Air Pollution in Delhi

Harshita Sharma(20171099)

Abhigyan Ghosh(20171089)

Zubair Abid(20171076)

CSE442: Disaster Management
Prof. Sunitha Palissey
April 29, 2021

Introduction

Delhi never ceases to be in the news. It is the capital of India, a hotspot of political drives and a hotbed of protest. However, the city has also been grabbing headlines for one more reason: its alarming level of pollution that has made it the most polluted city in the world. A report by the Ministry of Environment and Forests, India, in 1997 reviewed the environmental situation in Delhi over concerns of deteriorating conditions. Air pollution was one of the areas of concern identified in this study. It was estimated that about 3000 metric tons of air pollutants were emitted every day in Delhi, with a major contribution from vehicular pollution (67%), followed by coal-based thermal power plants (12%). There was a rising trend from 1989 to 1997 as monitored by the Central Pollution Control Board (CPCB). The concentrations of carbon monoxide from vehicular emissions in 1996 showed an increase of 92% over the values observed in 1989, consequent upon the increase in vehicular population.

Literature Review

Air pollution levels in Delhi. Delhi Air Quality Index generally hovers from moderate to worse. It is rarely satisfactory and never 'good.' During December to March, the months of winters when the Sun is hard to spot, it is the smog that affects the visibility to a great extent, and the quality reduces to very poor, severe and eventually hazardous. From October to December, the pollution level worsens exorbitantly due to stubble burning, dust storms, vehicle pollution, and gradually changing weather.

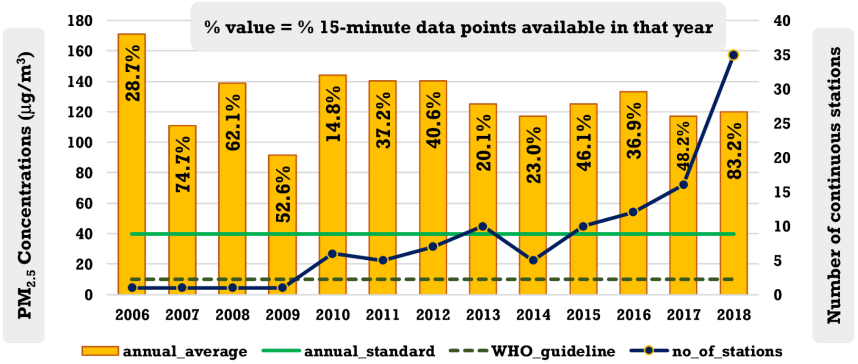


Figure 1. PM2.5 concentrations in Delhi from 2006 to 2018

The World Bank Development Research Group funded a study on air pollution in 1991-1994, which established that the TSP or the average total suspended particulate level in the capital are five times higher than the WHO's average annual standard. Another study on TSPs in Delhi revealed that the readings were above ninety-seven per cent higher than the WHO average limit every day.

The pollution level has always been a pain point for Delhi among its many issues, but of late, it has been drastic to the extent of becoming life-threatening. Even in the year 2011, the World Health Organisation reported in its urban air database that Delhi pollution levels have crossed the maximum PM 10 limit by more than 10X.

Great Smog of Delhi In 2017, Delhi earned the unenviable distinction of becoming the most polluted city on Earth this month, as air quality has reached epically bad proportions. Statistics revealed the true picture that $PM_{2.5}$ levels in Delhi on November 7, 2017, were 710 $\mu g/m^3$ and the smog reached its peak on November 8, 2017, which was a red-letter day in the history of Delhi as the AQI was 999, above the upper limit of worst category.

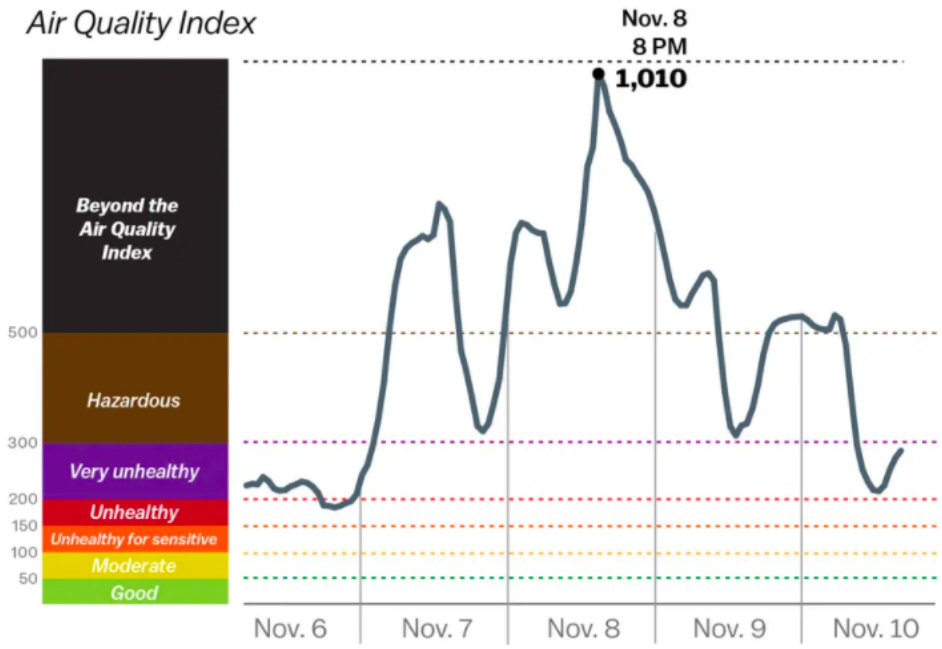


Figure 2. AQI during the Great Smog of Delhi

November 2017 is referred to as the Great Smog of Delhi as well because the air particulate hit the worst level, beyond the safe limit of 100. According to public health experts, this was equivalent to smoking 50 cigarettes/day and a public health emergency has been declared by the Indian Medical Association. United Airlines cancelled its flights to India's capital because of poor air quality. Visibility was so bad that cars crashed in pileups on highways and trains had to be delayed and cancelled.

The Ministry of Earth Sciences' research paper was published in October 2018 and identified the following pollutants in Delhi air:

- Vehicular emissions: 41 per cent
- Dust: 21.5 per cent
- Industries: 18 per cent
- Much of the pollution was found to be coming from farms in the nearby states of Punjab, Haryana, and Western Uttar Pradesh. With the rice harvest over, farmers burn crop stubble — specifically the remnants of the rice crop to prepare the fields to plant wheat and return nutrients to the soil.

As of November 21, 2017, the air quality index was at 326 — far below the peak on November 8, but still in the Hazardous category.

Reasons for pollution levels of Delhi. These are some of the reasons found by various researches for the high AQI and high pollutant concentrations in the air of Delhi:

IIT Kanpur conducted the CPCB (2010) study for Delhi and this 2015 study is considered an update, with a new sample collection at six locations and two seasons (summer and winter).

- The secondary sulfate aerosols, from the chemical conversion of SO2 emissions, are likely to originate from coal and diesel consumption. Similarly, nitrate aerosols, from the chemical conversion of NOx emissions, are likely to originate from coal, diesel, and petrol combustion.

- All the construction, soil, and road dust are clubbed into one dust category.
- Since the categories are listed along the fuel lines, we can interpret that all the diesel and petrol is linked to the vehicle exhaust and diesel generator sets.
- Biomass burning can be linked to crop residue burning (especially for the winter months) and biomass used for cooking and heating; coal could be consumed at industries, cooking, and heating.

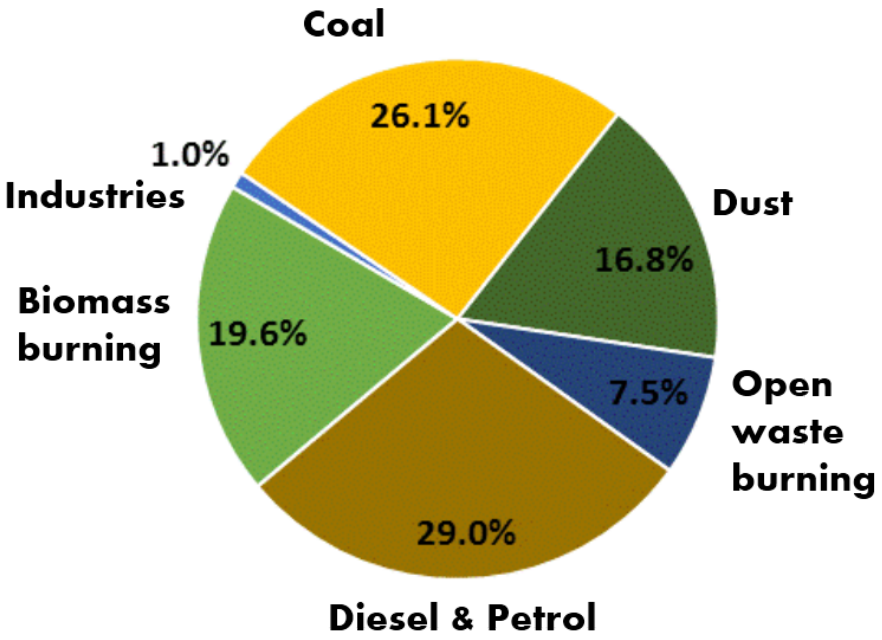


Figure 3. Reasons for increasing air pollution in Delhi

Some other reasons found from other researches were:

- Non-linear city structure which inhibits displacement of air.
- Tough to take the solid waste out of the city causes pollution when burnt within the city.
- Non-linear city structure means more transport routes leading to congestion.
- Weather factors (discussed later under Seasonality)

Designated Bodies for Environmental Concern and Policy Implementation. The Ministry of Environment and Forests and the Department of Environment of the Government of National Capital Territory are the nodal ministries for planning, promoting and overseeing India's environmental policies and programs. The Central Pollution Board is the state-level body to supervise and monitor the ecological concerns of a state.

Apart from the Delhi Pollution Board, Centre for Science and Environment, the Indian Association for Air Pollution Control, the Energy and Resources Institute and several representatives and members of the Society of Indian Automobile Manufacturers, Confederation of Indian Industry and Factories Inspectorate are involved in the brainstorming, suggestion and decision-making.

The research and academic bodies engaged in pollution control are- the Indian Institute of Chemical Engineers, the Indian National Science Academy, and the Indian Institute of Engineers, the Indian Institute of Technology, the National Environmental Engineering Research Institute and the Council of Scientific and Industrial Research Institution.

Measures taken to combat the rising air pollution levels in Delhi. Delhi air pollution has been so severe since time immemorial that the Supreme Court of India keeps intervening to ensure the implementation of preventive measures by the state and centre governments. The first such instance occurred in the year 1985 when the Supreme Court ordered the clampdown of hazardous industries, hot-mix plants, and brick kiln in response to a petition.

- The infrastructure has played a significant role in combating air pollution. The flyovers and subways ensure less clogging on roads and smoother transition.
- The introduction of CNG for public transport means, cabs and autos and rigorous phasing out of old commercial vehicles have led to better air quality to some extent.
- The Delhi Metro (DMRC) has proved to be Godsent in such times, and people from all walks of life take it. It connects each part of Delhi and is cost-effective as well as a time-saver.

- The government has also made PUC (Pollution under Control) mandatory for new and old vehicles. Besides, the vehicle owners need to comply with Bharat Stage II/ higher emission norms/ Euro-II stringently.
- Thermal power stations and industries are now following up stringent emission norms. A part of diesel sales is used to set up the Air Ambience Fund.
- A monthly Ambient Air Quality Monitoring is performed at various forty locations in Delhi to keep an eye on the pollution level across the capital. The countermeasures are taken accordingly.
- Delhi Chief Minister also launched the odd-even scheme as a traffic rationing measure in Delhi to reduce the burden on Delhi roads, to ensure hassle-free commute and reduce pollution. As a result, the streets were less occupied, and hourly particulate levels in the air were dropped by thirteen per cent.
- NASA's list of air-purifying plants comes in handy at these times to combat air pollution at a personal level. Plants like English ivy, spider plant, dragon tree, Barberton daisy, aloe vera, peace lily, rubber plant, ficus and tulsi can eliminate indoor pollutants like formaldehyde, ammonia, xylene, and toluene. These plants don't require direct sunlight and air, making them the ideal home accents for apartment living. The Central Pollution Control Board study reports improvement in overall well-being and respiratory status by spending about eight to ten hours indoors in clean air as well.

Lockdown effects. A lot of research on the impact of lockdown in India on air pollution exists. The air quality in India has been steadily deteriorating over the past few years. For example, A study conducted by Biswajit Bera et al. into the urban air pollution of Kolkata shows that pollutants like CO, NO_2 and SO_2 are significantly decreased, while the average level of O_3 has been slightly increased in 2020 during the lockdown due to the close-down of all industrial and transport activities. Meanwhile, around 17.5% was the mean reduction of PM10 and PM2.5 during lockdown compared with previous years owing to complete stop of vehicles movement, burning of biomass and dust particles from the construction works.

Another study conducted by a team of 10 interdisciplinary researchers from the University of Surrey's renowned Global Centre for Clean Air Research (GCARE) found that the lockdown reduced concentrations of harmful particles up to a 54% reduction in Delhi.

PM2.5 levels. The average concentration of PM2.5 before the lockdown is found to be higher in comparison with the concentration after the lockdown. The PM2.5 concentration in Kolkata is reduced by 34.52%, and 27.57% in Delhi, the capital of India. In general, PM2.5 is much higher throughout the year in the northern parts of India especially in the Indo-Gangetic Plains (IGP). Figure 2. shows the PM2.5 levels in the five major metropolitan cities in India: New Delhi (30 million), Mumbai (25 million), Hyderabad (12.91 million), Kolkata (15.6 million), and Chennai (10.96 million) - in the bracket, the populations are given as per the latest census.

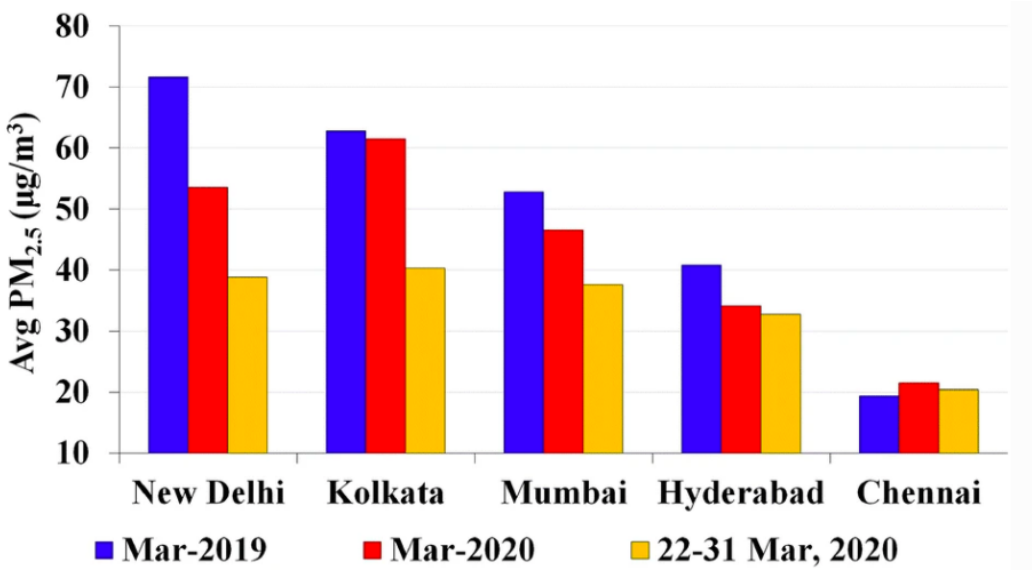


Figure 3. Variations of average PM2.5 during March 2019 (blue bar), March 2020 (red bar), and average during 22–31 March 2020

Nitrogen dioxide (NO_2) levels. Air pollution is not only seen to be prevented to rise rather it is deceased to multi-folds. Nitrogen dioxide emissions are one of the major air pollutants emitted from industrial and vehicular operation. As both the above operations have come to a substantial halt in many counties during this pandemic, NO_2 emissions are diminished, as visible from space.

Moreover, the cities in India full of a crowd on regular days i.e. Delhi and Mumbai were found to have a 40-50% reduction in nitrogen dioxide emission as compared to last year. Figure.4 (left) depicts the day-wise nitrogen dioxide emission level in Delhi. The findings revealed that before the lockdown, the level of nitrogen dioxide emission in Delhi remains between 30 and 65 (ug/m3). However, as the lockdown announced in India, the level of nitrogen dioxide emission showed a dramatic decline. In Delhi, the average level of nitrogen dioxide emission during the lockdown period remains 12 to 25 (ug/m3).

Objectives of the study

The primary aim of this project is to analyse air pollution in Delhi. This aim can be further broken down to provide a better understanding and analysis into a systemised set of objectives given below:

1. Analyse the pollutants contributing to the Air Quality Index(AQI) over the years 2015-2020(mid-July) and months(January - December). And analyse AQI over the years 2015-2020(mid-July) in Delhi.
2. Analysing the trend and seasonality in AQI and different pollutants
3. Analyse and study the effect of Lockdown on air quality and pollution in Delhi.
    a. Analyse pollutant concentrations and AQI of Delhi in India specifically in the year 2020
    b. Analyse pollutant concentrations and AQI of Delhi before and after the lockdown was imposed.
4. Construct models predicting the future of air quality of Delhi.
5. Study the different factors including environmental factors that have an effect on air quality and pollution of Delhi.
6. Data visualisation: Visualization of data and results to provide a better and deeper understanding.

Materials and Methodology

Dataset

Data was collected from **CPCB: Central Pollution and Control Board.** CPCB collects air quality data which is updated every week. At present, the Real-Time Ambient Air Quality Monitoring Network consists of 261 Continuous Ambient Air Quality Monitoring Stations in 134 Cities covering 23 States & Union Territories which are connected to the web-based system.

The data keeps track of major air pollutants such as CO, NO2, NO, O3, SO2, PM2.5 etc**.** which are shown below:

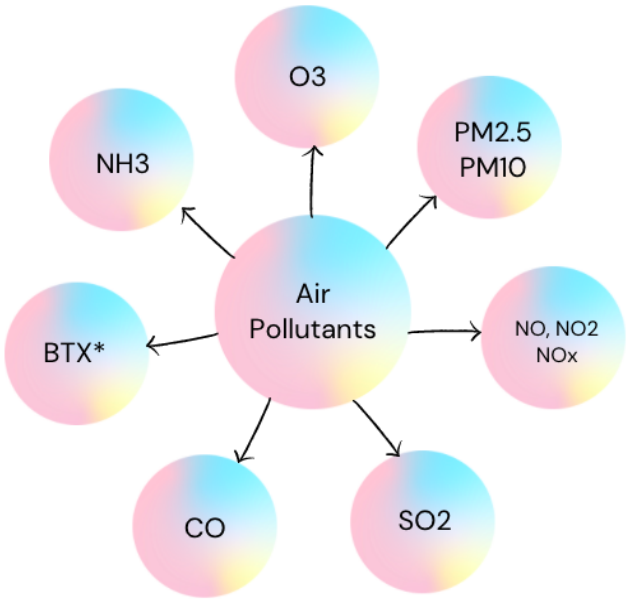


Figure 4. Air Pollutants in the CPCB dataset

- **Particulate matter (PM2.5 and PM10)**: Particulate matter is a mix of solids and liquids, including carbon, complex organic chemicals, sulphates, nitrates, mineral dust, and water suspended in the air. PM varies in size. Some particles, such as dust, soot, dirt or smoke are large or dark enough to be seen with the

naked eye. But the most damaging particles are the smaller particles, known as PM10 and PM2.5.

- **Nitrogen Oxides (NO, NO2, NOx):** Nitrogen oxides are a group of seven gases and compounds composed of nitrogen and oxygen, sometimes collectively known as NOx gases. The two most common and hazardous oxides of nitrogen are nitric oxide(NO) and nitrogen dioxide(NO2).
- **Sulphur Dioxide (SO2):** Sulfur dioxide or SO2 is a colourless gas with a strong odour, similar to a just-struck match. It is formed when fuel containing sulfur, such as coal and oil, is burned, creating air pollution.
- **Carbon Monoxide (CO):** Carbon monoxide is a colourless, highly poisonous gas. Under pressure, it becomes a liquid. It is produced by burning gasoline, natural gas, charcoal, wood, and other fuels.
- **Benzene, Toluene and Xylene (BTX):** Benzene, toluene, xylene, and formaldehyde are well-known indoor air pollutants, especially after house decoration. They are also common pollutants in the working places of the plastic industry, chemical industry, and leather industry.
- **Ammonia (NH3):** Ammonia pollution is pollution by the chemical ammonia (NH3) – a compound of nitrogen and hydrogen which is a byproduct of agriculture and industry.
- **Ozone(O3):** Ground-level ozone is a colourless and highly irritating gas that forms just above the earth's surface. It is called a "secondary" pollutant because it is produced when two primary pollutants react in sunlight and stagnant air. These two primary pollutants are nitrogen oxides (NOx) and volatile organic compounds (VOCs).

Along with these pollutants, the dataset also covers the AQI levels of each city.

Air Quality Index. The air quality index (AQI) is an index for reporting air quality daily. It is a measure of how air pollution affects one's health within a short period. A web-based system is designed to provide AQI on a real-time basis. It is an automated

system that captures data from continuous monitoring stations without human intervention and displays AQI based on running average values (e.g. AQI at 6 am on a day will incorporate data from 6 am on the previous day to the current day). For manual monitoring stations, an AQI calculator is developed wherein data can be fed manually to get AQI value.

Calculation of AQI. The AQI calculation uses 7 measures: PM2.5(Particulate Matter 2.5-micrometer), PM10, SO2, NOx, NH3, CO and O3 (ozone). For PM2.5, PM10, SO2, NOx and NH3 the average value in the last 24-hrs is used with the condition of having at least 16 values. For CO and O3 the maximum value in the last 8-hrs is used. Each measure is converted into a Sub-Index based on pre-defined groups. Sometimes measures are not available due to lack of measuring or lack of required data points. Final AQI is the maximum Sub-Index with the condition that at least one of PM2 and PM10 should be available and at least three out of the seven should be available.

1. The Sub-indices for individual pollutants at a monitoring location is calculated using its 24-hourly average concentration value (8-hourly in case of CO and O3) and health
   breakpoint concentration range. The worst sub-index is the AQI for that location.
2. All the eight pollutants may not be monitored at all the locations. Overall AQI is calculated only if data are available for a minimum of three pollutants out of which one should necessarily be either PM2.5 or PM10. Else, data are considered insufficient for calculating AQI. Similarly, a minimum of 16 hours' data is considered necessary for calculating the subindex.
3. The sub-indices for monitored pollutants are calculated and disseminated, even if data are inadequate for determining AQI. The Individual pollutant-wise sub-index will provide air quality status for that pollutant.

The air quality index (AQI) is defined as ratios of the measured concentration of the atmospheric pollutants to their standard prescribed values. A general formula to compute an AQI is the following:

$$\text{AQI pollutant} = \left(\frac{\textit{pollutant concentration reading}}{\textit{Standard Concentration}} \right) x\ 100$$

There are 6 categories of the air created in this air quality index:

Good (0–50)	Minimal Impact	**Poor (201–300)**	Breathing discomfort to people on prolonged exposure
Satisfactory (51–100)	Minor breathing discomfort to sensitive people	**Very Poor (301–400)**	Respiratory illness to the people on prolonged exposure
Moderate (101–200)	Breathing discomfort to the people with lung, heart disease, children and older adults	**Severe (>401)**	Respiratory effects even on healthy people

Daily and hourly city data, as well as daily and hourly Station data, is provided by CPCB. Station refers to the continuous pollution monitoring stations operated and maintained by the Central Pollution Control Board (CPCB) and the State Pollution Control Boards.

Exploring the data. To explore the dataset, first of all, we looked at how much data is available and what is its structure, then we looked at the time range and cities and pollutants covered. The following snippets and outputs show all the steps involved in the process of exploring the dataset.

1. **Loading associated libraries and the dataset**

```
import csv
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
import warnings
import os
import seaborn as sns
warnings.filterwarnings("ignore")
warnings.filterwarnings("ignore")
```

```
data = pd.read_csv(os.path.join(DATA_DIR,'city_day.csv'))
```

2. Time Range of the dataset

```
def getDurationOfData(data):
   print('DURATION OF DATA:\nThe data is between',
data['Date'].min(), 'and' , data['Date'].max())

getDurationOfData(data)
```

```
DURATION OF DATA:
The data is between 2015-01-01 and 2020-07-01
```

3. Cities covered by the dataset

```
def getCities(data):
   cities = data['City'].value_counts().to_frame()
   print('CITIES COVERED IN THE DATA:')
   print('Total number of cities in the dataset :', len(cities))
   cities = cities.sort_index().index
   for i in cities:
       print(i)

getCities(data)
```

```
CITIES COVERED IN THE DATA:
Total number of cities in the dataset : 26
Ahmedabad
Aizawl
Amaravati
Amritsar
Bengaluru
Bhopal
Brajrajnagar
Chandigarh
Chennai
Coimbatore
Delhi
Ernakulam
Gurugram
Guwahati
Hyderabad
Jaipur
Jorapokhar
Kochi
Kolkata
Lucknow
Mumbai
Patna
Shillong
Talcher
Thiruvananthapuram
Visakhapatnam
```

Data Preprocessing. The data is processed in several ways for many reasons which are shown and discussed below. The preprocessing is done in two stages:

Stage 1: To allow for better visualisation for the analysis of the data and to focus on Delhi since the aim is to look at the AQI and different pollutant concentrations in Delhi.

Stage 2: To avoid encountering these NULL values while working with machine learning models.

We start by describing the data preprocessing done in Stage 1.

Stage 1. Since a lot of pollutants have missing values, visualising them alone will do no good as they have no values. So, we start by merging some pollutants. Benzene, Toluene and Xylene are merged into one column(pollutant) which is named BTX to represent all three of them.

```
#merging columns
def mergeColumns(data):
    data['Date'] = pd.to_datetime(data['Date'])
    data['BTX'] = data['Benzene'] + data['Toluene'] + data['Xylene']
    data.drop(['Benzene','Toluene','Xylene'], axis=1)
    return data

data=mergeColumns(data)
```

Next, to analyse the data we subset the pollutants to focus only on some of the important and crucial pollutants i.e. PM2.5, PM10, NO2, CO, SO2, O3, BTX:

```
def subsetColumns(data):
    pollutants = ['PM2.5', 'PM10', 'NO2', 'CO','SO2', 'O3', 'BTX']
    columns =  ['City', 'AQI', 'AQI_Bucket'] + pollutants
    all_columns = ['Date'] + columns
    data = data[all_columns]
    print(data.columns)
    return data, columns, pollutants
data, columns, pollutants = subsetColumns(data)
```

Now, our dataset looks like this:

	Date	City	AQI	AQI_Bucket	PM2.5	PM10	NO2	CO	SO2	O3	BTX
0	2015-01-01	Ahmedabad	NaN	NaN	NaN	NaN	18.22	0.92	27.64	133.36	0.02
1	2015-01-02	Ahmedabad	NaN	NaN	NaN	NaN	15.69	0.97	24.55	34.06	12.95
2	2015-01-03	Ahmedabad	NaN	NaN	NaN	NaN	19.30	17.40	29.07	30.70	25.45
3	2015-01-04	Ahmedabad	NaN	NaN	NaN	NaN	18.48	1.70	18.59	36.08	15.57
4	2015-01-05	Ahmedabad	NaN	NaN	NaN	NaN	21.42	22.10	39.33	39.31	28.68
...	...	...	...	...	...	...	...	...	...	...	...
29526	2020-06-27	Visakhapatnam	41.0	Good	15.02	50.94	25.06	0.47	8.55	23.30	15.04
29527	2020-06-28	Visakhapatnam	70.0	Satisfactory	24.38	74.09	26.06	0.52	12.72	30.14	3.33
29528	2020-06-29	Visakhapatnam	68.0	Satisfactory	22.91	65.73	29.53	0.48	8.42	30.96	0.02
29529	2020-06-30	Visakhapatnam	54.0	Satisfactory	16.64	49.97	29.26	0.52	9.84	28.30	0.00
29530	2020-07-01	Visakhapatnam	50.0	Good	15.00	66.00	26.85	0.59	2.10	17.05	NaN

29531 rows × 11 columns

Cities and Pollution. Before we focus on Delhi, let us analyse the motivation behind keeping Delhi the focus of this air pollution project. To understand where Delhi stood in the country in terms of air pollution levels, we ranked the 26 cities available in the data, based on their Air Quality Index and the important pollutants. And these are the results.

To do so, we grouped each pollutant by city and calculated the mean concentration of the pollutant for each city and finally sorted the values in decreasing order. We have produced an output of the top 10 cities in each pollutant category by the mean concentration of the pollutant over the years.

```
def max_polluted_city(pollutant, data):
   x1=data[[pollutant,'City']].groupby(["City"]).mean().sort_values(
by=pollutant,ascending=False).reset_index()
   x1[pollutant] = round(x1[pollutant],2)
   return x1[:10].style.background_gradient(cmap='coolwarm')
```

This process is carried out for all pollutants and AQI, we have mentioned them below:

1. **Particulate Matter 2.5 (PM2.5)**

```
pm25 = max_polluted_city('PM2.5', data)
```

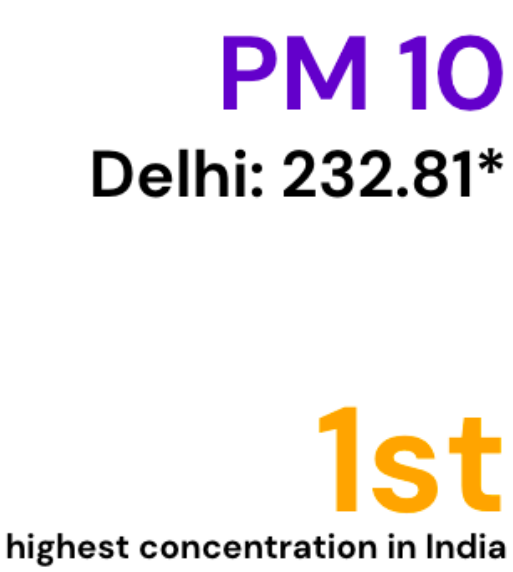
	City	PM2.5
0	Patna	123.500000
1	Delhi	117.200000
2	Gurugram	117.100000
3	Lucknow	109.710000
4	Ahmedabad	67.850000
5	Kolkata	64.360000
6	Jorapokhar	64.230000
7	Brajrajnagar	64.060000
8	Guwahati	63.690000
9	Talcher	61.410000

The average concentration of PM2.5 in Delhi is 117.2 micrograms per cubic meter. How bad is this? According to the National Ambient Air Quality Standards, PM2.5 levels up to 40-60 micrograms per cubic meter are considered safe in India.

2. **Particulate Matter 10 (PM10)**

```
pm10 = max_polluted_city('PM10', data)
```

	City	PM10
0	Delhi	232.810000
1	Gurugram	191.500000
2	Talcher	165.770000
3	Jorapokhar	149.660000
4	Patna	126.750000
5	Brajrajnagar	124.220000
6	Jaipur	123.480000
7	Bhopal	119.320000
8	Guwahati	116.600000
9	Kolkata	115.630000



The average concentration of PM10 in Delhi is 232.81 micrograms per cubic meter. How bad is this? According to the National Ambient Air Quality Standards, PM10 levels up to 60-100 ug/m3 are considered safe in India.

3. Nitrogen dioxide (NO_2)

```
no2 = max_polluted_city('NO2', data)
```

	City	NO2
0	Ahmedabad	59.030000
1	Delhi	50.790000
2	Kolkata	40.400000
3	Patna	37.490000
4	Visakhapatnam	37.190000
5	Lucknow	33.240000
6	Jaipur	32.420000
7	Bhopal	31.350000
8	Coimbatore	28.780000
9	Hyderabad	28.390000

The average concentration of NO2 in Delhi is 50.79 micrograms per cubic meter. According to the National Ambient Air Quality Standards, NO2 levels up to 40-80 ug/m3 are considered safe in India.

4. **Sulphur dioxide (SO_2)**

```
so2 = max_polluted_city('SO2', data)
```

	City	SO2
0	Ahmedabad	55.250000
1	Jorapokhar	33.650000
2	Talcher	28.490000
3	Patna	22.130000
4	Kochi	17.600000
5	Delhi	15.900000
6	Mumbai	15.200000
7	Guwahati	14.660000
8	Amaravati	14.260000
9	Bhopal	13.060000

The average concentration of SO2 in Delhi is 15.9 micrograms per cubic meter. According to the National Ambient Air Quality Standards, SO2 levels up to 50-80 ug/m3 are considered safe in India.

5. **Carbon monoxide(CO)**

```
co = max_polluted_city('CO', data)
```

	City	CO
0	Ahmedabad	22.190000
1	Lucknow	2.130000
2	Delhi	1.980000
3	Talcher	1.850000
4	Bengaluru	1.840000
5	Brajrajnagar	1.800000
6	Ernakulam	1.630000
7	Patna	1.530000
8	Kochi	1.300000
9	Gurugram	1.260000

The average concentration of CO in Delhi is 1.98 micrograms per cubic meter, which is almost safe as the acceptable concentration according to the National Ambient Air Quality Standards is below 2-4 micrograms per meter cube.

6. **Ozone (O3)**

```
o3 = max_polluted_city('O3', data)
```

	City	O3
0	Bhopal	59.850000
1	Delhi	51.320000
2	Jaipur	46.720000
3	Ahmedabad	39.160000
4	Amaravati	38.120000
5	Visakhapatnam	37.520000
6	Patna	37.250000
7	Lucknow	36.920000
8	Thiruvananthapuram	34.680000
9	Gurugram	34.410000

The average concentration of O3 in Delhi is 51.32 micrograms per cubic meter, which is safe as the acceptable concentration according to the National Ambient Air Quality Standards is below 100-180 ug/m3 is considered safe.

7. **BTX (Benzene, Toluene, Xylene)**

```
btx = max_polluted_city('BTX', data)
```

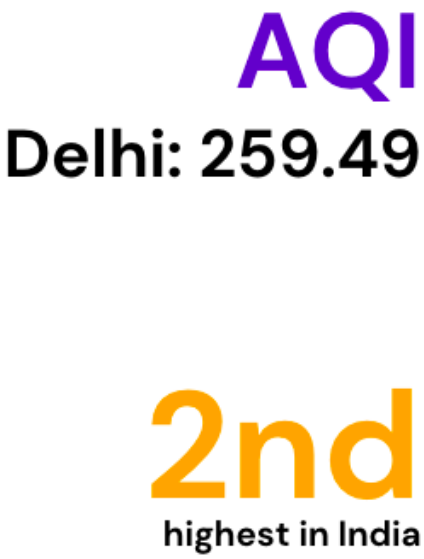
	City	BTX
0	Kolkata	38.230000
1	Ahmedabad	37.110000
2	Delhi	26.860000
3	Patna	17.430000
4	Visakhapatnam	15.030000
5	Gurugram	14.600000
6	Amritsar	14.580000
7	Hyderabad	10.730000
8	Chandigarh	9.090000
9	Amaravati	3.680000

The average concentration of BTX in Delhi is 26.86 micrograms per cubic meter.

8. AQI (Air Quality Index)

```
aqi = max_polluted_city('AQI', data)
```

	City	AQI
0	Ahmedabad	452.120000
1	Delhi	259.490000
2	Patna	240.780000
3	Gurugram	225.120000
4	Lucknow	217.970000
5	Talcher	172.890000
6	Jorapokhar	159.250000
7	Brajrajnagar	150.280000
8	Kolkata	140.570000
9	Guwahati	140.110000



We know that AQI 259.49 falls in the Poor category. And AQI levels up to 100 are considered satisfactory. So Delhi is 2.6x worse if we look at the satisfactory values.

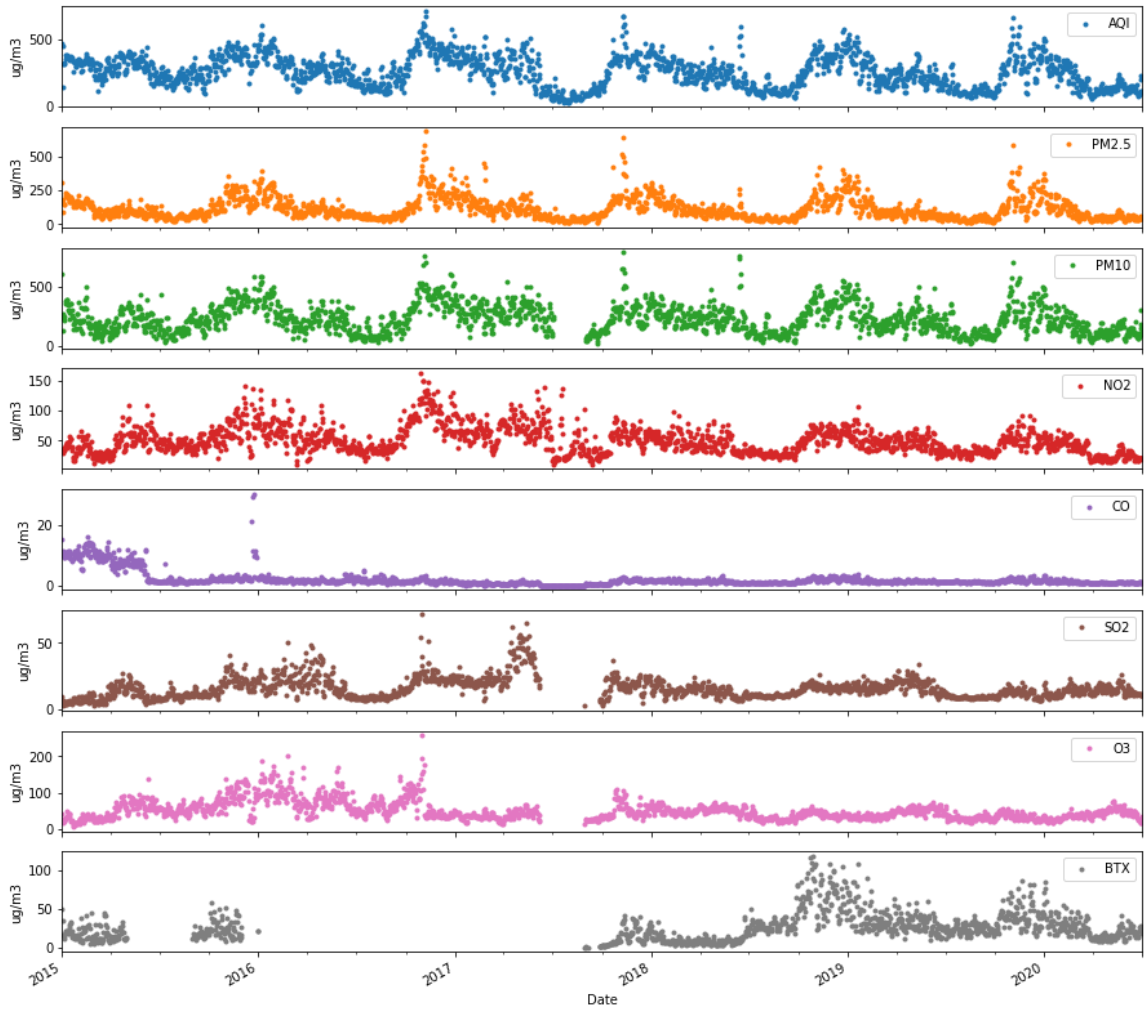
Continuing the Stage 1 processing, we perform the last step of extracting data of Delhi from the complete dataset:

```
delhi_data = data[data['City']=='Delhi']
delhi_data.set_index('Date',inplace=True, drop = False)
```

Visualisation of data after Stage 1 preprocessing. To understand the levels and concentrations of pollutants and the trends and seasonality they follow, we perform the following visualisations:

Visualising the concentration of pollutants in the air over the years. First, to get an overall bird's eye view of the entire Delhi dataset and pollutants we look at the concentration of pollutants in the air over the entire time period of the dataset i.e. 2015 - 2020. The concentrations are measured in ug/m^3.

```
def getEntireDataPlot(delhi_data):
    axes = delhi_data[columns].plot(marker='.', linestyle='None',
figsize=(15, 15), subplots=True)
    for ax in axes:
        ax.set_xlabel('Date')
        ax.set_ylabel('ug/m3')

getEntireDataPlot(delhi_data)
```



We notice a trend of increasing values over the time period in each pollutant from 2015 and the beginning of 2020, however, in the later part of 2020 i.e. after March 2020, we observe some decrease in the values of the pollutants while some like O_3 remain more or less the same. This observation confirms the effect of lockdown on the levels of these pollutants. We also observe the presence of missing plots implying missing values in the dataset.

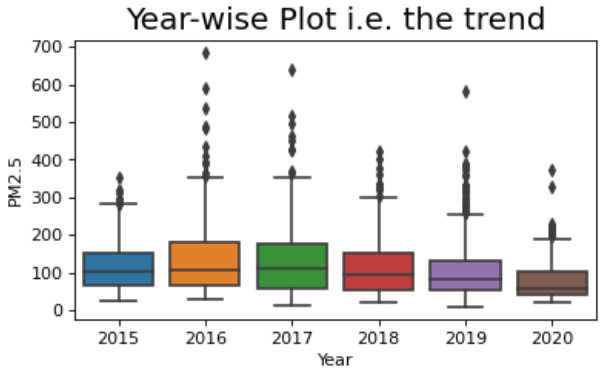
Visualising the amount of pollutants in air over the years and months. Next, to get a more detailed view of the trend and seasonality seen in each pollutant we look at the concentration of pollutants in the air over the entire time period of the dataset i.e. 2015 - 2020 and well as the months. The concentrations are measured in ug/m^3.

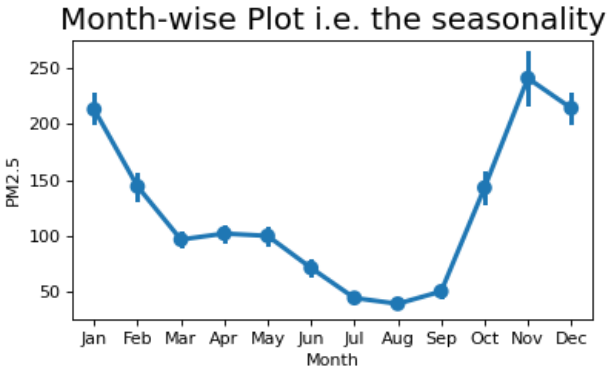
```
def trend_plot(delhi_data, value):
    data = delhi_data.copy()
    data['Year'] = [d.year for d in data.Date]
    data['Month'] = [d.strftime('%b') for d in data.Date]
    years = data['Year'].unique()
    fig, axes = plt.subplots(1, 2, figsize=(12,3), dpi= 80)
    sns.boxplot(x='Year', y=value, data=data, ax=axes[0])
    sns.pointplot(x='Month', y=value,
data=data.loc[~data.Year.isin([2015, 2020]), :])

    axes[0].set_title('Year-wise Plot i.e. the trend');
    axes[1].set_title('Month-wise Plot i.e. the seasonality')
    plt.show()
```

This process is carried out for all pollutants and AQI, we have mentioned the results below:

1. PM2.5

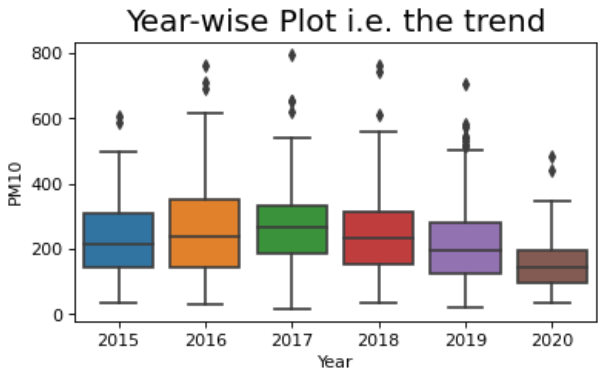
```
value='PM2.5'
trend_plot(delhi_data,value)
```

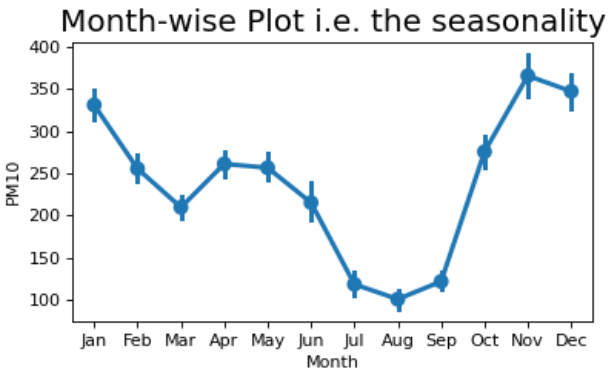




2. PM10

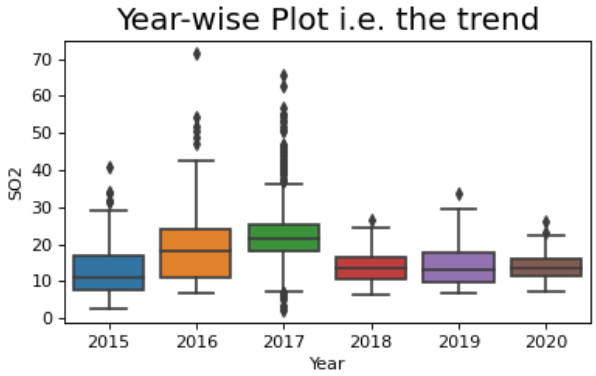
```
value='PM10'
trend_plot(data,value)
```

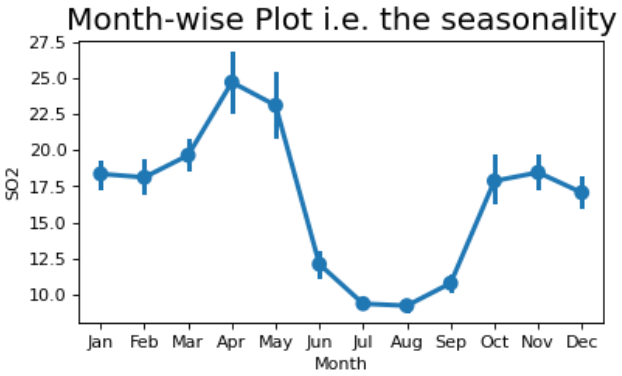




3. SO2

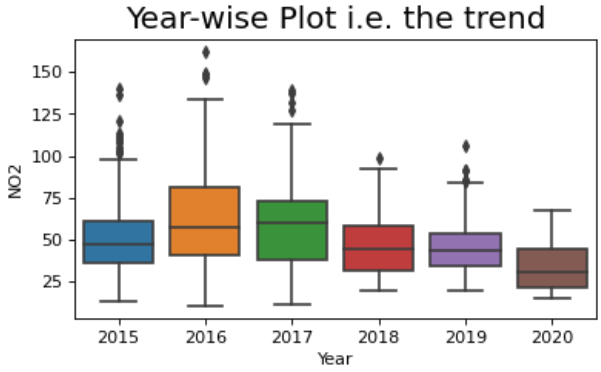
```
value='SO2'
trend_plot(data,value)
```

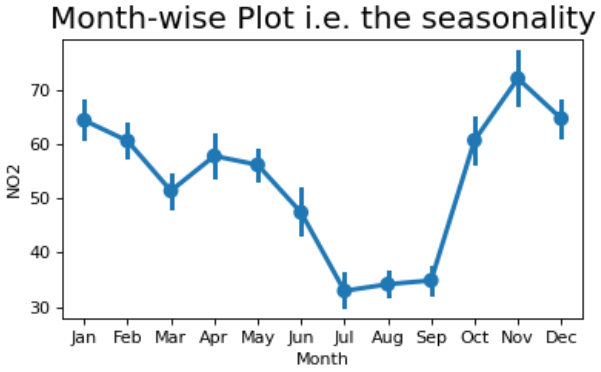




4. NO2

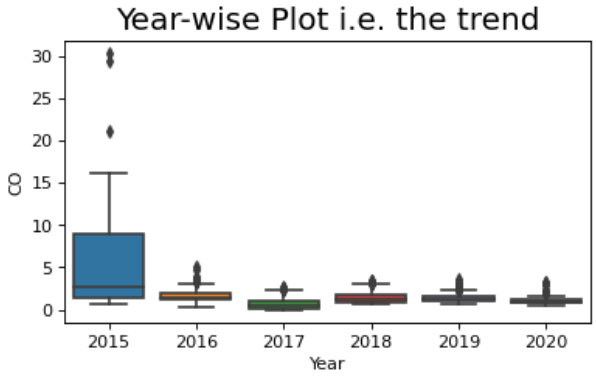
```
value='NO2'
trend_plot(data,value)
```

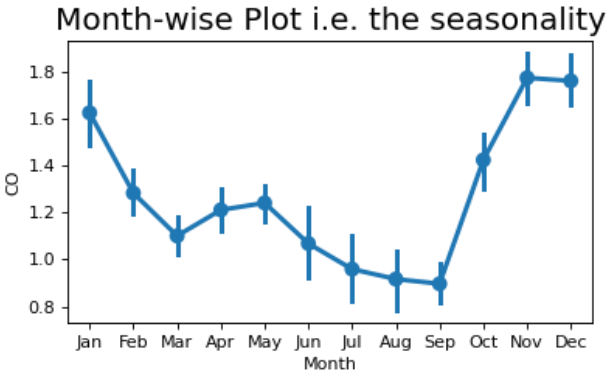




5. CO

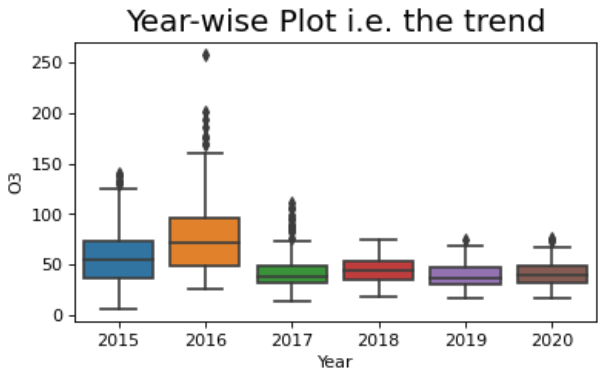
```
value='CO'
trend_plot(data,value)
```

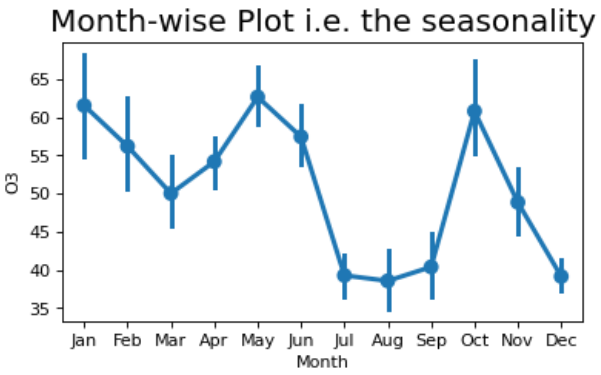




6. O3

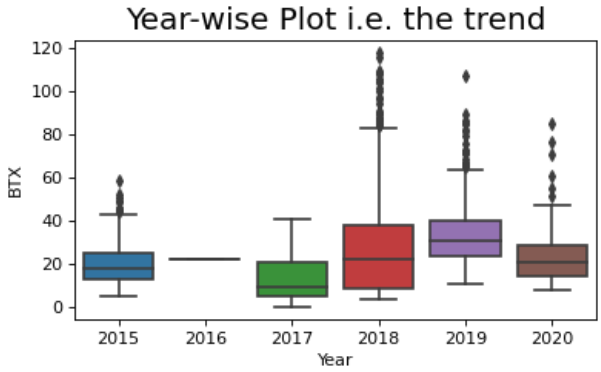
```
value='O3'
trend_plot(data,value)
```

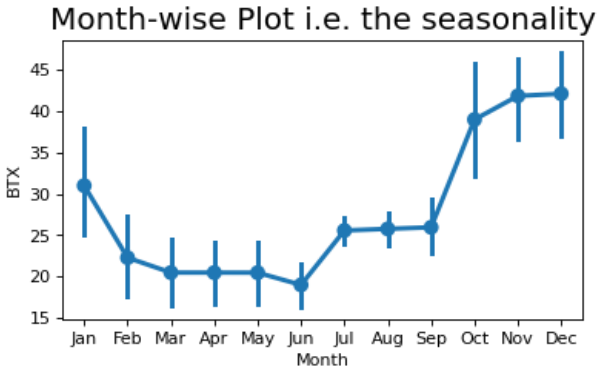




7. BTX

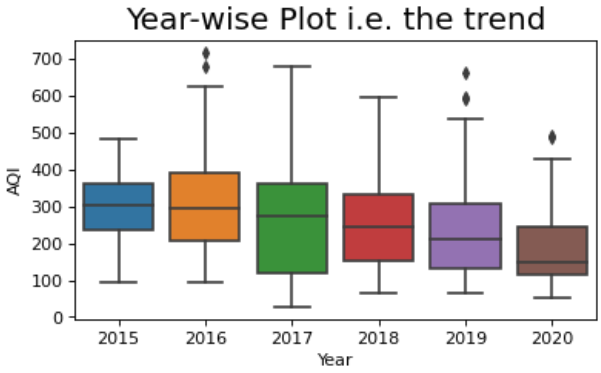
```
value='BTX'
trend_plot(data,value)
```

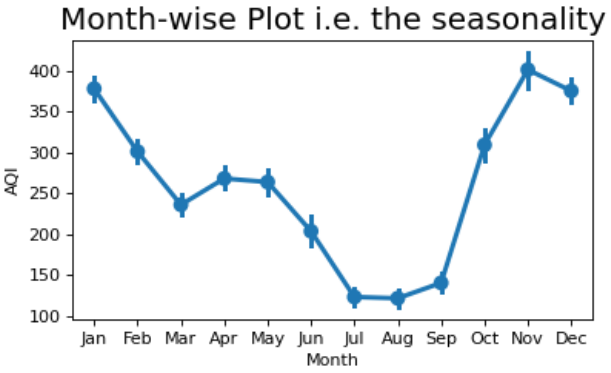




8. AQI

```
value='AQI'
trend_plot(data,value)
```





Delhi vs. pollutants. Next, for focused visualisation, we will look at Delhi's AQI and pollutant concentrations over a fixed time period. This will also help us look at the changes during the lockdown.

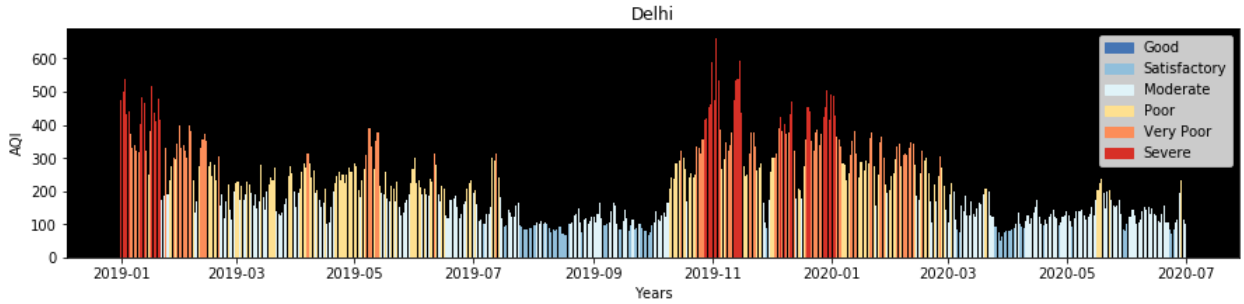
Visualisation of concentrations over the last two years. To do so, we have focused only on 2019 and 2020. To look at the concentrations for the last two years, we have used Bar Graphs, where each bar represents the pollutant concentration or AQI value for a particular day and is shown in different colours depending on the category of the quality of air for that day.

AQI: We already know the grading of AQI levels from Good to Severe based upon the values of the air quality index. This grading or types of air was used to colour the bars to categorise them into Good, Satisfactory, Moderate, Poor, Very Poor and Severe for each day from January 2019 to July 2020. The missing values were taken care of. bfill() is used to backward fill the missing values in the dataset. It will backward fill the NaN values that are present in the data.

```
def getColorBar(AQI_pivot):
    col = []
    for val in AQI_pivot:
        if val < 50:
            col.append('#4575b4')
        elif val > 50 and val < 101:
            col.append('#91bfdb') #cornflowerblue
        elif val > 100 and val < 201:
            col.append('#e0f3f8')
        elif val > 200 and val < 301:
            col.append('#fee090')
        elif val > 300 and val < 401:
            col.append('#fc8d59')
        else:
            col.append('#d73027')
    return col
```

```
def getAQILastTwoYears(delhi_data):
    filtered_city_day = delhi_data[delhi_data['Date'] >=
'2019-01-01']
    AQI = filtered_city_day['AQI']
    AQI.fillna(method='bfill',inplace=True)
    de = getColorBar(AQI)

    colors = {'Good':'#4575b4', 'Satisfactory':'#91bfdb',
'Moderate':'#e0f3f8', 'Poor':'#fee090', 'Very Poor':'#fc8d59',
'Severe':'#d73027'}
    labels = list(colors.keys())
    handles = [plt.Rectangle((0,0),1,1, color=colors[label]) for
```

```
label in labels]

    f, ((ax1)) = plt.subplots(1, 1, sharex='col', sharey='row',
figsize=(15,3))
    ax1.set_facecolor((0, 0, 0))
    ax1.bar(AQI.index, AQI, color = de, width = 0.75)
    ax1.legend(handles, labels, loc='upper right')
    ax1.title.set_text('Delhi')
    ax1.set_ylabel('AQI')
    ax1.set_xlabel('Years')

getAQILastTwoYears(delhi_data)
```



Pollutants: Since there is no grading available for each type of pollutant, we made use of the safe/acceptable values provided by the National Ambient Air Quality Standards to classify each day's pollutant concentration into either safe, almost safe or not safe. And using these three classes the bars were coloured.

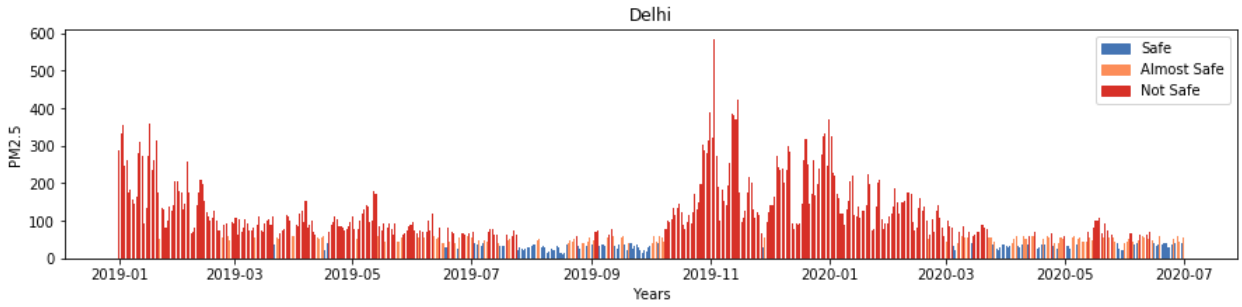
So, if NAAQS calls values below a-b ug/m^3 as acceptable, we have generated three classes, 0-a as Safe, a-b as almost safe and concentrations >b as not safe.

```
def getPollutantsLastTwoYears(delhi_data, cols, pollutant, safevalue,
almostsafe):
    filtered_city_day = delhi_data[delhi_data['Date'] >=
'2019-01-01']
    data_pollutants = filtered_city_day[cols]
    data_pollutants.fillna(method='bfill',inplace=True)
    col = []
```

```
    for val in data_pollutants[pollutant]:
        if val < safevalue:
            col.append('#4575b4')
        elif val < almostsafe:
            col.append('#fc8d59')
        else:
            col.append('#d73027')

    colors = {'Safe':'#4575b4', 'Almost Safe':'#fc8d59' ,'Not
Safe':'#d73027'}
    labels = list(colors.keys())
    handles = [plt.Rectangle((0,0),1,1, color=colors[label]) for
label in labels]

    f, ((ax1)) = plt.subplots(1, 1, sharex='col', sharey='row',
figsize=(15,3))
#     ax1.set_facecolor((0, 0, 0))
    ax1.bar(data_pollutants.index, data_pollutants[pollutant], color
= col, width = 0.75)
    ax1.legend(handles, labels, loc='upper right')
    ax1.title.set_text('Delhi')
    ax1.set_ylabel(pollutant)
    ax1.set_xlabel('Years')
```

a. PM2.5

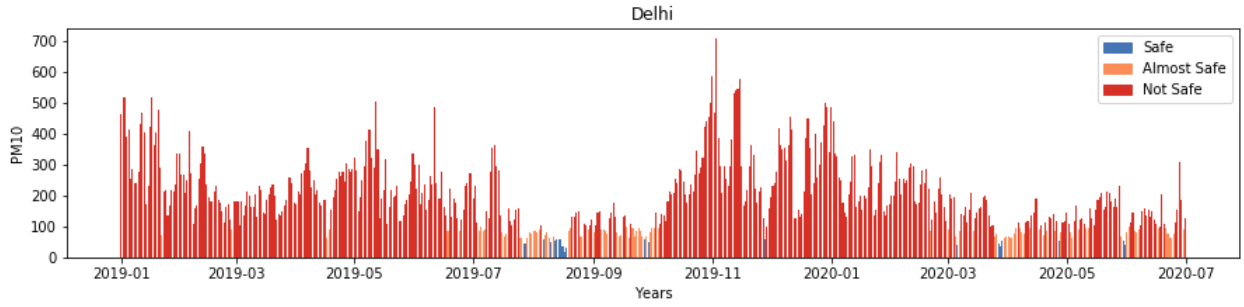
```
getPollutantsLastTwoYears(delhi_data, pollutants, 'PM2.5', 40, 60)
```



We see that the PM2.5 levels have been incredibly high in November 2019-January 2020 going up to 600. Additionally, we also see a decrease in the PM2.5 levels in 2020 after the lockdown had been imposed, however, these levels are still not very safe.

b. PM10

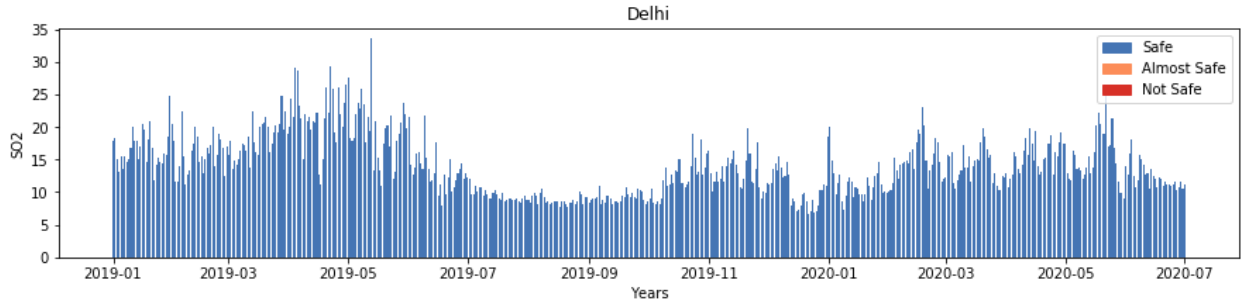
```
getPollutantsLastTwoYears(delhi_data, pollutants,'PM10', 60, 100)
```



We observe that the PM10 levels have been incredibly high throughout the two years but have spiked even more in November 2019 going up to 700. Additionally, we also see a decrease in the PM10 levels in 2020 after the lockdown had been imposed, however, these levels are still not safe.

c. SO2

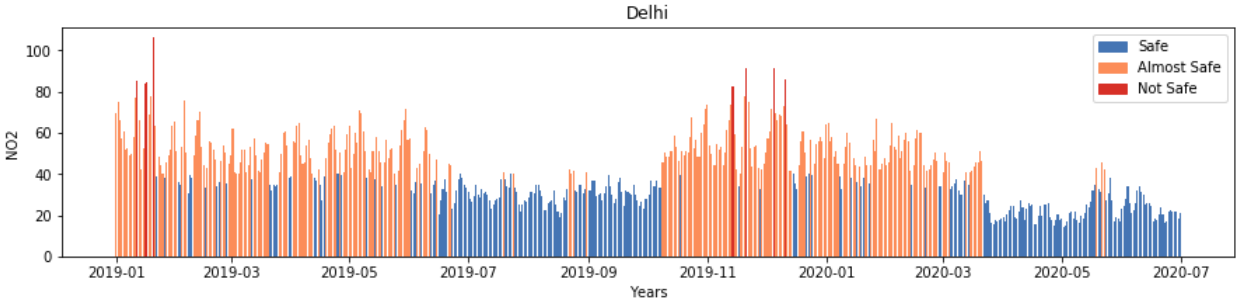
```
getPollutantsLastTwoYears(delhi_data, pollutants,'SO2', 50, 80)
```



SO2 levels have been safe throughout the two years. However, They seem to have been increasing in the year 2020.

d. NO2

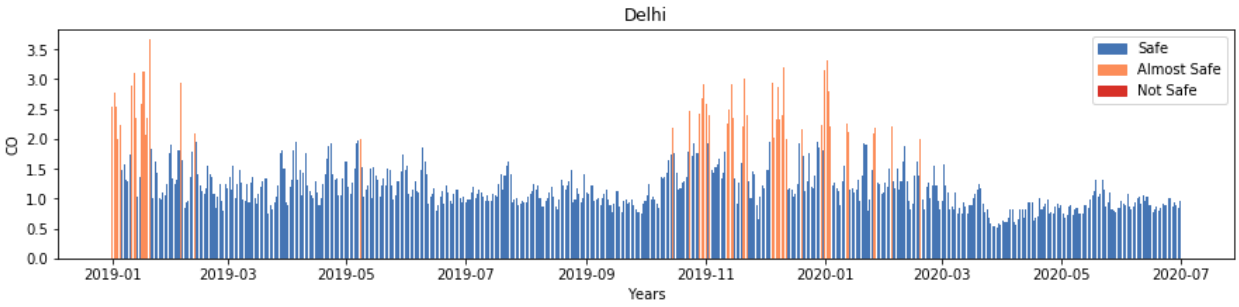
```
getPollutantsLastTwoYears(delhi_data, pollutants,'NO2', 40 , 80)
```



NO2 levels have been mostly safe in the two years in view. The levels seem to have increased in November 2019, however, we see a decrease in the NO2 levels starting March 2020, which was the onset of the lockdown period.

e. **CO**

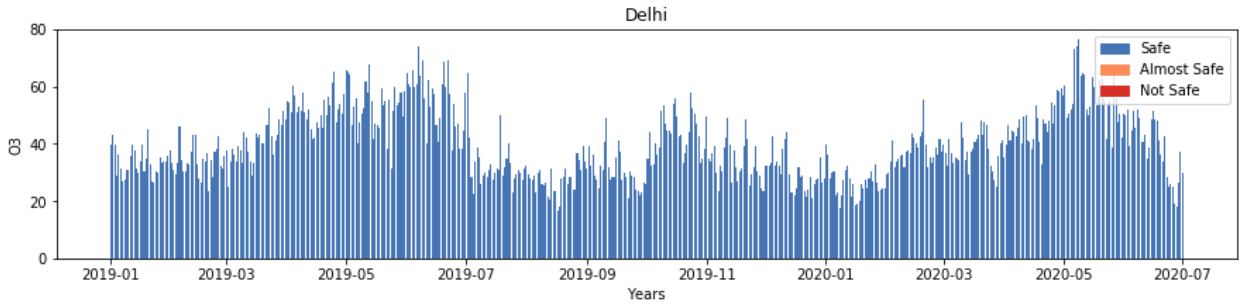
```
getPollutantsLastTwoYears(delhi_data, pollutants,'CO',2, 4)
```



CO concentration levels have increased in the January months of 2019 and 2020, along with November-December of 2019. But they have been low and safe since March 2020.

f. **O3**

```
getPollutantsLastTwoYears(delhi_data, pollutants,'O3', 100, 180)
```



Although Ozone concentration has been safe, it has seen a constant increase in the year 2020.

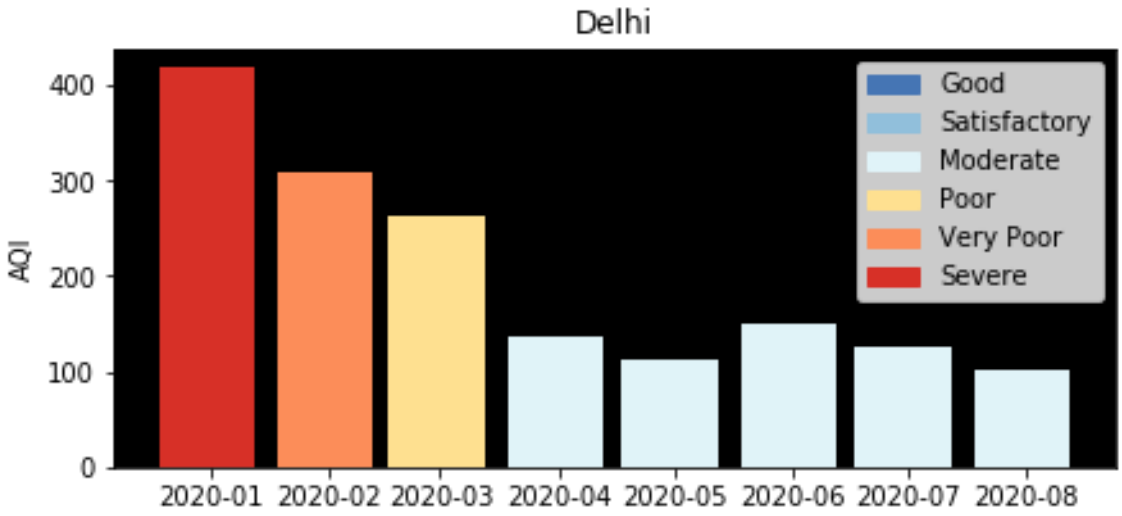
Now, to observe the effect of lockdown closely we perform two more types of visualisations.

Visualisation of concentrations in the year 2020. We again use bar graphs to show the concentration levels this time only for the year 2020. Where each bar represents the AQI values or pollutant concentrations for a particular month and is shown in different colours depending on the category of the quality of air for that month or acceptable/unacceptable level of pollutant concentration. The monthly concentrations are calculated by resampling the data with the frequency of the month.

AQI: Similar to the previous visualisation, AQI levels are represented using a bar graph and the bars are coloured based on the AQI category.

```
def get2020Data(delhi_data):
    filtered_city_day = delhi_data[delhi_data['Date'] >=
'2019-12-31']
    cols = ['Date', 'AQI'] + pollutants
    data_2020 = filtered_city_day[cols]
    data_2020 = data_2020.resample('M').mean()
    return data_2020

data_2020 = get2020Data(delhi_data)
data_2020
```

```
def getAQI2020(data_2020):
    fig, ((ax1)) =  plt.subplots(1, 1, sharex='col', sharey='row',
figsize=(7,3))
    ax1.set_facecolor((0, 0, 0))
    ax1.bar(data_2020.index, data_2020['AQI'], width = 25,
color=getColorBar(data_2020['AQI']))
    ax1.title.set_text('Delhi')
    ax1.set_ylabel('AQI')

    colors = {'Good':'#4575b4', 'Satisfactory':'#91bfdb',
'Moderate':'#e0f3f8', 'Poor':'#fee090', 'Very Poor':'#fc8d59',
'Severe':'#d73027'}
    labels = list(colors.keys())
    handles = [plt.Rectangle((0,0),1,1, color=colors[label]) for
label in labels]
    ax1.legend(handles, labels, loc='upper right')

getAQI2020(data_2020)
```



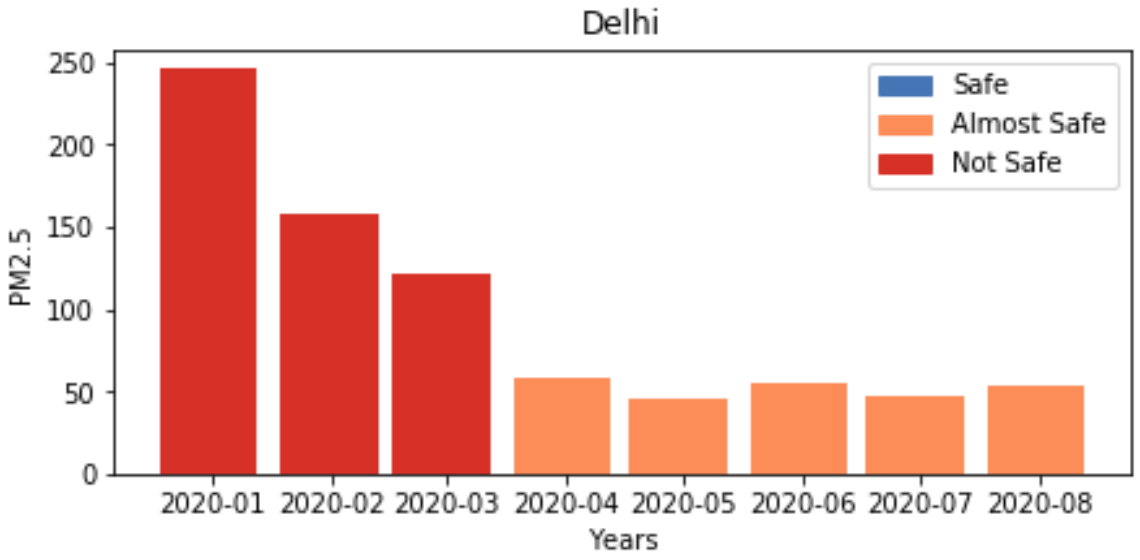
As we look at AQI in the year 2020 itself, specifically at the first seven months of 2020, we see that there has been a consistent improvement in the quality of air in Delhi.

Pollutants: Similar to the previous visualisation, pollutant concentration levels are represented using a bar graph and the bars are coloured based on the safe values for each pollutant.

```
def getPollutants2020(data_2020, pollutant, safevalue, almostsafe):
    pollutant_2020 = data_2020[pollutant]
    pollutant_2020.fillna(method='bfill',inplace=True)
    col = []
    for val in pollutant_2020:
        if val < safevalue:
            col.append('#4575b4')
        elif val < almostsafe:
            col.append('#fc8d59')
        else:
            col.append('#d73027')

    colors = {'Safe':'#4575b4', 'Almost Safe':'#fc8d59' ,'Not
Safe':'#d73027'}
    labels = list(colors.keys())
    handles = [plt.Rectangle((0,0),1,1, color=colors[label]) for
label in labels]

    f, ((ax1)) = plt.subplots(1, 1, sharex='col', sharey='row',
figsize=(7,3))
    ax1.bar(pollutant_2020.index, pollutant_2020, color = col, width
= 25)
    ax1.legend(handles, labels, loc='upper right')
    ax1.title.set_text('Delhi')
    ax1.set_ylabel(pollutant)
    ax1.set_xlabel('Years')
```

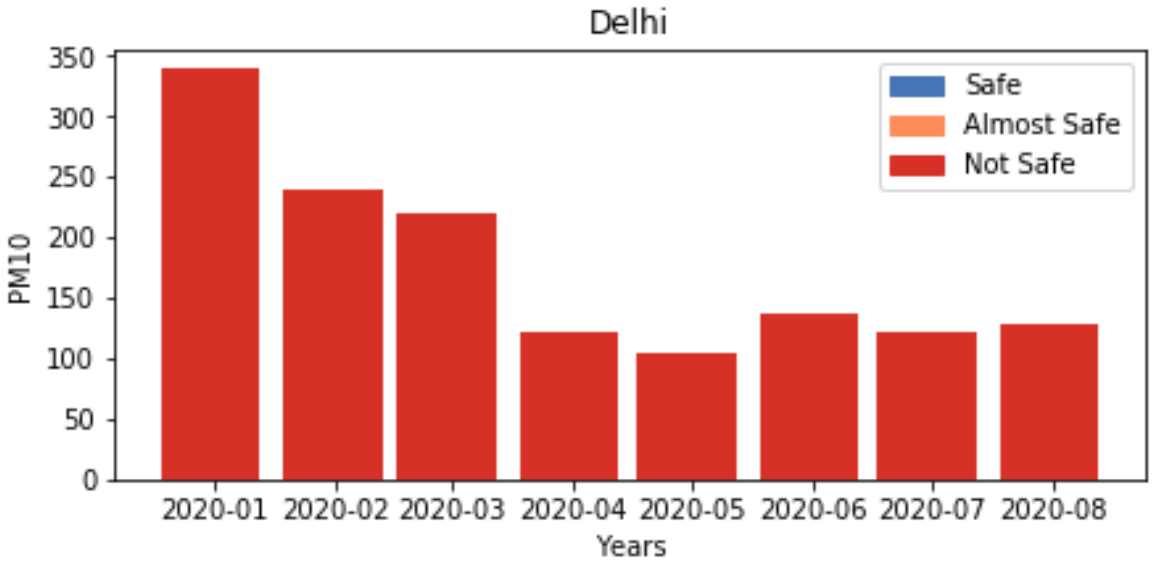
a. **PM2.5**

```
getPollutants2020(data_2020,'PM2.5', 40, 60)
```



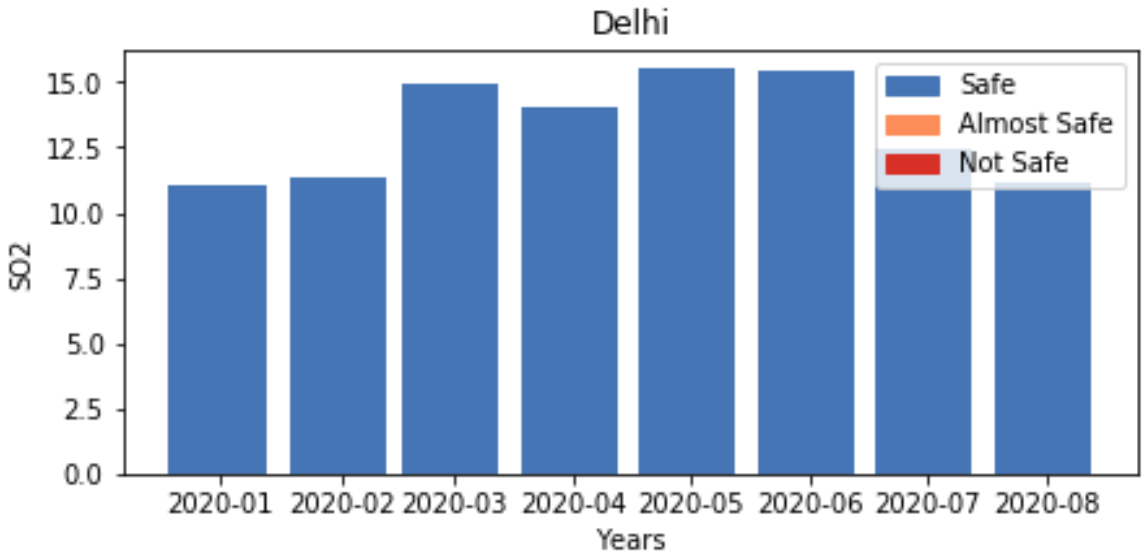
b. **PM10**

```
getPollutants2020(data_2020,'PM10', 60,100)
```



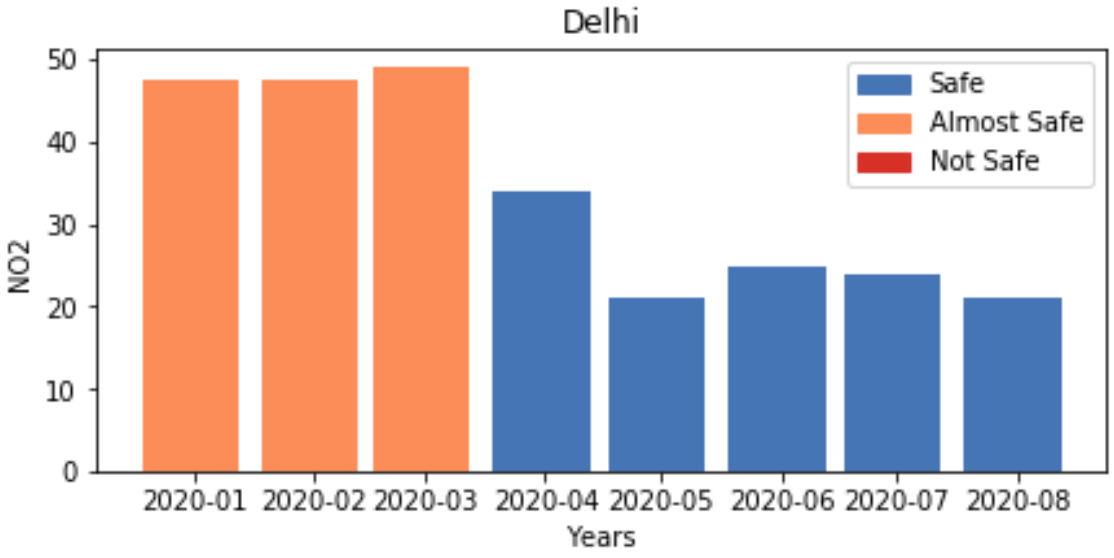
c. **SO2**

```
getPollutants2020(data_2020,'SO2', 50,80)
```



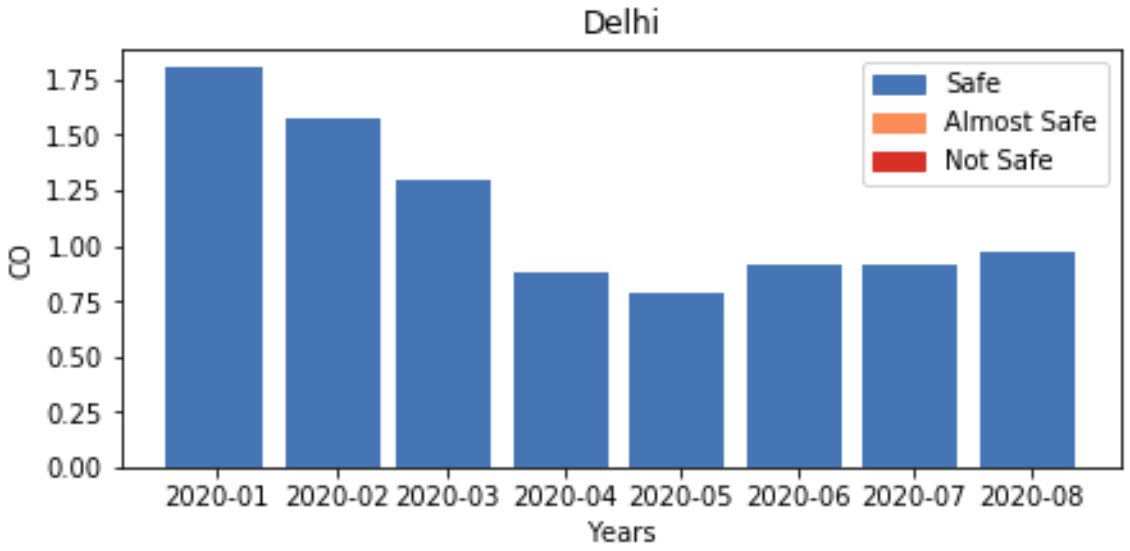
d. **NO2**

```
getPollutants2020(data_2020,'NO2', 40,80)
```



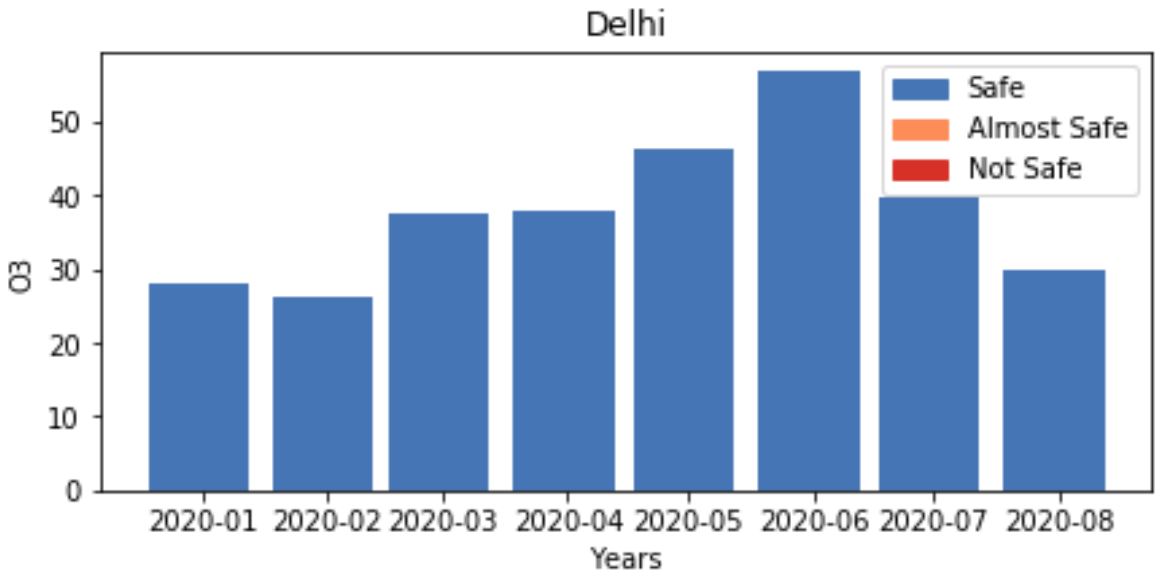
e. **CO**

```
getPollutants2020(data_2020,'CO', 2, 4)
```



f. **O3**

```
getPollutants2020(data_2020,'O3', 100, 180)
```



Visualisation of concentrations before and after Lockdown. We use bullet graphs (as information is presented in an easy to digest format) to show the AQI levels or pollutant concentrations and their category in two views: one view is of before lockdown

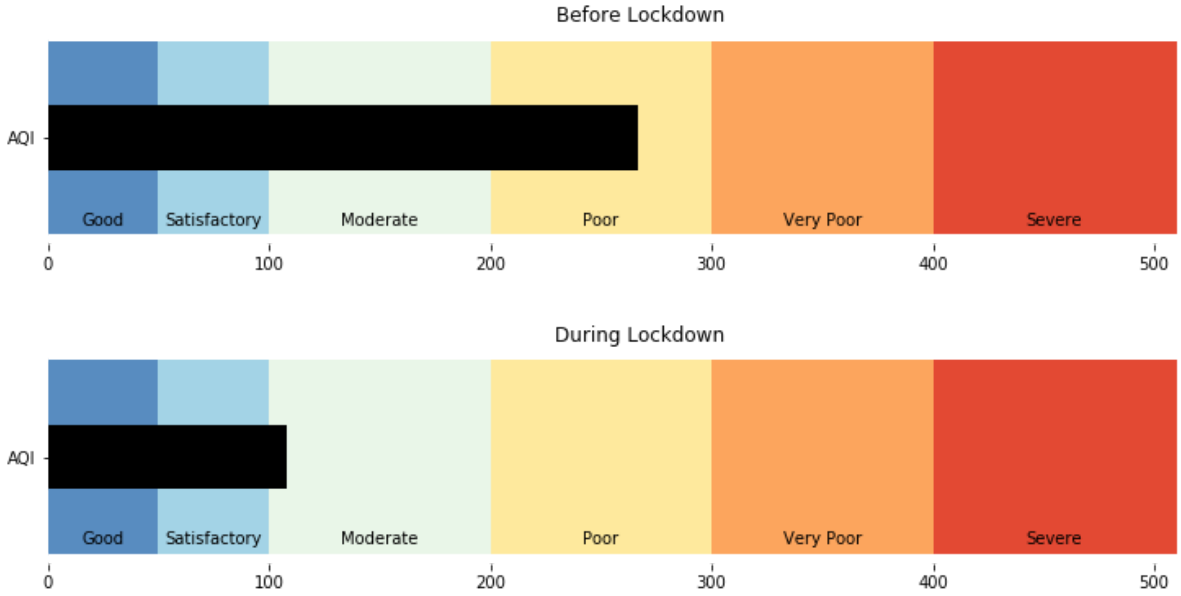
and the other is of after lockdown was imposed. To create these two views the data is separated into two sets: data_beforelockdown and data_afterlockdown.

```
def getBulletGraph(delhi_data, pollutant, limits, colours, labels):
    data_beforeLockdown = delhi_data['2015-01-01':'2020-03-22']
    data_afterLockdown = delhi_data['2020-03-23':'2020-05-01']
    palette = sns.color_palette(colours, len(limits))
    data_before = data_beforeLockdown[pollutant].mean()
    data_after = data_afterLockdown[pollutant].mean()
    fig, (ax1, ax2) = plt.subplots(2,1,figsize=(13, 6))
    plt.subplots_adjust(hspace = 0.5)
    ax1.set_yticks([1])
    ax1.set_yticklabels([pollutant])
    ax1.spines['bottom'].set_visible(False)
    ax1.spines['top'].set_visible(False)
    ax1.spines['right'].set_visible(False)
    ax1.spines['left'].set_visible(False)

    prev_limit = 0
    for idx, lim in enumerate(limits):
        ax1.barh([1], lim-prev_limit, left=prev_limit, height=15,
color=palette[idx])
        prev_limit = lim

    ax1.barh([1], data_before, color='black', height=5)

    # after lockdown
    ax2.set_yticks([1])
    ax2.set_yticklabels([pollutant])
    ax2.spines['bottom'].set_visible(False)
    ax2.spines['top'].set_visible(False)
    ax2.spines['right'].set_visible(False)
    ax2.spines['left'].set_visible(False)

    prev_limit = 0
    for idx, lim in enumerate(limits):
        ax2.barh([1], lim-prev_limit, left=prev_limit, height=15,
color=palette[idx])
```

```
        prev_limit = lim

    ax2.barh([1], data_after, color='black', height=5)

    ax1.set_title('Before Lockdown')
    ax2.set_title('During Lockdown')

    rects = ax1.patches

    for rect, label in zip(rects, labels):
        height = rect.get_height()
        ax1.text(
          rect.get_x() + rect.get_width() / 2,
          -height * .4,
          label,
          ha='center',
          va='bottom',
          color='black')
        ax2.text(
          rect.get_x() + rect.get_width() / 2,
          -height * .4,
          label,
          ha='center',
          va='bottom',
          color='black')
```

AQI. For AQI, the six categories that are displayed in different colours going from cool to warm analogous to from good to severe.

```
getBulletGraph(delhi_data, 'AQI',[50, 100, 200, 300, 400,
510],"RdYlBu_r",["Good", "Satisfactory", "Moderate", "Poor", 'Very
Poor', 'Severe'])
```



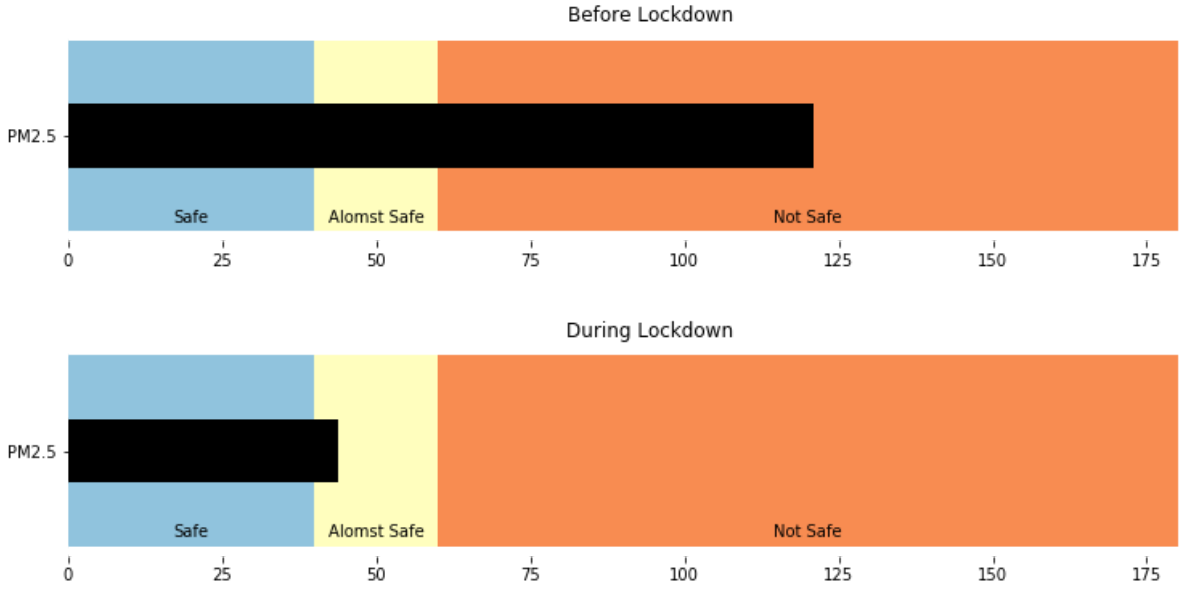
We see that in Delhi, before lockdown the air quality was poor and it reached almost satisfactory.

Pollutants. For pollutants, the National Ambient Air Quality Standards have given a range of values, below which the concentrations are declared acceptable. So, if NAAQS calls values below a-b ug/m^3 as acceptable, we have generated three classes, 0-a as Safe, a-b as almost safe and concentrations >b as not safe.

Here, these classes are used to generate bullet graphs for each pollutant before and after lockdown:

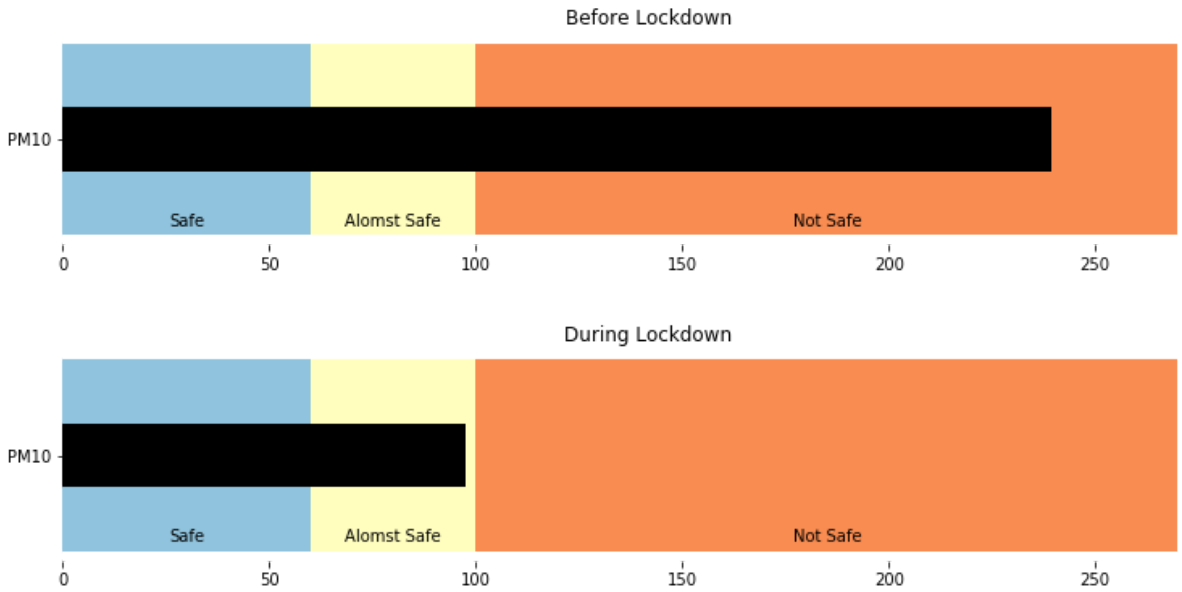
a. **PM2.5:** We notice that PM2.5 levels have significantly decreased during the lockdown.

```
getBulletGraph(delhi_data, 'PM2.5', [40, 60, 180],"RdYlBu_r",
["Safe", "Alomst Safe", "Not Safe"])
```



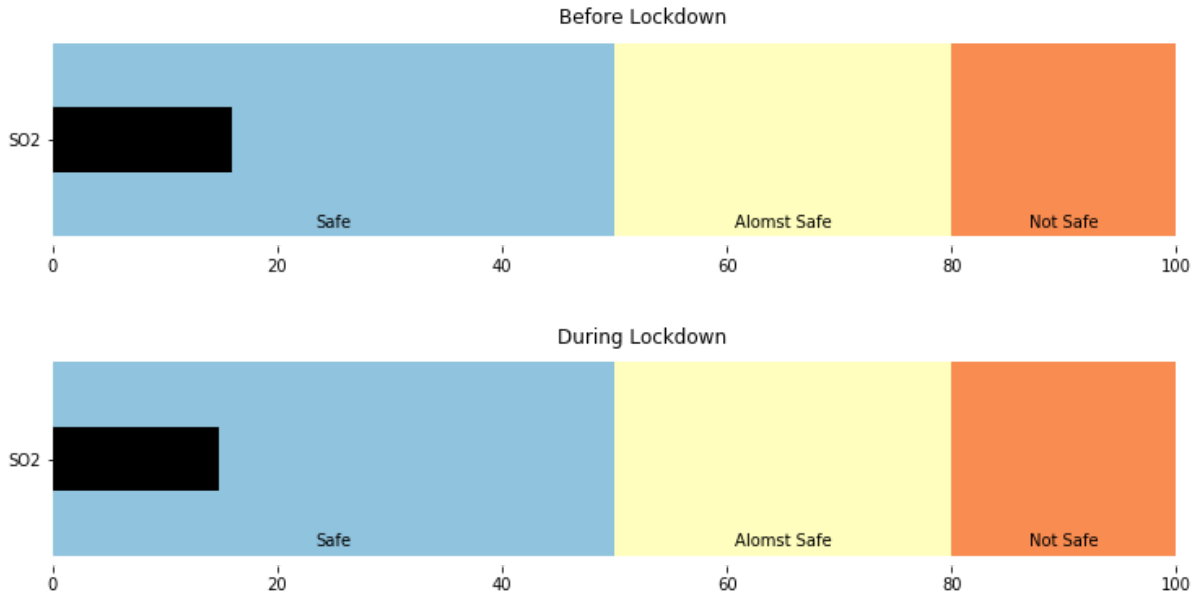
b. **PM10:** We notice that PM10 levels have also significantly decreased during the lockdown.

```
getBulletGraph(delhi_data, 'PM10', [60, 100, 270],"RdYlBu_r",
["Safe", "Alomst Safe", "Not Safe"])
```



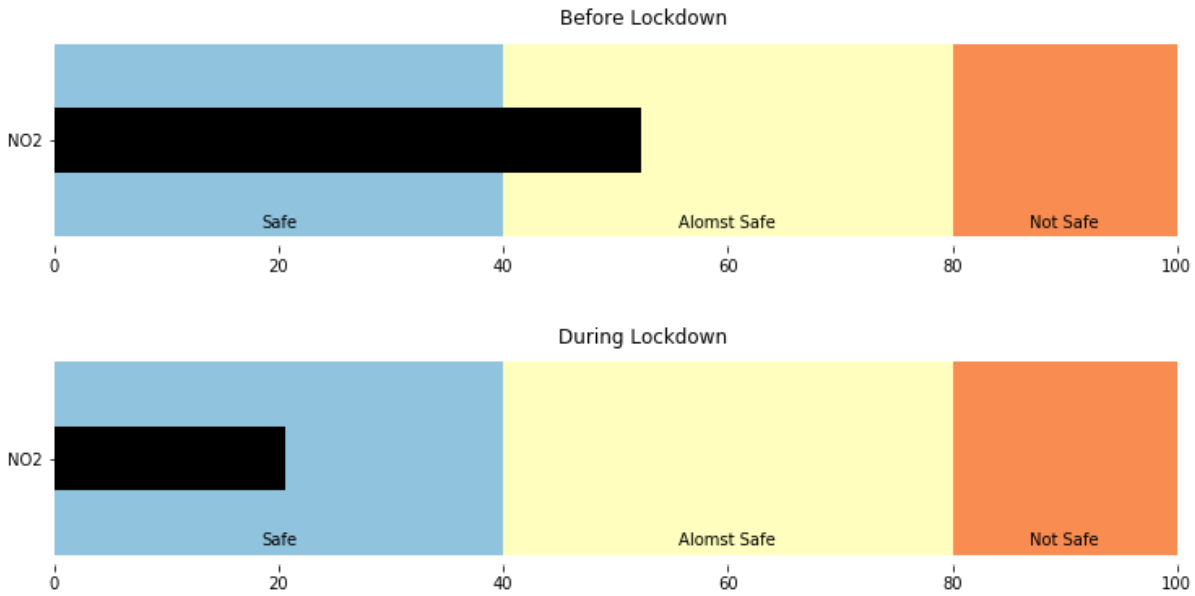
c. **SO2:** SO2 levels have remained almost the same.

```
getBulletGraph(delhi_data, 'SO2', [50, 80, 100],"RdYlBu_r", ["Safe",
"Alomst Safe", "Not Safe"])
```



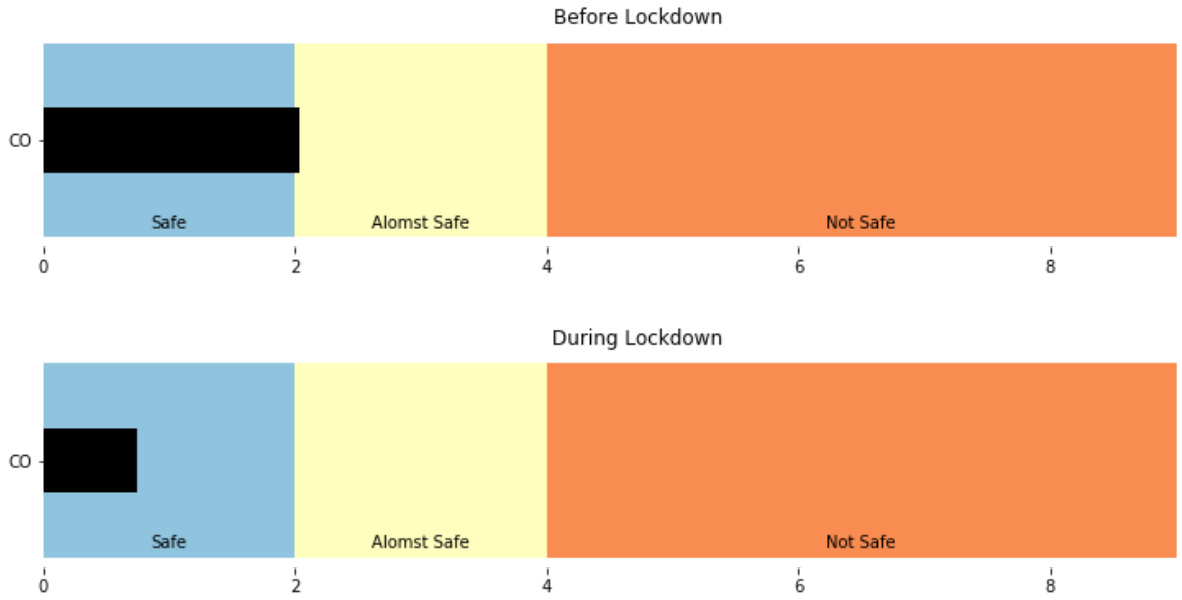
d. **NO2:** NO2 levels have also reduced during the lockdown.

```
getBulletGraph(delhi_data, 'NO2', [40, 80, 100],"RdYlBu_r", ["Safe",
"Alomst Safe", "Not Safe"])
```



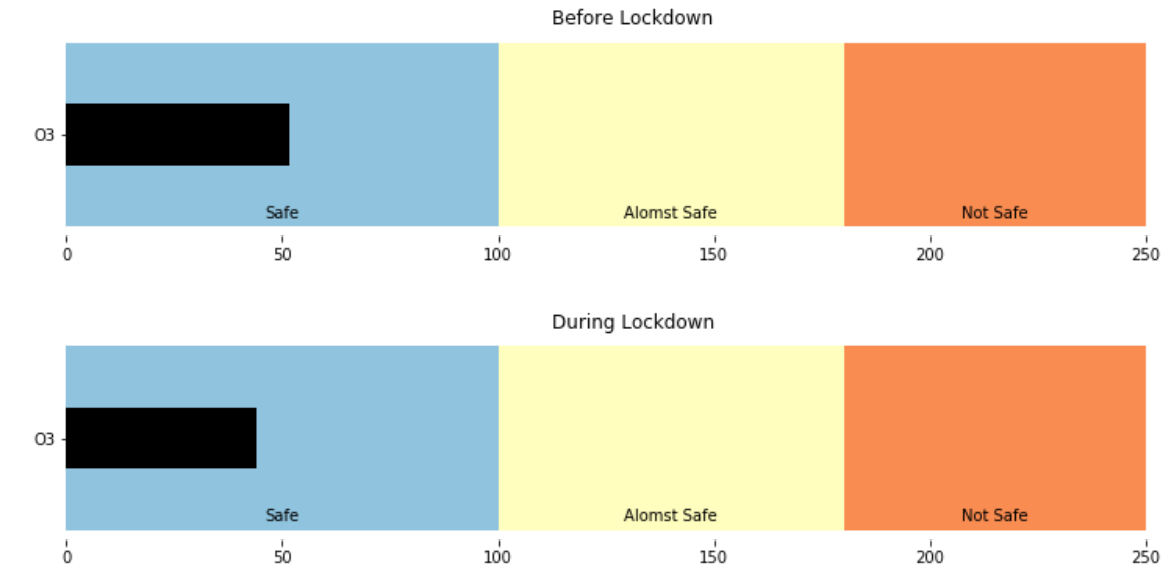
e. **CO:** CO levels have also reduced during the lockdown.

```
getBulletGraph(delhi_data, 'CO', [2,4, 9],"RdYlBu_r", ["Safe",
"Alomst Safe", "Not Safe"])
```



f. **O3:** O3 levels have stayed almost the same.

```
getBulletGraph(delhi_data, 'O3', [100, 180, 250],"RdYlBu_r", ["Safe",
"Alomst Safe", "Not Safe"])
```



We can very clearly see that lockdown had a pretty good effect on the quality of air.

Next, to predict the values of AQI and the concentration of pollutants in the air of Delhi, for the next year i.e. July 2020 - July 2021, we will look at a number of models, their training and testing and finally we will look at their results.

Methodology

Here, we will look at all the steps that were carried out to predict future air quality. We will start this section by discussing Stage 2 of data preprocessing. A quick recap of the two stages of data preprocessing:

Stage 1: To allow for better visualisation for the analysis of the data

Stage 2: To avoid encountering these NULL values while working with machine learning models.

Stage 2. Stage two consists of steps that are taken to allow proper functioning of the models and these steps also cater to the accuracy of the models.

Checking Missing Values: Before we start training models, we need to check if the dataset has any missing values and if there are any, we need to handle them so that we can train the models easily.

```
def getMissingValues(data):
    missing_val = data.isnull().sum()
    missing_val_percentage = 100 * data.isnull().sum() / len(data)
    missin_values_array = pd.concat([missing_val,
missing_val_percentage], axis=1)
    missin_values_array = missin_values_array.rename(columns =
                                                     {0 : 'Missing
Values', 1 : '% of Total Values'})
    missin_values_array = missin_values_array[
        missin_values_array.iloc[:,1] != 0].sort_values('% of Total
Values', ascending=False).round(1)
    return missin_values_array
```

```
missing_values = getMissingValues(delhi_data)
missing_values
```

	Missing Values	% of Total Values
BTX	781	38.9
SO2	110	5.5
O3	84	4.2
PM10	77	3.8
AQI	10	0.5
AQI_Bucket	10	0.5
PM2.5	2	0.1
NO2	2	0.1

Handling Missing Values: There are a number of methods to handle the presence of missing or NULL values. Some of them we have tried:

1. **Deleting rows with NULL values:** Let us look at the pros and cons associated with this method:
    - Complete removal of data with missing values results in a robust and highly accurate model.
    - Deleting a particular row or a column with no specific information is better since it does not have a high weightage.
    - Loss of information and data and works poorly if the percentage of missing values is high (say 30%), compared to the whole dataset

When we used this method, we found that the error generated by the models was less, but the data got distorted as the values started straying away from the trend they followed - due to missing data. Another issue with this approach was that some months

did not have any values at all - so when we grouped the data month-wise in the next step, those months now again had NULLs and hence this still increased the number of NULL values and thus, the model did not even run. So, this approach was discarded.

2. **Filling Missing Values in Database with most frequent values (i.e. Mode)**
3. **Filling Missing Values in Database with values separating the higher half from the lower half of a data sample (i.e. Median)**
4. **Filling Missing Values in Database with the average of all the values (i.e. Mean)**

All the above-mentioned methods have the following pros and cons:

- Prevent data loss which results in the removal of the rows and columns
- Imputing the approximations add variance and bias

Different models below discuss the method which was followed according to what gave the best results.

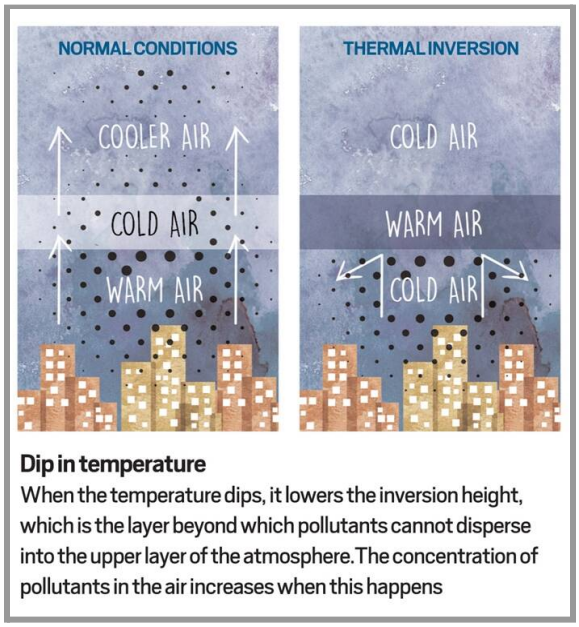
Models Used. We started with finding the most suitable models to train out time-series data on. And looking at the AQI plot of Delhi, we could make two inferences which we have also observed while visualising our data for AQI or even for the different pollutants.

- Presence of trend in AQI over the years
- Presence of a seasonal component that plays an important role

Next, we will look at what this seasonal component is and why it is observed.

Seasonality. The repeating short-term cycle in the data. There is an abnormal rise in the AQI values during the winter because of Winter inversion, the Valley effect and other factors such as dust storms, crop fires, burning of solid fuels for heating and firecracker-related pollution during Diwali or stubble burning.

Winter Inversion. In summer, the air in the planetary boundary layer is warmer and lighter and rises upwards more easily. This carries pollutants away from the ground and mixes them with cleaner air in the upper layers of the atmosphere however during winter, the planetary boundary layer is thinner as the cooler air near the earth's surface is dense. Mixing of the warn and cool air is limited during an inversion so pollutants are trapped inside the inversion layer. The cooler air is trapped under the warm air above it which forms a kind of atmospheric 'lid'. This phenomenon is called Winter Inversion. The figure below clearly explains this phenomenon. A similar and closely related phenomenon called the Valley Effect is also discussed.



Valley Effect. When the concentration of pollutants increases in low lying areas such as valleys because of certain weather conditions such as winter when cold air containing pollutants generated from vehicular emission becomes trapped by a layer of warmer air above the valley. This phenomenon is known as Valley Effect. The longer this air inversion lasts the worse the quality of air gets.

Wind Direction. October usually marks the withdrawal of monsoons in Northwest India. During monsoons, the prevalent direction of wind is easterly. These winds, which

travel from over the Bay of Bengal, carry moisture and bring rains to this part of the country. Once monsoon withdraws, the predominant direction of winds changes to north westerly. During summers, too, the direction of wind is north westerly and storms carrying dust from Rajasthan and sometimes Pakistan and Afghanistan.

Wind Speeds. High-speed winds are very effective at dispersing pollutants, but winters bring a dip in wind speed over all as compared to in summers. The combination of these meteorological factors makes the region prone to pollution. When factors such as farm fires and dust storms are added to the already high base pollution levels in the city, air quality dips further.

As we can observe from the figure above the “inversion” is very much literal. The first image in the figure shows the normal state of the atmosphere wherein the cool air lies above the warm air, however, in winter inversion, there is an inversion in the order of these layers i.e. the cool air now lies beneath the warm air - causing the concentration of pollutants to increase in winters. If we observe, the seasonality plot, plotted in the visualisation of pollutants we can see how in the winter months - i.e. January, February, March, November and December - the concentrations of pollutants are high.

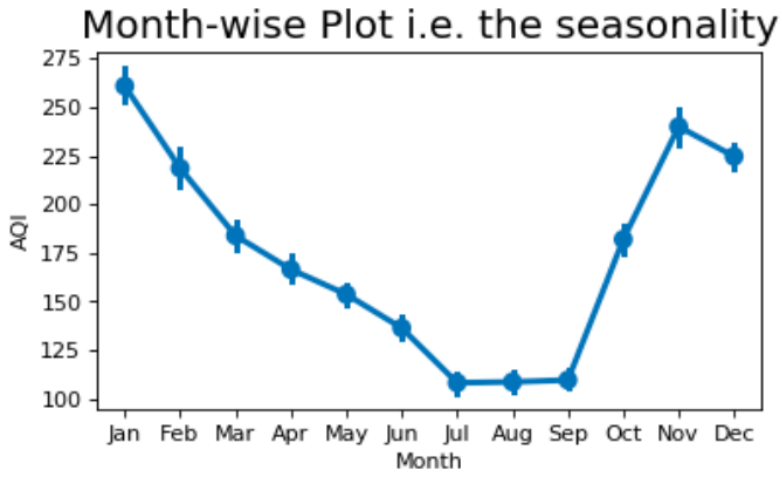


Figure. Seasonality in AQI

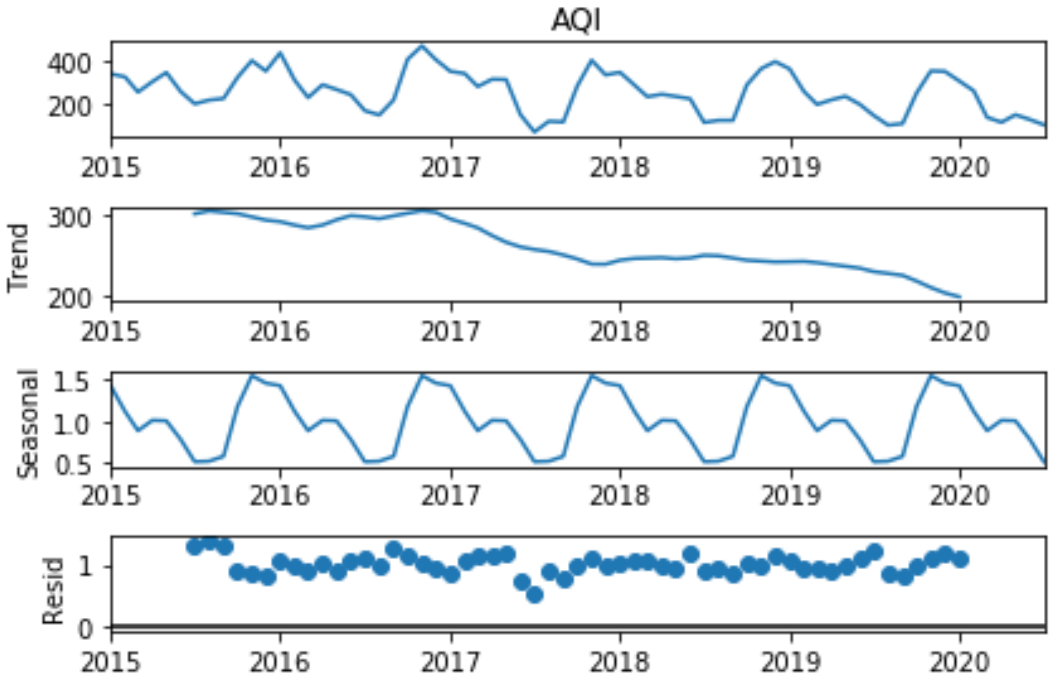
To check if the data has seasonality throughout, we perform seasonal decomposition on the data. Seasonal decomposition also describes the trends in the data. A multiplicative model is used to perform Seasonal Decomposition. It is Nonlinear, such as quadratic or exponential in which Changes increase or decrease over time. Why not a linear model? If a linear model is used, dimensions will increase hence complexity of the model will also increase. A multiplicative model suggests that the components are multiplied together as follows:

```
y(t) = Level * Trend * Seasonality * Noise
```

The seasonal decomposition of Delhi AQI data is shown below.

```
val = 'AQI'
final_data =
pd.DataFrame(index=np.arange('2015-01-01','2020-07-02',dtype='datetim
e64[D]'), columns = [val])
final_data[val] = delhi_data[val]
final_data=final_data.astype('float64')
final_data[val] =
final_data[val].fillna(final_data[val].mean(axis=0))
```

```
#THIS was performed after handling missing values
```



Looking at how seasonality and trend are consistently present in the data - we can now answer our question of which model to use. The idea here is to use a **Time Series Model**. We have identified all required parameter such as

- **Level**: The average value in the series.
- **Trend**: The increasing or decreasing value in the series.
- **Seasonality**: The repeating short-term cycle in the series.
- **Noise/Residue**: The random variation in the series.

Model 1. SARIMAX. Autoregressive Integrated Moving Average, or ARIMA, is one of the most widely used forecasting methods for univariate time series data forecasting. Although the method can handle data with a trend, it does not support time series with a

seasonal component. An extension to ARIMA that supports the direct modelling of the seasonal component of the series is called SARIMA. SARIMAX is an extension of the SARIMA model that also includes the modelling of exogenous variables (covariates and can be thought of as parallel input sequences that have observations at the same time steps as the original series).

We will now discuss the values for the parameters chosen: Configuring a SARIMA requires selecting hyperparameters for both the trend and seasonal elements of the series.

Trend Elements. There are three trend elements that require configuration. They are the same as the ARIMA model; specifically:

- p: Trend autoregression order.
- d: Trend difference order.
- q: Trend moving average order.

Seasonal Elements. There are four seasonal elements that are not part of ARIMA that must be configured; they are:

- **P**: Seasonal autoregressive order.
- **D**: Seasonal difference order.
- **Q**: Seasonal moving average order.
- **m**: The number of time steps for a single seasonal period.

Together, the notation for a SARIMA model is specified as:

```
SARIMA(p,d,q)(P,D,Q)m
```

Handling Missing Values. So, Mode and Median have a common problem that the most frequent value i.e. mode is NULL in some cases and the value separating the higher half from the lower half of a data sample (i.e. Median) is also NULL in most of the cases.

Using mode would not be a robust approach as the model will fail for most of the pollutants, it was also discarded. And for the same reason, the median was discarded too. So, finally, we used mean because that worked for most of the pollutants as the NULL values were filled with at least some values.

We were well-aware that using mean would incorporate a bias in the data so while evaluating our model we also calculated bias to understand its effects.

```
delhi_data = delhi_data.fillna(delhi_data.mean(axis=0))
delhi_data
```

Using SARIMAX. Coming back to our model, to use SARIMAX there are three steps, they are:

1. Define the model.
2. Fit the defined model.
3. Make a prediction with the fit model.

The snippets given below train the SARIMAX model on the Delhi_AQI data: First, the model is defined on Delhi City data and all Hyperparameters are initialized to zero with period interval(m) set as 12. auto_arima is used to find the Trend and Seasonal Element from which our model is prepared.

We will be training and looking at predictions in two phases to understand the effect of lockdown:

Phase 1. Without considering the effect of lockdown. For this phase, we trained the model on data from 2015-2018 and tested on 2019 data to test the accuracy and tune the parameters.

```
from statsmodels.tsa.statespace.sarimax import SARIMAX
from pmdarima import auto_arima;
```

```
auto_arima(y=Delhi_AQI,start_p=0,start_P=0,start_q=0,start_Q=0,seasonal=True, m=12)
```

```
#dividing into train and test:
train = Delhi_AQI[:41] #from 2015-2018
test = Delhi_AQI[42:54] #june 2018-july 2019
```

Next, after we have tuned our parameters depending on the predicted values, we go ahead to predict the future air quality i.e. the AQI for the year 2021 where we train the model on data from 2015-2020 and then predict 2020 July to 2021 July AQI values.

```
# Forming the model:
model=SARIMAX(train,order=(1,1,1),seasonal_order=(1,0,1,12),)
results=model.fit()
```

The predictions of the model for pollutant concentrations and AQI of Delhi for the year 2019, its evaluation and predictions for 2020- 2021 will be discussed in the next section.

Phase 2. Considering the effect of lockdown. For this phase, we trained the model on data from 2015-2019 and predicted the unknown values for 2021. (Since the model parameters have already been tuned by testing the model on 2019 data in the previous phase, that part is not repeated.) For this we only change the size of our training data , rest everything stays same:

```
#resizing training data:
train = Delhi_AQI #from 2015-2020
```

Model 2. LSTM. Long-short term memory is a common artificial recurrent neural network structure that is often used for time-series predictions. It stores past observations in memory, and while training it learns how to use this memory so as to not lose track of longer-term patterns. LSTMs are well regarded for a large variety of time-series tasks, although being vanilla neural network components some more configuration is required to set it up as compared to other models like ARIMA, SARIMAX, or Prophet. The LSTM is defined as can be seen below.

```
class LSTM(nn.Module):
 def __init__(self, input_size=1, hidden_layer_size=100, output_size=1):
   super().__init__()
   self.hidden_layer_size = hidden_layer_size

   self.lstm = nn.LSTM(input_size, hidden_layer_size)

   self.linear = nn.Linear(hidden_layer_size, output_size)
   self.hidden_cell = (torch.rand(1,1,self.hidden_layer_size).to(device),
                       torch.rand(1,1,self.hidden_layer_size).to(device))

 def forward(self, input_seq):
   lstm_out, self.hidden_cell = self.lstm(input_seq.view(len(input_seq) ,1,
-1), self.hidden_cell)
   predictions = self.linear(lstm_out.view(len(input_seq), -1))
   return predictions[-1]
```

In our case, we have opted for a single hidden layer with size of 100. Input and output size is one as there the input and output are always single-dimensional.

Being as vanilla as they are, some measures need to be taken to account for inconsistencies in data and accounting for random variations. Unlike Prophet, this system cannot simply handle outliers like holidays and festivals. In this case, our primary concern is dealing with the missing data points - missing out on invalid `Nans` can cause our model to fail - and account for the inconsistencies brought in through lockdown.

Input for LSTM. We need to define specific in-out sequences in order to express to

the LSTM the inputs and the proceeding result, which is the next day's result. We are operating with daily data here. For a sequence of `M` data points, if we want to pick a sequence with memory `N` from index `i`, we pick the index slice `[i:i+N]` for the input sequence, and index `[i+N+1]` for the output. For the task, we will be using the function as seen below: `create_inout_sequences`.

```
def create_inout_sequences(input_data, tw):
   inout_seq = []
   L = len(input_data)
   for i in range(L-tw):
       train_seq = input_data[i:i+tw]
       train_label = input_data[i+tw:i+tw+1]
       inout_seq.append((train_seq ,train_label))
   return inout_seq
```

Given the input data, it returns a list of tuples, where each tuple is a list of contiguous data points to the target window length specified and the element right after.

Handling missing values. LSTMs don't handle missing values on their own. A common issue encountered during training was when the LSTM kept giving Nan as the output for any input sequence where a single one of the entries was nan. So, in order to fix that we chose to simply eliminate the missing values altogether.

```
# Removing nans
 todel = []
 for index, item in enumerate(train_y):
   if math.isnan(item):
     # print(index, item)
     todel.append(index)
 train_y = np.delete(train_y, todel)
```

This way, there are no missing values in the dataset when we run our training or eval loops. Another option would be to replace the missing values with median, mode, or

a placeholder value. Quick experiments with PM2.5 concentration did not show substantial difference, and so we opted to keep it as it was.

Training and testing: First, we trained and tested our models on 2015-18 and 2019 data respectively to get parameters right and see how the accuracy results are turning out. Due to long training times, not much tuning could be done beyond a couple of runs, but the best results at this phase were taken for the next bit.

```
data = pd.read_csv(os.path.join(DATA_DIR,'city_day.csv'))
delhi_data = data[data['City']=='Delhi']
delhi_data['ds'] = pd.to_datetime(delhi_data['Date'])
delhi_data['y'] = delhi_data[gas]
```

```
temp_date = delhi_data.loc[delhi_data['ds'] <= '2019-07-01']
test, train = temp_date.loc[temp_date['ds']>='2018-07-01'],
temp_date.loc[temp_date['ds']<='2018-07-01']
```

```
# Creating the model
 model = LSTM().to(device)
 loss_function = nn.MSELoss()
 optimizer = torch.optim.Adam(model.parameters(), lr=0.0001)

 # Training the model
 epochs = 150

 for i in range(epochs):
   for seq, labels in train_inout_seq:
     optimizer.zero_grad()
     model.hidden_cell = (torch.rand(1, 1, model.hidden_layer_size).to(device),
                          torch.rand(1, 1, model.hidden_layer_size.to(device)))

     y_pred = model(seq)

     single_loss = loss_function(y_pred, labels)
     single_loss.backward()
     optimizer.step()
```

```
   if i%25 == 1:
     print(f'epoch: {i:3} loss: {single_loss.item():10.8f}')

 print(f'epoch: {i:3} loss: {single_loss.item():10.10f}')
```

```
model.eval()

 # It will predict for another year
 for i in range(future_window):
   seq = torch.FloatTensor(test_inputs[-train_window:]).to(device)
   with torch.no_grad():
     model.hidden = (torch.rand(1, 1, model.hidden_layer_size).to(device),
                     torch.rand(1, 1, model.hidden_layer_size).to(device))
     test_inputs.append(model(seq).item())
  actual_predictions =
scaler.inverse_transform(np.array(test_inputs[-future_window:] ).reshape(-1,
1))
```

Predictions for 2021. For this phase, we trained the model on data from 2015-2020 and predicted the unknown values for 2021. This requires a change in the split of the train and test, and the rest of the code remains unchanged.

```
temp_date = delhi_data
test, train = temp_date.loc[temp_date['ds']>='2020-07-01'],
temp_date.loc[temp_date['ds']<='2020-07-01']
```

Model 3. Prophet. Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well. It uses a simple, modular regression model that often works well with default parameters. That allows analysts to select the components relevant to their forecasting problem and easily make adjustments as needed.

Parameters in Prophet: There are many parameters that Prophet can consider while modelling a time series including national and regional holidays. The parameter we are most interested in is seasonality. Seasonalities are estimated in the Prophet using a partial Fourier sum.

Seasonal effects s(t) are approximated by the following function:

$$s(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi nt}{P}\right) + b_n \sin\left(\frac{2\pi nt}{P}\right) \right)$$

The number of terms N in the partial sum (the order) is a parameter that determines how quickly the seasonality can change. Increasing the number of Fourier terms allows the seasonality to fit faster-changing cycles, but can also lead to overfitting: N Fourier terms correspond to 2N variables used for modelling the cycle. The default Fourier order for yearly seasonality is 10. The default values are often appropriate, but they can be increased when the seasonality needs to fit higher-frequency changes, and generally be less smooth. The Fourier order can be specified for each built-in seasonality when instantiating the model, but we use the default only.

First we'll import the data and filter out the data only for Delhi:

```
import pandas as pd
data = pd.read_csv(os.path.join(DATA_DIR,'city_day.csv'))
delhi_data = data[data['City']=='Delhi']
```

We then split the data into training and test data:

```
test, train = delhi_data.loc[delhi_data['Date']>='2019-1-1'],
delhi_data.loc[delhi_data['Date']<='2019-1-1']
```

Note: We do not need to impute the data since Prophet understands there are missing points and fills the gaps as it trains.

Prophet follows the sklearn model API. We create an instance of the Prophet class and then call its fit and predict methods. The input to Prophet is always a dataframe with two columns: ds and y. The ds (datestamp) column should be of a format expected by Pandas, ideally YYYY-MM-DD for a date or YYYY-MM-DD HH:MM:SS for a timestamp. The y column must be numeric and represents the measurement we wish to forecast. We need to convert the time and date in the dataframe in the above format.

```
delhi_data['ds'] = pd.to_datetime(delhi_data['Date'])
delhi_data['y'] = delhi_data['NO2']
```

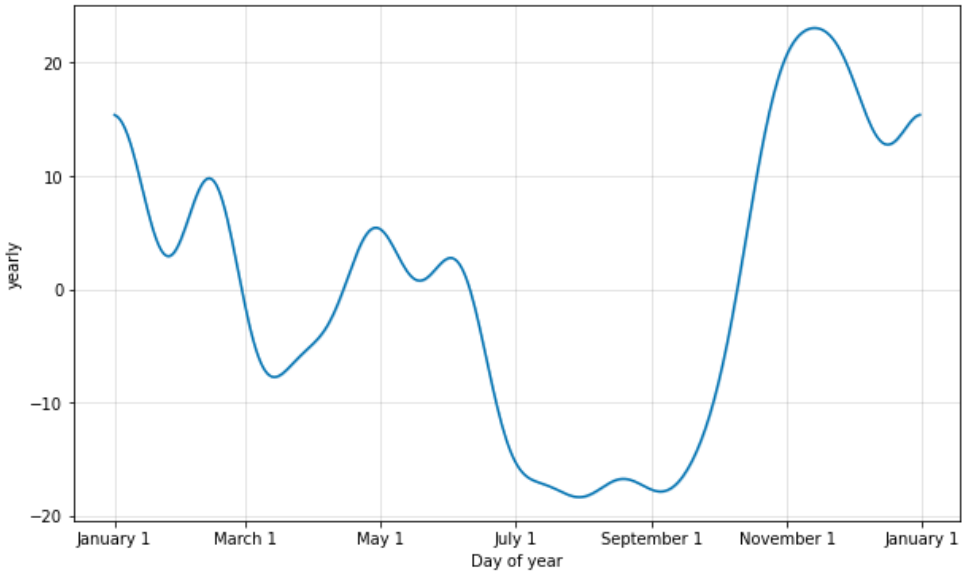
We fit the model by instantiating a new Prophet object. Any settings to the forecasting procedure are passed into the constructor. Then you call its fit method and pass in the historical dataframe. Even though we predict for the last 12 months, we pass the whole data for the training process. This is known as an in-sample forecast. Ideally, the model has seen the data before and would make a perfect prediction. Nevertheless, this is not the case as the model tries to generalize across all cases in the data.

```
m = Prophet(yearly_seasonality=2000)
m.fit(delhi_data)
```

We have plotted the seasonality with the partial Fourier of order 2000 a:

```
from prophet.plot import plot_yearly
a = plot_yearly(m)
```

The resultant plot is shown below:



Results and Discussion

Results. Now we will look at the predictions made by the models for the two tasks - one of predicting the AQI values for a year already present in the dataset and another for 2020-2021.

Model 1. SARIMAX. Here are the results for both phases of this model:

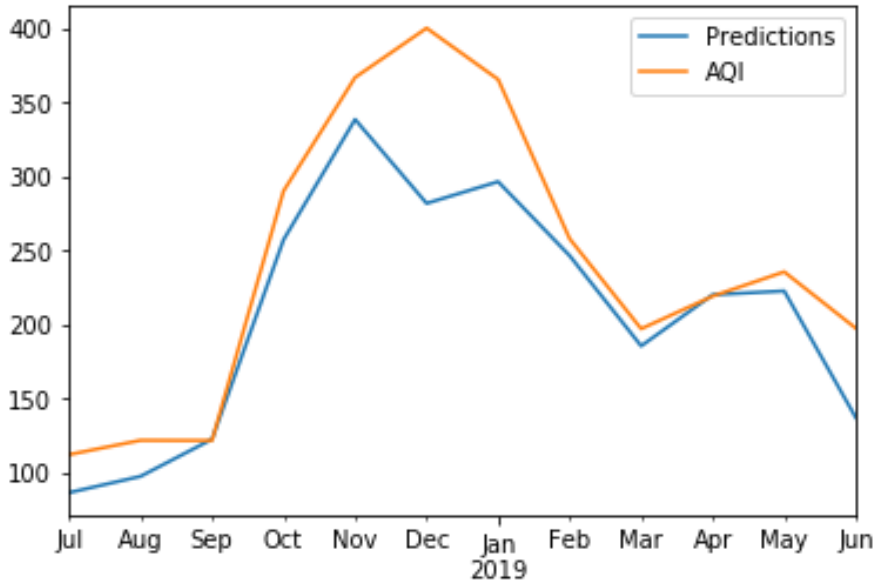
Phase 1. Without considering the effect of lockdown. For this phase, we trained the model on data from 2015-2018 and tested on 2019 data to test the accuracy and tune the parameters. We haven't considered the 2019-2020 data because it is an outlier due to lockdown and may give incorrect results.

First, we look at the predictions for the 2019(year already available) AQI values.

```
#predicted values:
predictions = results.predict(start=42, end=53,
typ='levels').rename('Predictions')
```

```
#Plotting predicted values with the true values:
```

```
predictions.plot(legend=True)
test.plot(legend=True);
```



The evaluation metric used is Root Mean Squared Error and Bias. We observe that the predicted mean AQI for Delhi in 2019 is 240.7142.

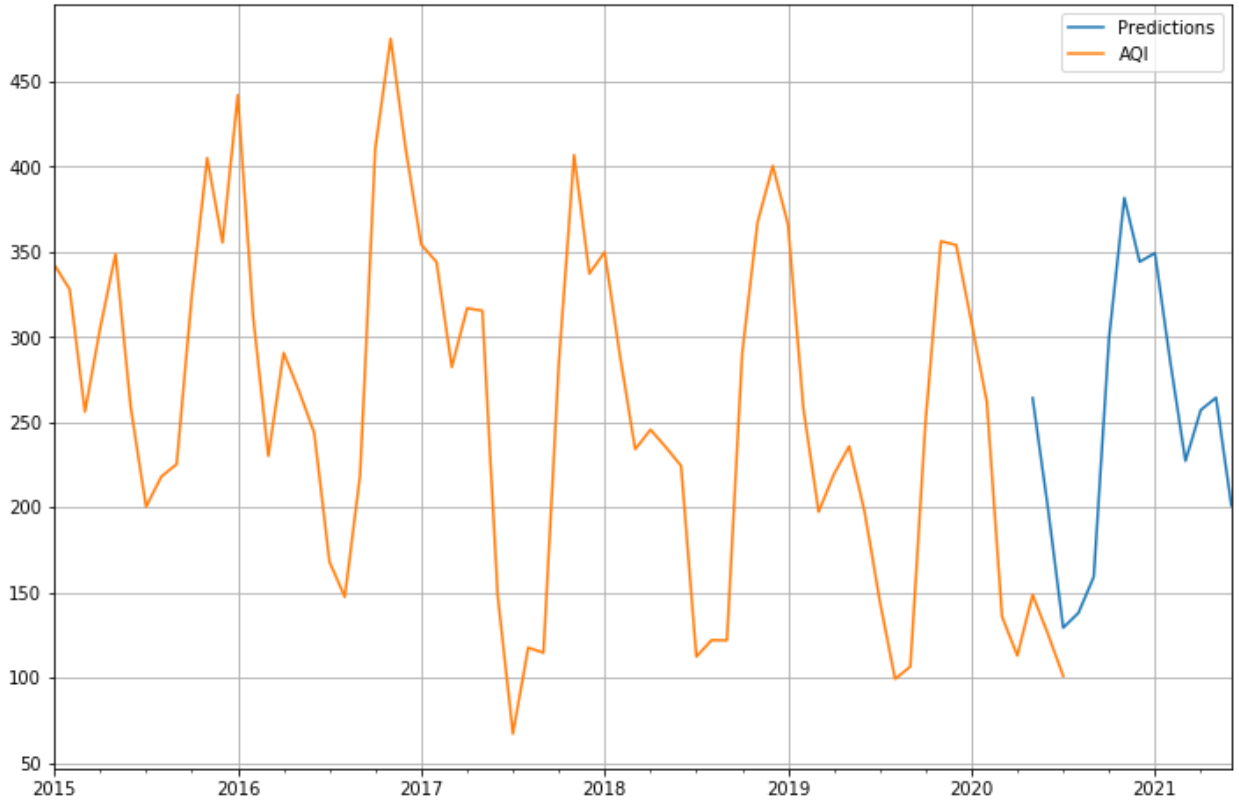
```
from sklearn.metrics import mean_squared_error
RMSE=np.sqrt(mean_squared_error(predictions,test))
print('Root Mean Squared Error: ', RMSE)
print('Mean AQI:',test.mean())
forecast_errors = [test[i]-predictions[i] for i in range(len(test))]
bias = sum(forecast_errors) * 1.0/len(test)
print('Bias: %f' % bias)
```

```
Root Mean Squared Error:  46.525126697298184
Mean AQI: 240.71425371223765
Bias: 32.755449
```

Next, we look at the predictions for the unknown i.e. 2021 AQI values without considering

the effect of lockdown.

```
# Forming the model:
final_model = SARIMAX(train,order=(1,1,1),seasonal_order=(1,0,1,12))
results = final_model.fit()
#Obtaining predicted values:
predictions = results.predict(start=64, end=77,
typ='levels').rename('Predictions')
#Plotting predicted values against the true values:
predictions.plot(legend=True)
Delhi_AQI.plot(legend=True,figsize=(12,8),grid=True);
```

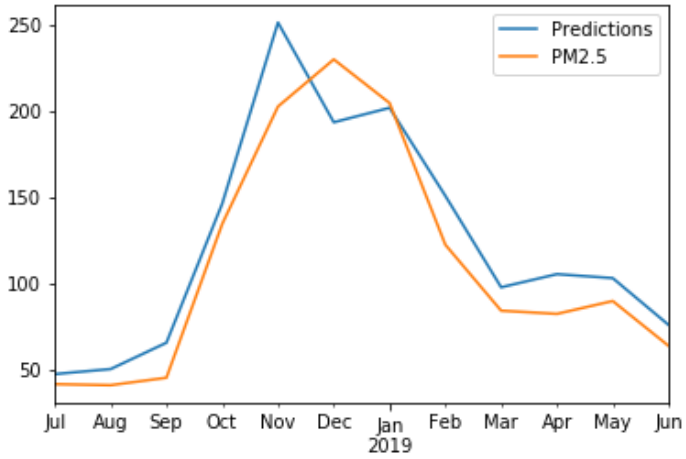


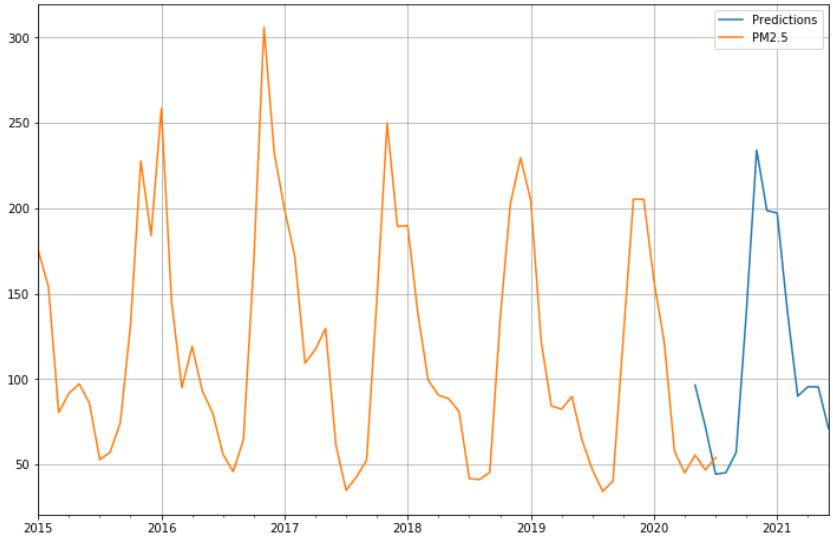
- We note that the 2021 prediction is that of very poor air quality i.e. approximately 380 AQI which is in the severe side of the Very Poor category.
- Even the lower bound on the AQI has increased from previous years.
- This plot represents the quality of air if there was no lockdown to begin with or if

the lockdown conditions were to completely disappear.

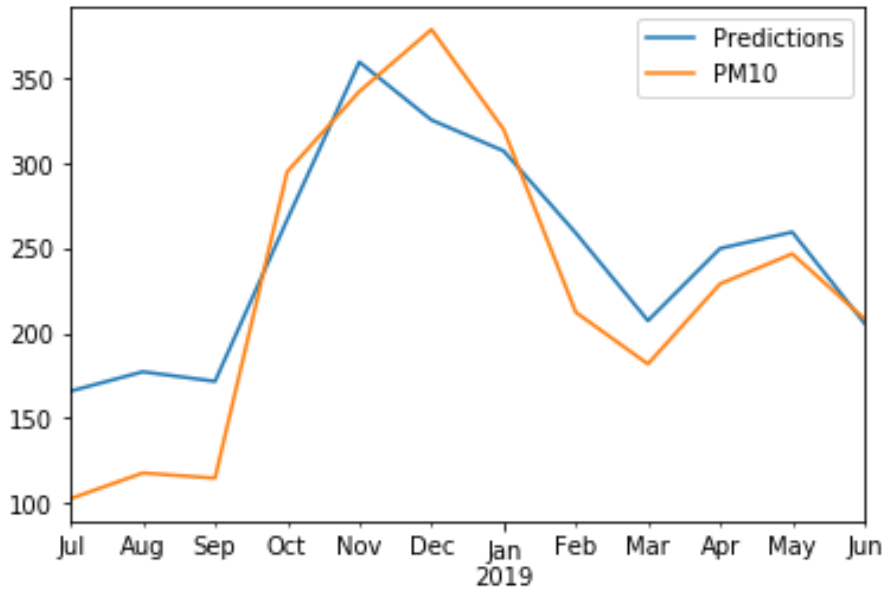
Predictions for some pollutants using SARIMAX (the first plot shows Pollutant concentration 2019 predictions and the second plot shows Pollutant concentration 2021 predictions) are shown below:

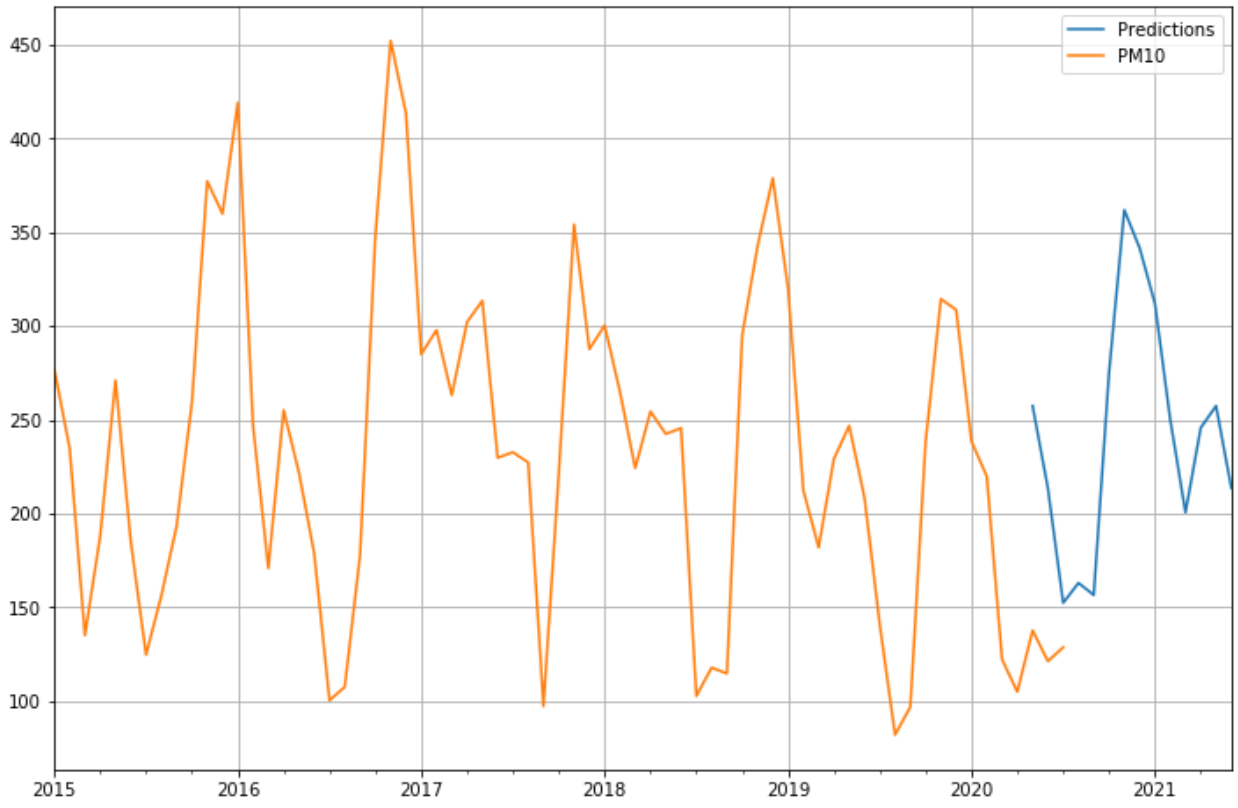
1. **PM 2.5**



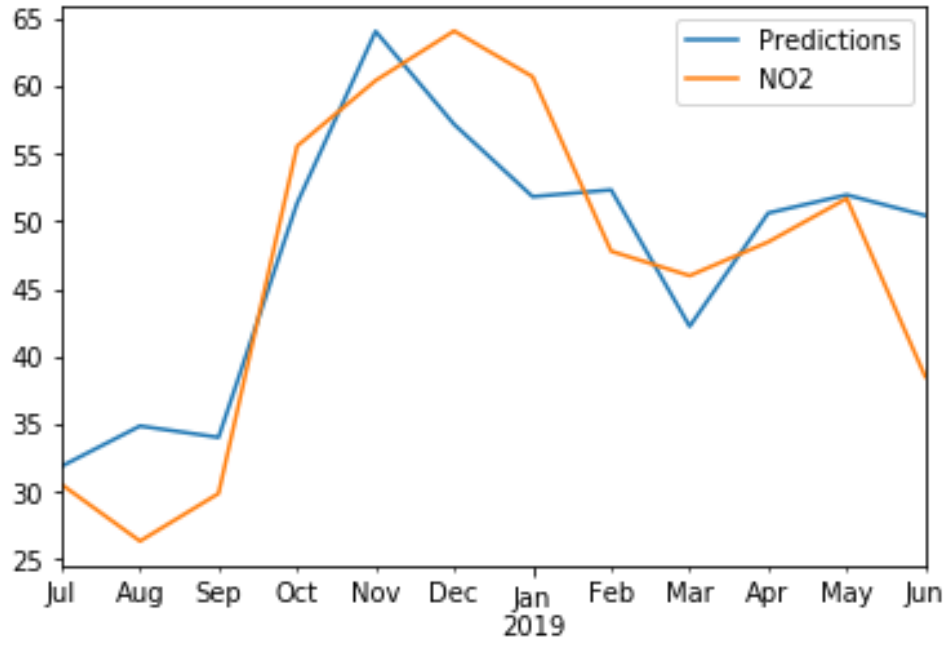


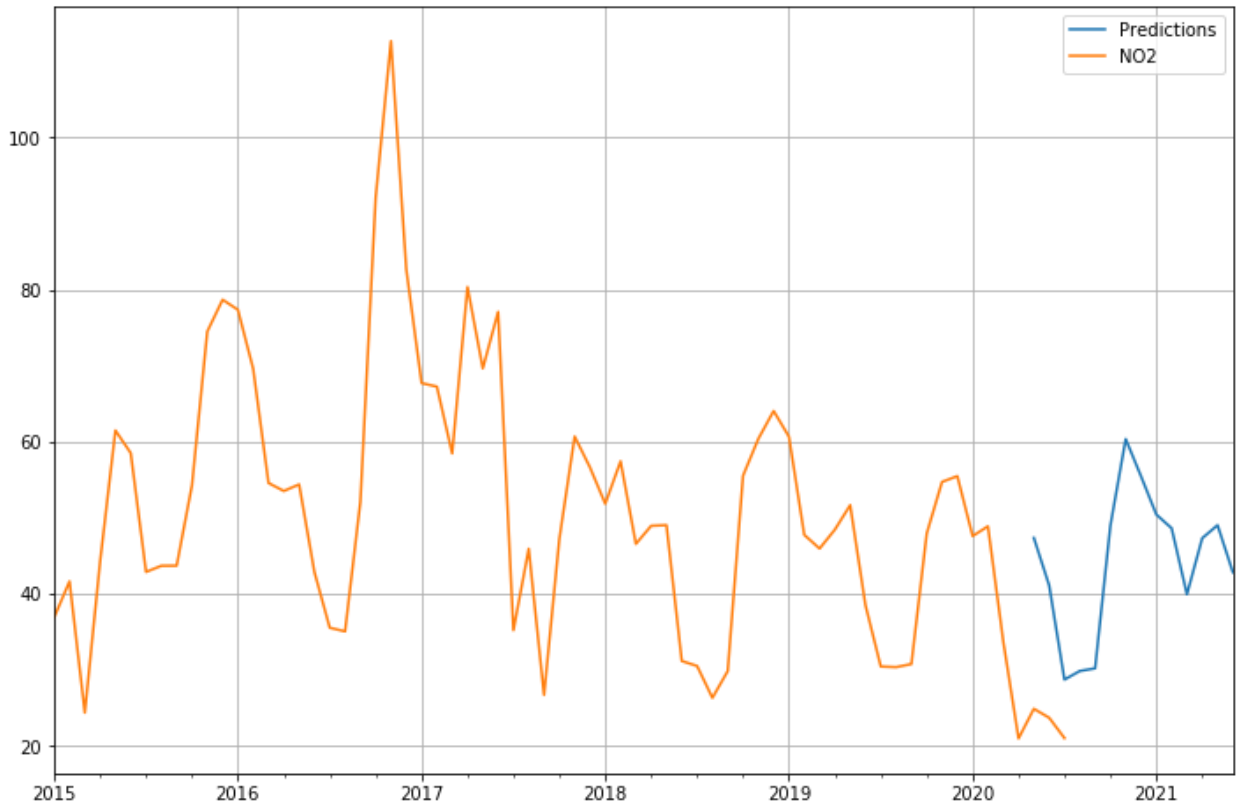
2. **PM10**


Predictions
PM10
350
300
250
200
150
100
Jul
Aug
Sep
Oct
Nov
Dec
Jan
2019
Feb
Mar
Apr
May
Jun

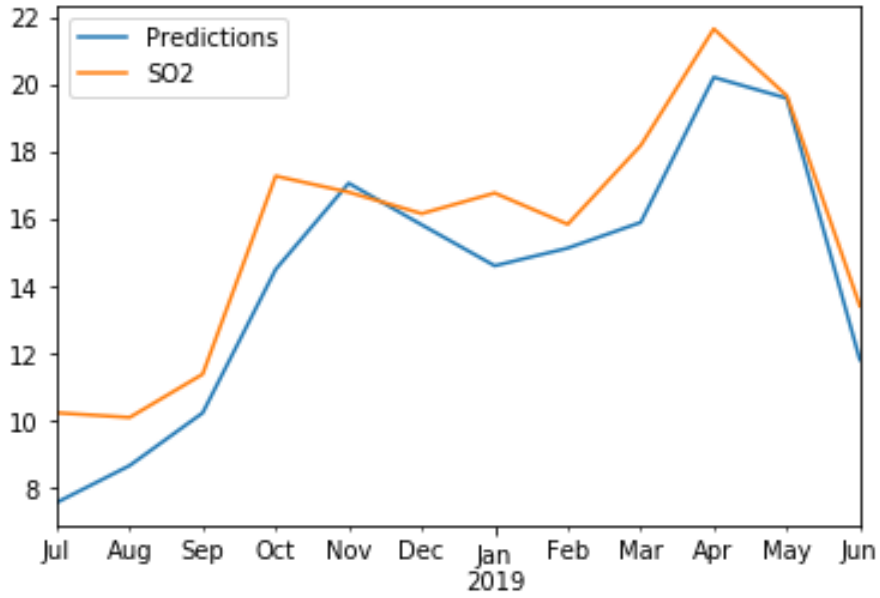

Predictions
PM10
450
400
350
300
250
200
150
100
2015
2016
2017
2018
2019
2020
2021

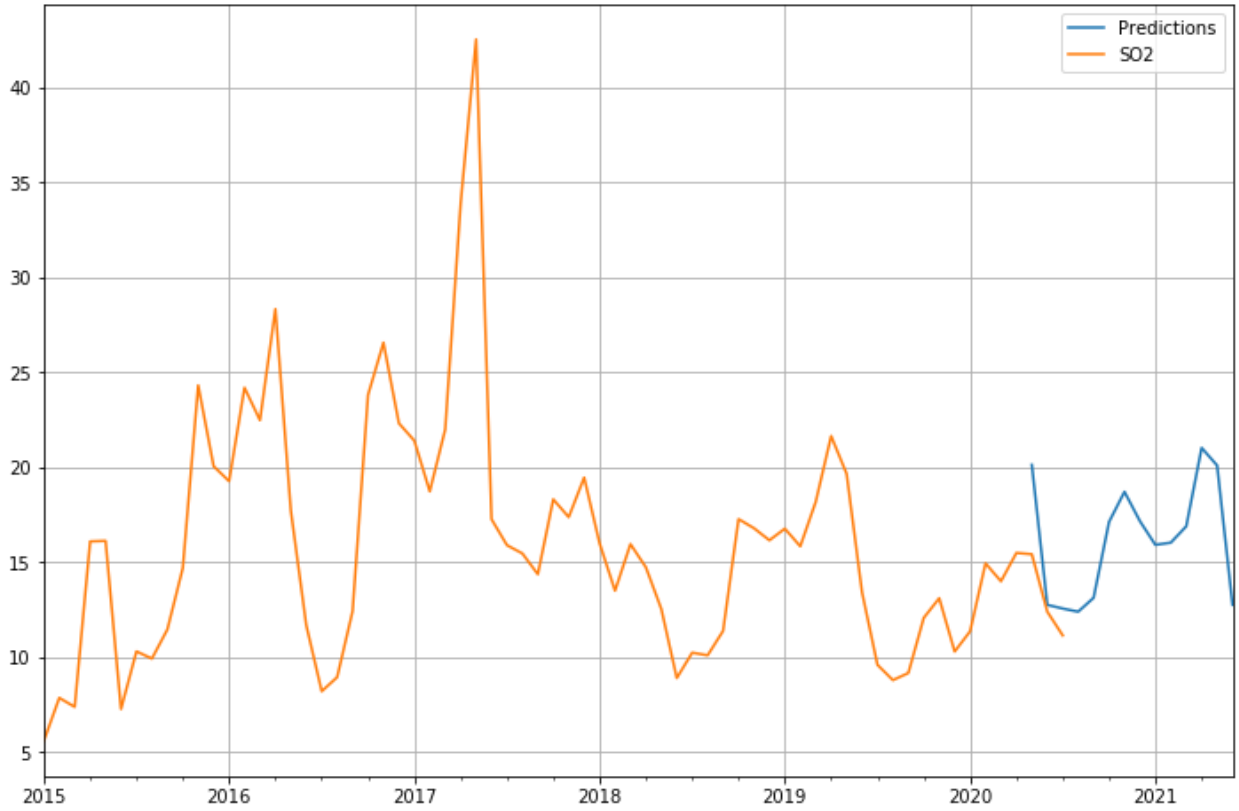
3. NO2


Predictions
NO2
65
60
55
50
45
40
35
30
25
Jul
Aug
Sep
Oct
Nov
Dec
Jan
2019
Feb
Mar
Apr
May
Jun

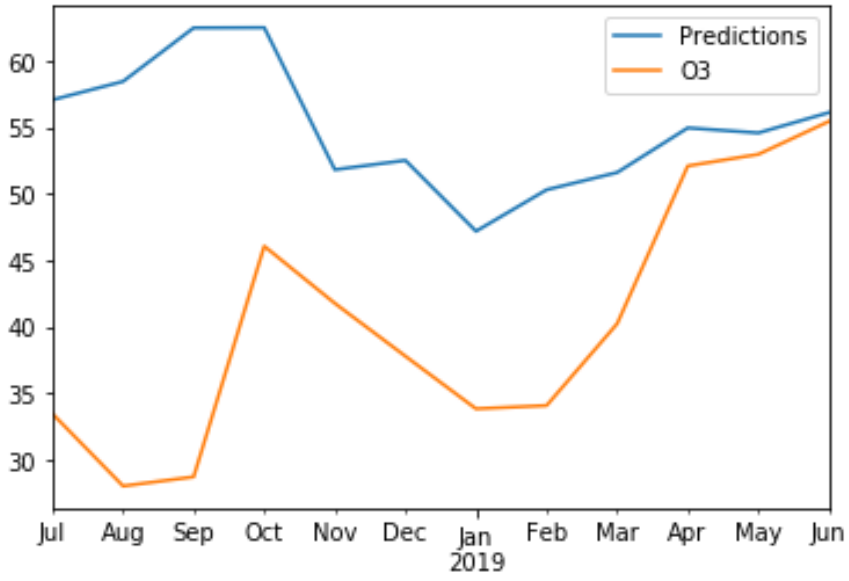

Predictions
NO2
100
80
60
40
20
2015
2016
2017
2018
2019
2020
2021

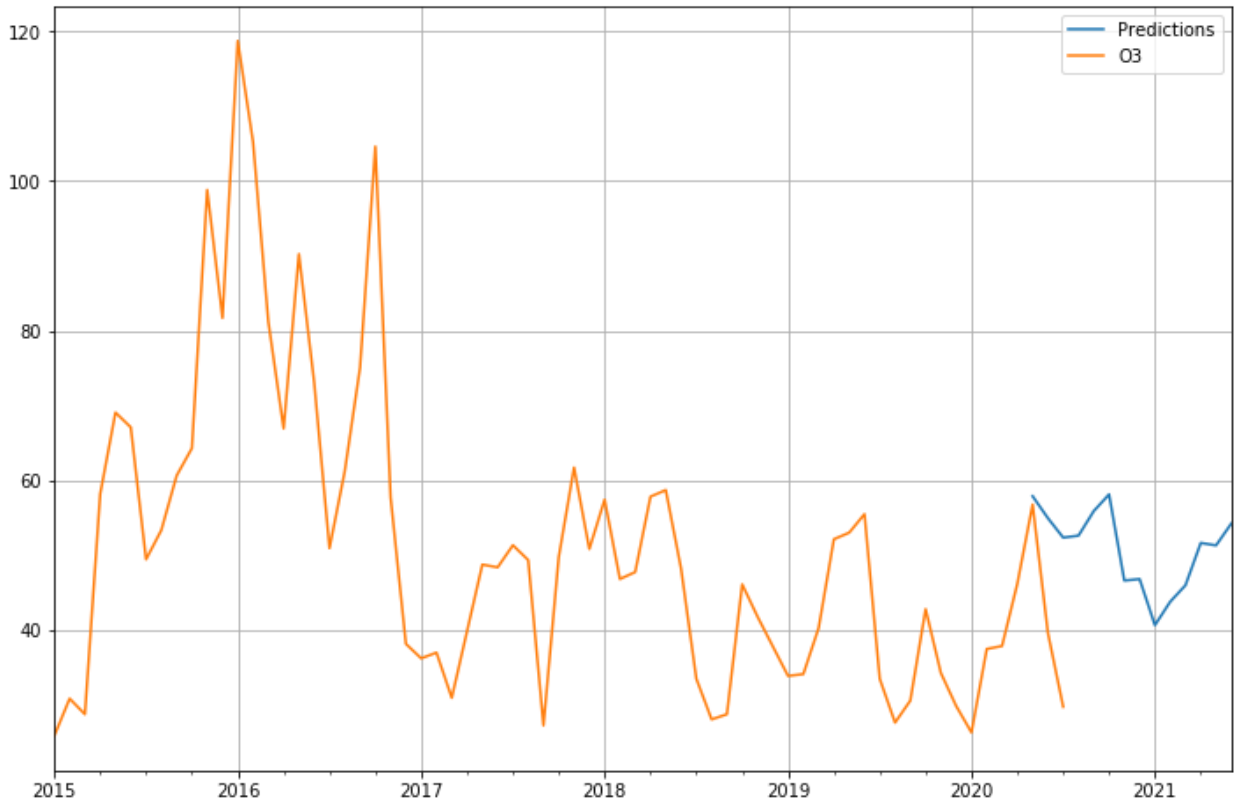
4. SO2


Predictions
SO2
22
20
18
16
14
12
10
8
Jul
Aug
Sep
Oct
Nov
Dec
Jan
2019
Feb
Mar
Apr
May
Jun


Predictions
SO2
40
35
30
25
20
15
10
5
2015
2016
2017
2018
2019
2020
2021

5. O3

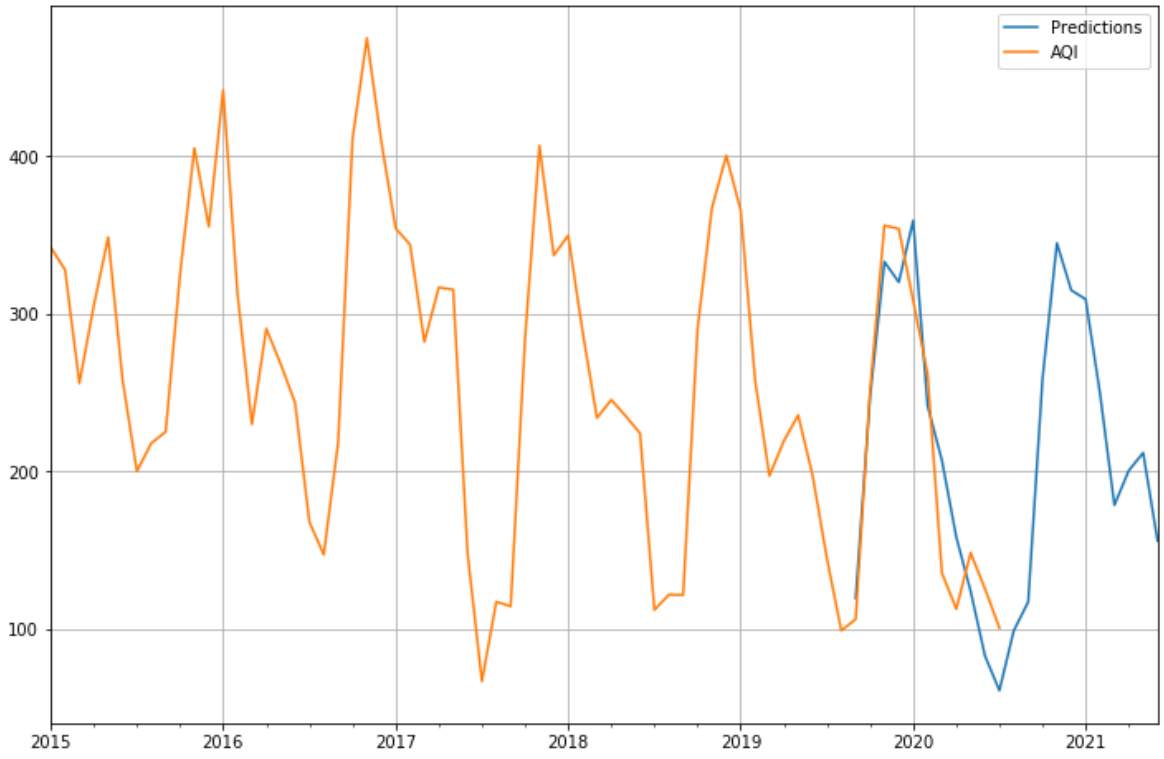

Predictions
O3
60
55
50
45
40
35
30
Jul
Aug
Sep
Oct
Nov
Dec
Jan
2019
Feb
Mar
Apr
May
Jun


Predictions
O3
120
100
80
60
40
2015
2016
2017
2018
2019
2020
2021

Phase 2. Considering the effect of lockdown. For this phase, we trained the model on data from 2015-2020 and predicted the unknown values for 2021. (Since the model parameters have already been tuned by testing the model on 2019 data in the previous phase, that part is not repeated.)

Next, we look at the predictions for the unknown i.e. 2020-2021 AQI values considering the effect of lockdown.

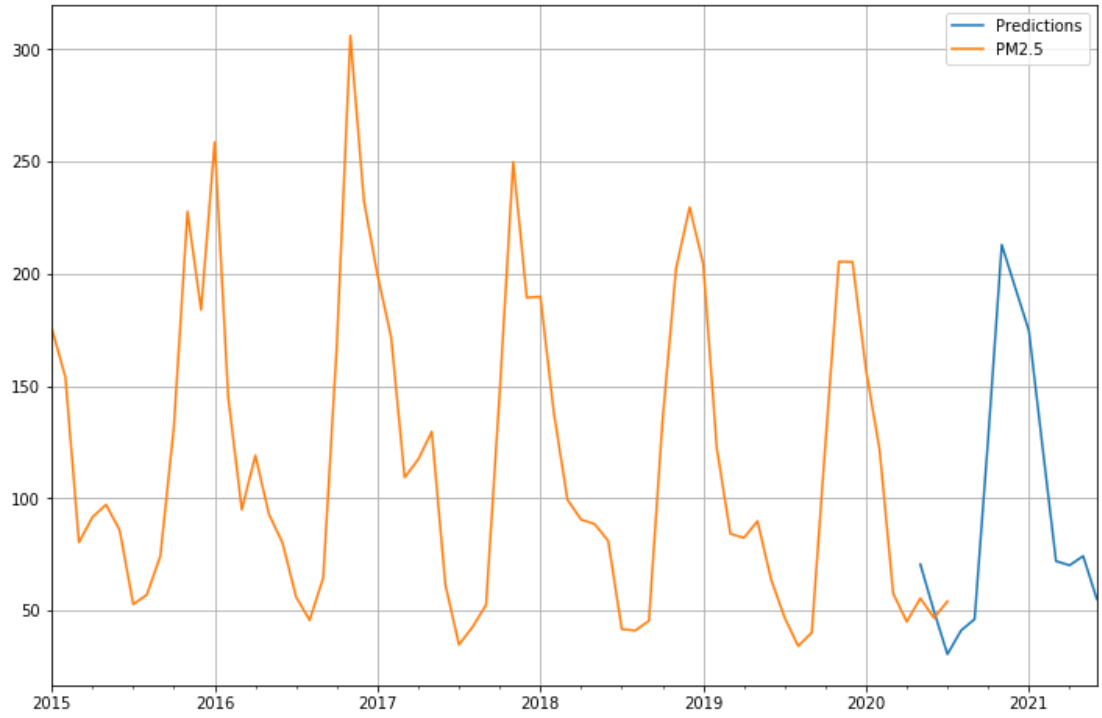
```
# Forming the model:
final_model =
SARIMAX(Delhi_AQI,order=(1,1,1),seasonal_order=(1,0,1,12))
results = final_model.fit()
#Obtaining predicted values:
predictions = results.predict(start=56, end=77,
typ='levels').rename('Predictions')
#Plotting predicted values against the true values:
predictions.plot(legend=True)
Delhi_AQI.plot(legend=True,figsize=(12,8),grid=True);
```



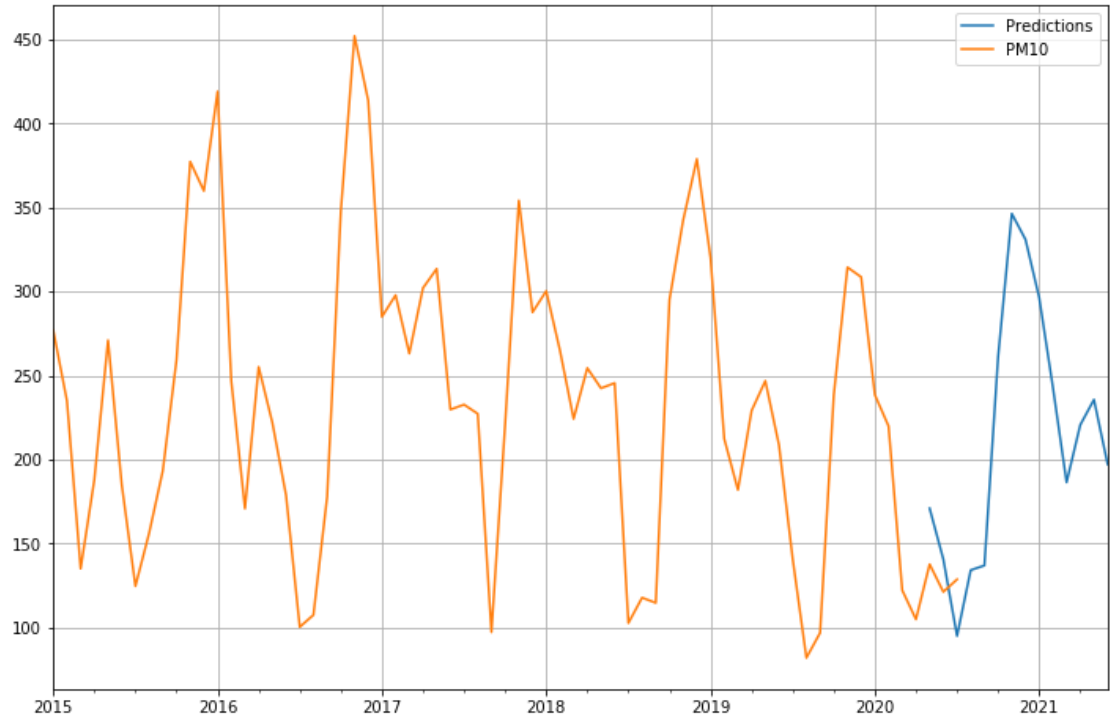
- We note that the 2021 prediction is not a highly optimistic one i.e. approximately 340 AQI which is in the Very Poor category. However, the peak has reduced a little.
- The decrease in the peak is purely due to the fact that 2020 is such an outlier. So, we can say that if lockdown were to continue, the air might have better quality.
- When the lockdown is removed, chances are, the pollution levels will follow the trend pre-2020 which would mean a bump in the AQI levels unless the city decides to keep the restrictions etc. as is, which is highly unlikely and can be seen in phase 1.

Predictions for some pollutants using SARIMAX (the plot shows the predicted pollutant concentrations for 2021 while considering the effect of lockdown) are shown below:

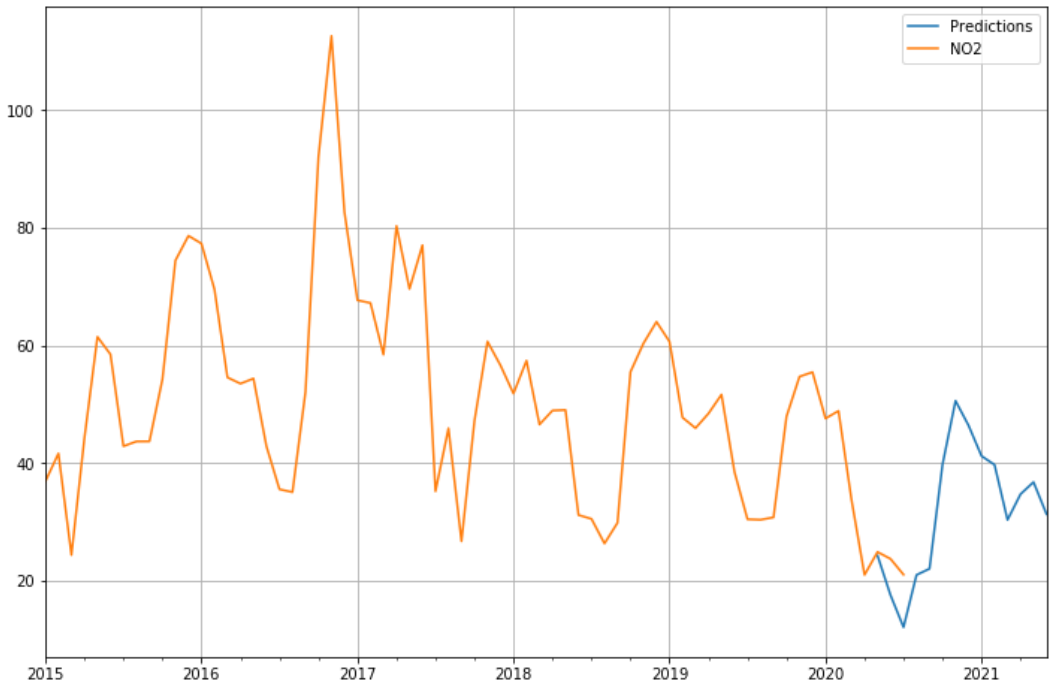
1. **PM 2.5**



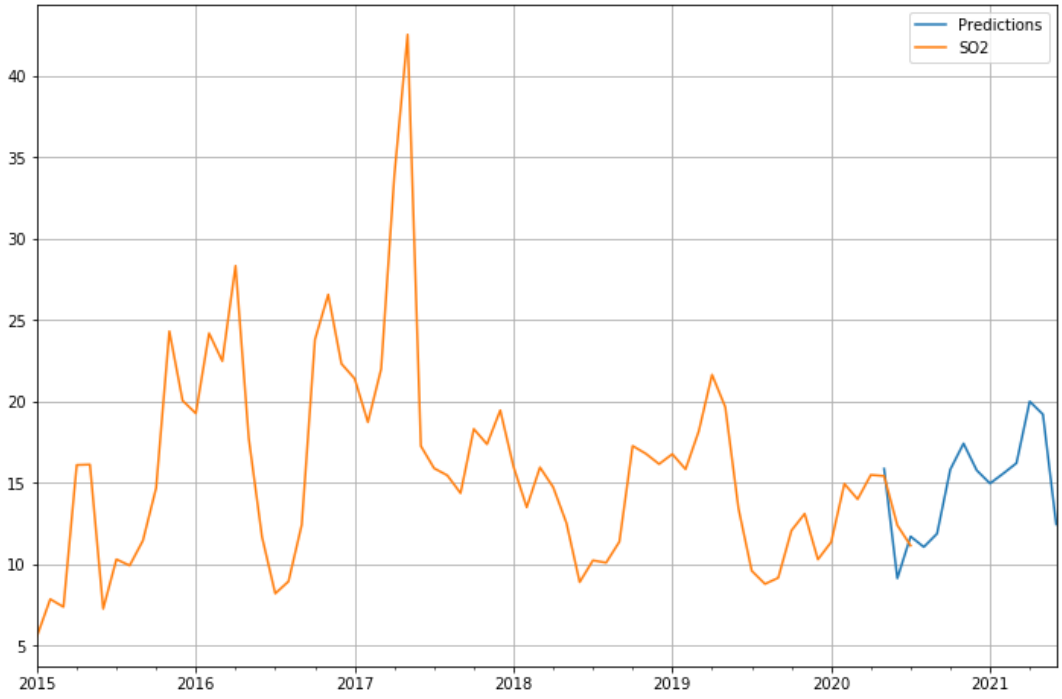
2. **PM10**


Predictions
PM10
450
400
350
300
250
200
150
100
2015
2016
2017
2018
2019
2020
2021

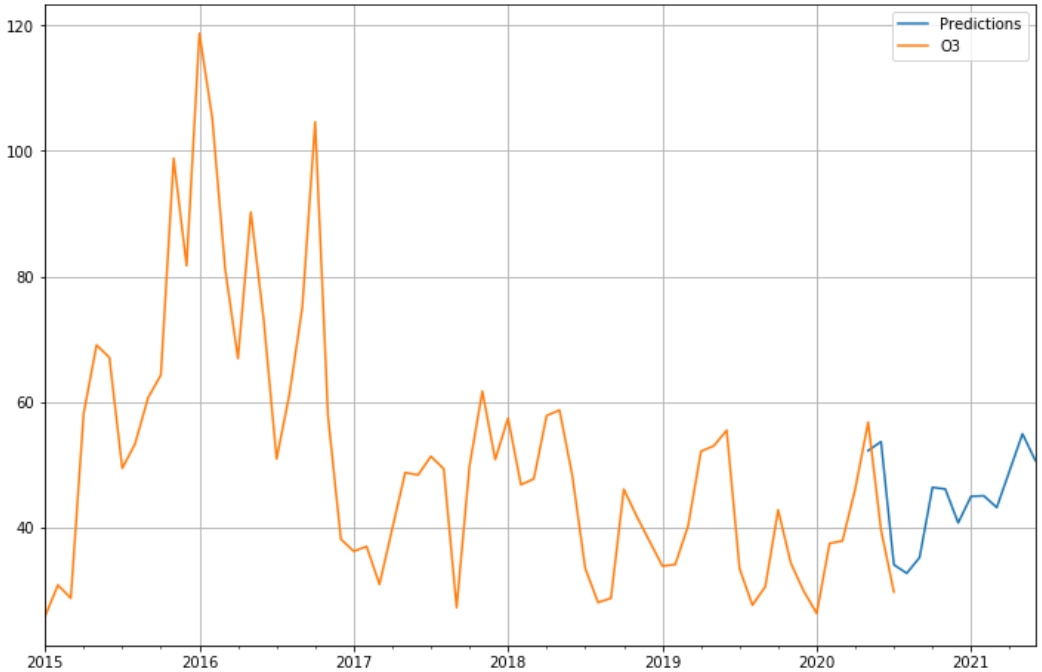
3. NO2


Predictions
NO2
100
80
60
40
20
2015
2016
2017
2018
2019
2020
2021

4. SO2


Predictions
SO2
40
35
30
25
20
15
10
5
2015
2016
2017
2018
2019
2020
2021

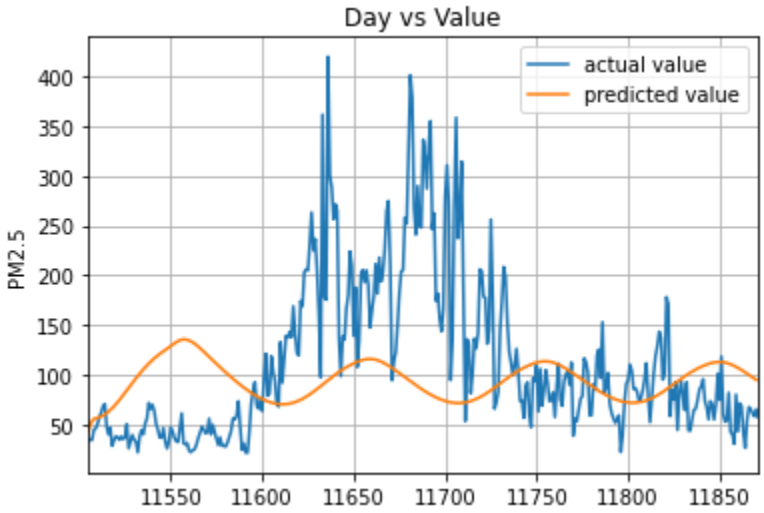
5. O3


Predictions
O3
120
100
80
60
40
2015
2016
2017
2018
2019
2020
2021

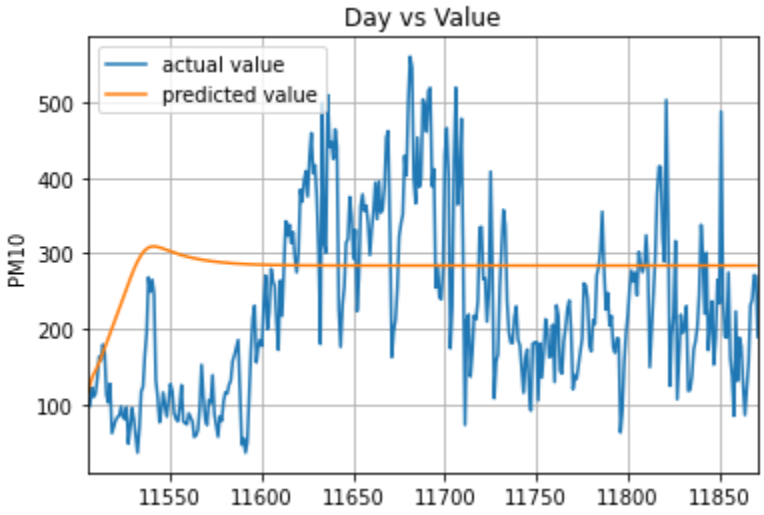
Model 2. LSTM. LSTMs provide the worst result of the lot, probably due to poor hyperparameters and their generalised nature. The graphs are shown as the metric on the y-axis vs index on the x-axis, and the index covers the period of one year from 2018 July to 2019 July in the first tuning stage, and 2020 July to 2021 July in the final inference stage.

Tuning the data. Training was done on 205-2018 data, and prediction on 2019 data. We present the results below:

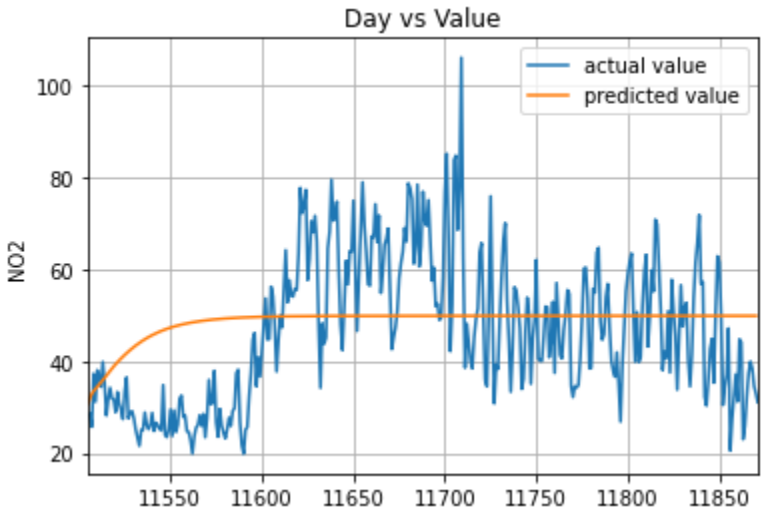
1. PM2.5



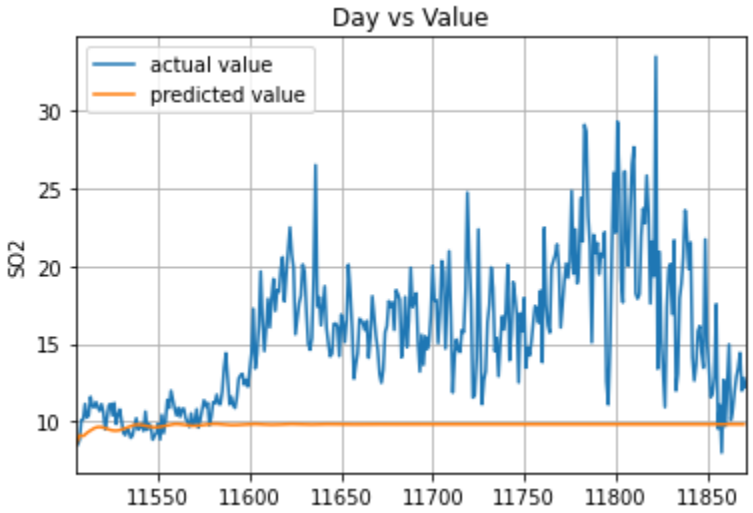
2. PM10


Day vs Value
actual value
predicted value
PM10
500
400
300
200
100
11550
11600
11650
11700
11750
11800
11850

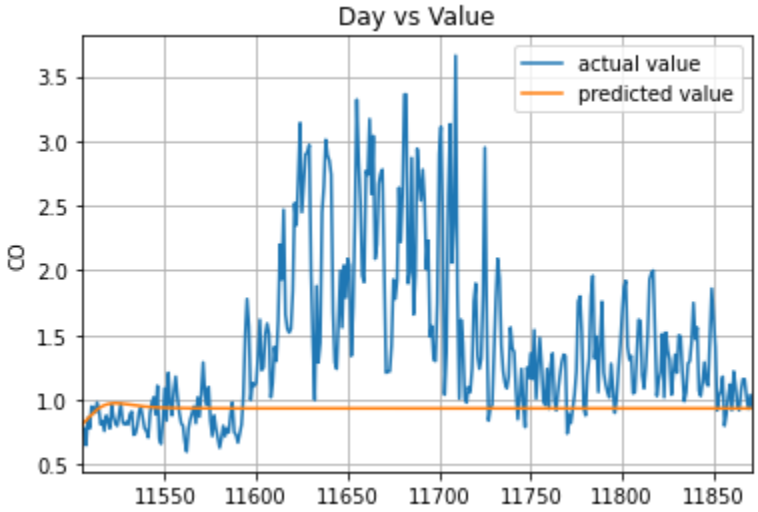
3. NO2


Day vs Value
actual value
predicted value
NO2
100
80
60
40
20
11550
11600
11650
11700
11750
11800
11850

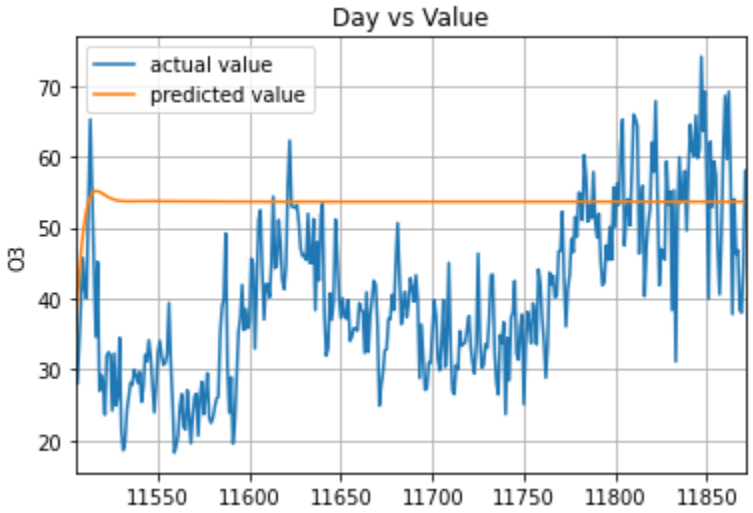
4. SO2


Day vs Value
actual value
predicted value
SO2
30
25
20
15
10
11550
11600
11650
11700
11750
11800
11850

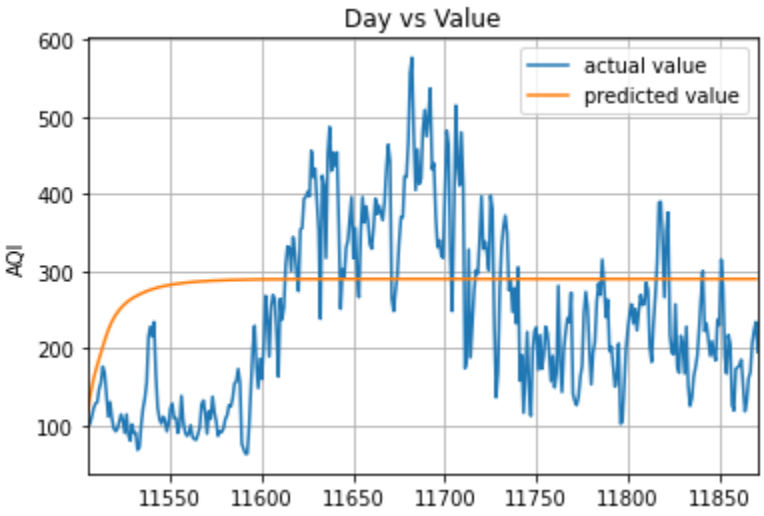
5. CO


Day vs Value
actual value
predicted value
CO
3.5
3.0
2.5
2.0
1.5
1.0
0.5
11550
11600
11650
11700
11750
11800
11850

6. O3

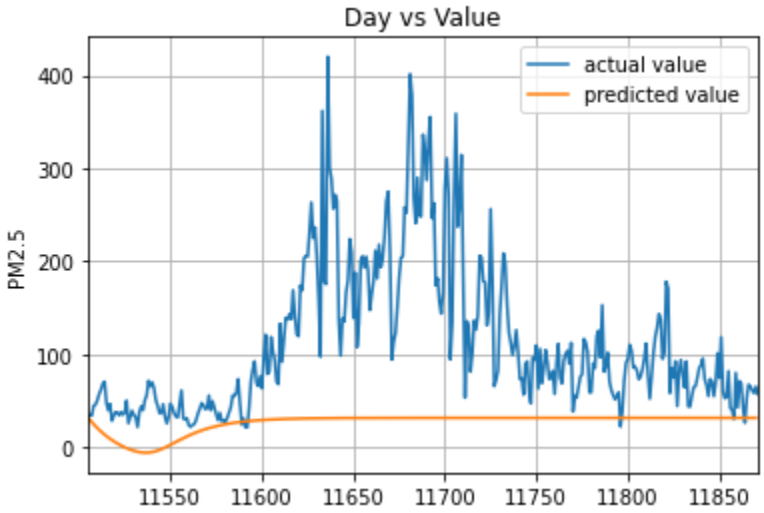


7. AQI

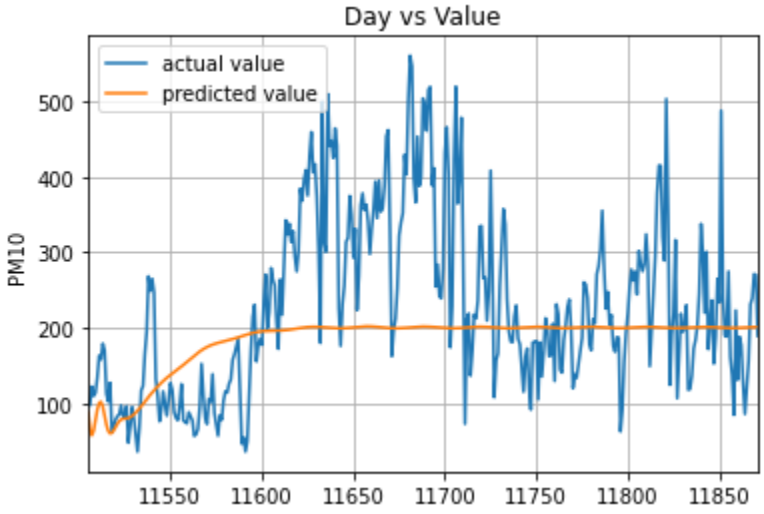


Prediction on 2021 data with all prior data. Training was done on 2015-2020 data, and predictions on 2021 data. We present the results below:

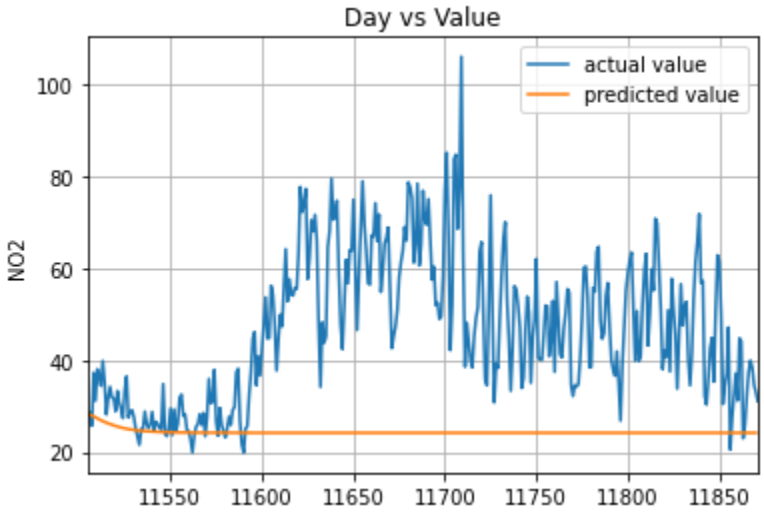
1. PM2.5


Day vs Value
actual value
predicted value
PM2.5
400
300
200
100
0
11550
11600
11650
11700
11750
11800
11850

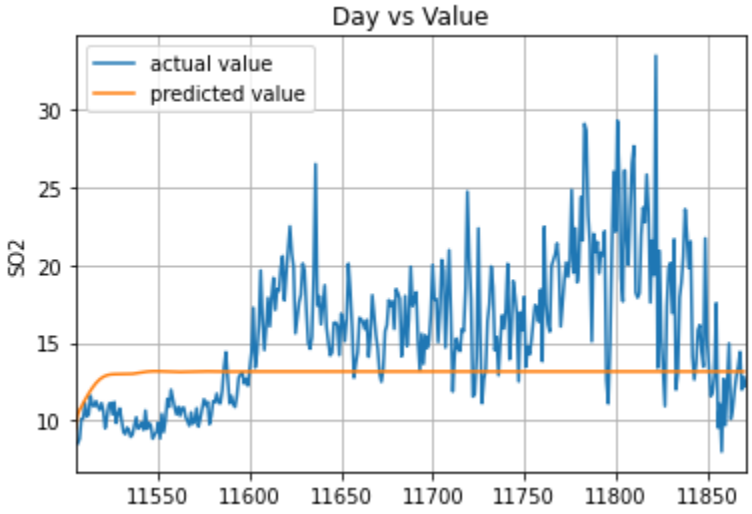
2. PM10


Day vs Value
actual value
predicted value
PM10
500
400
300
200
100
11550
11600
11650
11700
11750
11800
11850

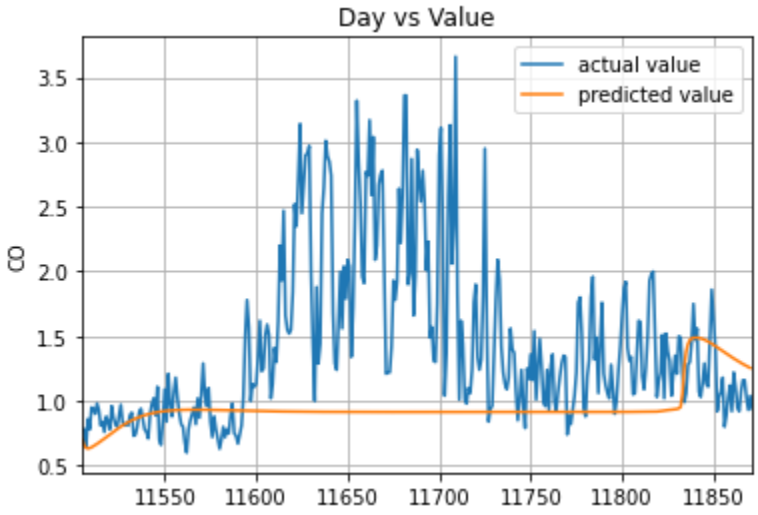
3. NO2


Day vs Value
actual value
predicted value
NO2
100
80
60
40
20
11550
11600
11650
11700
11750
11800
11850

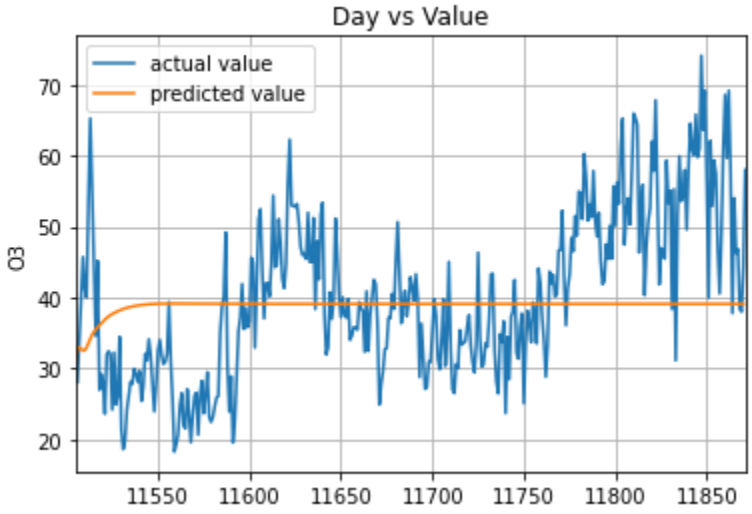
4. SO2


Day vs Value
actual value
predicted value
SO2
30
25
20
15
10
11550
11600
11650
11700
11750
11800
11850

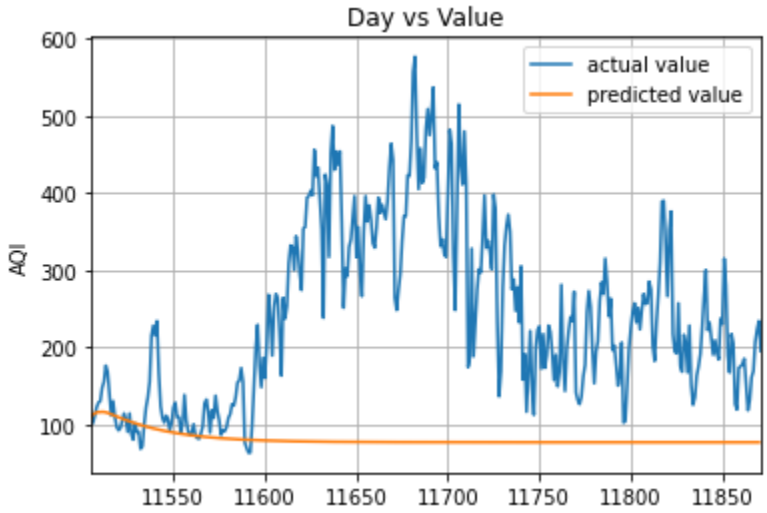
5. CO


Day vs Value
actual value
predicted value
CO
3.5
3.0
2.5
2.0
1.5
1.0
0.5
11550
11600
11650
11700
11750
11800
11850

6. O3


Day vs Value
actual value
predicted value
O3
70
60
50
40
30
20
11550
11600
11650
11700
11750
11800
11850

7. AQI



Model 3. Prophet.

First, let's see how well the model performed for the test data and then we will use it to predict future values.

The predict method will assign each row in future a predicted value which it names 'yhat'. If you pass in historical dates, it will provide an in-sample fit. The forecast object here is a new dataframe that includes a column 'yhat' with the forecast, as well as columns for components and uncertainty intervals which are yhat_lower and yhat_upper which are for the lower and upper bounds respectively.

First we

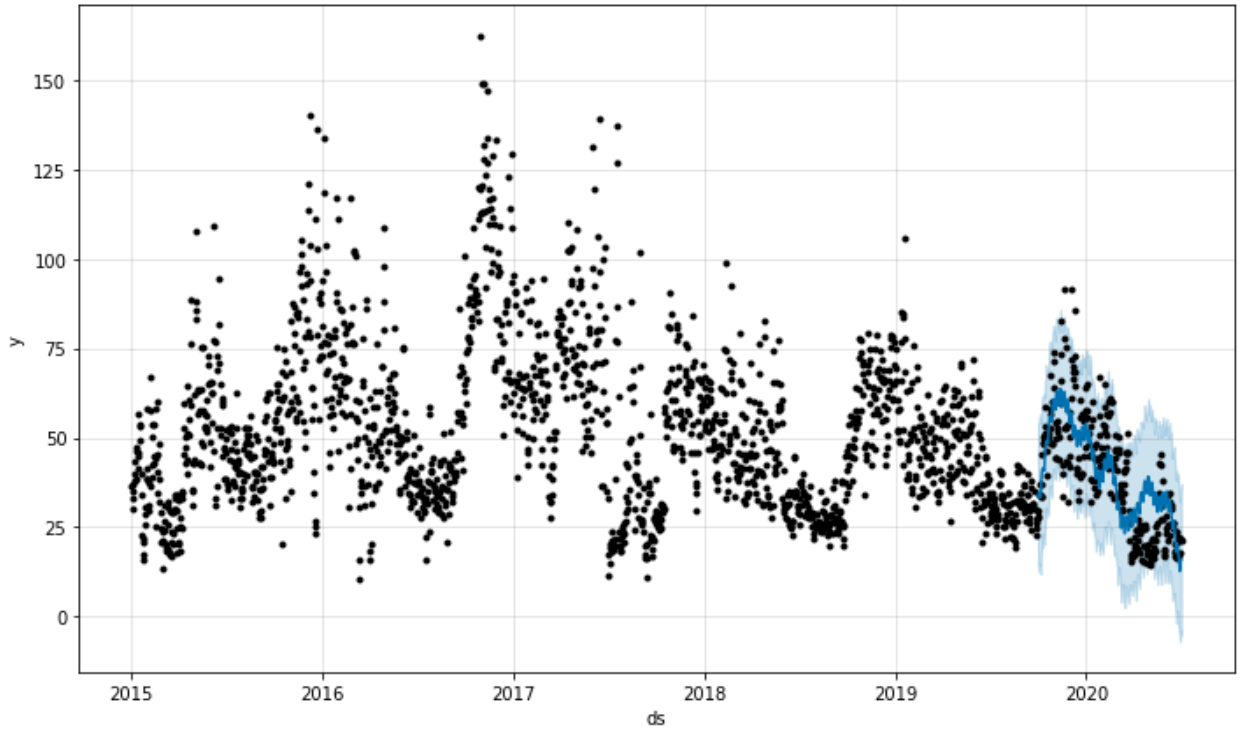
```
future = m.make_future_dataframe(periods=365)
forecast = m.predict(test)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

	ds	yhat	yhat_lower	yhat_upper

2369	2021-06-27	-1.621294	-22.503787	20.668015
2370	2021-06-28	0.418352	-20.807007	21.553200
2371	2021-06-29	1.391449	-21.551264	23.037396
2372	2021-06-30	1.332174	-21.595178	23.194338
2373	2021-07-01	0.197181	-21.969046	21.746198

You can plot the forecast by calling the Prophet.plot method and passing it in your forecast dataframe.

```
fig1 = m.plot(forecast)
```



- The black points represent the actual values.
- The blue line represents the predicted values.
- The light blue shading represents the range of values.

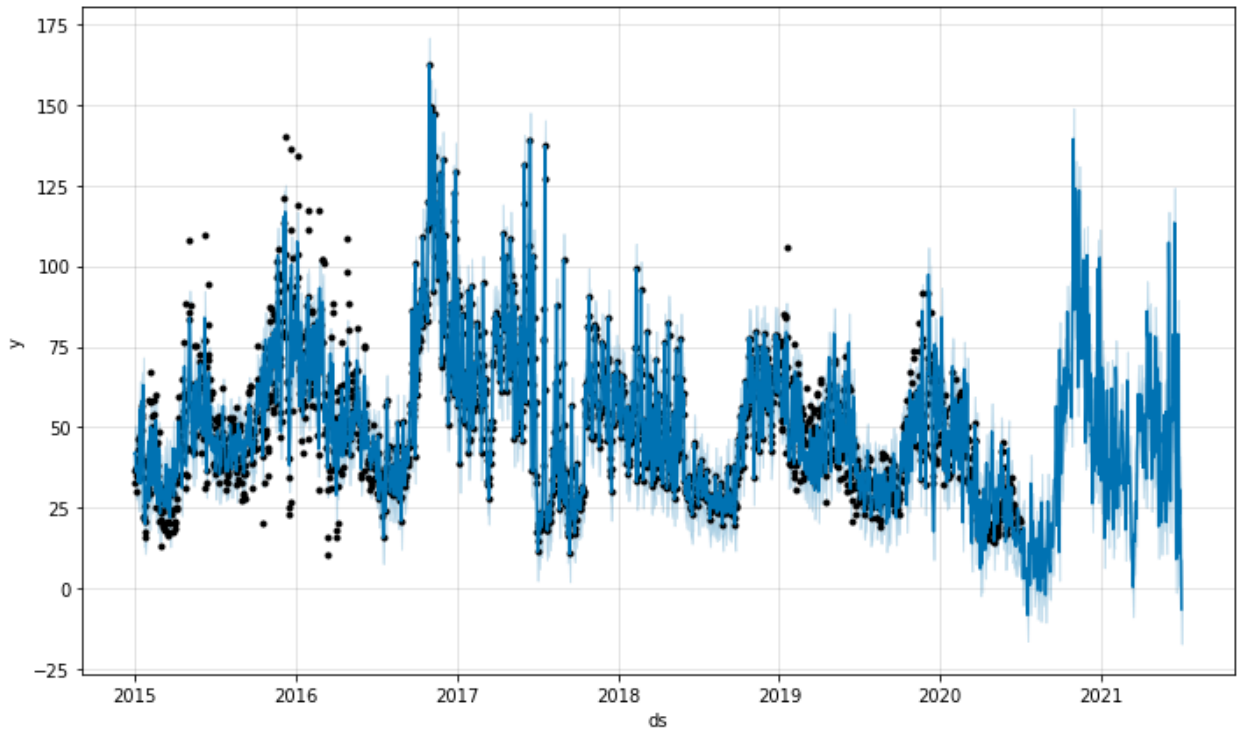
We calculated the RSME for the final predictions which was **9.054781512949596**

Next we make a dataset which is beyond the historical data available to us. This is known as an out-of-sample forecast. We can achieve this in the same way as an in-sample forecast and simply specify a different forecast period.

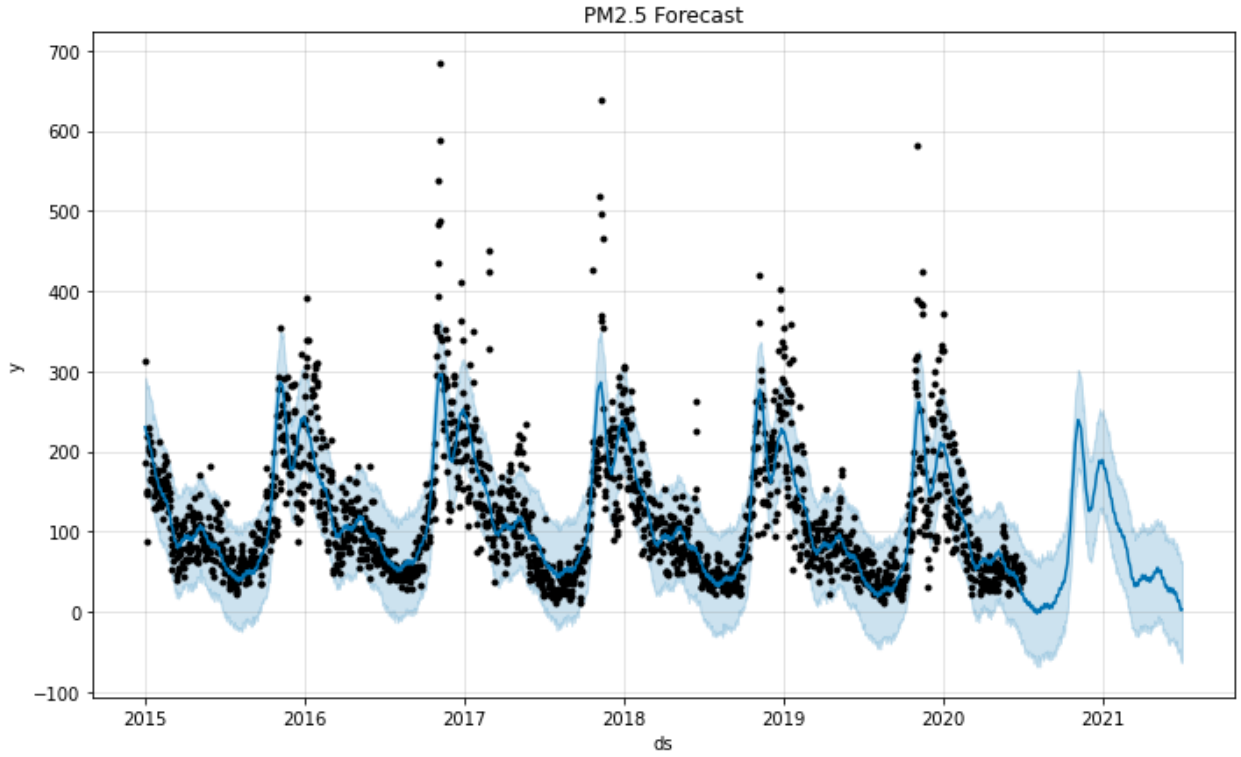
```
future = m.make_future_dataframe(periods=365)
forecast = m.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

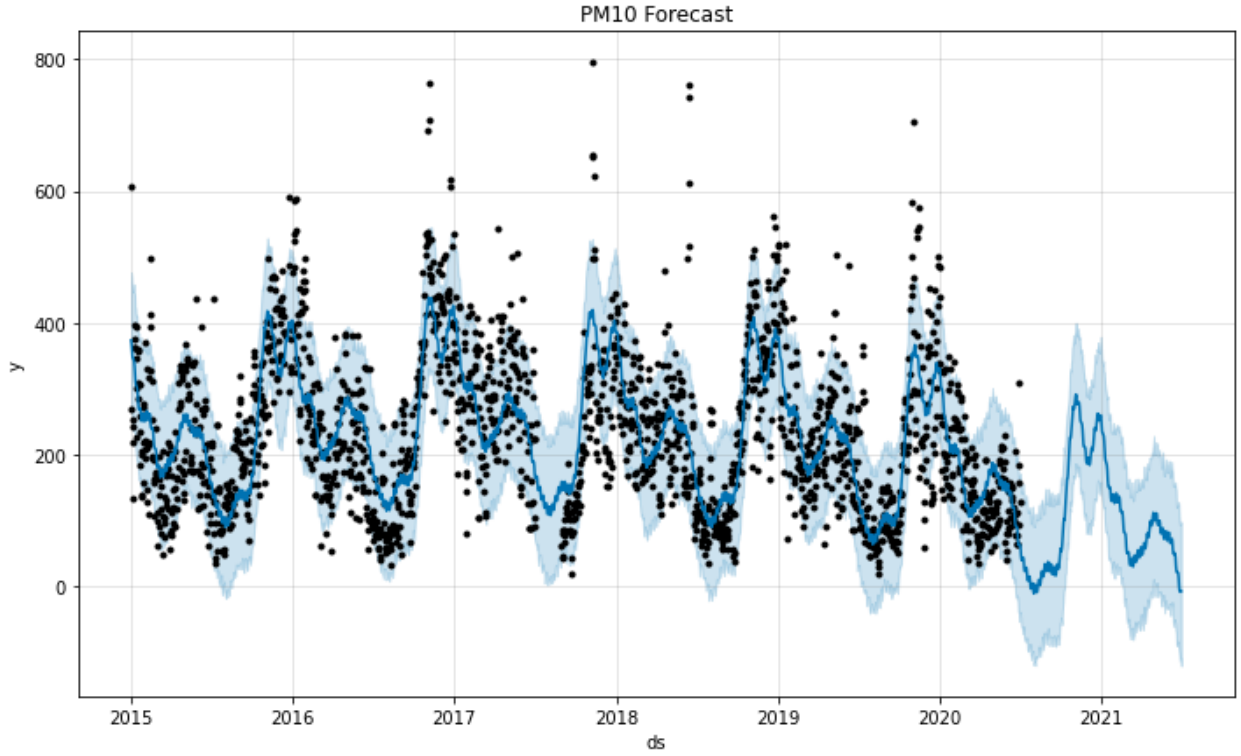
Next, we plot the forecast using the command

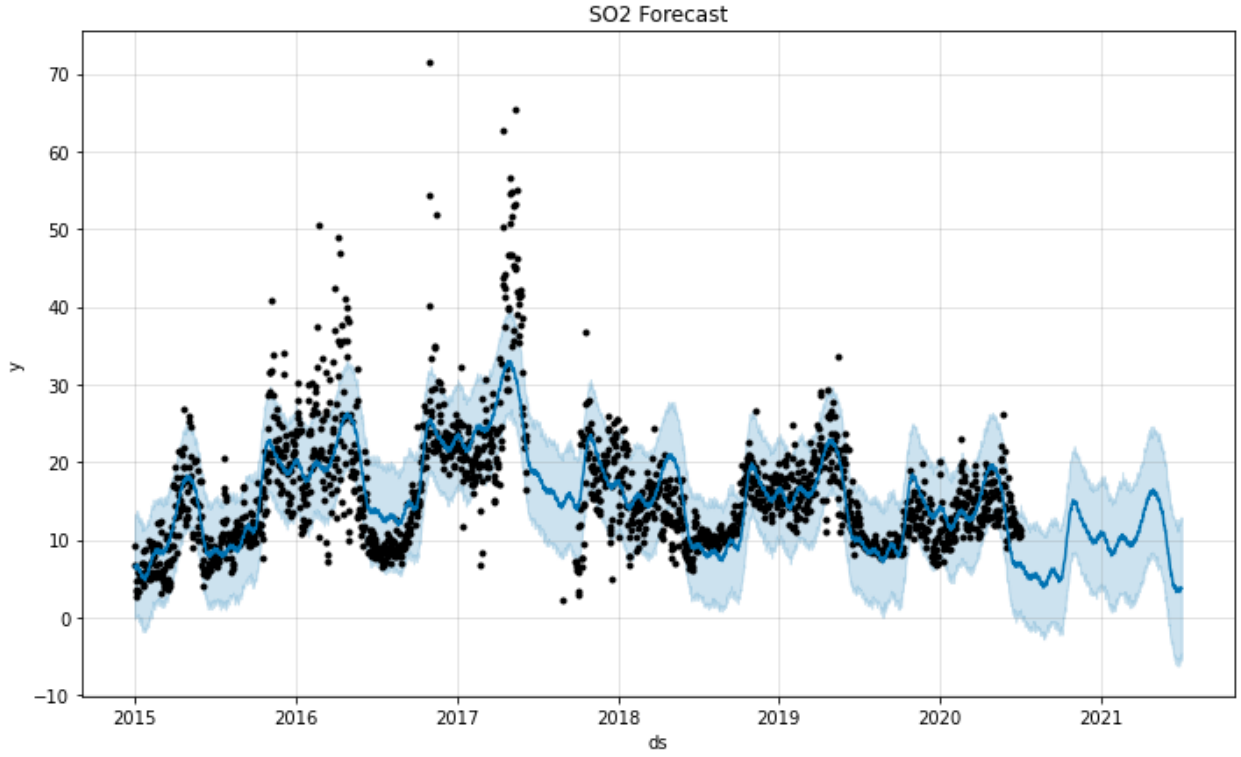
```
fig = m.plot(forecast)
```

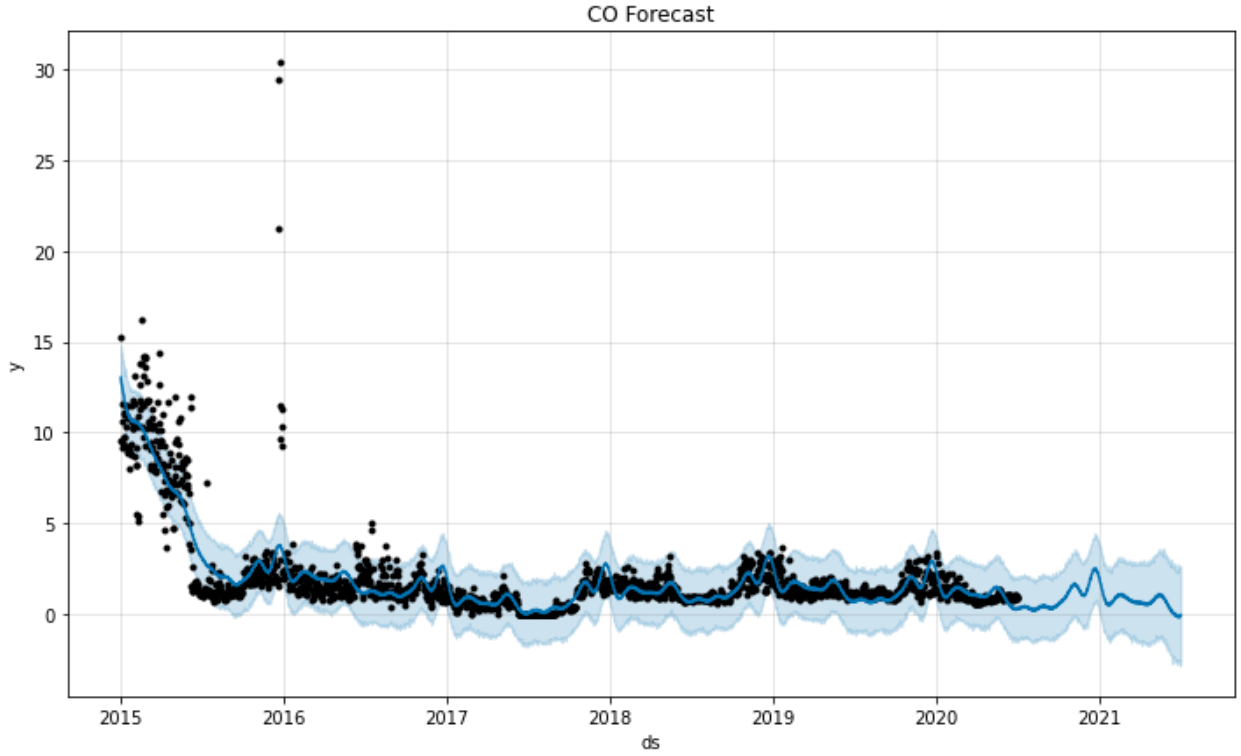


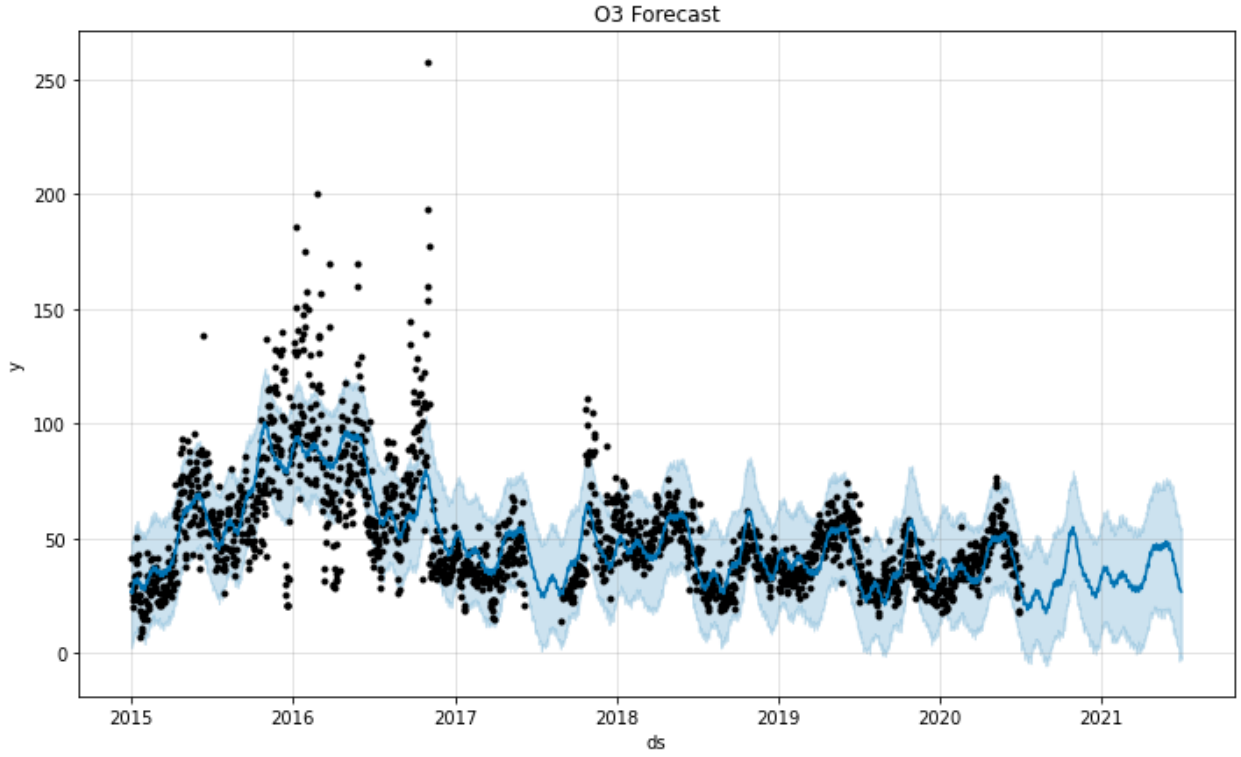
Similarly the plot for the rest of the pollutants is given below:

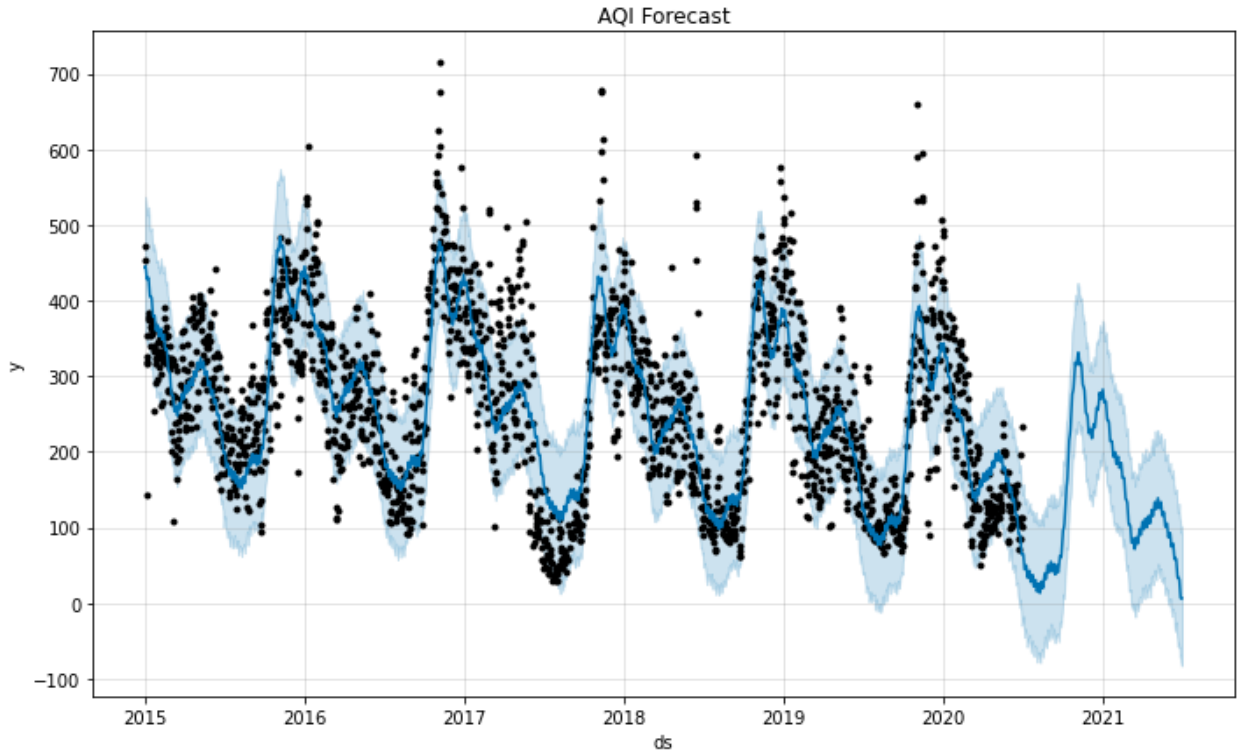



PM10 Forecast
800
600
400
200
0
y
2015
2016
2017
2018
2019
2020
2021
ds


SO2 Forecast
70
60
50
40
30
20
10
0
−10
y
2015
2016
2017
2018
2019
2020
2021
ds

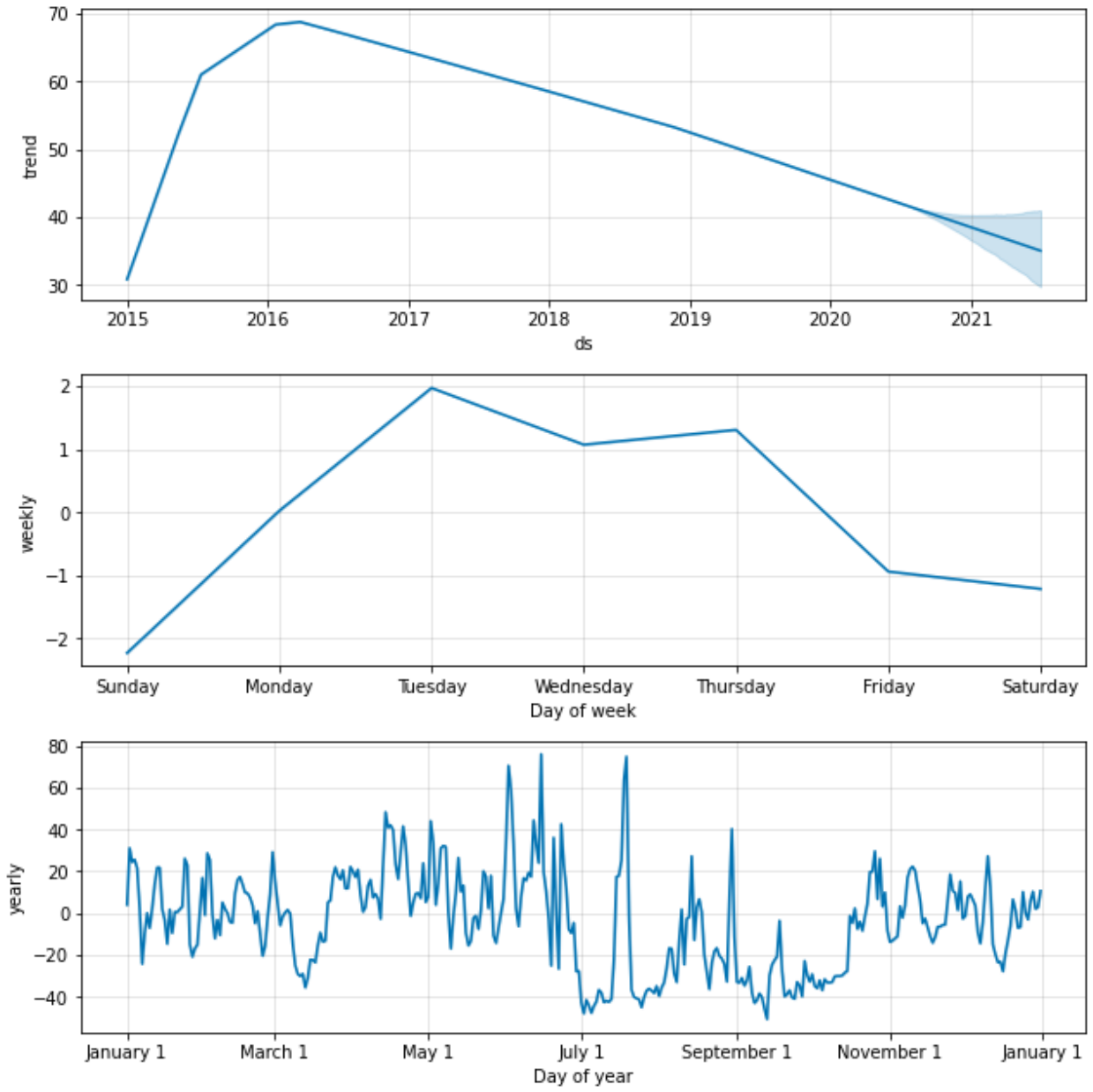

CO Forecast
30
25
20
15
10
5
0
y
2015
2016
2017
2018
2019
2020
2021
ds


O3 Forecast
250
200
150
100
50
0
y
2015
2016
2017
2018
2019
2020
2021
ds



We can also visualize some components in the data. By default, we'll see the trend, yearly seasonality, and weekly seasonality of the time series.

```
fig2 = m.plot_components(forecast)
```



Discussion. Comparing the models using the Root Mean Square Error found for each pollutant and AQI

RSME 2018-2019	SARIMAX	LSTM	Prophet (s=2000)

PM2.5	22.732492085387353	88.73265959287475	18.539922803525208
PM10	39.106712968850374	120.70019656609585	37.83781766940039
NO2	5.984225553581485	19.130738016389728	6.47750859505361
SO2	1.6676166027082113	14.521046452483802	1.770003651146963
CO	0.7235001719782407	0.7196393589289877	0.8667088674269894
O3	17.7873256051094	15.879392381766214	5.832394777954752
AQI	46.525126697298184	135.46684959995457	31.910006075802748

RSME 2019-2020	**SARIMAX**	**LSTM**	**Prophet (s=2000)**
PM2.5	28.744024821660677	123.07858102806442	33.030065352871304
PM10	76.52449137021613	138.53707223468663	55.526877872613944
NO2	11.286551147549964	28.806266720461227	9.054781512949596
SO2	4.3720068036053545	12.596179453761032	3.180200406436058
CO	0.2939873443351936	0.7343326671575078	1.3013884855513544
O3	24.952767741928778	18.713872628622738	14.218913942476874
AQI	69.23578918181191	201.1118009411132	47.014884350286245

From the results as seen in the table above, we can see that SARIMAX and Prophet trade blows when it comes to performance on predicting the future data. This performance does not seem to vary significantly across categories from the 2018-19 data, and the 2019-20 data.: SARIMAX performs better than Prophet on PM2.5, SO2, and CO, while the reverse is true for PM10, O3, AQI. The only value that switches around is NO2.

As for the differences in between the years wholesale, the lockdowns in the early part of 2020 (aka the end of the 2019-2020 data) means that the predictions shown by the models trained on “normal” circumstances are now significantly off. This is visible in the predictions, most notably in PM10 and AQI.

Conclusion. For pollutants that have seen almost constant levels, the error is low and the prediction is also similar and mostly lies in the safe category.

As far as the effect of lockdown is concerned it has impacted the pollution levels of the environment to improve the air quality in the short span owing to very less human activities. A complete lockdown forced people to live in their homes. The measures taken to ward off the threat of the virus are virtually identical to the measures that climate activists have been demanding for decades: less travel, less work and less environmental expropriation.

It is also evident how poor the quality of our air is. With or without lockdown, the concentrations of harmful pollutants like PM2.5 and PM10 remain far too high for comfort. The AQI tends to the high side of 200 or more, often reaching up to 380 at times; where it is recommended that “good air quality” be below 50 to ensure people have no troubles or health issues that arise from just breathing.

Acknowledgement

We would like to express our heartfelt gratitude to our Professor, Prof Sunitha Palissery who gave us the golden opportunity to do this project on the topic Air Pollution in Delhi, which helped us to understand the seriousness of the matter and also the far reaching consequences of Covid’19.

Further, we would like to thank the Kaggle Community for their Machine Learning related doubts and queries**.** CPCB, CIDM and NICB community for the dataset and information on pollutants and case studies.

References

1. Quick Start | Prophet
2. Time Series Forecasts using Facebook's Prophet (with Python & R codes)
3. What's Polluting Delhi's Air? -
4. Explained: Why does air pollution rise in October every year?
5. 141 Responses to How to Decompose Time Series Data into Trend and Seasonality.
6. How Delhi became the most polluted city on Earth
7. 1 thought on "Air pollution in Delhi Essay – Case Study!"
8. The great smog of Delhi.
9. "Air pollution in Delhi: Its Magnitude and Effects on Health"
10. Delhi Air Pollution: A Case Study
11. Fatal carbon monoxide poisoning: A lesson from a retrospective study at All India Institute of Medical Sciences, New Delhi
12. Analysis: India's CO2 emissions fall for first time in four decades amid coronavirus (Link)
13. National Air Quality Index, India (Link)
14. Kaggle(Link)
15. Wikipedia(Link)
16. Machine Learning Mastery(Link)
17. Medium for ML(Link)
18. ESA, 2020. ESA - air pollution drops in India following lockdown
19. CPCB online portal for air quality data dissemination (Link)
20. Air Pollution in India: Real-time Air Quality Index Visual Map (Link)