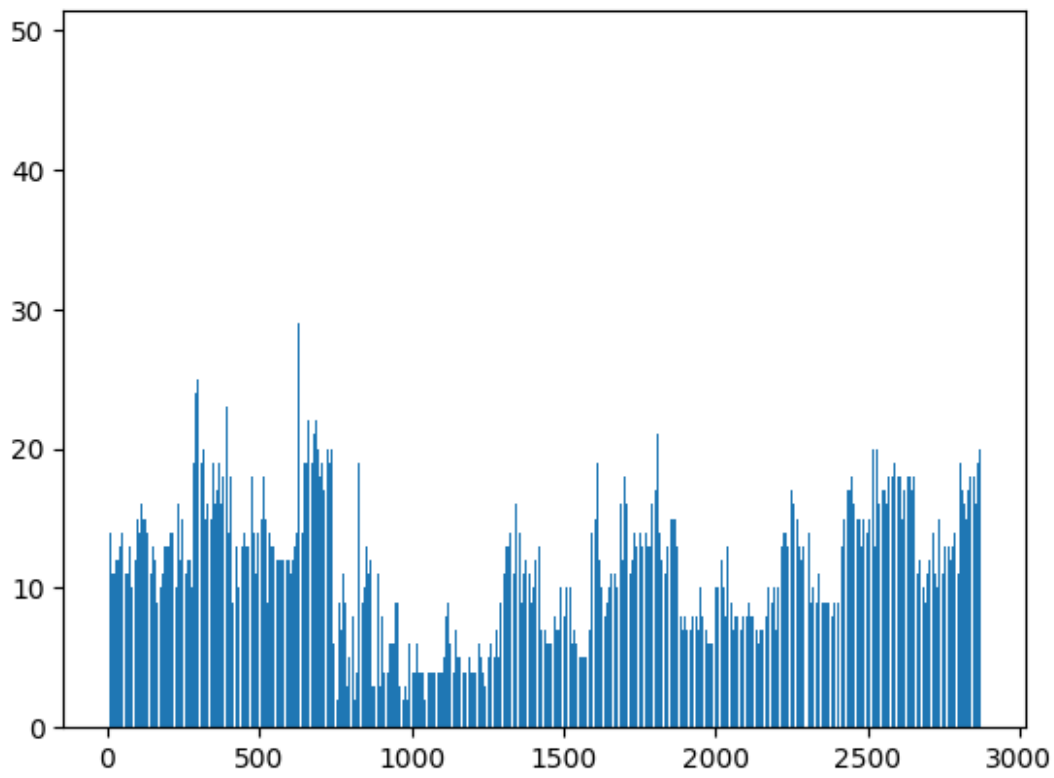
```
seaborn\matrix.py:459, in heatmap(data, vmin, vmax, cmap, center,
robust, annot, fmt, annot_kws, linewidths, linecolor, cbar, cbar_kws,
cbar_ax, square, xticklabels, yticklabels, mask, ax, **kwargs)
    457 if square:
    458     ax.set_aspect("equal")
--> 459 plotter.plot(ax, cbar_ax, kwargs)
    460 return ax

File ~\AppData\Local\Programs\Python\Python38\lib\site-packages\
seaborn\matrix.py:352, in _HeatMapper.plot(self, ax, cax, kws)
    350 # Annotate the cells with the formatted values
    351 if self.annot:
--> 352     self._annotate_heatmap(ax, mesh)

File ~\AppData\Local\Programs\Python\Python38\lib\site-packages\
seaborn\matrix.py:260, in _HeatMapper._annotate_heatmap(self, ax,
mesh)
    258 lum = relative_luminance(color)
    259 text_color = ".15" if lum > .408 else "w"
--> 260 annotation = ("{:" + self.fmt + "}").format(val)
    261 text_kwargs = dict(color=text_color, ha="center", va="center")
    262 text_kwargs.update(self.annot_kws)

ValueError: Invalid format specifier
```



```
plt.show()
sulphur=(data['SO2'])
plt.plot(xpos,sulphur,'r')
[<matplotlib.lines.Line2D at 0x24194770520>]
```

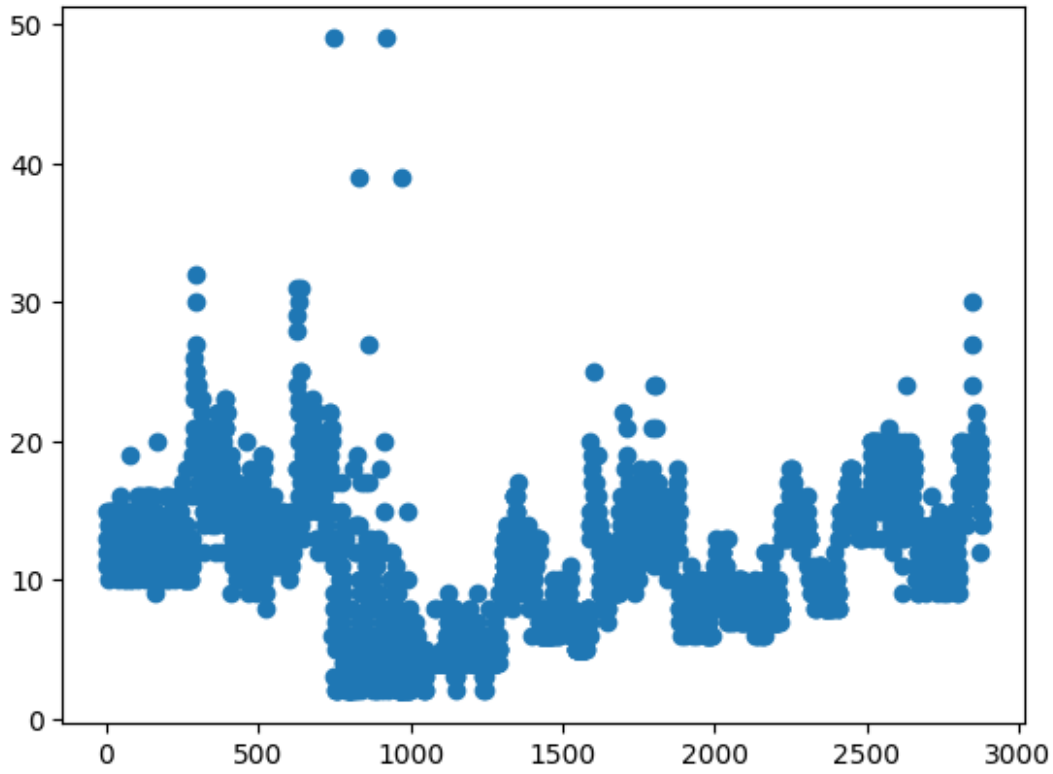



```
plt.bar(xpos,sulphur)
<BarContainer object of 2879 artists>
```

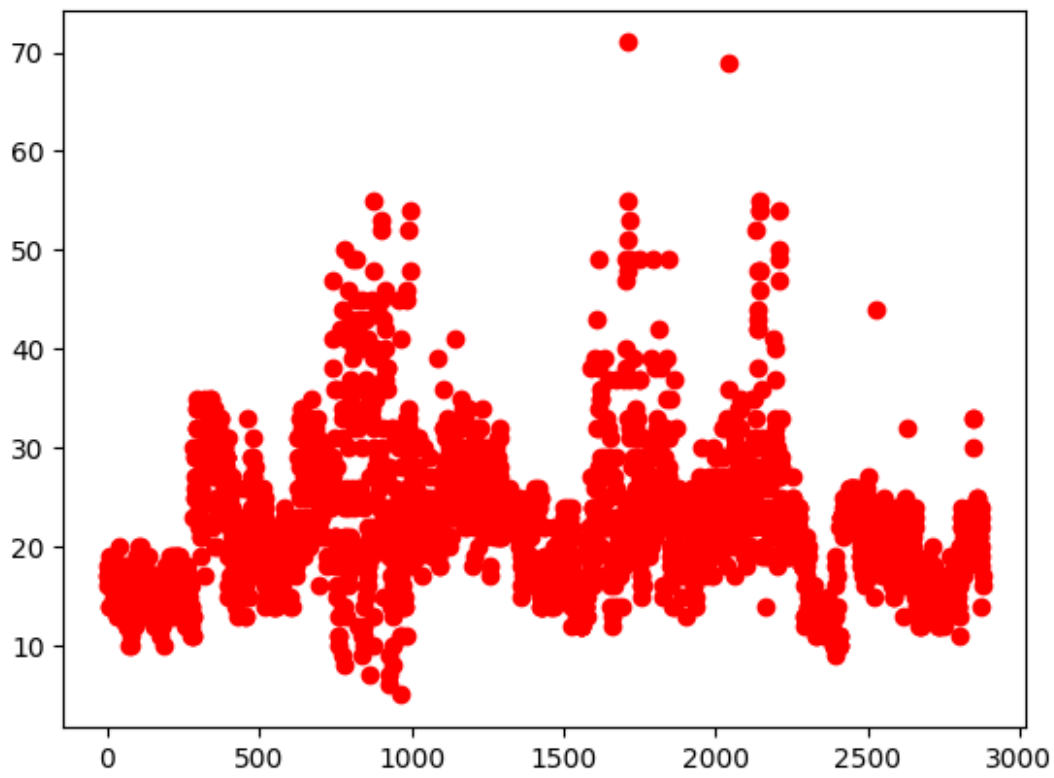



```
plt.scatter(xpos,sulphur)
```

```
<matplotlib.collections.PathCollection at 0x241975891f0>
```




```
nitrogen=(data['NO2'])

plt.scatter(xpos,nitrogen,c='r')

<matplotlib.collections.PathCollection at 0x24197436ee0>
```


70
60
50
40
30
20
10
0
500
1000
1500
2000
2500
3000