APPLIED DATA SCIENCE PROJECT PHASE 4

AIR QUALITY ANALYSIS AND PREDICTION

The project's aim is to examine and create visual representations of air quality information collected from monitoring stations in Tamil Nadu. Its purpose is to uncover patterns in air pollution, pinpoint regions with elevated pollution levels, and establish a forecasting model for estimating RSPM/PM10 levels by considering SO2 and NO2 levels. This endeavor encompasses setting clear goals, formulating an analysis strategy, choosing suitable visualization methods, and constructing a predictive model utilizing Python and pertinent libraries.

DEVELOPMENT PART 2:

STEP 1: Data Loading

The provided code imports necessary Python libraries for data visualization, sets the default figure size, suppresses warnings, reads a CSV file into a DataFrame while handling missing values, and displays the first few rows of the dataset. However, there is a redundant data import operation that can be removed.

```
 import numpy as np import pandas as pd # data processing, CSV file
I/O (e.g. pd.read_csv)

import matplotlib.pyplot as plt
import seaborn as sns
  plt.rcParams['figure.figsize'] = (10,
7)

# Warnings
import warnings
warnings.filterwarnings('ignore')

# Input data files are available in the "../input/" directory.
 import
os
data=pd.read_csv("C:\\Users\\Sandhya\\Downloads\\air
quality.csv") data.fillna(0, inplace=True) data.head()
import pandas as pd

# Read the CSV file into a DataFrame
data = pd.read_csv('C:\\Users\\Sandhya\\Downloads\\air quality.csv')
# Display the DataFrame as a table
print(data)
```

OUTPUT

	Stn Code	Date	State	...	NO2	RSPM	PM 2.5
0	38	01-02-2014	Tamil Nadu	...	17.0	55.0	NaN
1	38	01-07-2014	Tamil Nadu	...	17.0	45.0	NaN
2	38	21-01-2014	Tamil Nadu	...	18.0	50.0	NaN
3	38	23-01-2014	Tamil Nadu	...	16.0	46.0	NaN
4	38	28-01-2014	Tamil Nadu	...	14.0	42.0	NaN
...	...	...	...	...	...	...	...
2874	773	12-03-2014	Tamil Nadu	...	18.0	102.0	NaN
2875	773	12-10-2014	Tamil Nadu	...	14.0	91.0	NaN
2876	773	17-12-2014	Tamil Nadu	...	22.0	100.0	NaN
2877	773	24-12-2014	Tamil Nadu	...	17.0	95.0	NaN
2878	773	31-12-2014	Tamil Nadu	...	16.0	94.0	NaN

[2879 rows x 11 columns]

STEP 2: To calculate SI

The code calculates the Air Quality Index (AQI) for SO2 (Sulfur Dioxide) based on specific concentration ranges, stores the results in a DataFrame, and showcases different ways to display the DataFrame in tabular, markdown, and HTML formats. There is a redundant redefinition of the 'df' DataFrame that can be removed.

```
def calculate_si(SO2):
    si=0     if (SO2<=40):       si= SO2*(50/40)      if
(SO2>40 and SO2<=80):       si= 50+(SO2-40)*(50/40)      if
(SO2>80 and SO2<=380):       si= 100+(SO2-80)*(100/300)
if (SO2>380 and SO2<=800):       si= 200+(SO2-
380)*(100/800)      if (SO2>800 and SO2<=1600):       si=
300+(SO2-800)*(100/800)
```

```
    if (SO2>1600):
     si= 400+(SO2-1600)*(100/800)
return si
data['si']=data['SO2'].apply(calculate_si)
df= data[['SO2','si']] df.head()
# Assuming you already have 'df' defined df
= data[['SO2', 'si']]

# Print the DataFrame in a tabular format
print(df.to_string(index=False))
# Using to_markdown() to print the DataFrame as markdown
print(df.to_markdown(index=False))

# Using to_html() to print the DataFrame as HTML
print(df.to_html(index=False))
```

OUTPUT:

	SO2	**si**
0	11.0	13.75
1	13.0	16.25
2	12.0	15.00
3	15.0	18.75
4	13.0	16.25
…	…	…
2874	15.0	18.75
2875	12.0	15.00
2876	19.0	23.75
2877	15.0	18.75
2878	14.0	17.50

[2879 rows x 2 columns]

STEP 3: To calculate NI

The code defines a function calculate_ni(NO2) to calculate the Air Quality Index (AQI) for NO2 (Nitrogen Dioxide) based on specific concentration ranges. This function is applied to the 'NO2' column of a DataFrame 'data', and the resulting 'ni' values are stored in a new DataFrame 'df'. The code then showcases the first few rows of the 'df' DataFrame. However, there is a redundant redefinition of 'df' that can be removed.

```
def calculate_ni(NO2):
    ni=0     if(NO2<=40):
ni= NO2*50/40     elif(NO2>40
and NO2<=80):      ni= 50+(NO2-
14)*(50/40)     elif(NO2>80 and
NO2<=180):      ni= 100+(NO2-
80)*(100/100)     elif(NO2>180
and NO2<=280):       ni= 200+(NO-
180)*(100/100)     elif(NO2>280
and NO2<=400):      ni=
300+(NO2-280)*(100/120)
else:
     ni= 400+(NO2-400)*(100/120)
return ni
data['ni']=data['NO2'].apply(calculate_ni)
df= data[['NO2','ni']] df.head()
# Assuming you already have 'df' defined df
= data[['NO2', 'ni']]

print(df)
```

OUTPUT

	NO2	ni
0	17.0	21.25
1	17.0	21.25
2	18.0	22.50
3	16.0	20.00
4	14.0	17.50
...	...	...
2874	18.0	22.50

2875 14.0 17.50

2876 22.0 27.50

2877 17.0 21.25

2878 16.0 20.00

[2879 rows x 2 columns]

STEP 4: To calculate RPI

The code defines a function calculate_(RSPM) to calculate the Respirable Suspended Particulate Matter (RSPM) Index (rpi) based on specific concentration ranges. This function is applied to the 'RSPM' column of a DataFrame 'data', and the resulting 'rpi' values are stored in a new DataFrame 'df'. The code then displays the last few rows of the 'df' DataFrame, followed by a summary of the DataFrame.

```
def calculate_(RSPM):
    rpi=0     if(RSPM<=30):
rpi=RSPM*50/30     elif(RSPM>30
and RSPM<=60):
     rpi=50+(RSPM-30)*50/30
elif(RSPM>60 and RSPM<=90):
rpi=100+(RSPM-60)*100/30
elif(RSPM>90 and RSPM<=120):
rpi=200+(RSPM-90)*100/30
elif(RSPM>120 and RSPM<=250):
rpi=300+(RSPM-120)*(100/130)     else:
     rpi=400+(RSPM-250)*(100/130)
return rpi
data['rpi']=data['RSPM'].apply(calculate_)
df= data[['RSPM','rpi']] df.tail()
print(df)
```

OUTPUT

	RSPM	rpi
0	55.0	91.666667
1	45.0	75.000000
2	50.0	83.333333
3	46.0	76.666667

4 42.0 70.000000

... ...

2874 102.0 240.000000

2875 91.0 203.333333

2876 100.0 233.333333

2877 95.0 216.666667

2878 94.0 213.333333

STEP 5: To calculate AQI

The code defines a function calculate_aqi(si, ni, rpi) to calculate the Air Quality Index (AQI) based on the input parameters 'si' (Sulfur Dioxide Index), 'ni' (Nitrogen Dioxide Index), and 'rpi' (Respirable Suspended Particulate Matter Index). The code then applies this function to the DataFrame 'data' for each row, calculating the AQI for each row's respective values of 'si', 'ni', and 'rpi'. The results are stored in a new DataFrame 'df', which includes columns for 'Date', 'City/Town/Village/Area', 'si', 'ni', 'rpi', and the calculated 'AQI'. The code displays the first few rows of the 'df' DataFrame, followed by a summary of the DataFrame.

```
def calculate_aqi(si,ni,rpi):
    aqi=0     if(si>ni
and si>rpi):
     aqi=si     if(ni>si
and ni>rpi):
     aqi=ni     if(rpi>si
and rpi>ni):
     aqi=rpi
return aqi
data['AQI']=data.apply(lambda
x:calculate_aqi(x['si'],x['ni'],x['rpi']),axis=1)
df=
data[['Date','City/Town/Village/Area','si','ni','rpi','AQI']]
df.head() print (df)
```

OUTPUT

	Date	City/Town/Village/Area	si	ni	rpi	AQI
0	01-02-2014	Chennai	13.75	21.25	91.666667	91.666667
1	01-07-2014	Chennai	16.25	21.25	75.000000	75.000000
2	21-01-2014	Chennai	15.00	22.50	83.333333	83.333333

3 23-01-2014 Chennai 18.75 20.00 76.666667 76.666667

4 28-01-2014 Chennai 16.25 17.50 70.000000 70.000000

... ...

2874 12-03-2014 Trichy 18.75 22.50 240.000000 240.000000

2875 12-10-2014 Trichy 15.00 17.50 203.333333 203.333333

2876 17-12-2014 Trichy 23.75 27.50 233.333333 233.333333

2877 24-12-2014 Trichy 18.75 21.25 216.666667 216.666667 2878 31-12-2014

Trichy 17.50 20.00 213.333333 213.333333

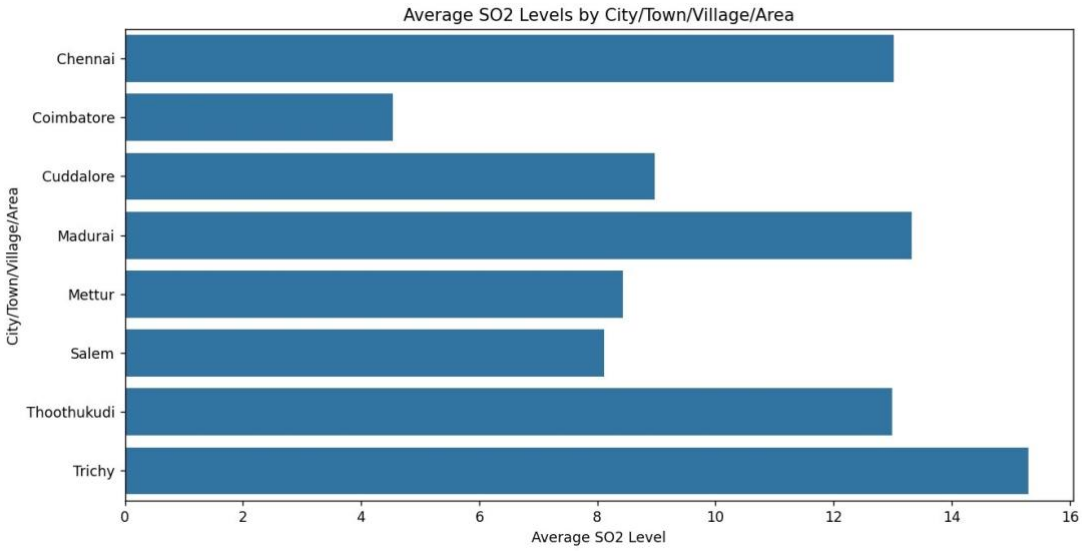
[2879 rows x 6 columns]

STEP 6: To calculate average SO2

The provided code groups air quality data by 'City/Town/Village/Area,' calculates the average levels of SO2 (Sulfur Dioxide) for each area, and visualizes the results in a horizontal bar plot using Seaborn. The plot represents the average SO2 levels for different areas, allowing for a comparison of air quality. However, there is duplicated code that can be removed.

```
# Group the data by 'City/Town/Village/Area' and calculate the average SO2
levels
average_so2_by_area =
data.groupby('City/Town/Village/Area')['SO2'].mean().reset_index()
# Plot the average SO2 levels
plt.figure(figsize=(12, 6))
sns.barplot(x='SO2', y='City/Town/Village/Area', data=average_so2_by_area,
orient='h')
plt.title('Average SO2 Levels by
City/Town/Village/Area') plt.xlabel('Average SO2 Level')
plt.ylabel('City/Town/Village/Area') plt.show()
```

OUTPUT

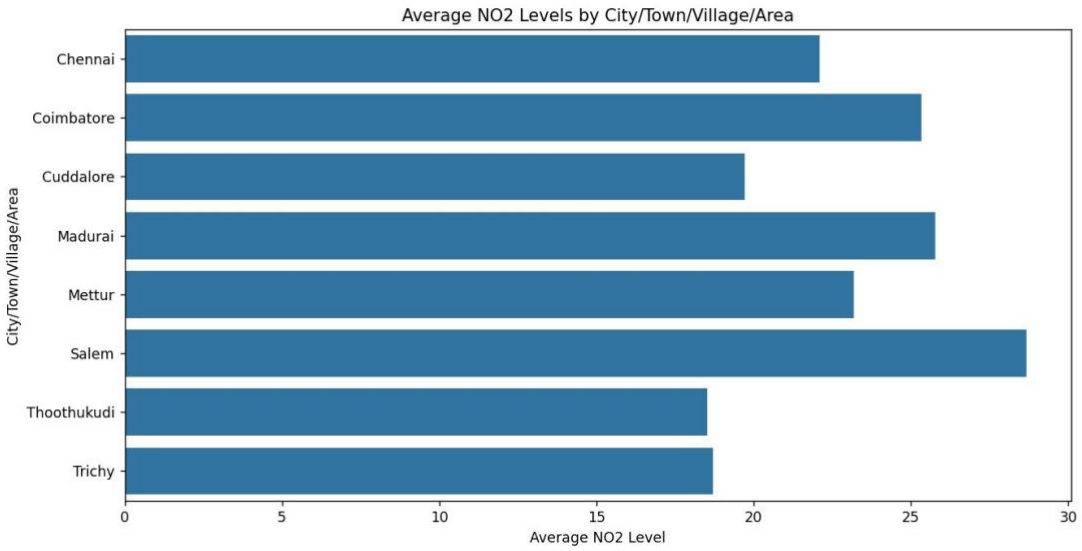


STEP 7: To calculate average NO2

The code groups air quality data by 'City/Town/Village/Area,' calculates the average NO2 (Nitrogen Dioxide) levels for each area, and visualizes the results in a horizontal bar plot using Seaborn. This plot allows for a comparison of average NO2 levels across different areas.

```
# Group the data by 'City/Town/Village/Area' and calculate the average NO2
levels
average_no2_by_area =
data.groupby('City/Town/Village/Area')['NO2'].mean().reset_index()
# Plot the average NO2 levels
plt.figure(figsize=(12, 6))
sns.barplot(x='NO2', y='City/Town/Village/Area', data=average_no2_by_area,
orient='h')
plt.title('Average NO2 Levels by
City/Town/Village/Area') plt.xlabel('Average NO2 Level')
plt.ylabel('City/Town/Village/Area') plt.show()
```

OUTPUT

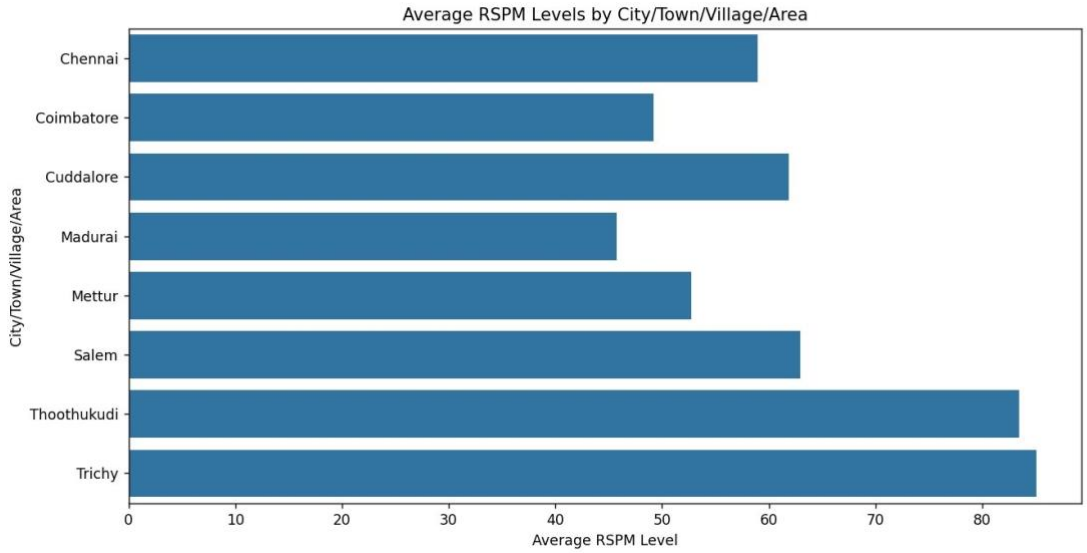


STEP 8: To calculate average RPSM

The code groups air quality data by 'City/Town/Village/Area,' calculates the average RSPM (Respirable Suspended Particulate Matter) levels for each area, and visualizes the results in a horizontal bar plot using Seaborn. This plot allows for a comparison of average RSPM levels across different areas.

```
# Group the data by 'City/Town/Village/Area' and calculate the average RSPM
levels
average_rpsm_by_area =
data.groupby('City/Town/Village/Area')['RSPM'].mean().reset_index()
# Plot the average RSPM levels
plt.figure(figsize=(12, 6))
sns.barplot(x='RSPM', y='City/Town/Village/Area',
data=average_rpsm_by_area, orient='h')
plt.title('Average RSPM Levels by
City/Town/Village/Area') plt.xlabel('Average RSPM Level')
plt.ylabel('City/Town/Village/Area') plt.show()
# Group the data by 'City/Town/Village/Area' and calculate the average AQI
levels
average_aqi_by_area =
data.groupby('City/Town/Village/Area')['AQI'].mean().reset_index()
```

OUTPUT

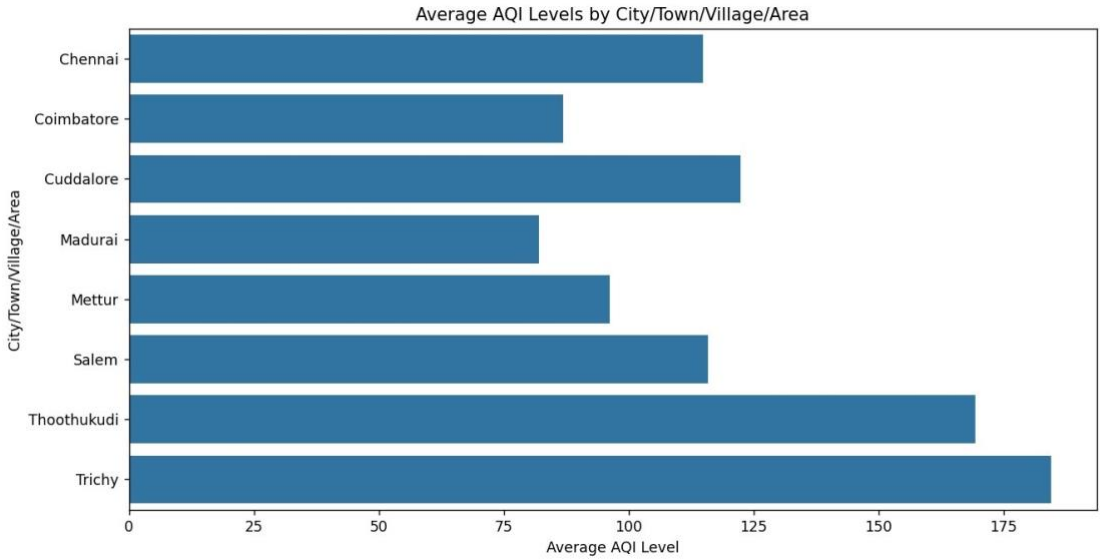


STEP 9: To calculate average AQI

This procces involves the calculation of average API across different stations and it is used to find the most affected citities in Tamil Nadu. This code groups the data by city or area, calculates the average AQI levels for each area, and presents this information in a horizontal bar plot for easy comparison of air quality by location.

```
# Group the data by 'City/Town/Village/Area' and calculate the average AQI
levels
average_aqi_by_area =
data.groupby('City/Town/Village/Area')['AQI'].mean().reset_index()
# Plot the average AQI levels
plt.figure(figsize=(12, 6))
sns.barplot(x='AQI', y='City/Town/Village/Area', data=average_aqi_by_area,
orient='h')
plt.title('Average AQI Levels by
City/Town/Village/Area') plt.xlabel('Average AQI Level')
plt.ylabel('City/Town/Village/Area') plt.show()
```

OUTPUT



STEP 10: To identify pollution trends with high pollution levels

The code provides a comprehensive analysis of AQI data, including visualization of trends, a naive forecasting approach, and a recalculation of AQI based on different components.

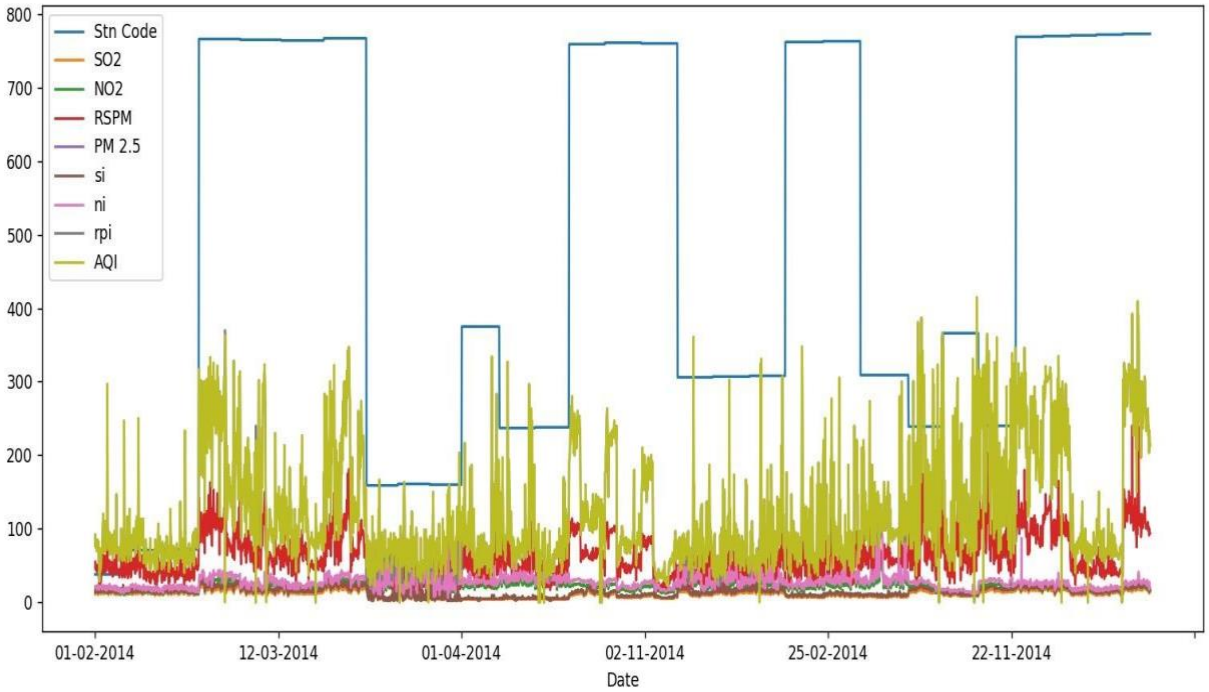
```
df =
data[['AQI','Date']].groupby(["Date"]).median().reset_index().sort_values(b
y='Date',ascending=False) f,ax=plt.subplots(figsize=(15,10))
sns.pointplot(x='Date', y='AQI', data=df) train = pd.DataFrame(data) test =
pd.DataFrame(data) dd= np.asarray(train.AQI) y_hat = test.copy()
y_hat['naive'] = dd[len(dd)-1]
plt.figure(figsize=(12,8))
plt.plot(train.index, train['AQI'], label='Train')
plt.plot(test.index,test['AQI'], label='Test')
plt.plot(y_hat.index,y_hat['naive'], label='Naive Forecast')
plt.legend(loc='best')
plt.title("Naive Forecast",fontsize=20)

plt.legend(["actual ","predicted"])
```

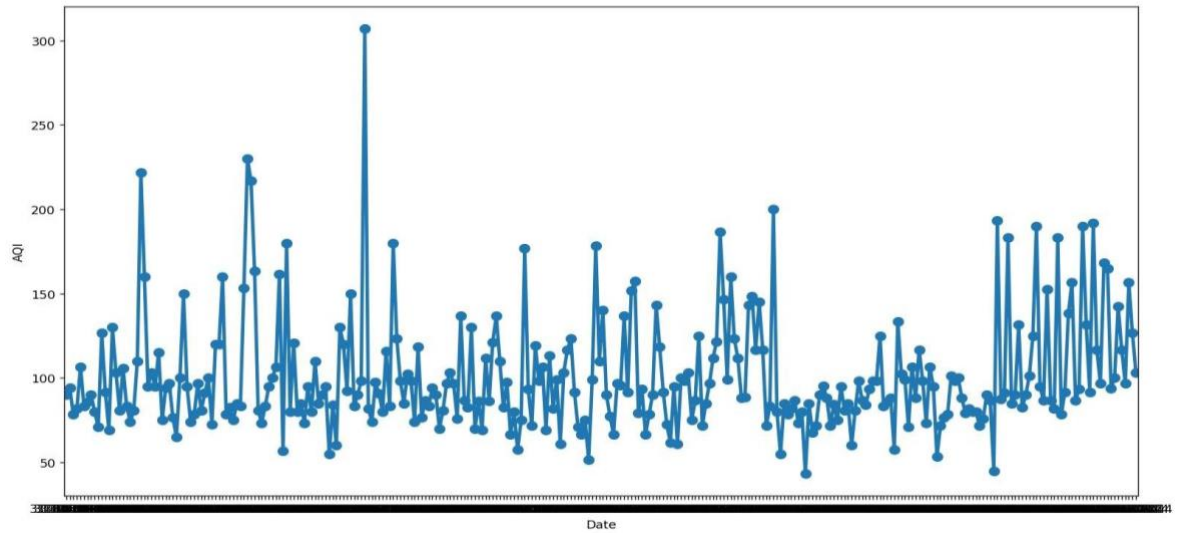
```
plt.xlabel("Date",fontsize=20)
plt.ylabel("AQI",fontsize=20)
plt.tick_params(labelsize=20) plt.show()
data['AQI']=data.apply(lambda
x:calculate_aqi(x['si'],x['ni'],x['rpi']),axis=1)
df= data[['Date','City/Town/Village/Area','si','ni','rpi','AQI']]
df.head() print (df)
df=data.set_index('Date')
df.sort_values(by='Date',ascending=False)
df.plot(figsize=(15, 6)) plt.show()
y=df.AQI
```

OUTPUT

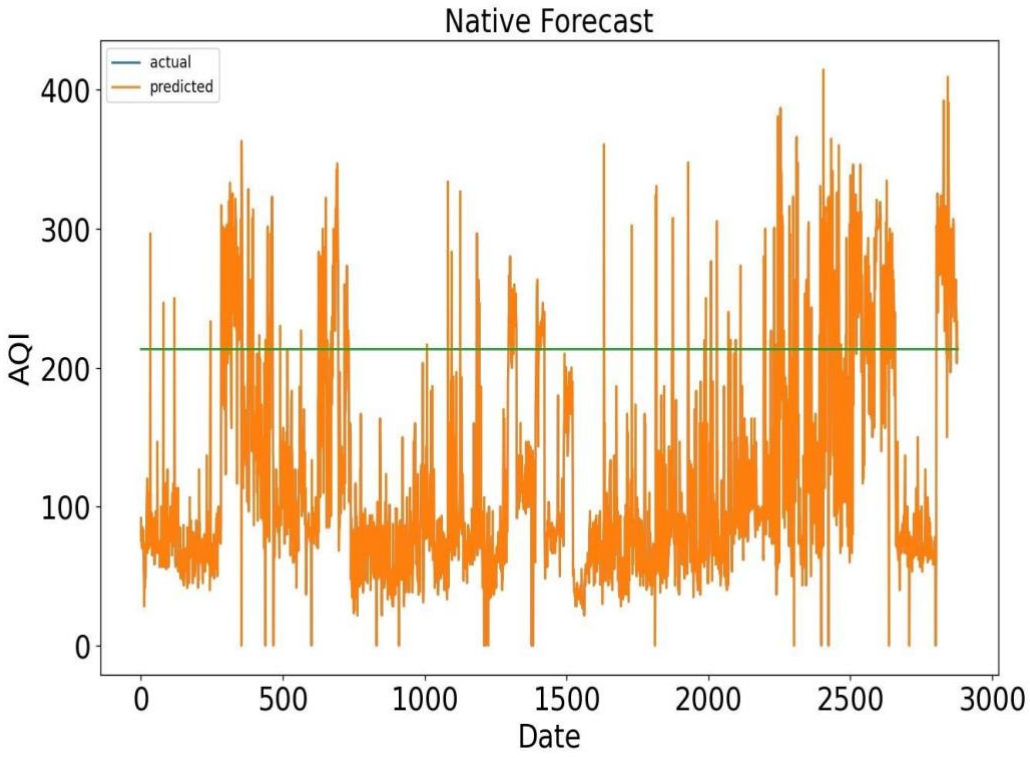
a)Overall Plot



b)AQI



c) TREND



In summary

a) Chennai, Thoothukudi, Trichy and Madurai has higher SO2 levels.
b) Salem, Coimbatore and Madurai has higher NO2 levels.
c) Trichy and Thoothukudi has higher RPSM levels
d) Overall, the higher average AQI has been found in Trichy and Thoothukudi .

These are the cities with high air pollution levels and are highly prone to health hazards