Air Pollution Levels And Lockdown Measures: A Meta-Analysis

Harshita Sharma(20171099)

Jalees Jahanzaib (2018101001)

Ojaswi Binnani(20161006)

CSE 596: Environmental Science and Technology
Prof. Rama Chandra Pillutla
November 28, 2020

Introduction

The outbreak of viral disease labelled as Novel Coronavirus started in Wuhan, China in December 2019. The virus spread in almost every part of the world and was declared a global pandemic in March 2020 by the World Health Organization (WHO). The pandemic has given insight into the mess humans are and have been making of the planet. The ongoing pandemic of COVID-19 has forced several countries of the world to observe complete lockdown forcing people to live in their homes. These 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. India also faced the phase of total lockdown for 21 days(in the first phase) to avoid the spread of coronavirus to the maximum possible extent. This lockdown impacted the pollution levels of the environment and improved air and water quality in the short span owing to very less human activities. The Air Quality Index (AQI) is a pretty robust metric which reflects to be changing in favour of Mother Nature.

Literature Review

India before Lockdown. According to the World Air Quality Report 2019 compiled by IQAir Air Visual, India had topped an annual list of cities with the worst air quality in the world, while Chinese cities have continued to show improvements from the previous year. India was the 5 most polluted country in 2019, with Ghaziabad in the National Capital Region ranked as the most polluted city in the world. Twenty-one of the world's 30 cities with the worst air pollution are in India, according to the same report, with six in the top ten. Whilst cities in India, on average, exceed the World Health Organisation target for annual PM2.5 exposure by 500%, national air pollution decreased by 20% from 2018 to 2019, with 98% of cities experiencing improvements. India had launched a National Clean Air Programme in 2019 that commits to reducing air pollution in 102 most polluted cities by a maximum of 30% by 2024.

Rank	Country/Region	2019 AVG	2018 AVG	Population
1	Bangladesh	83.30	97.10	166,368,149
2	Pakistan	65.81	74.27	200,813,818
3	Mongolia	62.00	58.50	3,121,772
4	Afghanistan	58.80	61.80	36,373,176
5	India	58.08	72.54	1,354,051,854
6	Indonesia	51.71	42.01	266,794,980

Figure 1. India ranked 5th among the list of most polluted countries

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 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 across all five cities, from a 10% reduction in Mumbai up to a 54% reduction in Delhi. These reductions in PM2.5 were found to be comparable to reductions in other cities across the world, such as in Vienna (60%) and Shanghai (42%). The study also estimated the value of the lives saved by the reduction of air pollution is $690 million.

The air quality of India. The air quality index (AQI) is defined as ratios of the measured concentration of the atmospheric pollutants to their standard prescribed values. It is a robust

metric representing the quality of air of a certain region. Some studies show the following effects of the nationwide lockdown on the levels and concentration of various pollutants in the atmosphere and hence on the air quality of India.

PM2.5 levels. The average concentration of PM2.5 before the lockdown is found to be higher in comparison with the concentration after lockdown. The PM2.5 concentration in Kolkata is reduced by 34.52%, and 27.57% in Delhi, 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.

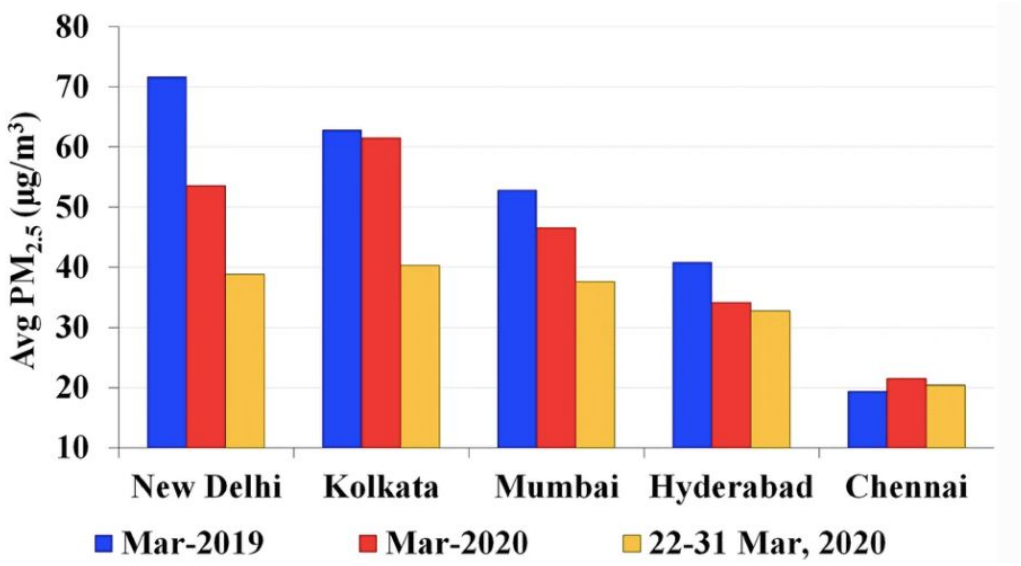


Figure 2. Variations of average PM2.5 during March 2019 (blue bar), March 2020 (red bar), and average during 22–31 March 2020

In Mumbai, Chennai, and Hyderabad, PM2.5 was reduced by 19.25%, 5.40%, and 3.99%, respectively. The dominance of westerly wind from arid and semi-arid regions and lower temperature along the Indo-Gangetic plains in the month of March, the average concentration of PM2.5 remains higher in comparison to other cities. The proximity of Mumbai and Chennai to the sea, the air mass mostly reaches from the sea surface during March and the PM2.5 is lower in comparison with Delhi and Kolkata.

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 the 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.

Figure.3(right) represents the environmental condition in India from the period of January 01, 2020, to March 24, 2020 (before lockdown) and March 25, 2020, to April 20, 2020 (after lockdown). It also makes a comparison with the same dates of 2019. This image is taken through the *Copernicus Sentinel-5P* satellite. The findings revealed a significant reduction in nitrogen dioxide concentrations. Figure 3(left) shows the Maritime route of the commercial ship in the Indian Ocean before and after lockdown. The findings indicate a faint trail of nitrogen dioxide emission in the Indian Ocean at the Maritime route. The shipping lanes looked like a straight line as the ships follow the same tracks.

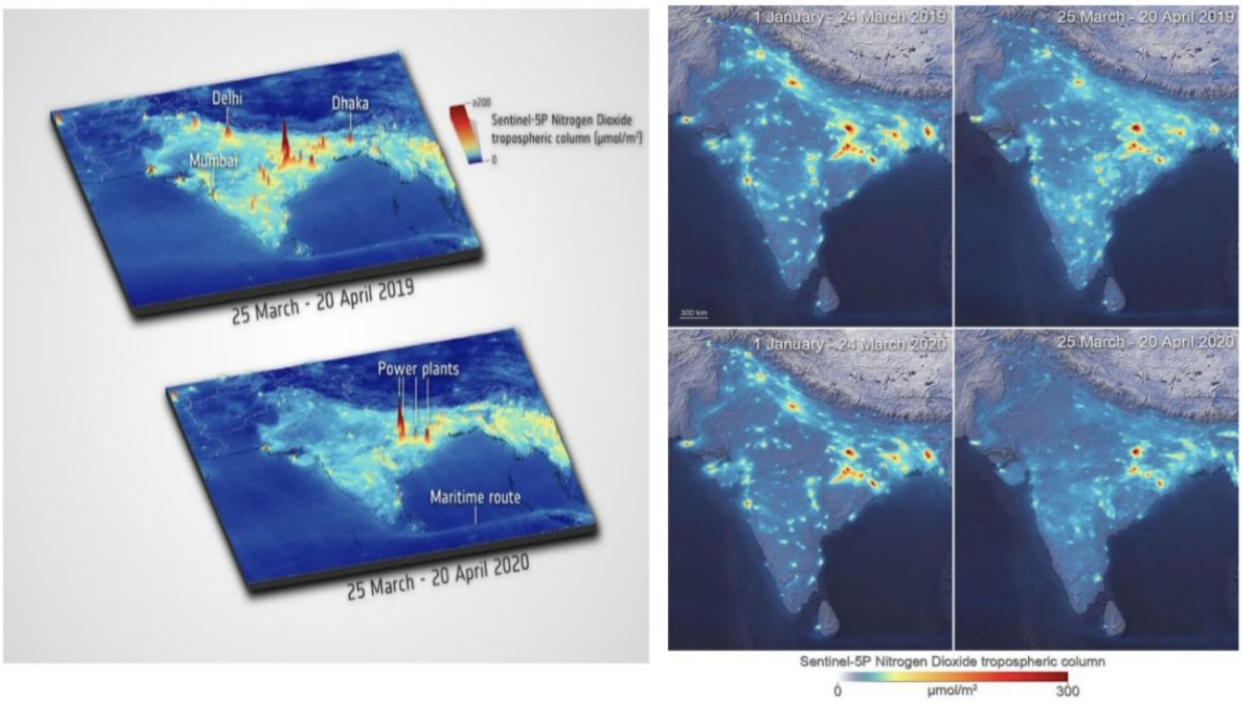


Figure 3. Nitrogen dioxide emission before and after a lockdown in India

Moreover, the cities in India with full of a crowd in 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).

The daily level of Nitrogen Dioxide emission in Mumbai can be seen in Figure 4(right). The results show that before the lock-down period, the level of nitrogen dioxide emission remains from 28 to 62 (ug/m3). Nonetheless, it faced an enormous deterioration with the beginning of lockdown in India. This study indicates that nitrogen dioxide emission remains low during the complete lockdown in Mumbai. Consequently, these statistics verified that lockdown in India has imperatively improved environmental quality. Also, it confirmed the accuracy of the information generated through satellite images.

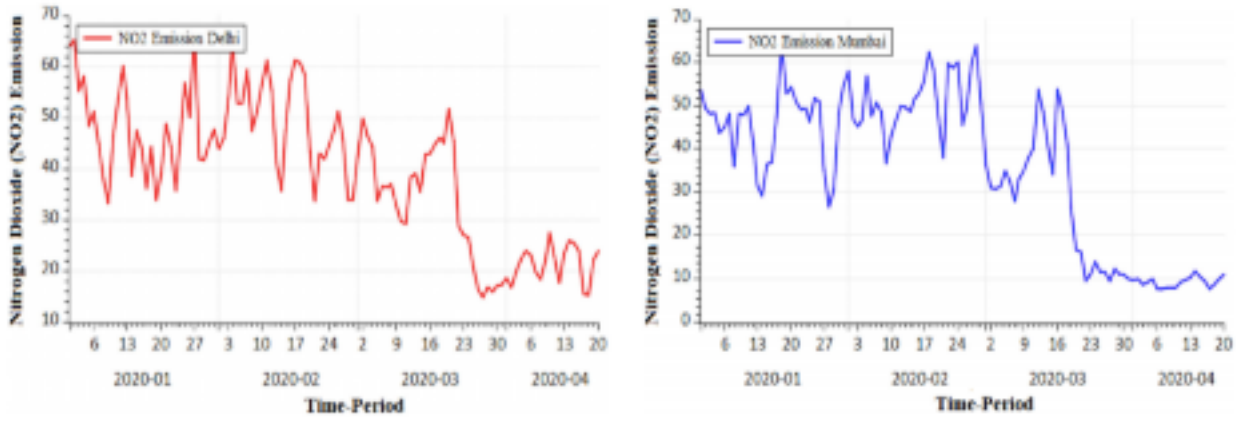


Figure 4. Nitrogen dioxide emission before and during the lockdown in Delhi(left) and Mumbai(right)

Carbon dioxide (CO_2) emissions. Daily global CO_2 emissions decreased by –17% (–11 to –25% for ±1σ) (σ stands for uncertainty) by early April 2020 compared with the mean 2019

levels, just under half from changes in surface transport. At their peak, emissions in individual countries decreased by –26% on average. The impact on 2020 annual emissions depends on the duration of the confinement, with a low estimate of –4% (–2 to –7%) if pre-pandemic conditions return by mid-June, and a high estimate of –7% (–3 to –13%) if some restrictions remain worldwide until the end of 2020. Government actions and economic incentives post-crisis will likely influence the global CO_2 emissions path for decades.

As with the global CO_2 impact of the pandemic, the longer-term outlook for India's emissions was shaped, to a significant degree, by the government response to the crisis and will have major long-term implications for India's CO_2 emissions and air quality trajectory.

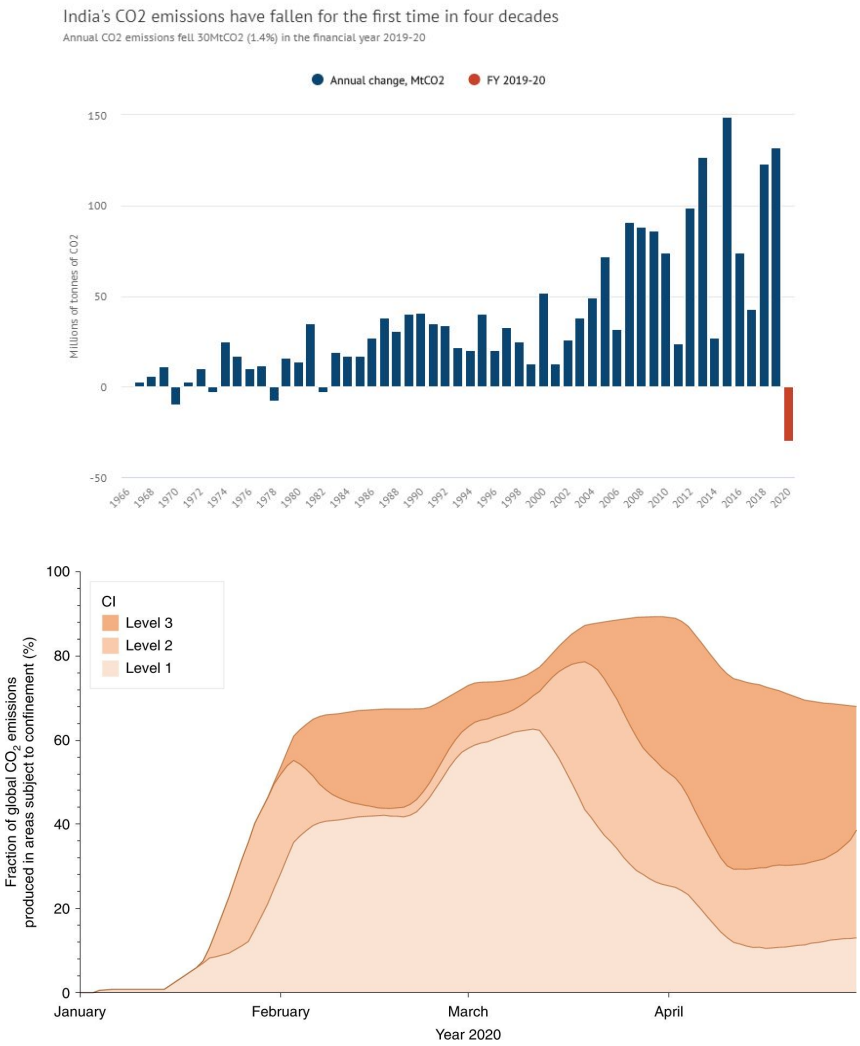


Figure. 5 Carbon dioxide emission before and during the lockdown Globally

Objectives of the study

The primary aim of this project is to find and analyse the effect of lockdown on the Air Quality Index(AQI) of India and its cities. 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 the trend and seasonality in the same.
2. Analyse the various stages of Lockdown on air quality and pollution in different places in India.
    a. Analyse AQI over the years 2015-2020(mid-July)
    b. Analysing the trend and seasonality in AQI
    c. Analyse AQI of the different cities in India specifically in the year 2020
    d. Analyse AQI of cities in India before and after the lockdown was imposed.
3. Construct models predicting the future of air quality if the lockdown continues.
4. Study the different environmental factors that have an effect on air quality and pollution.
5. Using the models, analyse the long term effects of the lockdown on the air quality and pollution and compare it to the case if lockdown was removed.
6. Data visualisation: Visualization of data and results to provide a better and deeper understanding.

Materials and Methodology

Dataset

The **Central Pollution Control Board (CPCB)**, a statutory organisation which collects air quality data which is updated every week. The data keeps track of major air pollutants such as CO, NO2, NO, O3, SO2, PM2.5 etc**.** which are shown below:

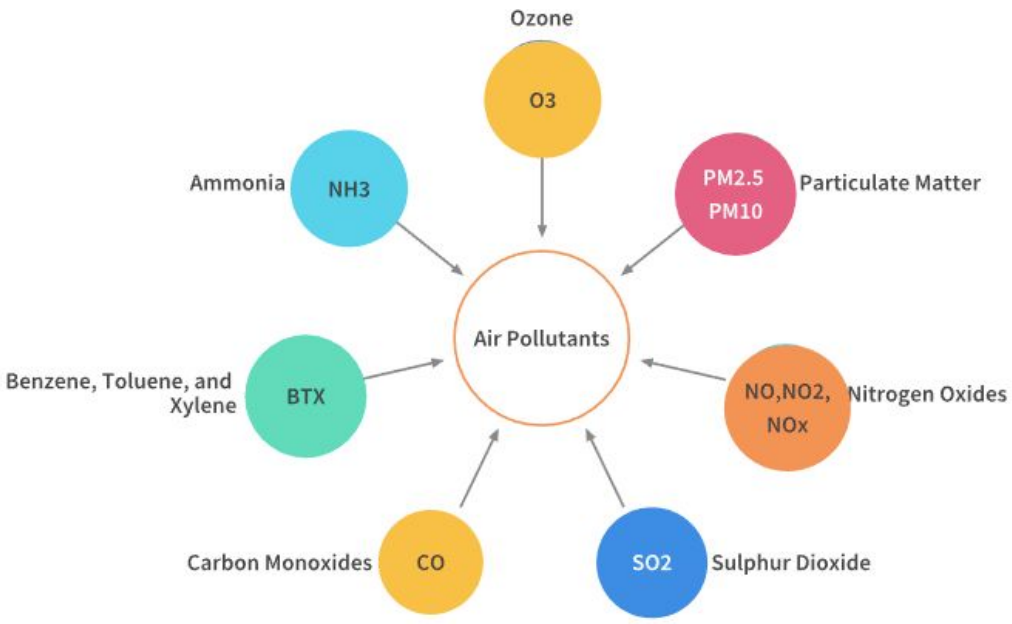


Figure. Air Pollutants

- **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).

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. We will be working with the daily city data in this project since the aim is to look at the AQI in different cities of the country.

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 matplotlib.pyplot as plt
from datetime import datetime
```

```
import warnings
import seaborn as sns
warnings.filterwarnings("ignore")
warnings.filterwarnings("ignore")

citydailydata = pd.read_csv('../data/city_day.csv')
```

2. **Extracting the structure of the available dataset**

```
citydailydata.info()
```

```
RangeIndex: 29531 entries, 0 to 29530
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   City        29531 non-null  object 
 1   Date        29531 non-null  object 
 2   PM2.5       24933 non-null  float64
 3   PM10        18391 non-null  float64
 4   NO          25949 non-null  float64
 5   NO2         25946 non-null  float64
 6   NOx         25346 non-null  float64
 7   NH3         19203 non-null  float64
 8   CO          27472 non-null  float64
 9   SO2         25677 non-null  float64
 10  O3          25509 non-null  float64
 11  Benzene     23908 non-null  float64
 12  Toluene     21490 non-null  float64
 13  Xylene      11422 non-null  float64
 14  AQI         24850 non-null  float64
 15  AQI_Bucket  24850 non-null  object 
dtypes: float64(13), object(3)
memory usage: 3.6+ MB
```

3. **Time Range of the dataset**

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

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

4. **Cities covered by the dataset - <u>Study area</u>**

```
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)
```

```
--------------------------------------------------------------------------------
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
```

Checking for missing values in the dataset. Sometimes the levels of some or all pollutants are not available due to lack of measuring or lack of required data points. This may lead to presence of NULL values in the dataset. So in the next step we check for missing values and what percentage of values are missing or not defined.

```
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)

    print('MISSING VALUES IN DATA:\n', missin_values_array, '\n')
    return missin_values_array
```

```
-------------------------------------------------
MISSING VALUES IN DATA:
-------------------------------------------------
             Missing Values  % of Total Values
Xylene                18109               61.3
PM10                  11140               37.7
NH3                   10328               35.0
Toluene                8041               27.2
Benzene                5623               19.0
AQI                    4681               15.9
AQI_Bucket             4681               15.9
PM2.5                  4598               15.6
NOx                    4185               14.2
O3                     4022               13.6
SO2                    3854               13.1
NO2                    3585               12.1
NO                     3582               12.1
CO                     2059                7.0
```

There are many pollutants that show missing values and the percentage of total values go up to 61.3%, therefore analyzing these pollutants is unreliable. So we will do a little processing to the data to try to avoid encountering these NULL values while working with machine learning models.

Data Preprocessing The data is preprocessor in a number of ways for a number of 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

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 and PM2.5 and PM10 are merged into a single pollutant which is named Particulate Matter to represent the two.

```
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)
    data['Particulate_Matter'] = data['PM2.5'] + data['PM10']
    return data
```

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

```
def subsetColumns(data):
    pollutants = ['Particulate_Matter', 'NO2', 'CO','SO2', 'O3',
'BTX']
    columns =  ['Date', 'City', 'AQI', 'AQI_Bucket'] + pollutants
    data = data[columns]
```

```
    return data, pollutants
```

Now, after the Stage 1 preprocessing the dataset looks like this:

```
--------------------------------------------------------------------------------
UPDATED DATA:
--------------------------------------------------------------------------------
```

	Date	City	AQI	AQI_Bucket	Particulate_Matter	NO2	CO	SO2	O3	BTX
0	2015-01-01	Ahmedabad	NaN	NaN	NaN	18.22	0.92	27.64	133.36	0.02
1	2015-01-02	Ahmedabad	NaN	NaN	NaN	15.69	0.97	24.55	34.06	12.95
2	2015-01-03	Ahmedabad	NaN	NaN	NaN	19.30	17.40	29.07	30.70	25.45
3	2015-01-04	Ahmedabad	NaN	NaN	NaN	18.48	1.70	18.59	36.08	15.57
4	2015-01-05	Ahmedabad	NaN	NaN	NaN	21.42	22.10	39.33	39.31	28.68
...	...	...	...	...	...	...	...	...	...	...
29526	2020-06-27	Visakhapatnam	41.0	Good	65.96	25.06	0.47	8.55	23.30	15.04
29527	2020-06-28	Visakhapatnam	70.0	Satisfactory	98.47	26.06	0.52	12.72	30.14	3.33
29528	2020-06-29	Visakhapatnam	68.0	Satisfactory	88.64	29.53	0.48	8.42	30.96	0.02
29529	2020-06-30	Visakhapatnam	54.0	Satisfactory	66.61	29.26	0.52	9.84	28.30	0.00
29530	2020-07-01	Visakhapatnam	50.0	Good	81.00	26.85	0.59	2.10	17.05	NaN

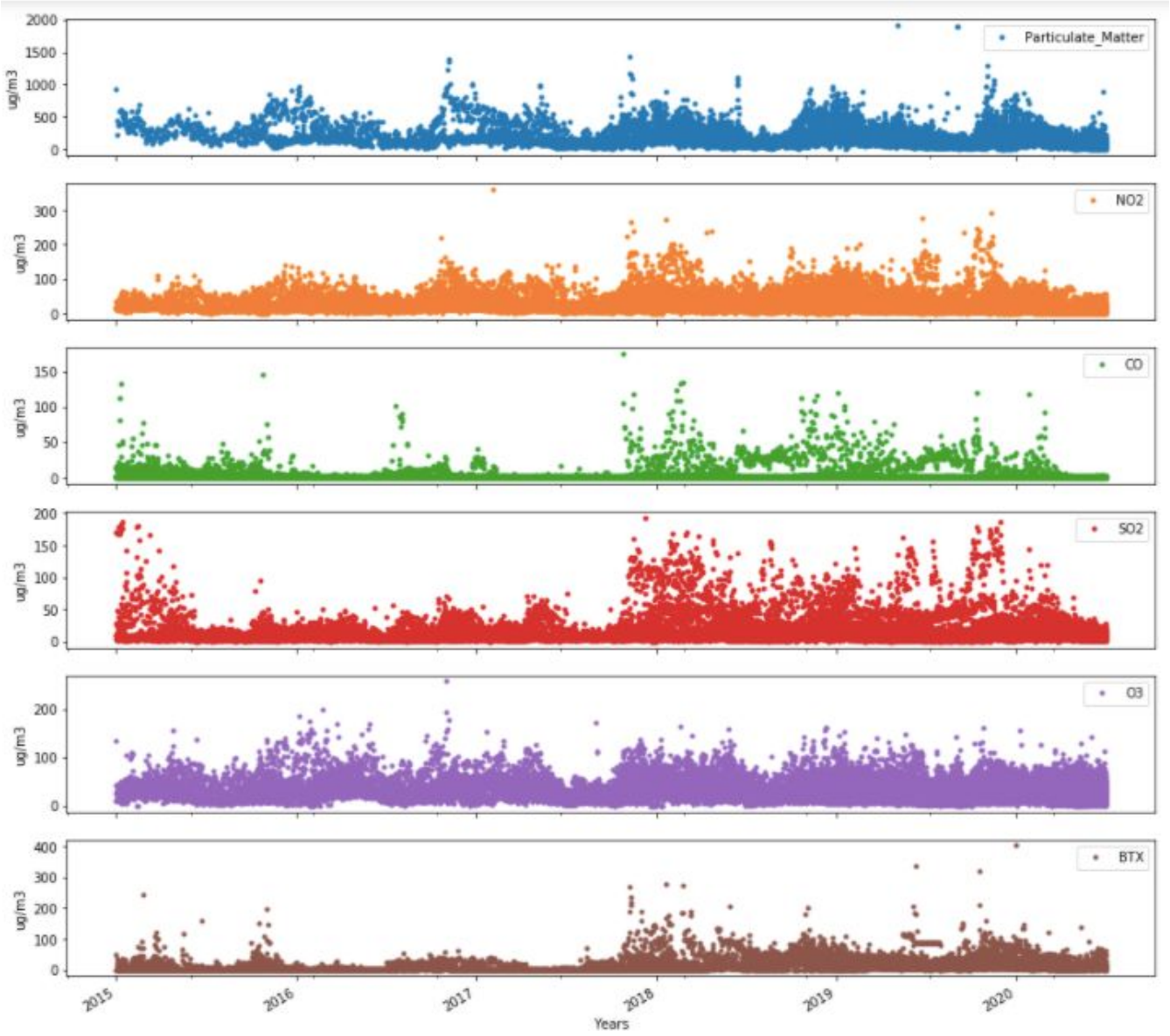
29531 rows × 10 columns

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 amount of pollutants in air over the years. First to get an overall bird-eye's view of the entire 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 visualisePollutants(data, columns):
    data.set_index('Date',inplace=True)
    axes = data[columns].plot(marker='.', linestyle='None',
figsize=(15, 15), subplots=True)
```

```
    for ax in axes:
        ax.set_xlabel('Years')
        ax.set_ylabel('ug/m3')
```



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.

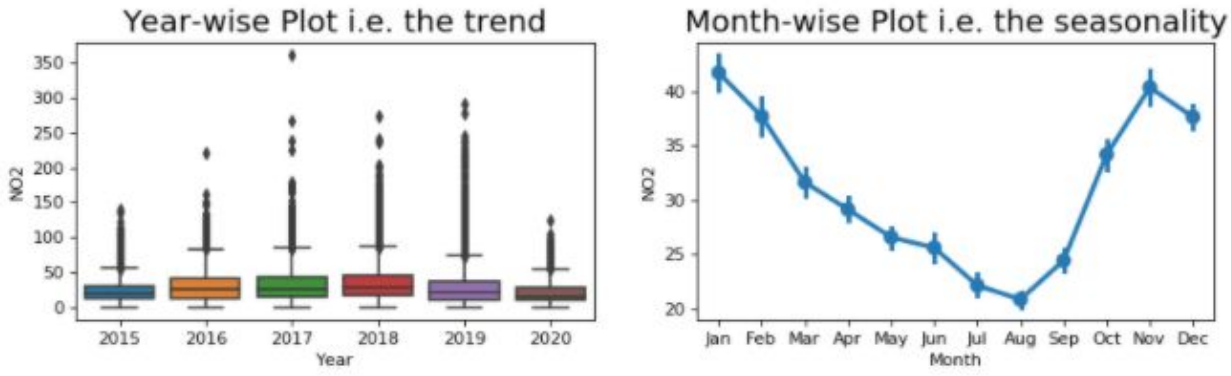
Visualising the amount of pollutants in air over the years and months. Next, to get a more detailed view of the trend and seasonality seen in each pollutant we look at the concentration of pollutants in the air over the entire time period of the dataset i.e. 2015 - 2020 and well as the months. The concentrations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

```
def trend_plot(updatedCityData, value):
    data = updatedCityData.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, we have mentioned some of them below:

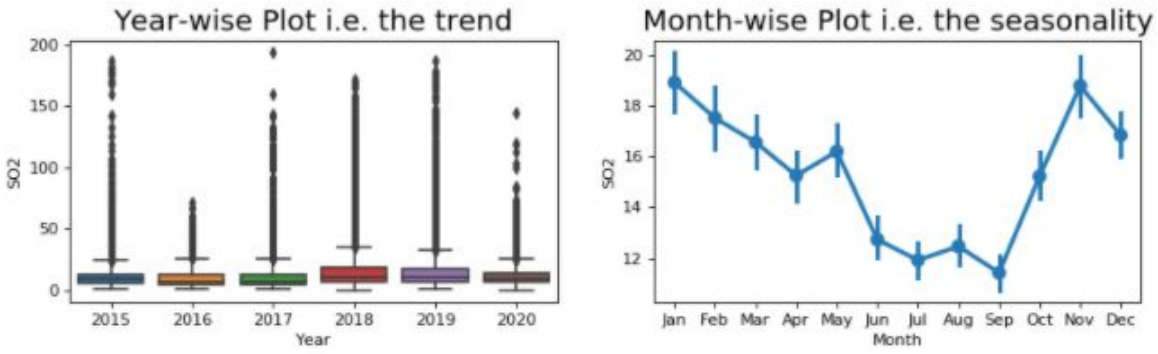
1. NO2

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



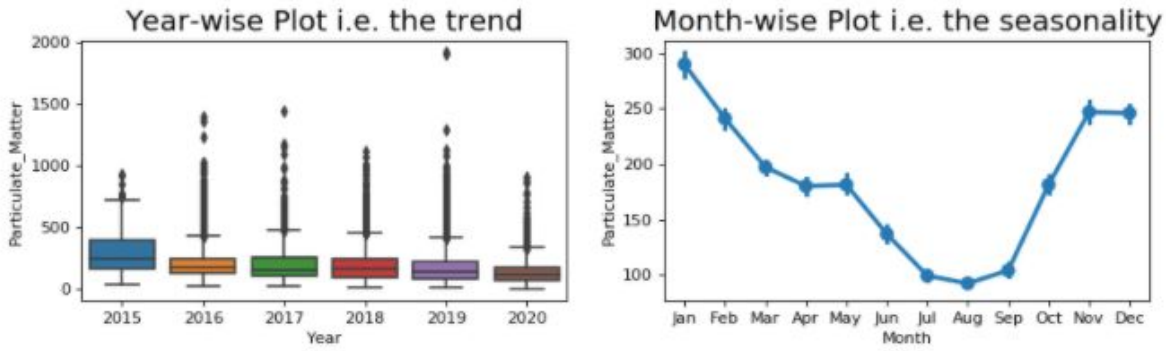
2. SO2

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



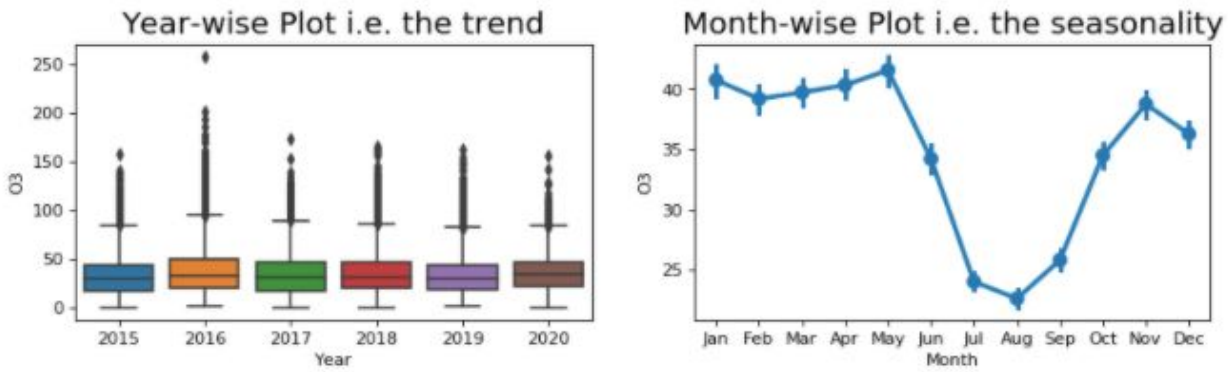
3. Particulate Matter

```
value='Particulate_Matter'
trend_plot(updatedCityData,value)
```



4. O3

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



Visualising cities with respect to each pollutant. Next, we look at which city is the most polluted when it comes to a certain pollutant. 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 mean concentration of the pollutant over the years.

```
def max_polluted_city(pollutant, data):
    x1 =
data[[pollutant,'City']].groupby(["City"]).mean().sort_values(by=poll
utant,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, we have mentioned some of them below:

1. Particulate Matter

```
pm = max_polluted_city('Particulate_Matter', updatedCityData)
```

	City	Particulate_Matter
0	Delhi	352.480000
1	Gurugram	289.980000
2	Talcher	233.790000
3	Jorapokhar	198.450000
4	Patna	192.580000
5	Brajrajnagar	189.680000
6	Kolkata	179.990000
7	Guwahati	179.890000
8	Jaipur	178.000000
9	Amritsar	173.950000

Delhi tops the list when it comes to PM, showing an average concentration of 352.48 ug/m3.

2. NO2

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

	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

Ahmedabad tops the list when it comes to PM, showing an average concentration of 52.03 ug/m3.

3. SO2

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

	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

Ahmedabad tops the list when it comes to PM, showing an average concentration of 55.25 ug/m3.

Visualising AQI over the years. Moving onto the main part of the project, the Air Quality Index. Let us first look at what AQI is, how it is calculated and how it determines the quality of air.

Air Quality Index. The **air quality index** (**AQI**) is an index for reporting air quality on a daily basis. It is a measure of how air pollution affects one's health within a short time 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 6am on a day will incorporate data from 6am on 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 are 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 minimum 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 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:

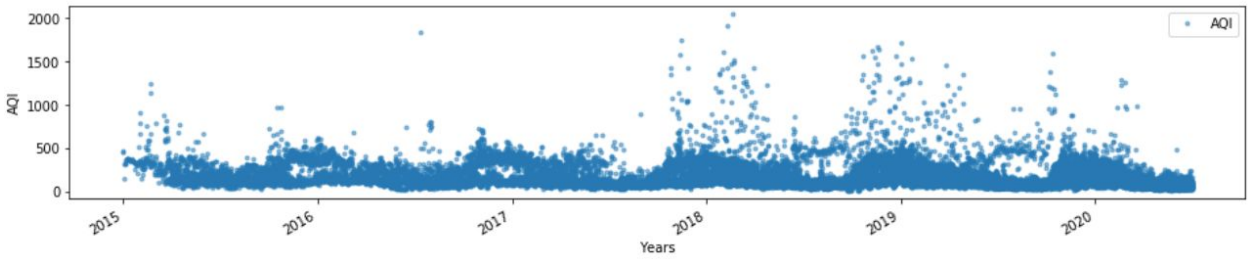
$$\text{AQI pollutant} = \left(\frac{\textit{pollutant concentration reading}}{\textit{Standard Concentration}} \right) x\ 100$$

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

Now, after understanding what AQI is and its significance, we plot and visualise the AQI over the years which spreads over 2015 to 2020 till mid-July:

```
def visualiseAQI(udata, columns):
    data = udata.copy()
    data.set_index('Date',inplace=True)

    axes = data[columns].plot(marker='.', alpha=0.5,
linestyle='None', figsize=(16, 3), subplots=True)
    for ax in axes:
        ax.set_xlabel('Years')
        ax.set_ylabel('AQI')
```



Here, if we look at the plot we can observe that the air quality has been taking a much greater downfall since later 2017 and AQI has been increasing ever since then till towards the end of 2019 and even in the beginning of 2020. However, in the later period of 2020 the AQI has decreased that means it has improved and we can safely assume that the reason behind the same is the nationwide lockdown due to the spread of the coronavirus.

Visualising cities with respect to AQI. Next, we look at which city has the worst air quality using AQI. To do so, we grouped each pollutant by city and calculated the mean AQI 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 mean AQI over the years.

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

	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

This shows that Ahmedabad has a very high AQI and we have seen that AQI> 400 is marked in the severe category which means the city has a very bad quality air. Following Ahmedabad is Delhi with an AQI of 259.49 - which is categorised into poor quality of air.

Additionally we also find the AQi of some major cities of India:

14	Chennai	114.500000
15	Hyderabad	109.210000
16	Mumbai	105.350000
20	Bengaluru	94.320000

Major Cities and AQI. Next, for focused visualisation we will look at some major cities and their AQI: Ahmedabad, Delhi, Mumbai, Kolkata, Hyderabad and Chennai.

Visualisation of AQI over the last two years. To do so, we have focused only on 2019 and 2020 for the six cities mentioned above.

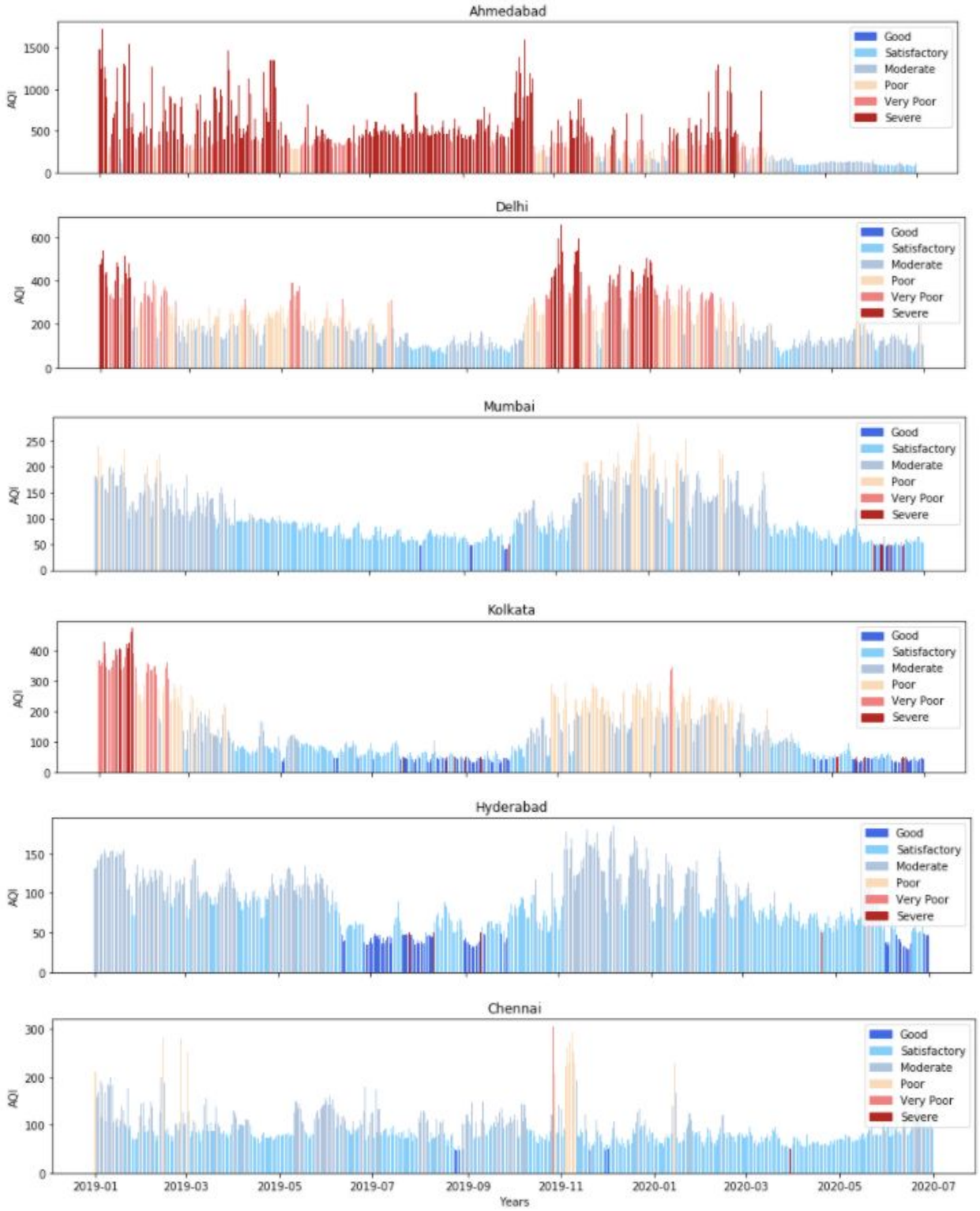
```
cities =
['Ahmedabad','Delhi','Mumbai','Kolkata','Hyderabad','Chennai']
filtered_city_day = updatedCityData[updatedCityData['Date'] >=
'2019-01-01']
```

```
AQI =
filtered_city_day[filtered_city_day.City.isin(cities)][['Date','City'
,'AQI','AQI_Bucket']]
```

To look at the AQI of the last two years, we have used Bar Graphs, where each bar represents the AQI for a particular day and is shown different colours depending on the category of the quality of air for that day.

```
def getColorBar(city):
    col = []
    for val in AQI_pivot[city]:
        if val < 50:
            col.append('royalblue')
        elif val > 50 and val < 101:
            col.append('lightskyblue') #cornflowerblue
        elif val > 100 and val < 201:
            col.append('lightsteelblue')
        elif val > 200 and val < 301:
            col.append('peachpuff')
        elif val > 300 and val < 401:
            col.append('lightcoral')
        else:
            col.append('firebrick')
    return col


ah = getColorBar('Ahmedabad')
de = getColorBar('Delhi')
mu = getColorBar('Mumbai')
ko = getColorBar('Kolkata')
hy = getColorBar('Hyderabad')
ch = getColorBar('Chennai')

colors = {'Good':'royalblue', 'Satisfactory':'lightskyblue',
'Moderate':'lightsteelblue', 'Poor':'peachpuff', 'Very
Poor':'lightcoral', 'Severe':'firebrick'}
labels = list(colors.keys())
handles = [plt.Rectangle((0,0),1,1, color=colors[label]) for label in
```

```
labels]

f, ((ax1, ax2, ax3, ax4, ax5, ax6)) = plt.subplots(6, 1,
sharex='col', sharey='row', figsize=(15,18))
ax1.bar(AQI_pivot.index, AQI_pivot['Ahmedabad'], color = ah, width =
0.75)
ax2.bar(AQI_pivot.index, AQI_pivot['Delhi'], color = de, width =
0.75)
ax3.bar(AQI_pivot.index, AQI_pivot['Mumbai'], color = mu, width =
0.75)
ax4.bar(AQI_pivot.index, AQI_pivot['Kolkata'], color = ko, width =
0.75)
ax5.bar(AQI_pivot.index, AQI_pivot['Hyderabad'], color = hy, width =
0.75)
ax6.bar(AQI_pivot.index, AQI_pivot['Chennai'], color = ch, width =
0.75)

ax1.legend(handles, labels, loc='upper right')
ax2.legend(handles, labels, loc='upper right')
ax3.legend(handles, labels, loc='upper right')
ax4.legend(handles, labels, loc='upper right')
ax5.legend(handles, labels, loc='upper right')
ax6.legend(handles, labels, loc='upper right')

ax1.title.set_text('Ahmedabad')
ax2.title.set_text('Delhi')
ax3.title.set_text('Mumbai')
ax4.title.set_text('Kolkata')
ax5.title.set_text('Hyderabad')
ax6.title.set_text('Chennai')

ax1.set_ylabel('AQI')
ax2.set_ylabel('AQI')
ax3.set_ylabel('AQI')
ax4.set_ylabel('AQI')
ax5.set_ylabel('AQI')
ax6.set_ylabel('AQI')
```

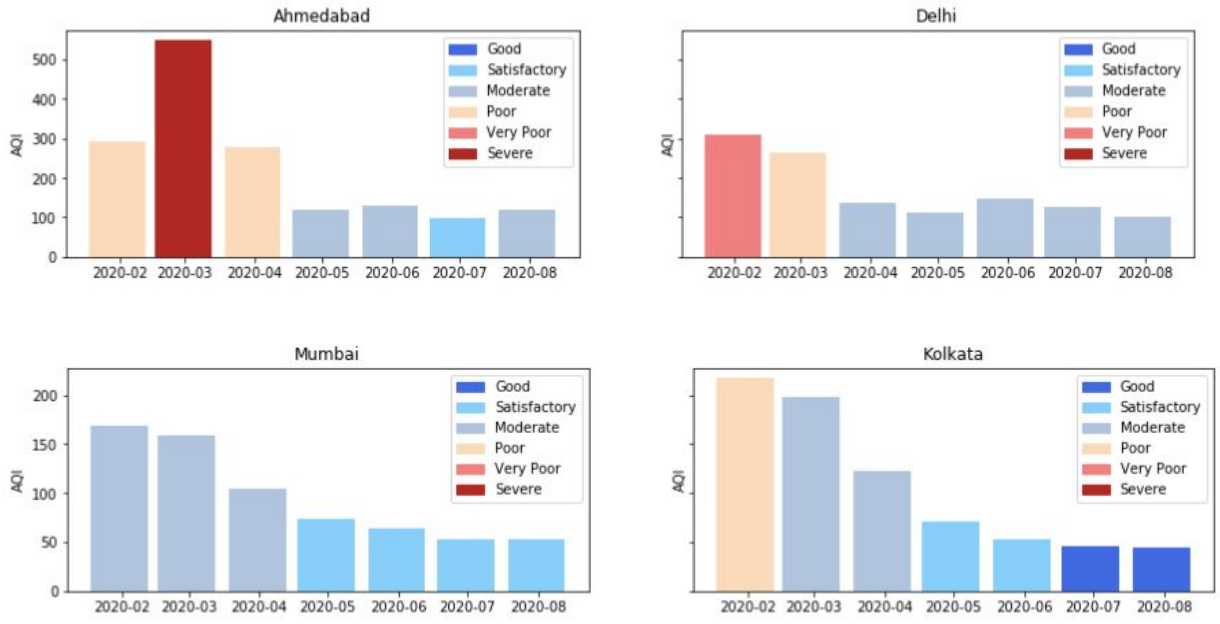

Ahmedabad
Delhi
Mumbai
Kolkata
Hyderabad
Chennai
AQI
Good
Satisfactory
Moderate
Poor
Very Poor
Severe
2019-01
2019-03
2019-05
2019-07
2019-09
2019-11
2020-01
2020-03
2020-05
2020-07
Years

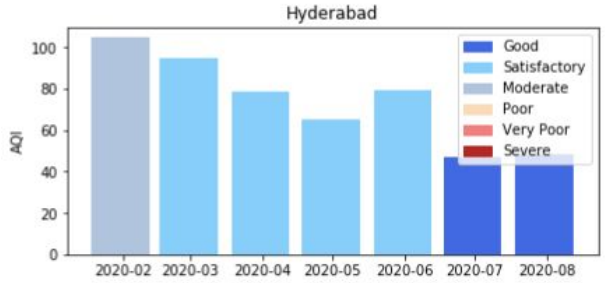
Now, to observe the effect of lockdown closely we perform two more types of visualisations.

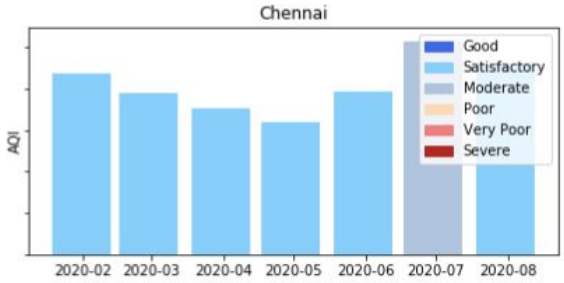
Visualisation of AQI in the year 2020 - City-wise. We again use bar graphs to show the AQI levels this time only for the year 2020. Where each bar represents the AQI for a particular month and is shown different colours depending on the category of the quality of air for that month. The monthly AQI is calculated by resampling the data with the frequency of month.

```
AQI_2020 = AQI_pivot[AQI_pivot.index > '2019-12-31']
AQI_2020 = AQI_2020.resample('M').mean()

def getColorBar(city):
    col = []
    for val in AQI_2020[city]:
        if val < 50:
            col.append('royalblue')
        elif val > 50 and val < 101:
            col.append('lightskyblue') #cornflowerblue
        elif val > 100 and val < 201:
            col.append('lightsteelblue')
        elif val > 200 and val < 301:
            col.append('peachpuff')
        elif val > 300 and val < 401:
            col.append('lightcoral')
        else:
            col.append('firebrick')
    return col

for i in range(0, 6, 2):
    city_1 = cities[i]
    city_2 = cities[i+1]
    fig, ((ax1, ax2)) =  plt.subplots(1, 2, sharex='col',
sharey='row', figsize=(15,3))
    ax1.bar(AQI_2020.index, AQI_2020[city_1], width = 25,
color=getColorBar(city_1))
    ax1.title.set_text(city_1)
    ax1.set_ylabel('AQI')
```

```
    colors = {'Good':'royalblue', 'Satisfactory':'lightskyblue',
'Moderate':'lightsteelblue', 'Poor':'peachpuff', 'Very
Poor':'lightcoral', 'Severe':'firebrick'}
    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')

    ax2.bar(AQI_2020.index, AQI_2020[city_2], width = 25,
color=getColorBar(city_2))
    ax2.title.set_text(city_2)
    ax2.set_ylabel('AQI')

    colors = {'Good':'royalblue', 'Satisfactory':'lightskyblue',
'Moderate':'lightsteelblue', 'Poor':'peachpuff', 'Very
Poor':'lightcoral', 'Severe':'firebrick'}
    labels = list(colors.keys())
    handles = [plt.Rectangle((0,0),1,1, color=colors[label]) for
label in labels]
    ax2.legend(handles, labels, loc='upper right')
```







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 Ahmedabad and in Delhi which can also be seen in the other cities like Mumbai and Kolkata.

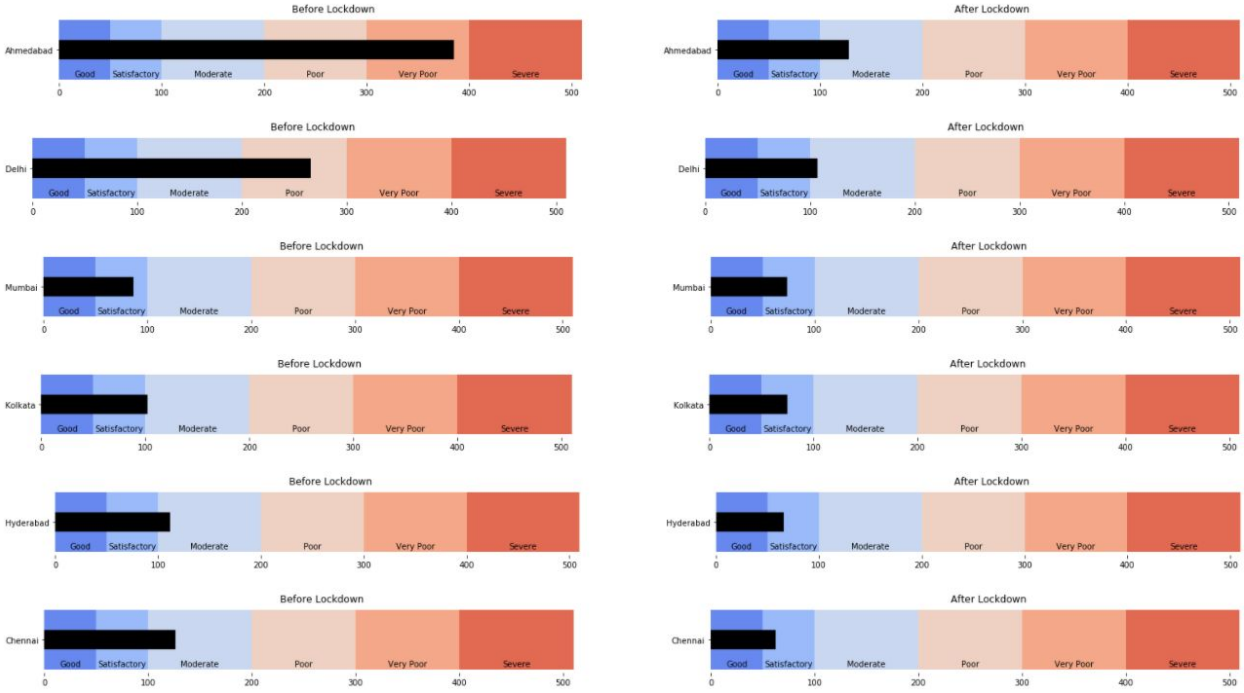
Kolkata and Ahmedabad have shown very good effects on AQI - so, Ahmedabad has gone from severe to satisfactory and moderate and Kolkata has gone from poor to good quality of air. And even in cities like Hyderabad where the quality of air was moderate, it has improved to even come in the good category.

Visualisation of AQI before and after Lockdown. We use bullet graphs (as information is presented in an easier to digest format) to show the AQI levels and their category in two views: one view is of before lockdown and the other is of after lockdown was imposed. Two create these two views the data is separated into two sets: AQI_beforelockdown and AQI_afterlockdown. The six categories that are in a displayed in different colours going from cool to warm analogous to from good to severe.

```
AQI_beforeLockdown = AQI_pivot['2015-01-01':'2020-03-25']
AQI_afterLockdown = AQI_pivot['2020-03-26':'2020-05-01']
limits = [50, 100, 200, 300, 400, 510]
palette = sns.color_palette("coolwarm", len(limits))
for city in cities:
    aqi_before = AQI_beforeLockdown[city].mean()
    aqi_after = AQI_afterLockdown[city].mean()
    fig, (ax1, ax2) = plt.subplots(1,2,figsize=(27, 2))
    ax1.set_yticks([1])
    ax1.set_yticklabels([city])
    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], aqi_before, color='black', height=5)

    # after lockdown
    ax2.set_yticks([1])
    ax2.set_yticklabels([city])
    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], aqi_after, color='black', height=5)

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

    rects = ax1.patches
    labels=["Good", "Satisfactory", "Moderate", "Poor", 'Very Poor',
'Severe']

    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')
```



Ahmedabad had a very poor air-quality however as it went to moderate during the lockdown. Similarly in other cities like Delhi, the air quality was poor and it reached almost satisfactory.

We can see that lockdown had a pretty good effect on the quality of air. Now, to understand the effects of lockdown if it were to continue we have trained two models which will be covered in the next section.

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 the 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.

Handling Missing Values: Even after we subset the pollutants and merged some columns, for future prediction of AQI we need to only focus on the AQI column and AQI_bucket column in our dataset and these two columns still have missing values in them. We start by subsetting our dataset to include only the required columns.

```
updatedCityData = updatedCityData[['City','Date','AQI','AQI_Bucket']]
```

	City	Date	AQI	AQI_Bucket
0	Ahmedabad	2015-01-01	NaN	NaN
1	Ahmedabad	2015-01-02	NaN	NaN
2	Ahmedabad	2015-01-03	NaN	NaN
3	Ahmedabad	2015-01-04	NaN	NaN
4	Ahmedabad	2015-01-05	NaN	NaN

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 for some cities, but we noticed that the data got distorted as the AQI 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 for most of the cities. 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 removal of the rows and columns
- Imputing the approximations add variance and bias

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 gave the best results for the very few cities it worked but since it would not be a robust approach as the model will fail for most of the

cities, 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 cities 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 evaluation our model we also calculated bias to understand its effects.

```
updatedCityData['AQI'] =
updatedCityData['AQI'].fillna(updatedCityData['AQI'].mean(axis=0))
updatedCityData
```

	City	Date	AQI	AQI_Bucket
0	Ahmedabad	2015-01-01	166.463581	NaN
1	Ahmedabad	2015-01-02	166.463581	NaN
2	Ahmedabad	2015-01-03	166.463581	NaN
3	Ahmedabad	2015-01-04	166.463581	NaN
4	Ahmedabad	2015-01-05	166.463581	NaN
...	...	...	...	...
29526	Visakhapatnam	2020-06-27	41.000000	Good
29527	Visakhapatnam	2020-06-28	70.000000	Satisfactory
29528	Visakhapatnam	2020-06-29	68.000000	Satisfactory
29529	Visakhapatnam	2020-06-30	54.000000	Satisfactory
29530	Visakhapatnam	2020-07-01	50.000000	Good

Changing data structure to City-wise Monthly data: For ease in observing data for a particular city and observing data for a particular year the data structure was changed.

```
#using cities for Comparision
cities=pd.unique(updatedCityData['City'])
column1= cities+'_AQI'
column2=cities+'_AQI_Bucket'
columns=[*column1,*column2]

#Converting to Monthly data for easier usage
```

```
final_data=
pd.DataFrame(index=np.arange('2015-01-01','2020-07-02',dtype='datetim
e64[D]'), columns=column1)
for city,i in zip(cities, final_data.columns):
    n = len(np.array(updatedCityData[updatedCityData['City'] ==
city]['AQI']))
    final_data[i][-n:] =
np.array(updatedCityData[updatedCityData['City']==city]['AQI'])

final_data=final_data.astype('float64')
final_data=final_data.resample(rule='MS').mean()
```

	Ahmedabad_AQI	Aizawl_AQI	Amaravati_AQI	Amritsar_AQI	Bengaluru_AQI	Bhopal_AQI	Brajrajnagar_AQI	Chandigarh_AQI	Chennai_AQI	Coimbat
2020-03-01	344.645161	66.105263	52.548387	95.387097	90.741935	122.709677	152.354839	57.096774	70.290323	85
2020-04-01	121.900000	40.300000	44.400000	59.866667	68.533333	108.400000	140.666667	44.233333	63.500000	82
2020-05-01	129.774194	24.193548	59.096774	77.677419	73.161290	104.451613	144.161290	74.129032	78.677419	57
2020-06-01	98.066667	20.800000	47.866667	101.533333	55.166667	71.666667	113.033333	66.500000	103.066667	39
2020-07-01	119.000000	20.000000	54.000000	78.000000	43.000000	69.000000	76.000000	66.000000	92.000000	33

As we conclude the second stage of data preprocessing, here are some examples showing how the data has become more accessible and model friendly:

1. Observing data for a particular city: Now we can easily call our model to be trained on the data for a particular city and predict its future AQI values.

```
delhi_aqi = final_data['Delhi_AQI']
delhi_aqi.tail()
```

```
2020-03-01    135.838710
2020-04-01    113.000000
2020-05-01    148.645161
2020-06-01    125.733333
2020-07-01    101.000000
Freq: MS, Name: Delhi_AQI, dtype: float64
```

2. Observing data for a particular year: To train and test our model we can easily fetch data for particular years and create train and test datasets.

```
#Obesrving the data for Last Year
data_2019 = final_data['2019-01-01':'2019-12-31']
data_2019
```

	Ahmedabad_AQI	Amritsar_AQI	Bengaluru_AQI	Brajrajnagar_AQI	Chennai_AQI	Delhi_AQI
2019-01-01	755.516129	96.950438	116.806452	259.967742	131.000000	365.741935
2019-02-01	560.535714	94.892857	106.250000	206.766556	118.071429	258.178571
2019-03-01	601.741935	101.885922	122.129032	186.870968	101.419355	197.354839
2019-04-01	586.813025	112.000000	124.300000	179.316338	87.533333	219.400000
2019-05-01	373.042513	113.258065	105.387097	133.948089	103.612903	235.838710
2019-06-01	395.133333	106.000000	70.533333	111.261811	112.666667	197.766667
2019-07-01	470.402051	73.709677	63.741935	99.851314	88.709677	145.096774
2019-08-01	516.870968	61.402051	59.354839	126.050419	85.548387	99.290323
2019-09-01	436.179691	79.833333	72.966667	94.966667	103.633333	106.533333
2019-10-01	606.660116	149.161290	81.129032	111.156591	102.903226	251.000000
2019-11-01	451.000000	192.766667	96.200000	158.197572	121.966667	356.100000
2019-12-01	293.548387	137.935484	81.774194	141.548387	80.161290	353.903226

3. Observing data for a particular city and year:

```
# do not use Amritsar's AQI
data_2019 = data_2019.drop(['Amritsar_AQI'],axis=1)
data_2019
```

	Ahmedabad_AQI	Bengaluru_AQI	Brajrajnagar_AQI	Chennai_AQI	Delhi_AQI
2019-01-01	755.516129	116.806452	259.967742	131.000000	365.741935
2019-02-01	560.535714	106.250000	206.766556	118.071429	258.178571
2019-03-01	601.741935	122.129032	186.870968	101.419355	197.354839
2019-04-01	586.813025	124.300000	179.316338	87.533333	219.400000
2019-05-01	373.042513	105.387097	133.948089	103.612903	235.838710
2019-06-01	395.133333	70.533333	111.261811	112.666667	197.766667
2019-07-01	470.402051	63.741935	99.851314	88.709677	145.096774
2019-08-01	516.870968	59.354839	126.050419	85.548387	99.290323
2019-09-01	436.179691	72.966667	94.966667	103.633333	106.533333
2019-10-01	606.660116	81.129032	111.156591	102.903226	251.000000
2019-11-01	451.000000	96.200000	158.197572	121.966667	356.100000
2019-12-01	293.548387	81.774194	141.548387	80.161290	353.903226

Models Used. We started with finding the most suitable models to train out time-series data on. And looking at the AQI plot of India, 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

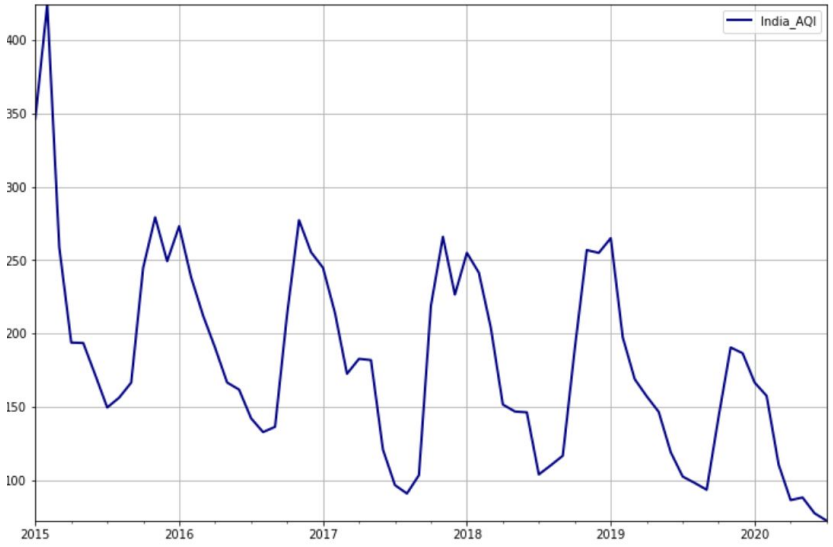
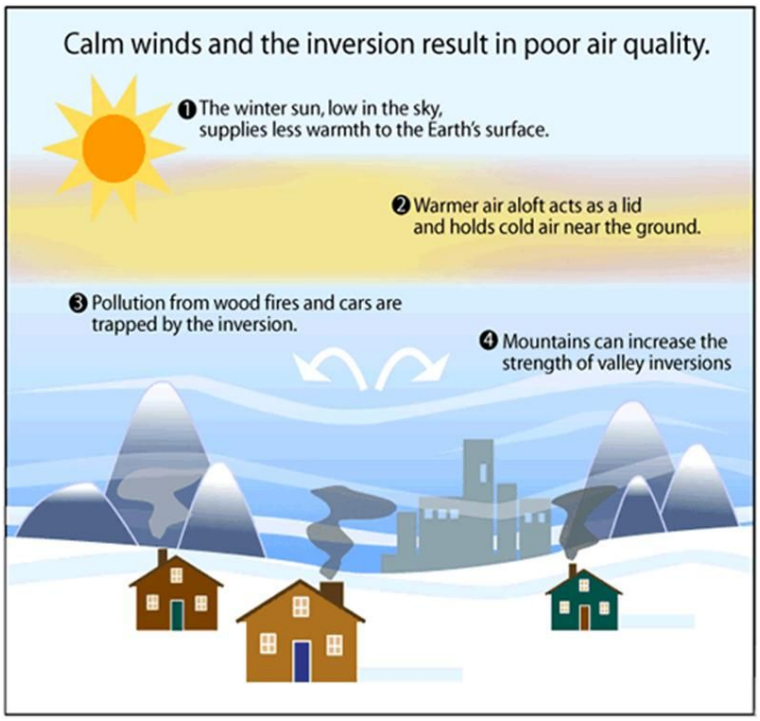


Fig. AQI plot of India

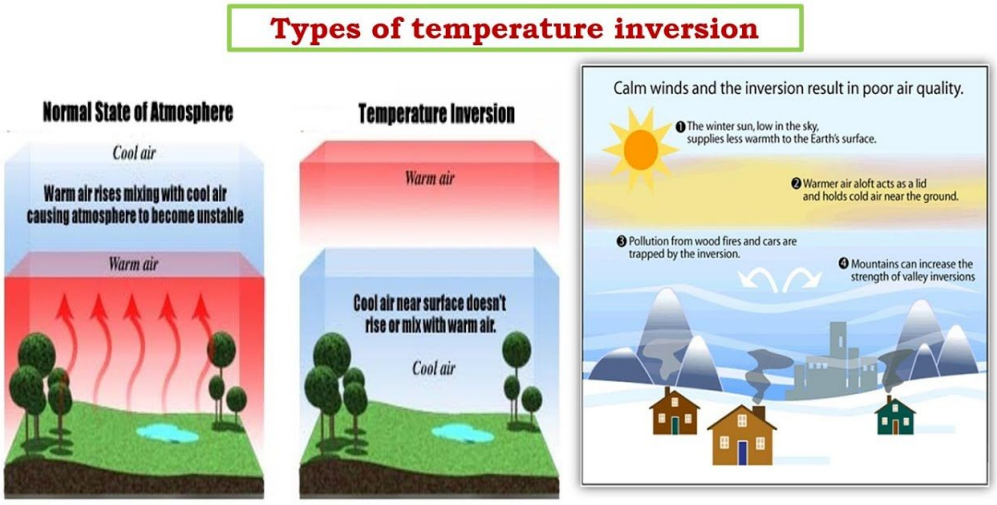
Next, we will look at what this seasonal component is and why it is observed.

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, 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.



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.

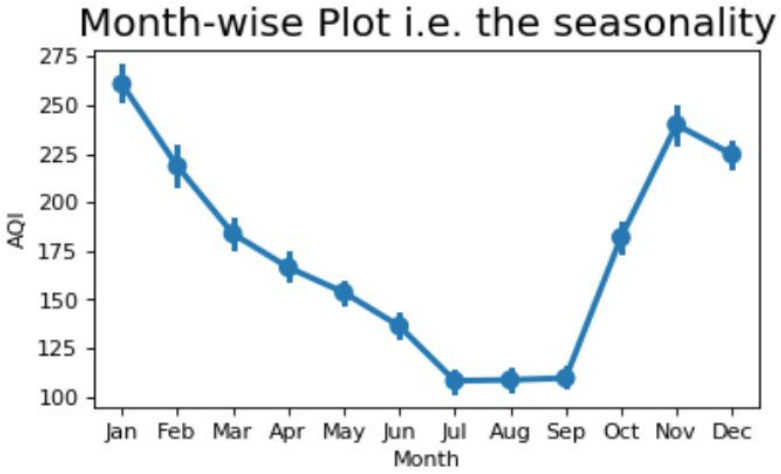


Fig. Seasonality in AQI

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 for 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 Hyderabad AQI data is shown below.

```
#seasonal decomposition of Delhi AQI
from statsmodels.tsa.seasonal import seasonal_decompose
Hyderabad_AQI = final_data['Hyderabad_AQI']
result = seasonal_decompose(Hyderabad_AQI, model='multiplicative')
result.plot();
Hyderabad_AQI
```

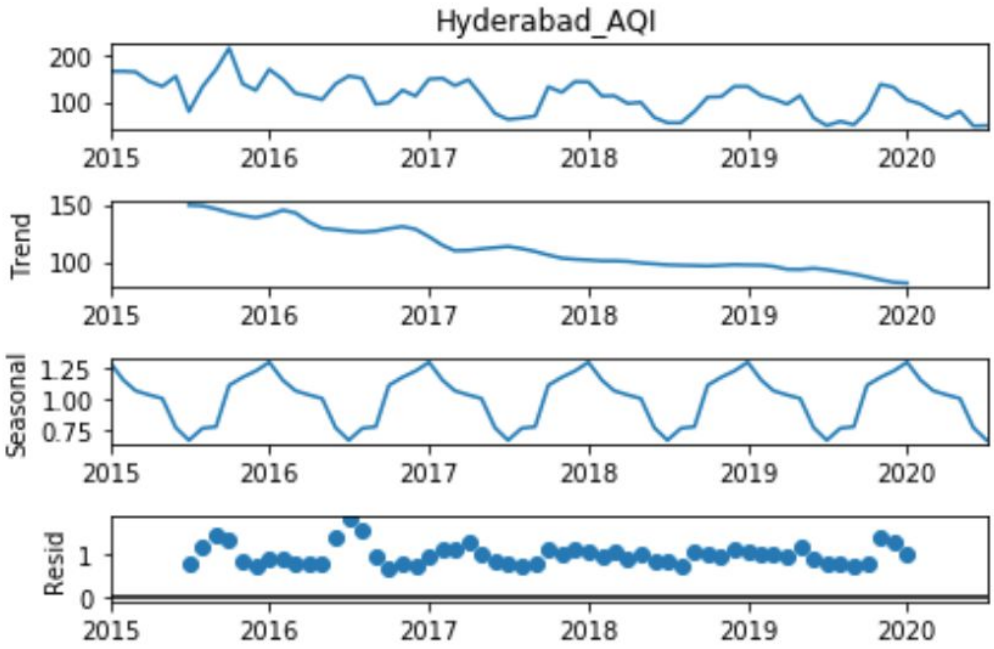


Fig. Seasonal Decomposition of Hyderabad AQI

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
```

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 snippet given below trains the SARIMAX model on the Hyderabad_AQI data: First , the model is defined on Hyderabad 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.This Model is then trained 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=Hyderabad_AQI,start_p=0,start_P=0,start_q=0,start_Q=0,se
asonal=True, m=12)

#dividing into train and test:
train = Hyderabad_AQI[:41] #from 2015-2018
test = Hyderabad_AQI[42:54] # june-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:
final_model =
SARIMAX(Hyderabad_AQI,order=(1,1,1),seasonal_order=(1,0,1,12))
results = final_model.fit()
results.summary()
```

The results of the model will be discussed in the next section.

Model 2. Recurrent Neural Network. A type of Neural Network which is used for time-based/frequency-based/memory-based data like text data, speech, time series etc. We will

be using a particular cell type LSTM (Long Short term memory).

Advantages:

- LSTM networks are particularly meant to keep particular information for a longer term as compared to regular RNN's.

Disadvantages:

- RNN is a black-box method, which means there is little transparency in the model and how it trains.
- Also, there is the high complexity of hyperparameters(initial parameters before training, more).

Step 1. Min Max Scaler: Transform features by scaling each feature to a given range. Here we are scaling and translating each feature individually such that it comes inside the range zero and one.

```
Hyderabad_AQI=Hyderabad_AQI.reset_index()
Hyderabad_AQI.columns=['ds','y']
Hyderabad_AQI=Hyderabad_AQI.set_index('ds')
train=Hyderabad_AQI[:-24]
test=Hyderabad_AQI[-24:-12]

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(train)
scaled_train = scaler.transform(train)
scaled_test = scaler.transform(test)
```

Step 2. Time Series Generator: Using this we are creating an instance of the class where input and output series are same which is scaled_input. Now, we will use it to train our neural network model. This class both informs what the model will learn and how we intend to use the model in the future when making predictions.

```
from keras.preprocessing.sequence import TimeseriesGenerator
n_input = 24
```

```
n_features = 1
generator = TimeseriesGenerator(scaled_train, scaled_train,
length=n_input, batch_size=1)
X,y = generator[0]
print(f'Given the Array: \n{X.flatten()}')
print(f'Predict this y: \n {y}')
```

```
Given the Array:
[0.68919577 0.68919577 0.67998959 0.55262191 0.48603131 0.62205931
 0.14880127 0.48325738 0.7115316  1.         0.52153094 0.43293045
 0.71428571 0.57854102 0.39350109 0.35619425 0.3085001  0.52071197
 0.62631266 0.5949689  0.24865991 0.2706414  0.43553926 0.35744006]
Predict this y:
 [[0.58450565]]
```

Step 3. Creating and Fitting the Model: Here we are fitting our model with the time-series generator created earlier. We have used 250 epochs. Here epochs represent the number of passes of the entire training dataset that our model has done to get trained.

```
### Creating the model:
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
# defining the model:
model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(n_input,
n_features)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')

model.summary()
```

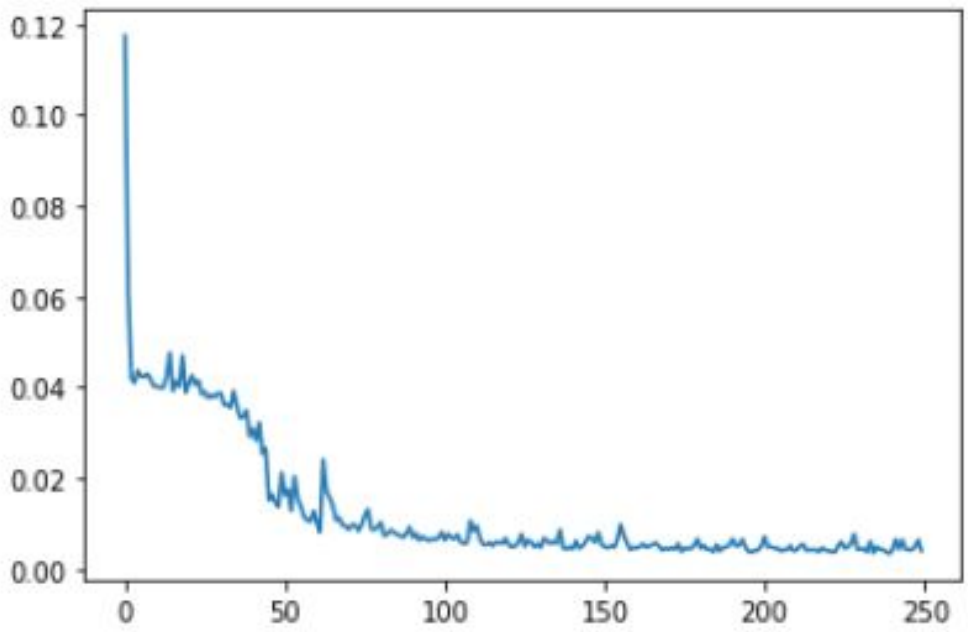
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 50)                10400
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 51
=================================================================
Total params: 10,451
Trainable params: 10,451
Non-trainable params: 0
_________________________________________________________________
```

```
# Fitting the model with the generator object:
model.fit_generator(generator,epochs=250)
```

```
Epoch 1/250
43/43 [==============================] - 0s 11ms/step - loss: 0.0517
Epoch 2/250
43/43 [==============================] - 0s 10ms/step - loss: 0.0374
Epoch 3/250
43/43 [==============================] - 0s 10ms/step - loss: 0.0361
Epoch 4/250
43/43 [==============================] - 0s 11ms/step - loss: 0.0341
Epoch 5/250
43/43 [==============================] - 0s 10ms/step - loss: 0.0328
Epoch 6/250
43/43 [==============================] - 0s 10ms/step - loss: 0.0324
Epoch 7/250
43/43 [==============================] - 0s 11ms/step - loss: 0.0302
Epoch 8/250
43/43 [==============================] - 0s 9ms/step - loss: 0.0295
Epoch 9/250
43/43 [==============================] - 0s 10ms/step - loss: 0.0283
```

```
We observe that as each epoch gets over we observed that loss (error
over the training set) reduces:
loss_per_epoch = model.history.history['loss']
plt.plot(range(len(loss_per_epoch)),loss_per_epoch)
test_predictions = []

first_eval_batch = scaled_train[-n_input:]
current_batch = first_eval_batch.reshape((1, n_input, n_features))
```

```
for i in range(len(test)):
    current_pred = model.predict(current_batch)[0]
    test_predictions.append(current_pred)
    current_batch =
np.append(current_batch[:,1:,:],[[current_pred]],axis=1)
```



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.

```
scaler.fit(Hyderabad_AQI)
scaled_City_AQI=scaler.transform(Hyderabad_AQI)
generator = TimeseriesGenerator(scaled_City_AQI, scaled_City_AQI,
length=n_input, batch_size=1)
test_predictions = []

first_eval_batch = scaled_City_AQI[-n_input:]
```

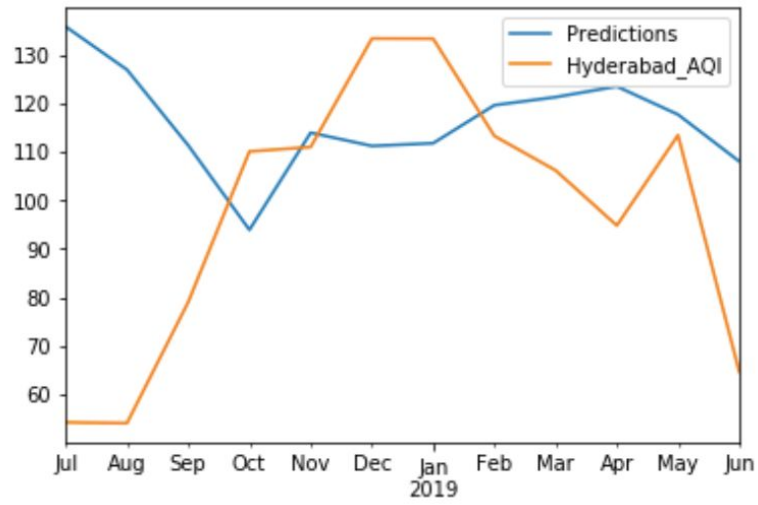
```
current_batch = first_eval_batch.reshape((1, n_input, n_features))

for i in range(len(test)):
    current_pred = model.predict(current_batch)[0]
    test_predictions.append(current_pred)
    current_batch =
np.append(current_batch[:,1:,:],[[current_pred]],axis=1)
```

Results and Discussion

Now we will look at the predictions made by the two models for the two tasks - one of predicting the AQI values for 2019 ans another for 2021.

Model 1. SARIMAX. First, we look at the predictions for the 2019 AQI values.

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



The evaluation metric used is Root Mean Squared Error and Bias. We observe that the predicted

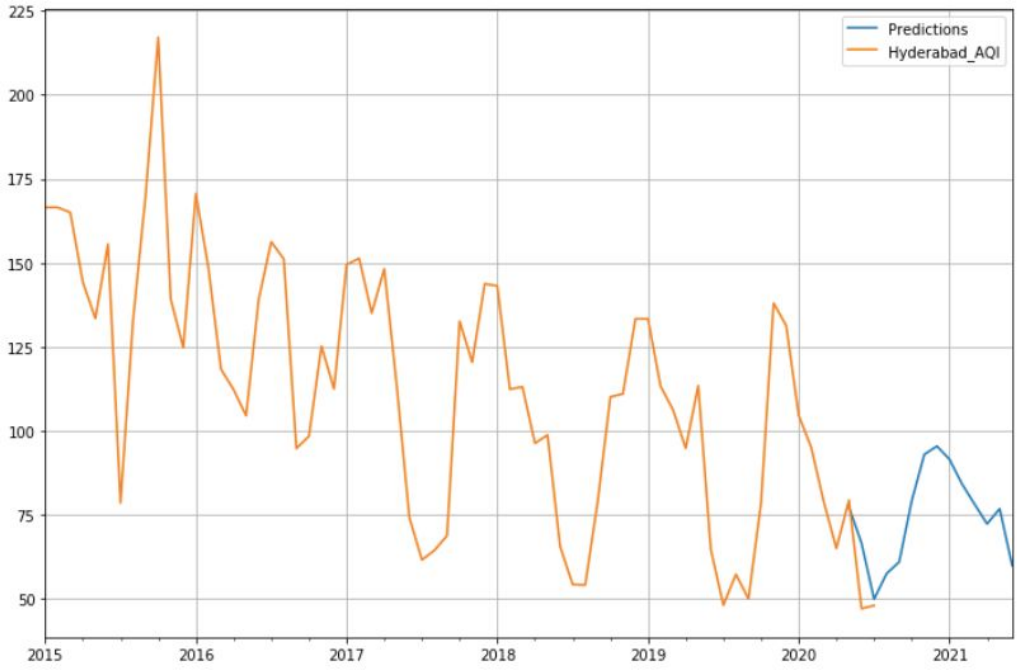
mean AQI for Hyderabad in 2019 is 97.277.

```
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:  37.85122379249102
Mean AQI: 97.27746415770609
Bias: -18.947500
```

Next, we look at the predictions for the unknown i.e. 2021 AQI values.

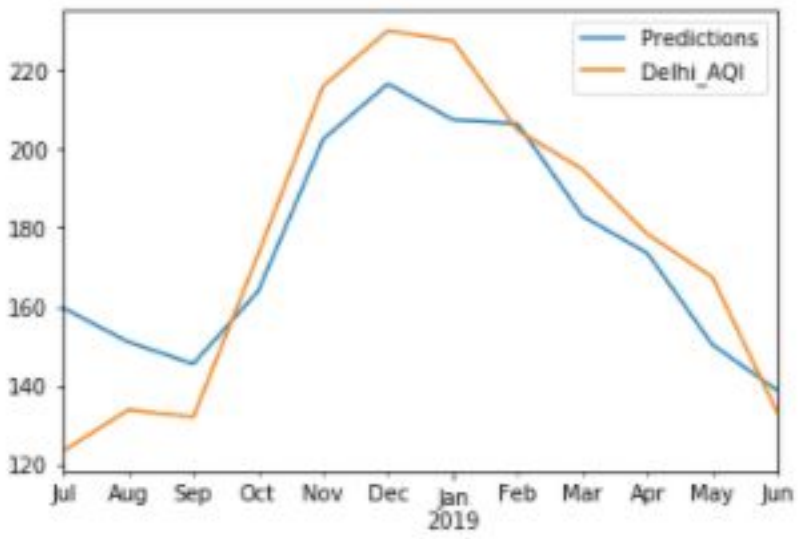
```
#Obtaining predicted values:
predictions = results.predict(start=64, end=77,
typ='levels').rename('Predictions')

#Plotting predicted values against the true values:
predictions.plot(legend=True)
Hyderabad_AQI.plot(legend=True,figsize=(12,8),grid=True);
```

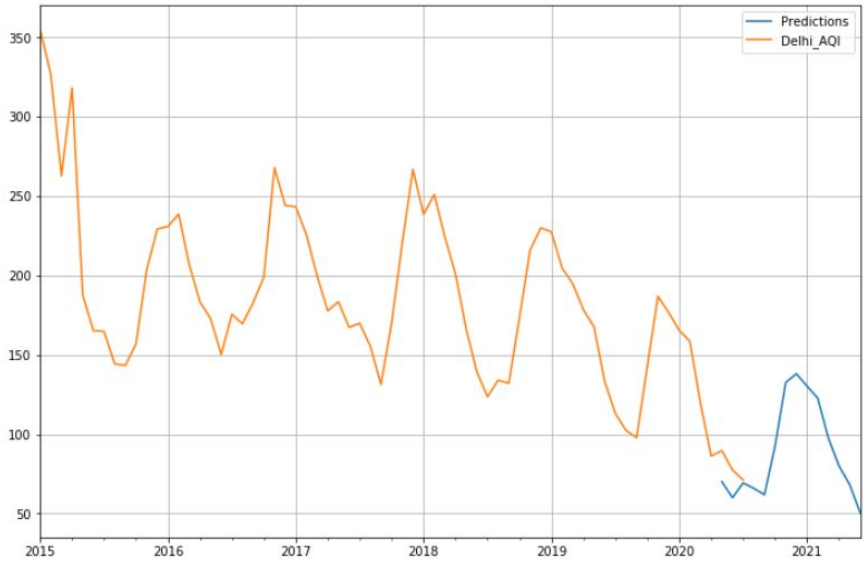


- We can see the predictions plotted are in continuation with 2020.
- We also note that the 2021 prediction is a highly optimistic one i.e. approximately 70 AQI which is close enough to Good category. We also observe that for some months the predicted AQI values go as low as 50.
- That is purely due to the fact that 2020 is such an outlier. So, we can say that if lockdown were to continue, the air will continue to 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.

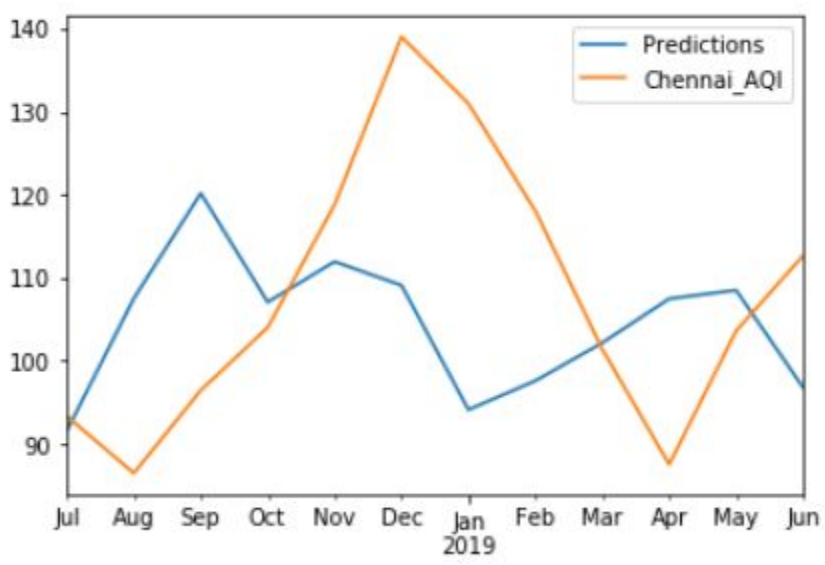
Predictions for some other cities using SARIMAX (the first plot shows AQI 2019 predictions and the second plot shows AQI 2021 predictions)are shown below:

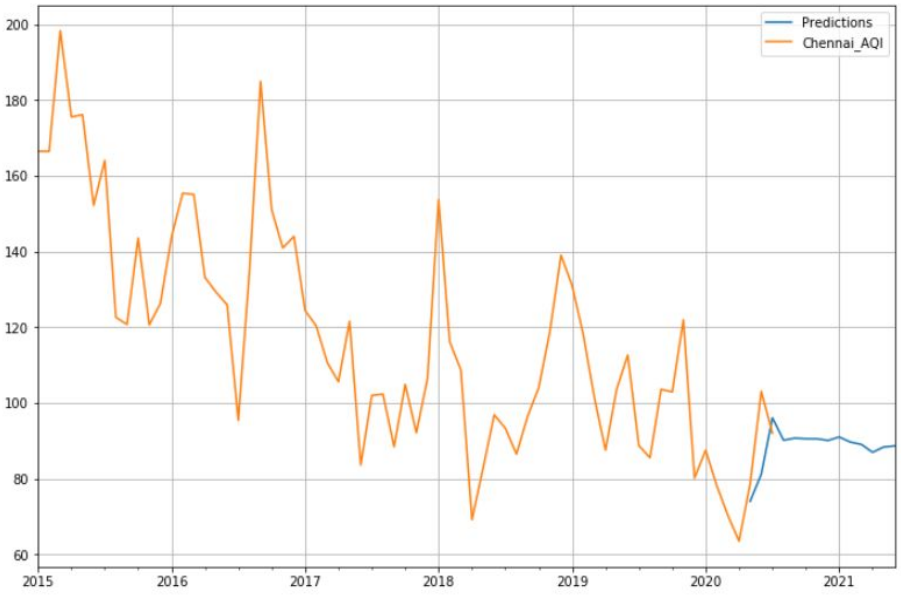
1. Delhi


Predictions
Delhi_AQI
220
200
180
160
140
120
Jul
Aug
Sep
Oct
Nov
Dec
Jan
2019
Feb
Mar
Apr
May
Jun

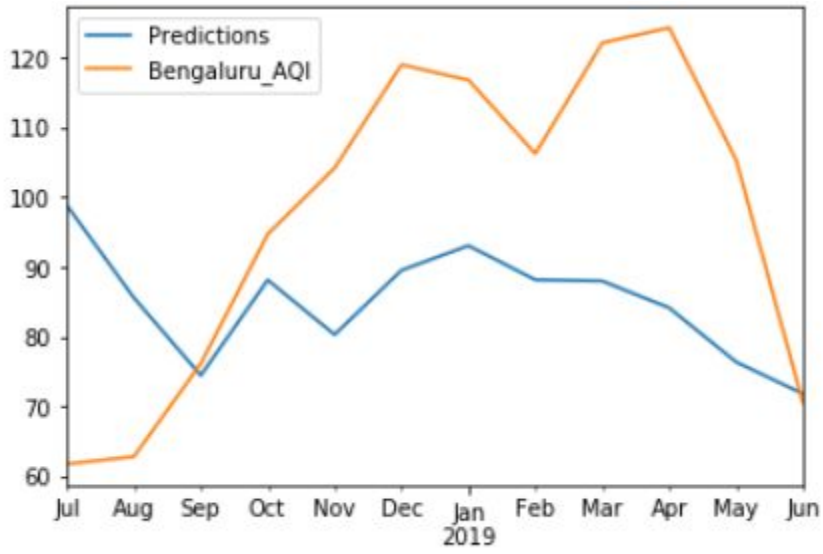

Predictions
Delhi_AQI
350
300
250
200
150
100
50
2015
2016
2017
2018
2019
2020
2021

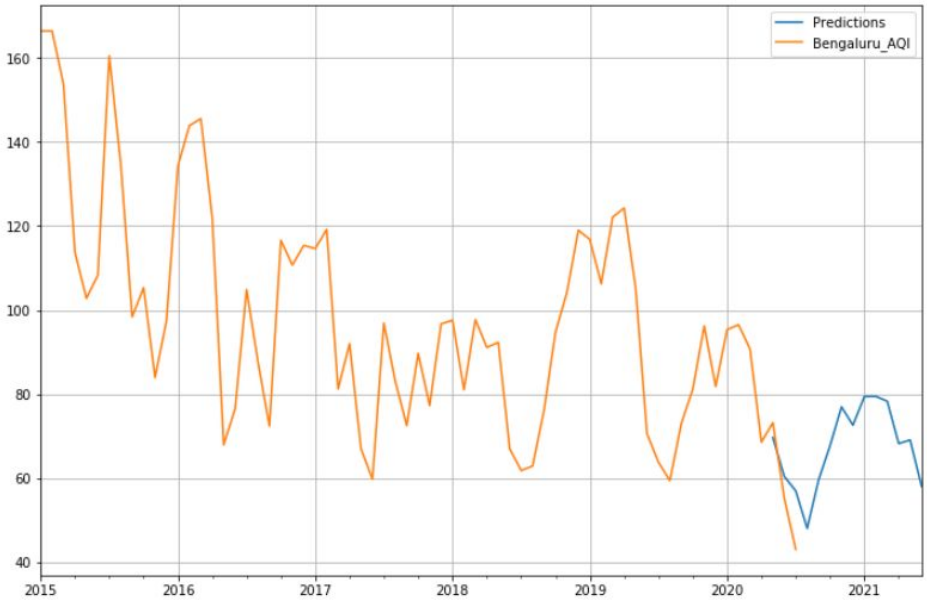
2. Chennai


Predictions
Chennai_AQI
140
130
120
110
100
90
Jul
Aug
Sep
Oct
Nov
Dec
Jan
2019
Feb
Mar
Apr
May
Jun


Predictions
Chennai_AQI
200
180
160
140
120
100
80
60
2015
2016
2017
2018
2019
2020
2021

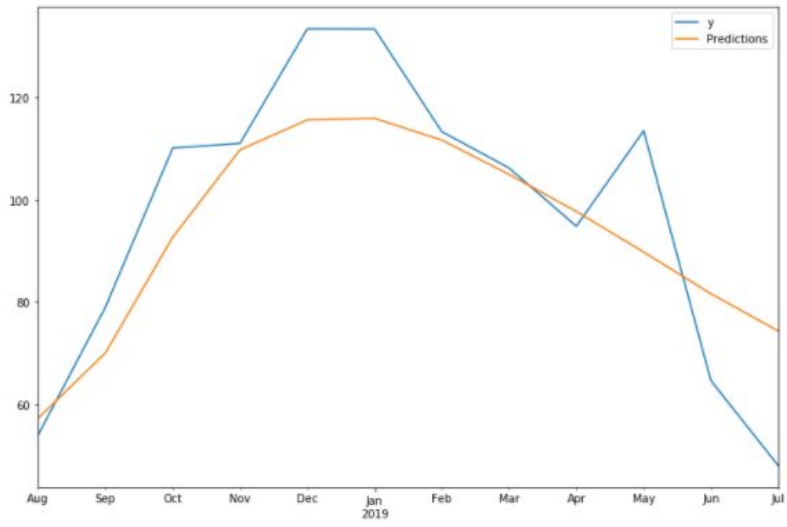
3. Bengaluru





Model 2. RNN. First, we look at the predictions for the 2019 AQI values.

```
true_predictions = scaler.inverse_transform(test_predictions)
test['Predictions'] = true_predictions
test.plot(figsize=(12,8))
plt.plot(true_predictions)
```



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

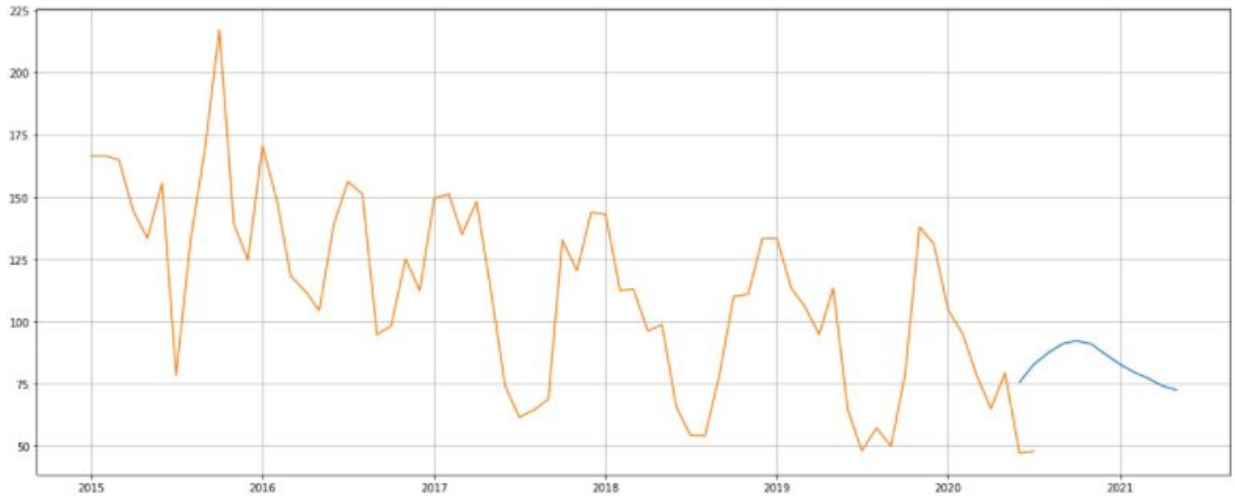
```
RMSE=np.sqrt(mean_squared_error(test['y'],test['Predictions']))
print('RMSE = ',RMSE)
print('Hyderabad_AQI=',Hyderabad_AQI['y'].mean())
forecast_errors = [test[i]-predictions[i] for i in range(len(test))]
bias = sum(forecast_errors) * 1.0/len(test)
print('Bias: %f' % bias)
```

```
RMSE =  16.625953834298937
```

Next, we look at the predictions for the unknown i.e. 2021 AQI values.

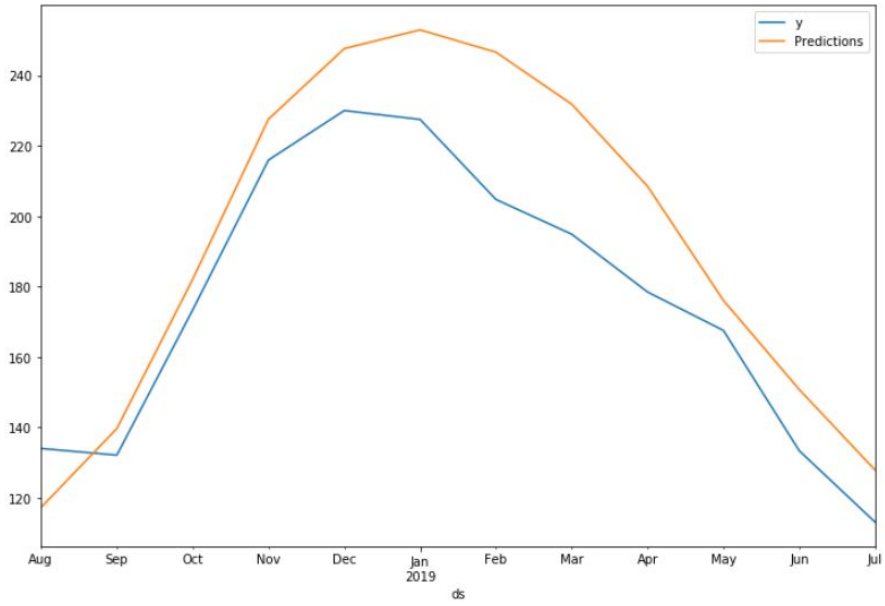
```
true_predictions = scaler.inverse_transform(test_predictions)
true_predictions=true_predictions.flatten()
true_preds=pd.DataFrame(true_predictions,columns=['Forecast'])
true_preds=true_preds.set_index(pd.date_range('2020-06-01',periods=12
,freq='MS'))
plt.figure(figsize=(20,8))
plt.grid(True)
plt.plot( true_preds['Forecast'])
```

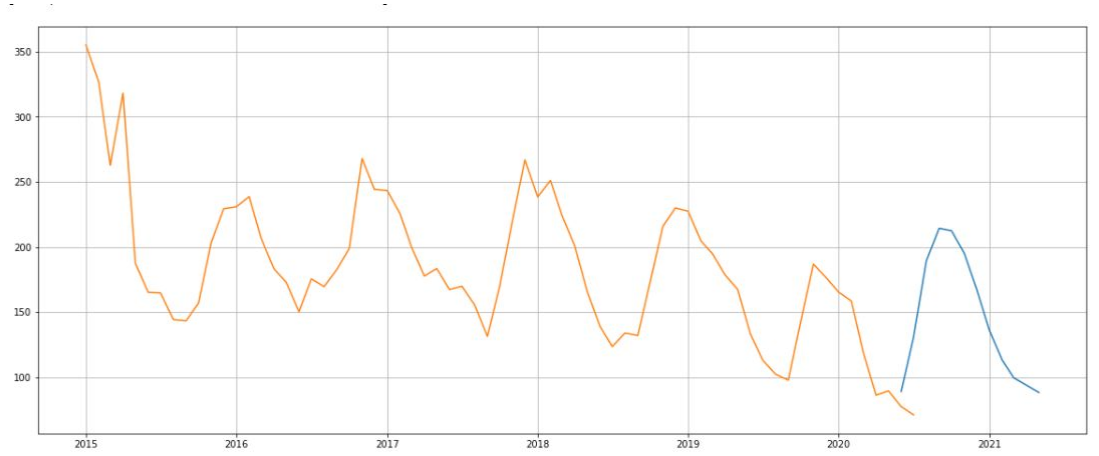
```
plt.plot( Hyderabad_AQI['y'])
```



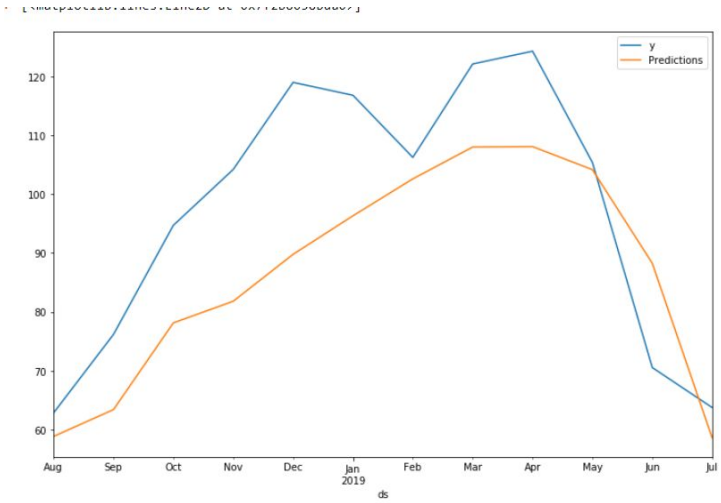
Predictions for some other cities using RNN are shown below(the first plot shows AQI 2019 predictions and the second plot shows AQI 2021 predictions):

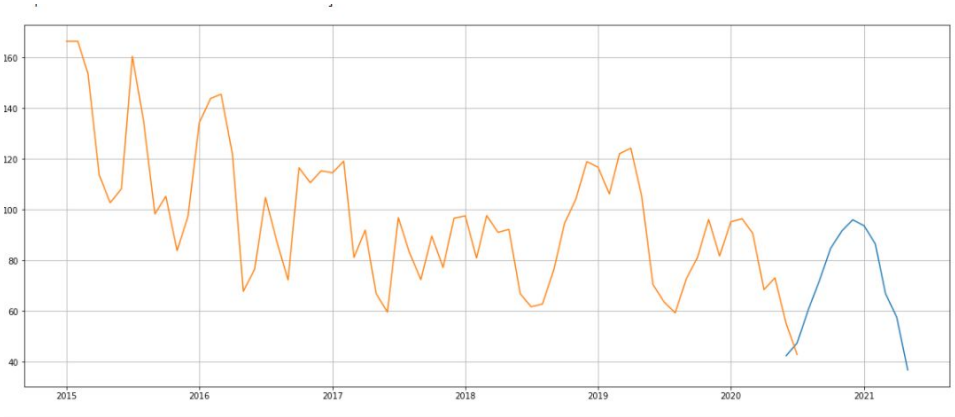
1. Delhi





2. Bangaluru





Discussion. If we compare the models we see that:

- It is obvious that RMSE for Delhi is very high using RNN but if apply the model on a different city perform better than SARIMAX.
- Ex. Bangalore
    - SARIMAX RSME = 25
    - RNN RSME =22
- Hyderabad
    - SARIMAX RSME = 37
    - RNN RSME = 16
- The RNN plots/predictions for 2021 are very vague and not clear, maybe due to the lack of huge data.
- If we were to say which one performed better :
    - For smaller ranges, RNN Model performs better
    - For big ranges, SARIMAX performs better

Conclusion. The lockdown has impacted the pollution levels of the environment and improved air quality in the short span owing to very less human activities. 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. So it shows that by at least trying to avoid unnecessary travel, and limiting the working of factories and Lessening the pollution - a great impact can be made on the air quality - and the lockdown has been a living proof for the same - showing us that our bit matters.

Acknowledgement

We would like to express our heartfelt gratitude to our Professor, Prof Rama Chandra Pillutla who gave us the golden opportunity to do this project on the topic Effect of Covid'19 on AQI, which also helped us to understand far reaching consequences of Covid'19.

Further, we would like to thank the Kaggle Community for their Machine Learning related doubts and queries.

References

BERA B, BHATTACHARJEE S, SHIT PK, SENGUPTA N, SAHA S. Significant impacts of COVID-19 lockdown on urban air pollution in Kolkata (India) and amelioration of environmental health (Link)

MAHATO, S., PAL, S., GHOSH, K.G., 2020. Effect of lock-down amid COVID-19 pandemic on air quality of the megacity Delhi, India. Sci. Total Environ. 730, 139086.

SINGH, R.P., CHAUHAN, A. Impact of lockdown on air quality in India during COVID-19 pandemic. Air Qual Atmos Health 13, 921–928 (2020). (Link)

ESA, 2020. ESA - air pollution drops in India following lockdown

CPCB online portal for air quality data dissemination (Link)

Air Pollution in India: Real-time Air Quality Index Visual Map (Link)

LE QUÉRÉ, C., JACKSON, R.B., JONES, M.W. *et al.* Temporary reduction in daily global CO2 emissions during the COVID-19 forced confinement (Link)

Analysis: India's CO2 emissions fall for first time in four decades amid coronavirus (Link)

National Air Quality Index, India (Link)

Kaggle(Link)

Wikipedia(Link)

Machine Learning Mastery(Link)

Medium for ML(Link)

World's Most Polluted Cities in 2019 - PM2.5 Ranking | AirVisual

Air pollution drops in India following lockdown