AIR QUALITY MONITORING

PHASE 3: DEVELOPMENT PART 1

START BUILDING THE IOT AIR QUALITY MONITORING SYSTEM.

PROJECT: AIR QUALITY MONITORING

INTRODUCTION:

Air is a basic requirement for the survival and development of all lives on Earth. It affects healthand influences the development of the economy. Today, due to the development of industrialization,the increase in the number of private cars, and the burning of fossil fuels, air quality is decreasing, withincreasingly serious air pollution. There are many pollutants in the atmosphere, such as SO_2, NO_2,CO_2, NO, CO, NO_x, PM2.5, and PM10. Internationally, a large number of scholars have conductedresearch on air pollution and air quality forecasts, concentrating on the forecasting of contaminants.Air pollution affects the life of a society, and even endangers the survival of mankind. During theIndustrial Revolution, there was a dramatic increase in coal use by factories and households, andthe smog caused significant morbidity and mortality, particularly when combined with stagnantatmospheric conditions. During the Great London Smog of 1952, heavy pollution for 5 days caused atleast 4000 deaths [1,2]. This episode highlighted the relationship between air pollution and humanhealth, yet air pollution continues to be a growing problem in cities and households around the world.

Air pollution is made up of a mixture of gases and particles in harmful amounts that are releasedinto the atmosphere due to either natural or human activities [3]. The sources of pollutants can bedivided into two categories:

1) Natural sources
2) Anthropogenic sources

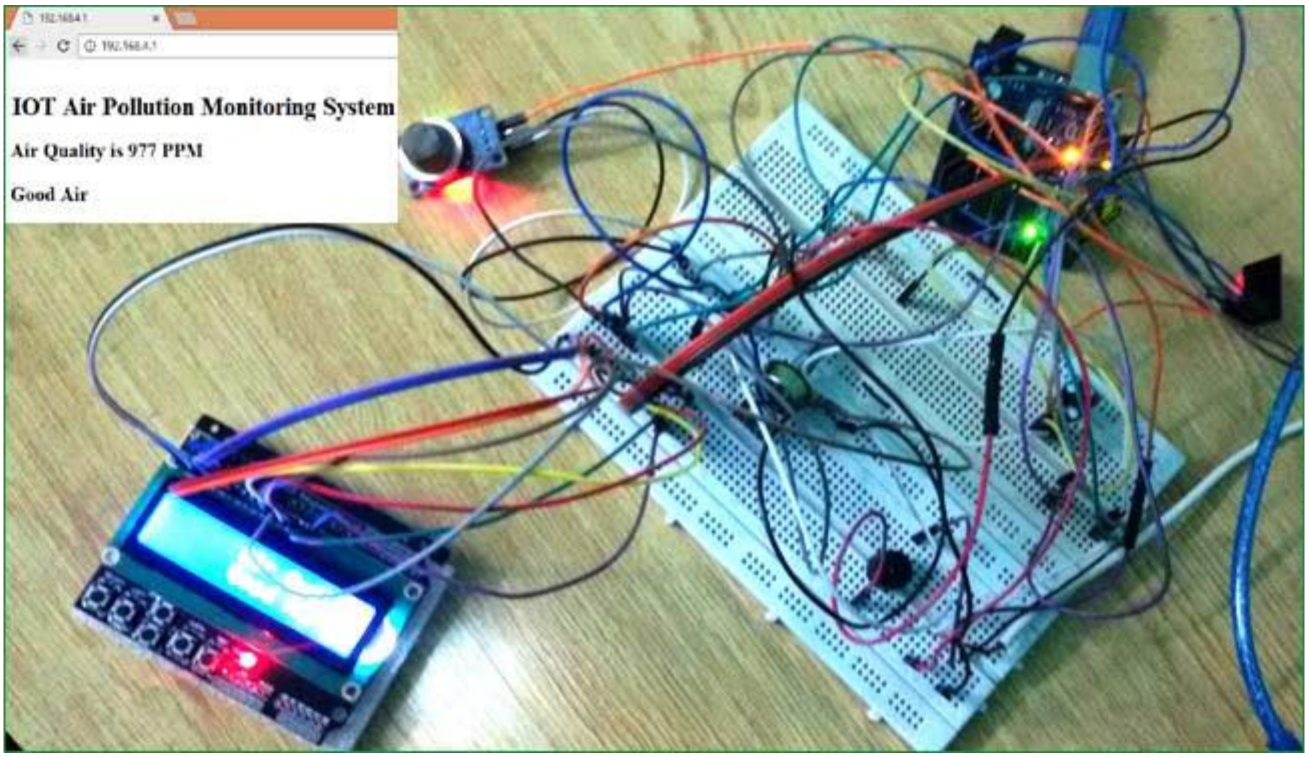
(1) Natural sources:

Natural pollution sources are natural phenomena that discharge harmful substances or haveharmful effects on the environment. Natural phenomena, such as volcanic eruptions and forest fires,will result in air pollutants, including SO2, CO2, NO2, CO, and sulfate.

(2) Anthropogenic (man-made) sources:

Man-made sources such as the burning of fuels, discharges from industrial production processes,and transportation emissions are the main sources of air pollution. There are many kinds of pollutantsemitted by man-made pollution sources, including hydrogen, oxygen,nitrogen, sulfur, metal compounds,and particulate matter.With the increasing world population and the developing world economy, the demand for energyin the world has increased dramatically. The large-scale use of fossil energy globally has also ledto a series of environmental problems that have received much attention due to their detrimentaleffects on human health and the environment [3–5]. Air pollution is a fundamental problem in manyparts of the world, with two important concerns: the impact on human health, such as cardiovasculardiseases, and the impact on the environment, such as acid rain, climate change, and global warming.

IOT based Air Pollution Monitoring System using Arduino:



In this project we are going to make an **IoT Based Air Pollution Monitoring System** in which we will **monitor the Air Quality over a webserver using internet** and will trigger a alarm when the air quality goes down beyond a certain level, means when there are sufficient amount of harmful gases are present in the air like CO2, smoke, alcohol, benzene and NH3. It will show the air quality in PPM on the LCD and as well as on webpage so that we can monitor it very easily.

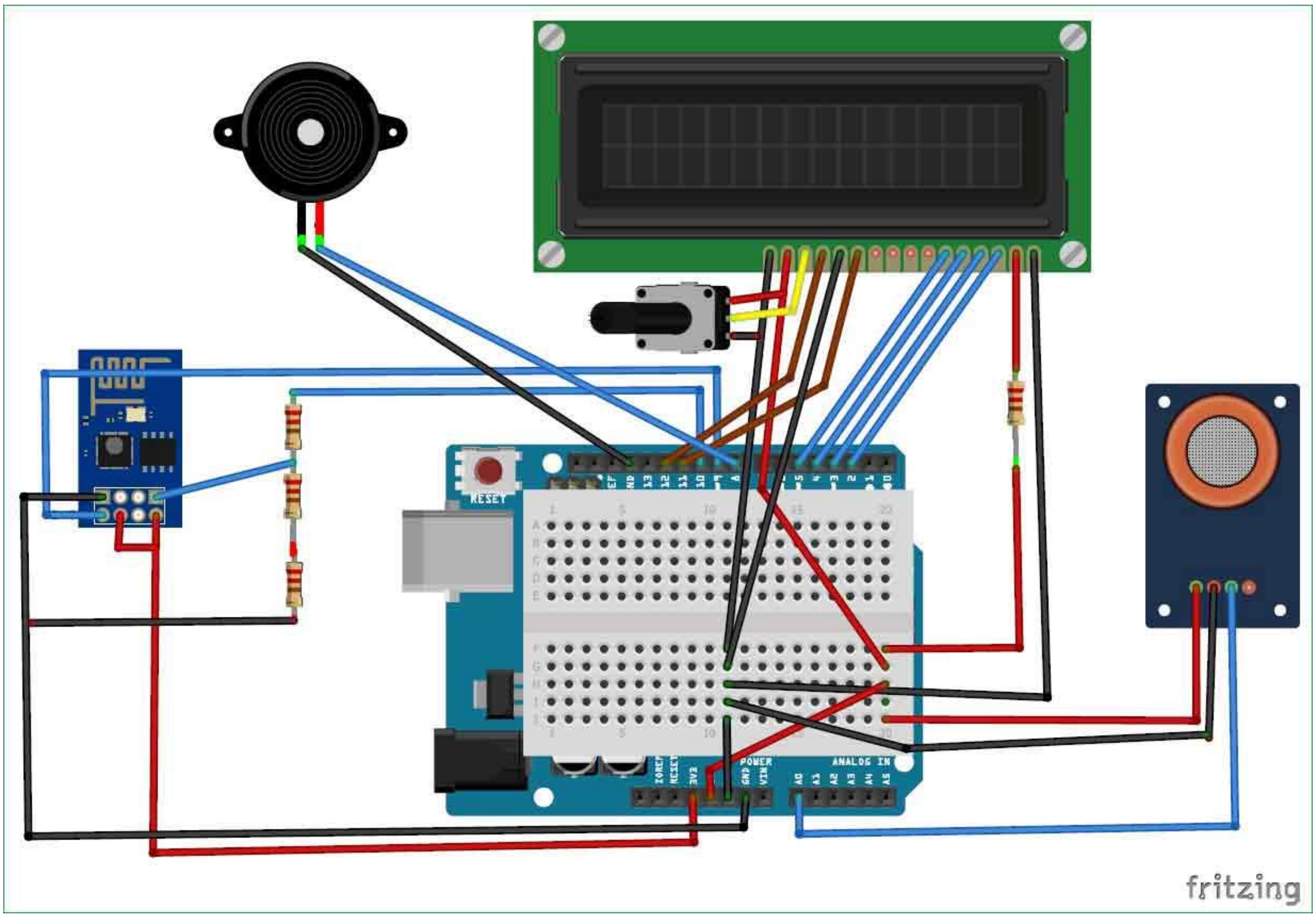
Previously we have built the LPG detector using MQ6 sensor, Smoke detector using MQ2 sensor, and Air Quality Analyser but this time we have used MQ135 sensor as the air quality sensor which is the best choice for monitoring Air Quality as it can detects most harmful gases and can measure their amount accurately. In this IOT project, you can monitor the pollution level from anywhere using your computer or mobile. We can install this system anywhere and can also trigger some device when pollution goes beyond some level, like we can switch on the Exhaust fan or can send alert SMS/mail to the user.

Required Components:

- MQ135 Gas sensor
- Arduino Uno
- Wi-Fi module ESP8266
- 16X2 LCD
- Breadboard
- 10K potentiometer
- 1K ohm resistors
- 220 ohm resistor
- Buzzer

You can buy all the above components from here.

CIRCUIT DIAGRAM:



Circuit Diagram and Explanation:

First of all we will connect the **ESP8266 with the Arduino**. ESP8266 runs on 3.3V and if you will give it 5V from the Arduino then it won't work properly and it may get damage. Connect the VCC and the CH_PD to the 3.3V pin of Arduino. The RX pin of ESP8266 works on 3.3V and it will not communicate with the Arduino when we will connect it directly to the Arduino. So, we will have to make a voltage divider for it which will convert the 5V into 3.3V. This can be done by connecting three resistors in series like we did in the circuit. Connect the TX pin of the ESP8266 to the pin 10 of the Arduino and the RX pin of the esp8266 to the pin 9 of Arduino through the resistors.

ESP8266 Wi-Fi module gives your projects **access to Wi-Fi or internet**. It is a very cheap device and make your projects very powerful. It can communicate with any microcontroller and it is the most leading devices in the IOT platform. Learn more about using ESP8266 with Arduino here.

Then we will connect the **MQ135 sensor with the Arduino**. Connect the VCC and the ground pin of the sensor to the 5V and ground of the Arduino and the Analog pin of sensor to the A0 of the Arduino.

Connect a buzzer to the pin 8 of the Arduino which will start to beep when the condition becomes true.

In last, we will connect LCD with the Arduino. The connections of the LCD are as follows

- Connect pin 1 (VEE) to the ground.

- Connect pin 2 (VDD or VCC) to the 5V.
- Connect pin 3 (V0) to the middle pin of the 10K potentiometer and connect the other two ends of the potentiometer to the VCC and the GND. The potentiometer is used to control the screen contrast of the LCD. Potentiometer of values other than 10K will work too.
- Connect pin 4 (RS) to the pin 12 of the Arduino.
- Connect pin 5 (Read/Write) to the ground of Arduino. This pin is not often used so we will connect it to the ground.
- Connect pin 6 (E) to the pin 11 of the Arduino. The RS and E pin are the control pins which are used to send data and characters.
- The following four pins are data pins which are used to communicate with the Arduino.

Connect pin 11 (D4) to pin 5 of Arduino.

Connect pin 12 (D5) to pin 4 of Arduino.

Connect pin 13 (D6) to pin 3 of Arduino.

Connect pin 14 (D7) to pin 2 of Arduino.

- Connect pin 15 to the VCC through the 220 ohm resistor. The resistor will be used to set the back light brightness. Larger values will make the back light much more darker.
- Connect pin 16 to the Ground.

Working Explanation:

The MQ135 sensor can sense NH3, NOx, alcohol, Benzene, smoke, CO2 and some other gases, so it is perfect gas sensor for our **Air Quality Monitoring Project**. When we will connect it to Arduino then it will sense the gases, and we will get the Pollution level in PPM (parts per million). MQ135 gas sensor gives the output in form of voltage levels and we need to convert it into PPM. So for converting the output in PPM, here we have used a library for MQ135 sensor, it is explained in detail in “Code Explanation” section below.

Sensor was giving us value of 90 when there was no gas near it and the safe level of air quality is 350 PPM and it should not exceed 1000 PPM. When it exceeds the limit of 1000 PPM, then it starts cause Headaches, sleepiness and stagnant, stale, stuffy air and if exceeds beyond 2000 PPM then it can cause increased heart rate and many other diseases.

When the value will be less than 1000 PPM, then the LCD and webpage will display “Fresh Air”. Whenever the value will increase 1000 PPM, then the buzzer will start beeping and the LCD and webpage will display “Poor Air, Open Windows”. If it will increase 2000 then the buzzer will keep beeping and the LCD and webpage will display “Danger! Move to fresh Air”.

Air Pollution Assessment:

In recent years, air pollution accidents have occurred frequently, which have damaged the
economy and human life. To assess the extent of the damage, air pollution control must be evaluatedin order to have a quantitative understanding of pollution.Int. J. Environ. Res. Public Health **2018**, 15, 780 6 of 44The assessment of air pollution is identify and measure the degree

and scope of damage causedby environmental pollution cover the economic, legal, technical and other means reasonably [35–37].

Two of the more mature assessment methods will be described. The market value methodis a type of cost benefit analysis method. It uses the change of product yield and profit causedby the environmental quality change to measure the economic loss related to the environmentalquality change.Environmental pollution and damage caused by air pollution can be prevented, restored, orreplaced by the existing environmental functions. Therefore, the cost of preventing, restoring, orreplacing the original functional protection facilities can be used to estimate the loss caused bypollution or damage to the environment. This method is called the engineering cost method.The main equation and the meaning of the variables in those methods are given in Table 1, andthe flowchart of the assessment methods is given in Figure 2.

Two of the more mature assessment methods will be described. The market value method is a typeof cost benefit analysis method. It uses the change of product yield and profit caused by theenvironmental quality change to measure the economic loss related to the environmental qualitychange.

Environmental pollution and damage caused by air pollution can be prevented, restored, orreplaced by the existing environmental functions. Therefore, the cost of preventing, restoring, orreplacing the original functional protection facilities can be used to estimate the loss caused bypollution or damage to the environment. This method is called the engineering cost method.

CODING:

```
webpage += "</h2></p></body>";

    String cipSend = "AT+CIPSEND=";

    cipSend += connectionId;

    cipSend += ",";

    cipSend +=webpage.length();

    cipSend +="\r\n";



    sendData(cipSend,1000,DEBUG);

    sendData(webpage,1000,DEBUG);



    cipSend = "AT+CIPSEND=";

    cipSend += connectionId;

    cipSend += ",";

    cipSend +=webpage.length();
```

```
     cipSend +="\r\n";


     String closeCommand = "AT+CIPCLOSE=";

     closeCommand+=connectionId; // append connection id

     closeCommand+="\r\n";


     sendData(closeCommand,3000,DEBUG);

    }

  }

lcd.setCursor (0, 0);

lcd.print ("Air Quality is ");

lcd.print (air_quality);

lcd.print (" PPM ");

lcd.setCursor (0,1);

if (air_quality<=1000)

{

lcd.print("Fresh Air");

digitalWrite(8, LOW);

}

else if( air_quality>=1000 && air_quality<=2000 )

{

lcd.print("Poor Air, Open Windows");

digitalWrite(8, HIGH );

}

else if (air_quality>=2000 )

{

lcd.print("Danger! Move to Fresh Air");
```

```
digitalWrite(8, HIGH);   // turn the LED on
}
lcd.scrollDisplayLeft();
delay(1000);
}
String sendData(String command, const int timeout, boolean debug)
{
    String response = "";
    esp8266.print(command); // send the read character to the esp8266
    long int time = millis();
    while( (time+timeout) > millis())
    {
      while(esp8266.available())
      {
        // The esp has data so display its output to the serial window
        char c = esp8266.read(); // read the next character.
        response+=c;
      }
    }
    if(debug)
    {
      Serial.print(response);
    }
    return response;
}
```

DATASET:

The dataset contains air quality data and AQI (Air Quality Index) at hourly and daily level of various stations across multiple cities in India. This project deals with exploration of the data and modelling to predict the classes namely : Good, Satisfactory, moderate, poor, very poor or severe cases of the air quality.

Import and load the data:

Import necessary packages: Here I have imported packages needed for preprocessing and modelling

```
import pandas as pd
import os
import zipfile
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objs as go
import numpy as np
from plotly.offline import init_notebook_mode,iplot
import missingno as msno
from sklearn.impute import KNNImputer
from sklearn import preprocessing
import pylab
import scipy.stats as stats
from scipy.special import boxcox1p
import pylab
import scipy.stats as stats
%matplotlib inline
# Transformation and modelling packages
from sklearn.model_selection import train_test_split , KFold
from sklearn.metrics import accuracy_score
from sklearn. preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier
from collections import Counter
from sklearn.model_selection import GridSearchCV ,RandomizedSearchCV
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from sklearn.svm import SVC
```

The dataset upon loading and observation, it's important and easy to process the data with smaller size.

- The data types are changed while loading.
- String data is converted into categorical values, float64 to float32 etc. * *
- This process place vital role while handling dataset with larger size.

```
'City':'category',
   'Datetime' : 'category',
   'PM2.5' : 'float32',
   'PM10' : 'float32',
   'NO' : dtypes1 = {
   'float32',
   'NO2' : 'float32',
   'NOx' : 'float32',
   'NH3' : 'float32',
   'CO' : 'float32',
   'SO2' : 'float32',
   'O3' : 'float32',
   'Benzene' : 'float32',
   'Toluene' : 'float32',
   'Xylene' : 'float32',
   'AQI' : 'float32',
   'AQI_Bucket': 'category'
}
```

Loop through the zip files and extract files.

```
path = '/content/drive/MyDrive/Projects/air-quality-data-in-india.zip'

with zipfile.ZipFile(path, 'r') as f:
  for name in f.namelist():
    if name=='city_hour.csv':
      df_c = pd.read_csv(f.open(name), dtype=dtypes1)
```

Save the files for in csv format for direct access. Load the csv files using pandas 'read_csv'

```
# Save the dtyp changed csv files
fp1 = '/content/drive/MyDrive/Projects/city_hour.csv'
df_c.to_csv(fp1, index=False)
df_c = pd.read_csv('/content/drive/MyDrive/Projects/city_hour.csv',dtype = dtypes1)
```

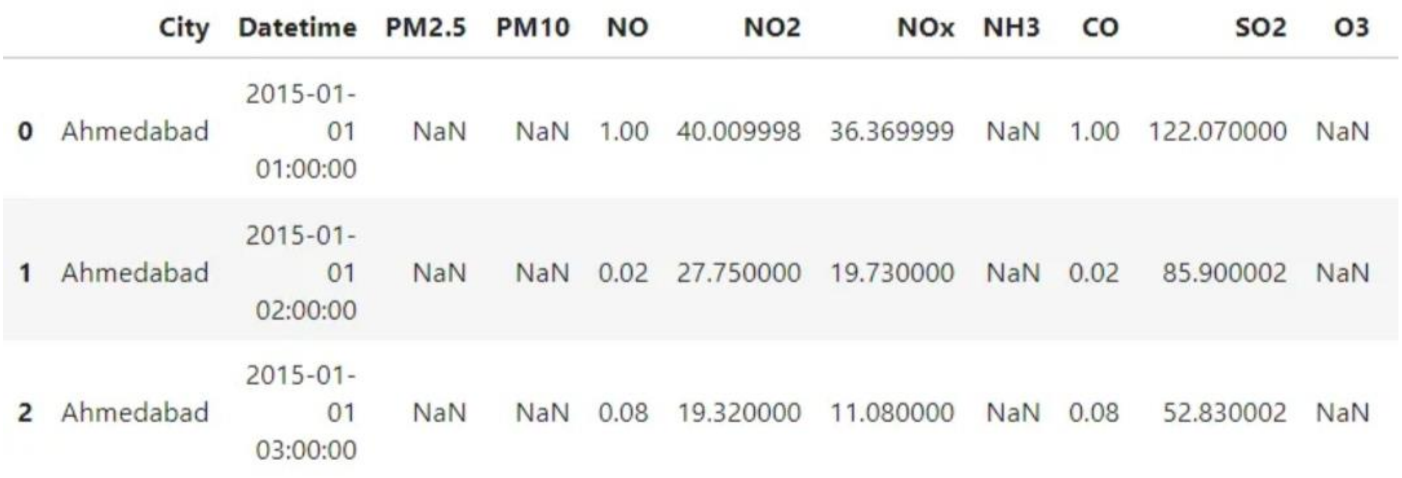
The function 'basic_exploration' explorates the routine exploration done on any tabular data. Functionizing makes the task easier and saves a lot of time.

```
def space():
  print(" ")
  print("---------------------------------------------------------------")
  print(" ")

def basic_exploration(steps):
  for i in steps:
    print(i)
    space()

steps = [df.shape, df.duplicated().any(), df.isnull().sum()]
basic_exploration(steps)

(2589083, 16)

---------------------------------------------------------------

False

---------------------------------------------------------------

StationId        0
Datetime         0
PM2.5       647689
PM10       1119252
NO          553711
NO2         528973
NOx         490808
NH3        1236618
CO          499302
SO2         742737
O3          725973
Benzene     861579
Toluene    1042366
Xylene     2075104
AQI         570190
AQI_Bucket  570190
dtype: int64
```

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

The dataset has numerous missing values.

```
df_c.head(3)
```

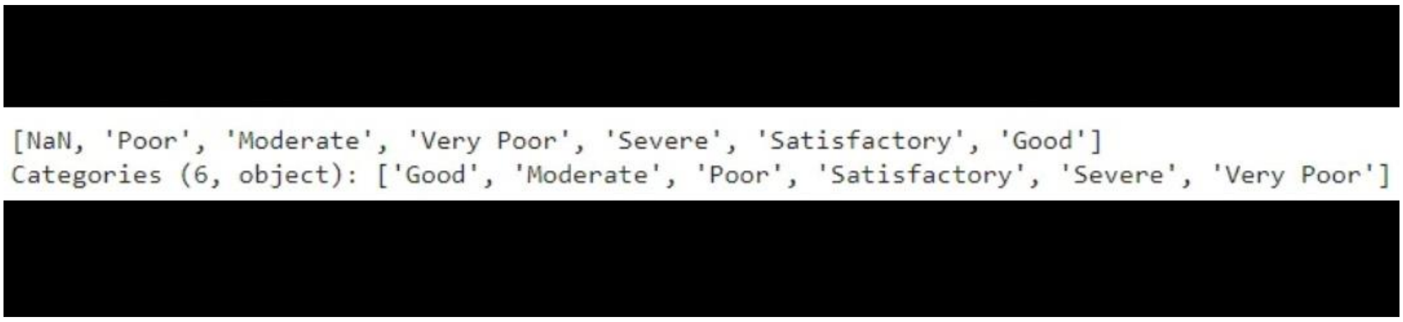


	City	Datetime	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3
0	Ahmedabad	2015-01-01 01:00:00	NaN	NaN	1.00	40.009998	36.369999	NaN	1.00	122.070000	NaN
1	Ahmedabad	2015-01-01 02:00:00	NaN	NaN	0.02	27.750000	19.730000	NaN	0.02	85.900002	NaN
2	Ahmedabad	2015-01-01 03:00:00	NaN	NaN	0.08	19.320000	11.080000	NaN	0.08	52.830002	NaN

Data Preprocessing

Convert to datetime

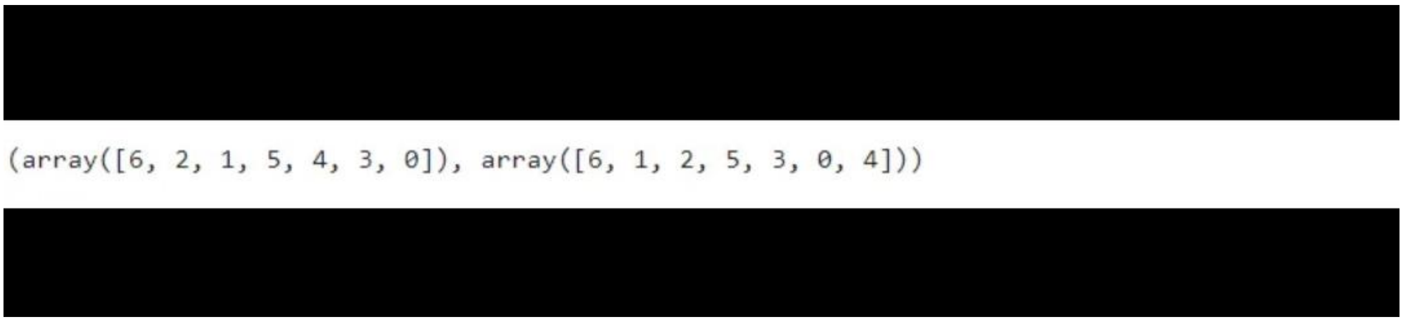
```
df_c['Datetime'] = pd.to_datetime(df_c['Datetime'])
df_c['AQI_Bucket'].unique()
```



```
[NaN, 'Poor', 'Moderate', 'Very Poor', 'Severe', 'Satisfactory', 'Good']
Categories (6, object): ['Good', 'Moderate', 'Poor', 'Satisfactory', 'Severe', 'Very Poor']
```

Label encode the ‘AQI_Bucket’ which is the target column for our classification model. Label Encoder assign integers to the classes.

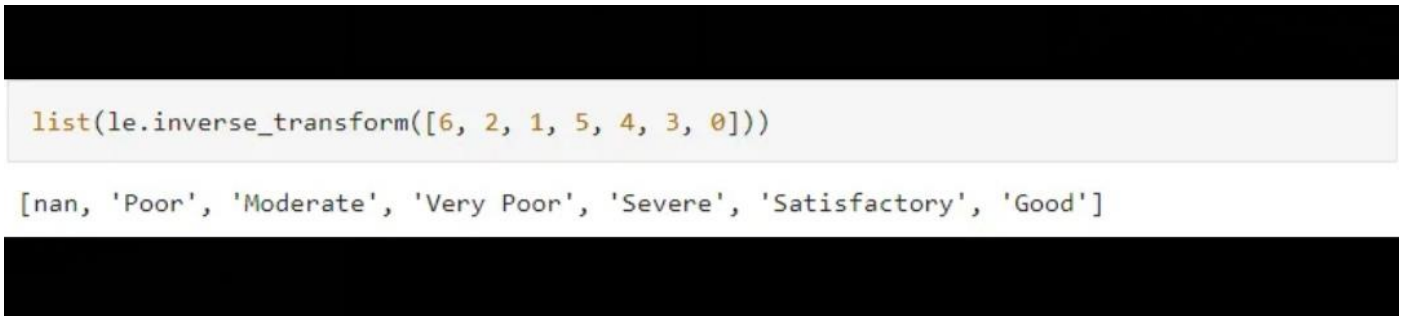
```
le = preprocessing.LabelEncoder()
df_c['AQI_Bucket'] = le.fit_transform(df_c['AQI_Bucket'])
```


[redacted]

```
(array([6, 2, 1, 5, 4, 3, 0]), array([6, 1, 2, 5, 3, 0, 4]))
```

[redacted]

Here we apply inverse transform function to the label encoder inorder to store t he mappings done by the encoder.


[redacted]

```
list(le.inverse_transform([6, 2, 1, 5, 4, 3, 0]))
```

```
[nan, 'Poor', 'Moderate', 'Very Poor', 'Severe', 'Satisfactory', 'Good']
```

[redacted]

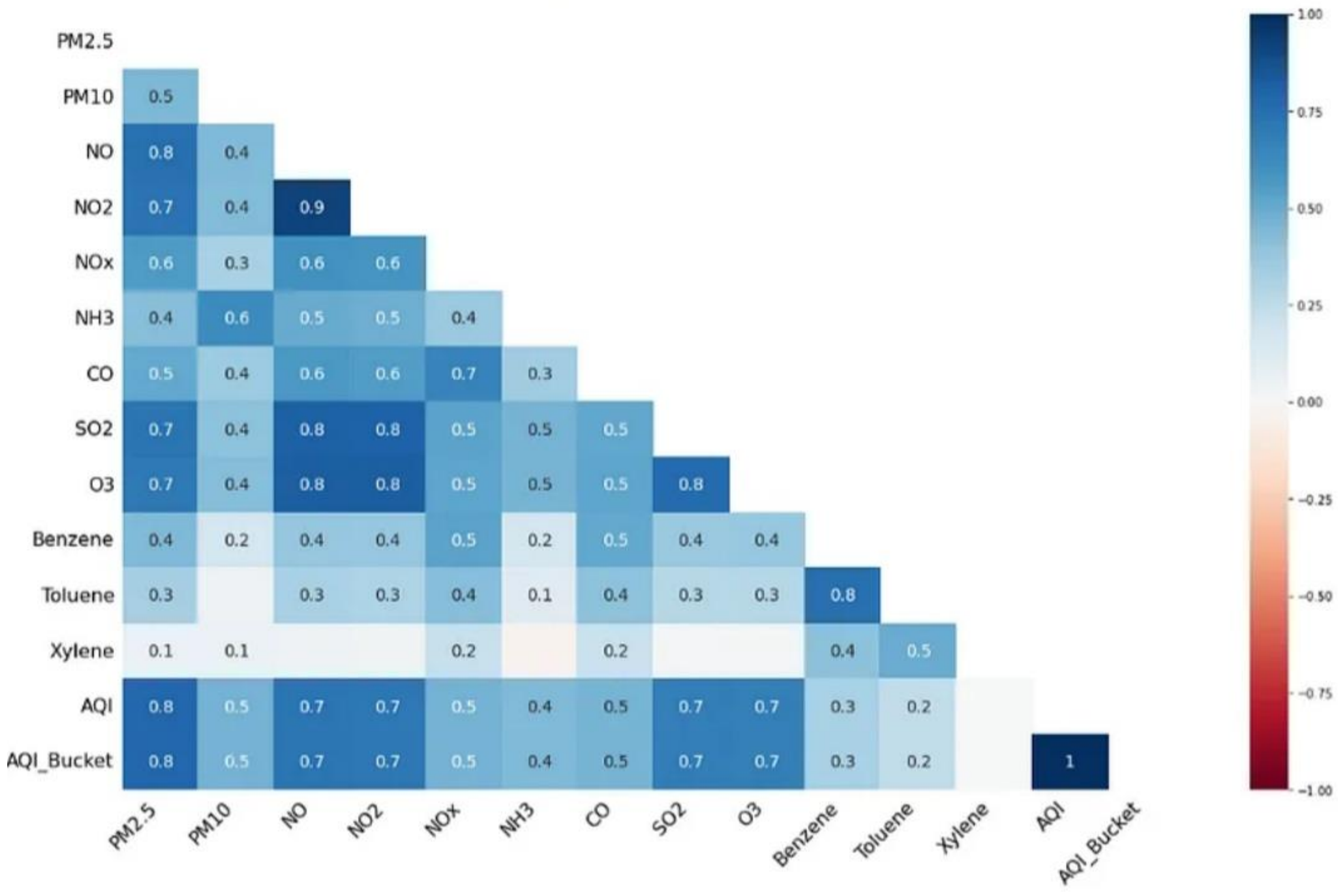
```
mappings = {
    'nan': 6,
    'Poor': 2,
    'Moderate': 1,
    'Very Poor': 5,
    'Severe': 4,
    'Satisfactory': 3,
    'Good': 0
}
```

Pattern of Missing Values

‘missingno’ is the python library which has functions for analysing the pattern of the missing values. According to the missing value pattern suitable imputation can be decided

```
msno.heatmap(df_c)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1e6419ced0>
```



- From the above observation of distribution of missing values there is significant common pattern between the data columns. We observe that NO and NO2 are strongly correlated. As this is reading of air polluting gases, it can also be the case where industrial pollutants emit certain type of gases and vehicle pollutants have different set of gases. Hence there is correlation with missing values.

Imputation:

- We apply forward imputation to the given dataset. **Linear interpolation** is an imputation technique that assumes a linear relationship between data points and utilizes non-missing values from adjacent data points to compute a value for a missing data point.

Detailed guide for choice of imputation techniques can be found here here.

- Choose the columns to interpolate the missing data.
- Apply bfill to initial values with 'Nan' as there isn't any value prior to them to interpolate.

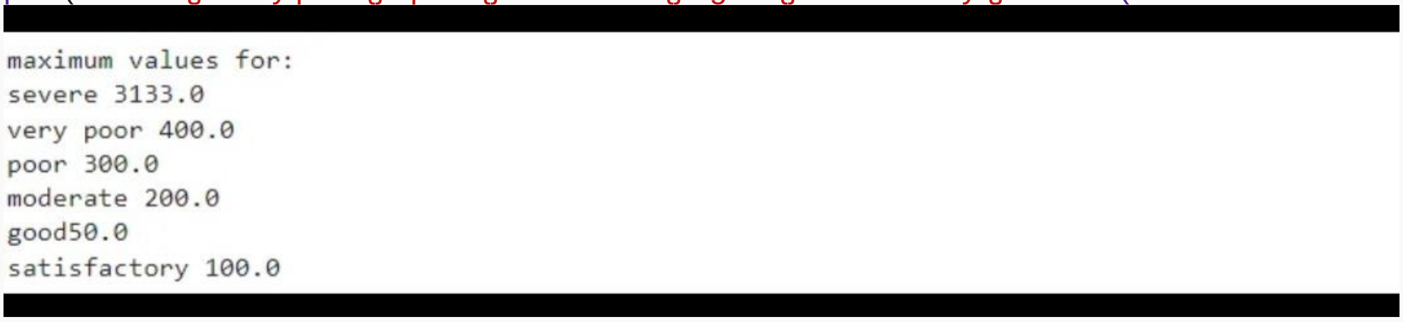
```
cols = ['PM2.5','PM10','NO','NO2', 'NOx','NH3','CO', 'SO2', 'O3', 'Benzene', 'Toluene', 'Xylene', 'AQI',
'AQI_Bucket']

df_c_imp = df_c[cols]
df_c_imp.interpolate(limit_direction='forward', inplace=True);

df_c_1 = df_c[['City', 'Datetime']]
df_c_2 = pd.concat([df_c_1, df_c_imp], axis=1, join='inner')
df_c_2['AQI'].fillna(method='bfill', inplace=True)
```

Extract the range of AQI_Bucket labels, ie, based on AQI (Air Quality Index) corresponding classes are assigned in AQI_bucket. Hence knowing the range of AQI is important to assign the corresponding classes to 'Nan' values in target label(AQI_Bucket). Here we:

- Write conditions
- Extract the upper limit of the range
- Use the upper limit to add the mappings

```
modc = df_c['AQI_Bucket'] == 1
satc = df_c['AQI_Bucket'] == 3
vpc = df_c['AQI_Bucket'] == 5
prc = df_c['AQI_Bucket'] == 2
gdc = df_c['AQI_Bucket'] == 0
src = df_c['AQI_Bucket'] == 4

severec = np.max(df_c[src]['AQI'])
```

```
very_poorc = np.max(df_c[vpc]['AQI'])
satisfactoryc = np.max(df_c[satc]['AQI'])
poorc = np.max(df_c[prc]['AQI'])
moderatec = np.max(df_c[modc]['AQI'])
goodc = np.max(df_c[gdc]['AQI'])

print('maximum values for:')
print("severe {}\nvery poor {}\npoor {}\nmoderate {}\ngood{}\nsatisfactory {} ".format(seve
```

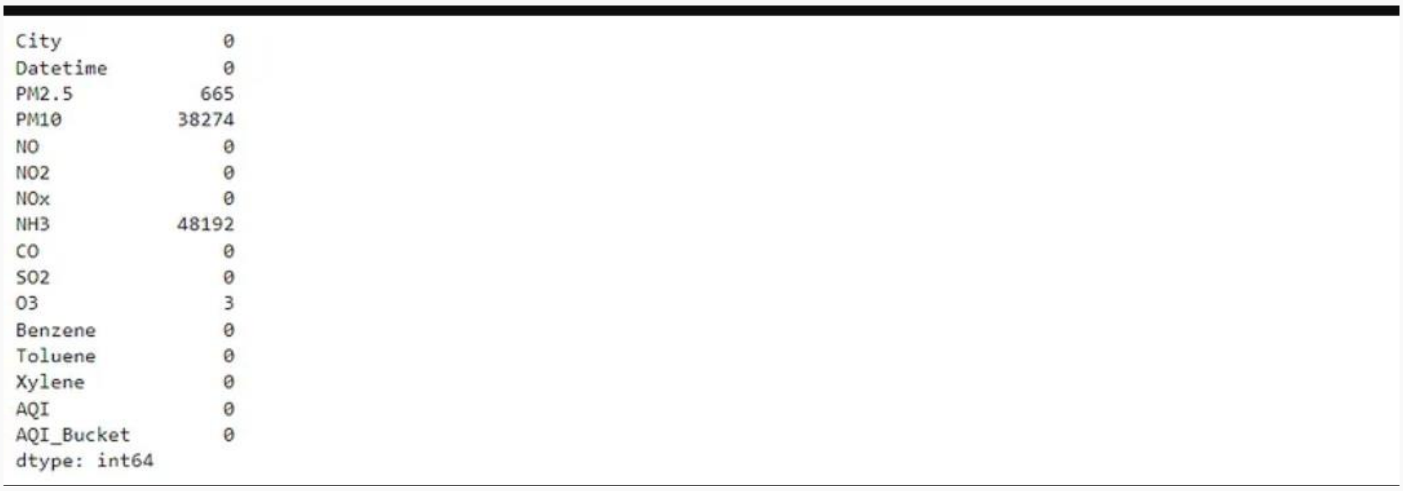


```
maximum values for:
severe 3133.0
very poor 400.0
poor 300.0
moderate 200.0
good50.0
satisfactory 100.0
```

```
(sr_cond = df_c_2['AQI'] > 400
vp_cond = df_c_2['AQI'] <=400
pr_cond = df_c_2['AQI'] <= 300
md_cond = df_c_2['AQI'] <= 200
st_cond = df_c_2['AQI'] <= 100
gd_cond = df_c_2['AQI'] <= 50

df_c_2.loc[sr_cond, 'AQI_Bucket'] = 4
df_c_2.loc[vp_cond, 'AQI_Bucket'] = 5
df_c_2.loc[pr_cond, 'AQI_Bucket'] = 2
df_c_2.loc[md_cond, 'AQI_Bucket'] = 1
df_c_2.loc[st_cond, 'AQI_Bucket'] = 3
df_c_2.loc[gd_cond, 'AQI_Bucket'] = 0
```

Check for Null and remove null

```
df_c_2.isnull().sum()
```



```
City              0
Datetime          0
PM2.5           665
PM10          38274
NO                0
NO2               0
NOx               0
NH3           48192
CO                0
SO2               0
O3                3
Benzene           0
Toluene           0
Xylene            0
AQI               0
AQI_Bucket        0
dtype: int64
```

```
df_c_2['PM2.5'] = df_c_2['PM2.5'].fillna(0)
df_c_2['O3'] = df_c_2['O3'].fillna(0)
df_c_2['NH3'] = df_c_2['NH3'].fillna(0)
df_c_2['PM10'] = df_c_2['PM10'].fillna(0)
df_c_2.isnull().sum()
```

Save the preprocessed data:

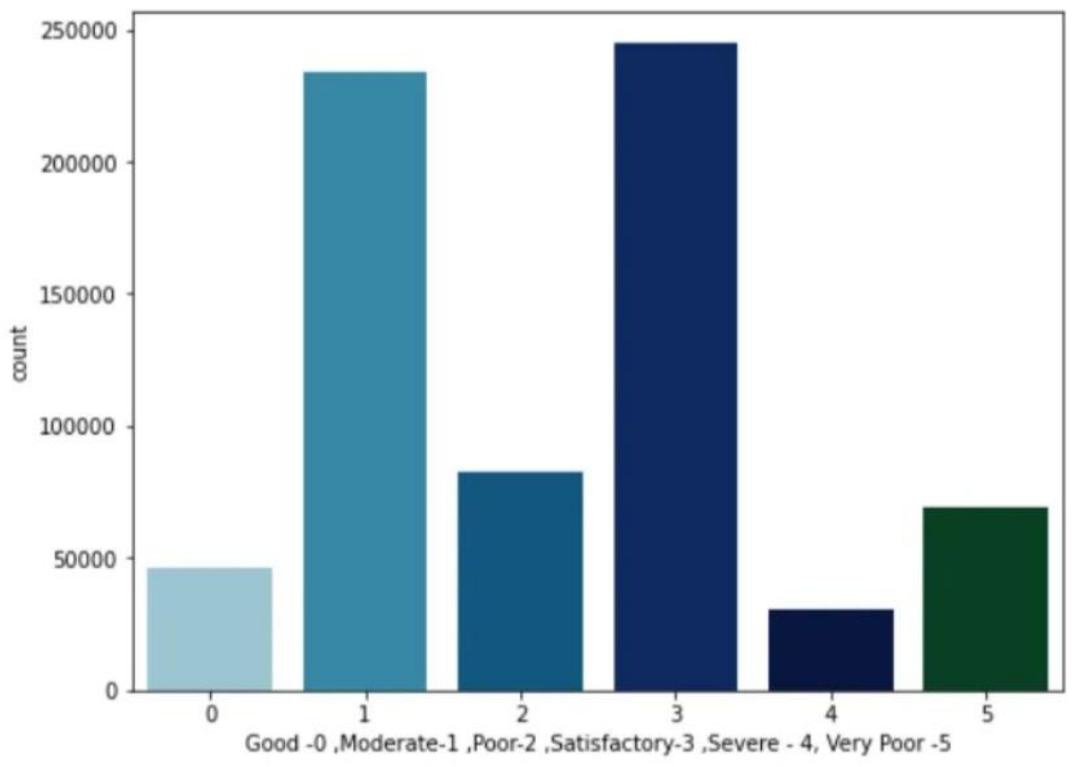
```
c = '/content/drive/MyDrive/Projects/imp_city_hour.csv'
df_c_2.to_csv(fp_c, index fp_=False)
```

Exploratory Data Analysis

Distribution of classes in the Data

Countplot is a univariate plot ie. single variable under consideration, showing the number of data points representing the categorical variables.

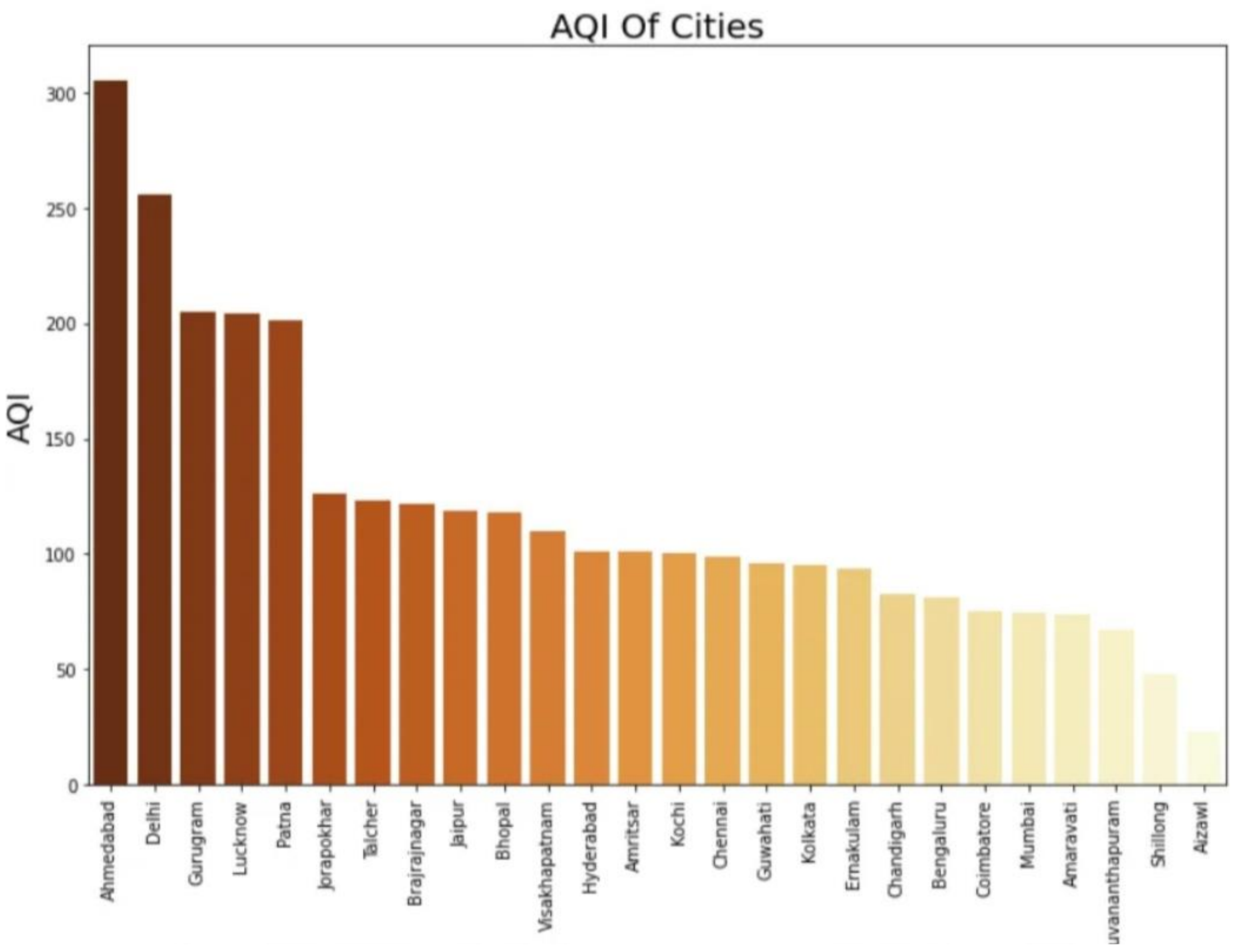
```
def countplot(x_val):
  plt.figure(figsize = (8, 6))
  sns.countplot(x = x_val, palette = 'ocean_r');
  plt.xlabel("Good -0 ,Moderate-1 ,Poor-2 ,Satisfactory-3 ,Severe - 4, Very Poor -5")
countplot(df_ch['AQI_Bucket'])
```



- Most cities fall nder moderate and satisfactory range.
- There are few representation for good air quality and very poor air quality.

Cities vs AQI Indexes

```
"""Select city, AQI from df_ch ORDER BY AQI DESC"""
# do average of every city aqi value,
city_vs_AQI = df_ch[['City', 'AQI']].groupby(['City']).median().sort_values('AQI',
ascending=False).reset_index()

def barplot(a,b, title, x_label, y_label, palet):
  plt.figure(figsize=(12,8))
  sns.barplot(x = a, y = b, palette =palet, order=a)
  plt.xticks(rotation = 90)
  plt.title(title, fontsize = 20)
  plt.xlabel(x_label, fontsize = 18)
  plt.ylabel(y_label, fontsize = 18)

barplot(city_vs_AQI['City'], city_vs_AQI['AQI'], "AQI Of Cities","Cities", "AQI","YlOrBr_r”)
```

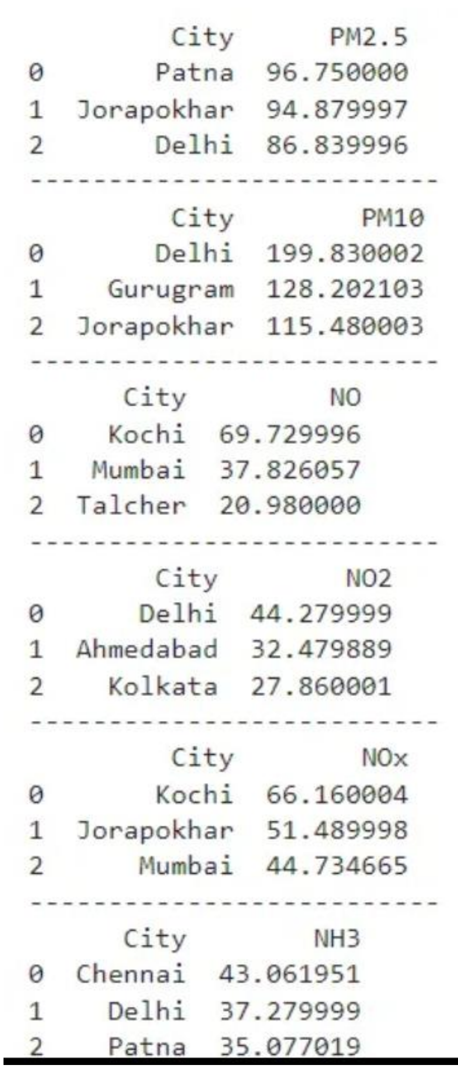


Ahmedabad has AQI in the range of 300 followed by Delhi. 300 indicates poor quality.

Gases and top 3 cities with maximum of that particular gas

Different cities are extracted which has maximum amount of gas type. Different gases have different cities in top 3.

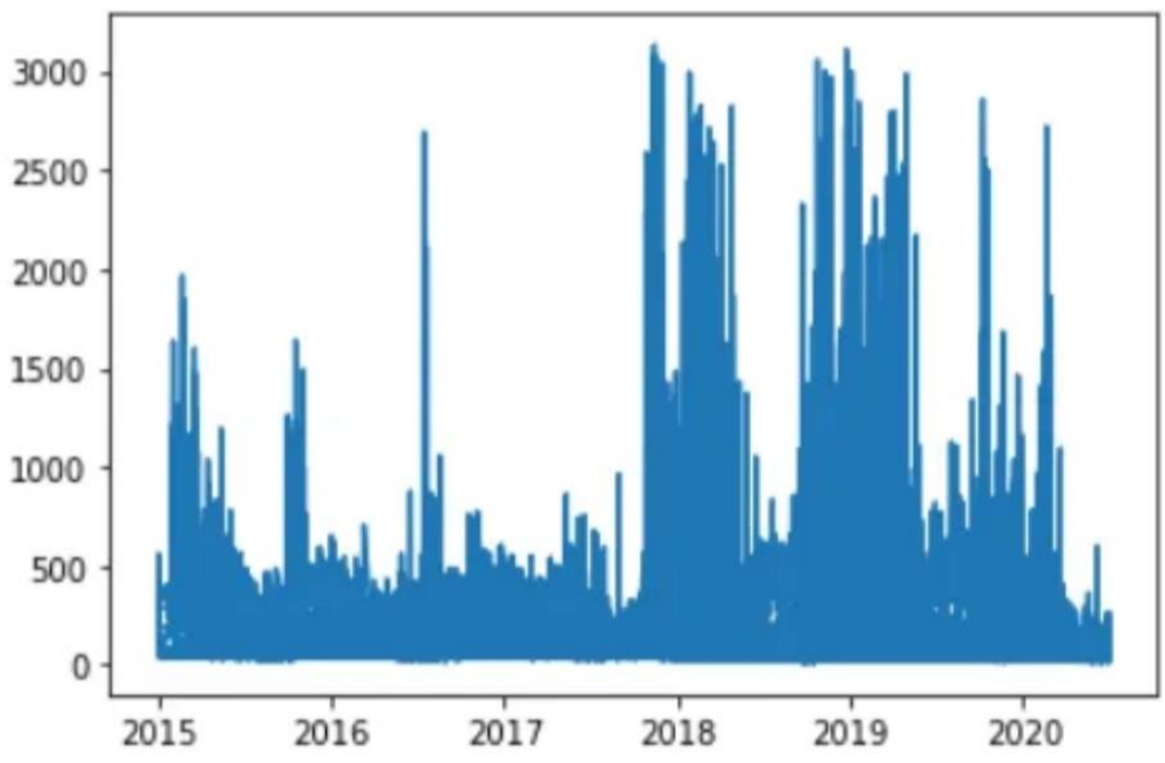
```
cols = ['PM2.5','PM10','NO','NO2', 'NOx','NH3','CO', 'SO2', 'O3', 'Benzene', 'Toluene', 'Xylene', 'AQI',
'AQI_Bucket']

# Analyze the states with highest particular gases
dff = []
for i in cols:
  data = df_ch[[i, 'City']].groupby(['City']).median().sort_values(i, ascending =
False).iloc[0:3].reset_index()
  dff.append(data)

for i in range(len(cols)):
  print(dff[i])
  print("-------------------------")
```



```
          City       PM2.5
0        Patna   96.750000
1   Jorapokhar   94.879997
2        Delhi   86.839996
-------------------------
          City        PM10
0        Delhi  199.830002
1     Gurugram  128.202103
2   Jorapokhar  115.480003
-------------------------
      City         NO
0    Kochi  69.729996
1   Mumbai  37.826057
2  Talcher  20.980000
-------------------------
        City        NO2
0      Delhi  44.279999
1  Ahmedabad  32.479889
2    Kolkata  27.860001
-------------------------
          City        NOx
0        Kochi  66.160004
1   Jorapokhar  51.489998
2       Mumbai  44.734665
-------------------------
      City        NH3
0  Chennai  43.061951
1    Delhi  37.279999
2    Patna  35.077019
```

Datetime Vs AQI

```
plt.plot(df_ch['Datetime'], df_ch['AQI'])
```



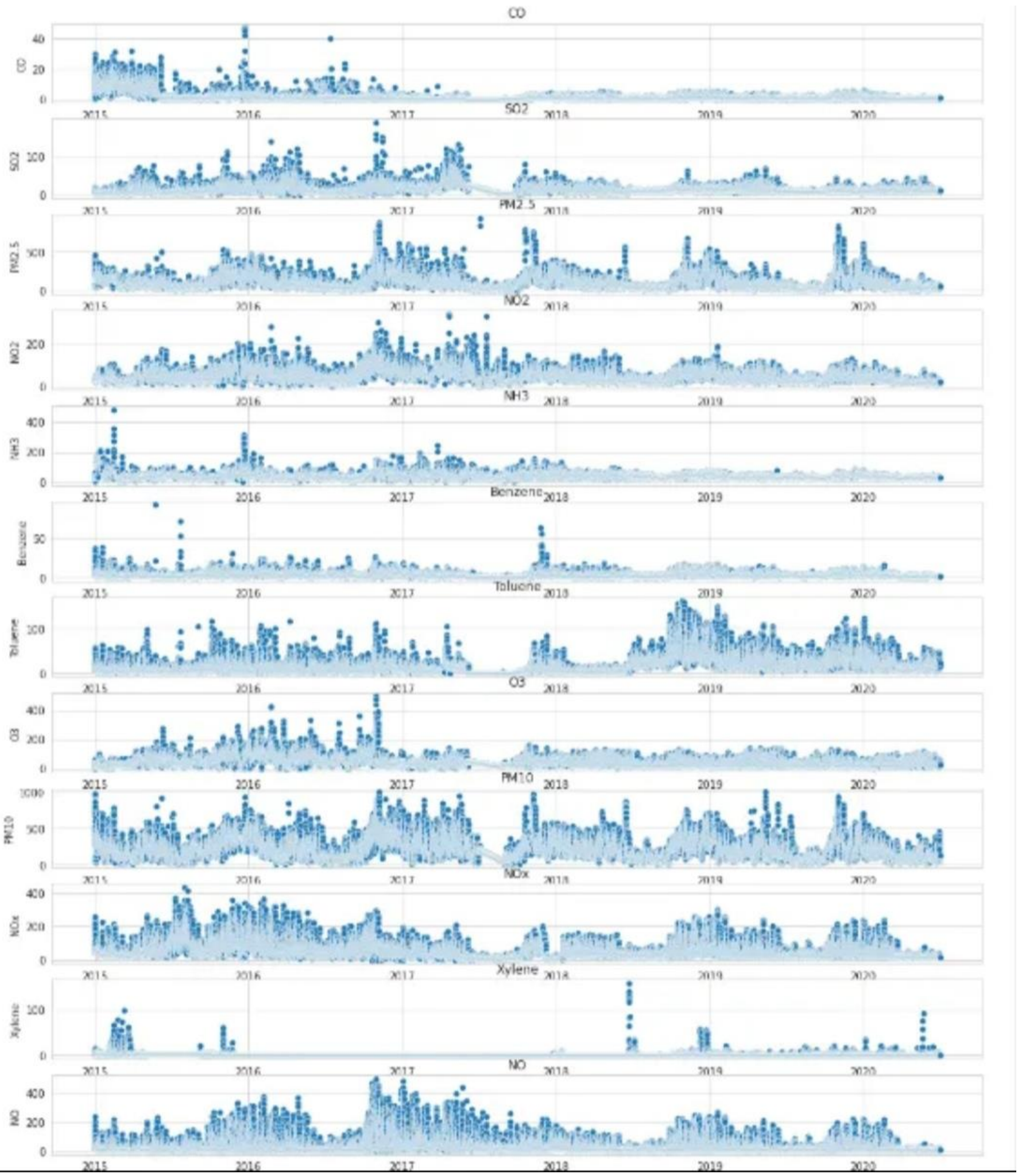
We observe that AQI is rising to severe in year 2018–2019 and slightly reduced in 2020 due to lockdown.

Analyzing Pollution in Delhi

Select data values corresponding to city= Delhi. Scatterplot is used for representation.

```
dli = df_ch[df_ch.City == 'Delhi'].reset_index()

sns.set_style("whitegrid")

fig, axes = plt.subplots(nrows = 12,
                    ncols = 1,
                    figsize = (16,20))
axes[0].set_title('CO')
sns.scatterplot(x=dli['Datetime'], y =dli['CO'],data = df_ch,ax = axes[0], alpha = 1 )
axes[1].set_title('SO2')
sns.scatterplot(x=dli['Datetime'], y =dli['SO2'],data = df_ch, ax = axes[1], alpha = 1)

axes[2].set_title('PM2.5')
sns.scatterplot(x=dli['Datetime'], y =dli['PM2.5'],data = df_ch,ax = axes[2], alpha = 1 )
axes[3].set_title('NO2')
sns.scatterplot(x=dli['Datetime'], y =dli['NO2'],data = df_ch, ax = axes[3], alpha = 1)

axes[4].set_title('NH3')
```

```
    sns.scatterplot(x=dli['Datetime'], y =dli['NH3'],data = df_ch,ax = axes[4], alpha = 1 )
    axes[5].set_title('Benzene')
    sns.scatterplot(x=dli['Datetime'], y =dli['Benzene'],data = df_ch, ax = axes[5], alpha = 1)

    axes[6].set_title('Toluene')
    sns.scatterplot(x=dli['Datetime'], y =dli['Toluene'],data = df_ch,ax = axes[6], alpha = 1 )
    axes[7].set_title('O3')
    sns.scatterplot(x=dli['Datetime'], y =dli['O3'],data = df_ch, ax = axes[7], alpha = 1)

    axes[8].set_title('PM10')
    sns.scatterplot(x=dli['Datetime'], y =dli['PM10'],data = df_ch,ax = axes[8], alpha = 1 )
    axes[9].set_title('NOx')
    sns.scatterplot(x=dli['Datetime'], y =dli['NOx'],data = df_ch, ax = axes[9], alpha = 1)

    axes[10].set_title('Xylene')
    sns.scatterplot(x=dli['Datetime'], y =dli['Xylene'],data = df_ch,ax = axes[10], alpha = 1 )
    axes[11].set_title('NO')
    sns.scatterplot(x=dli['Datetime'], y =dli['NO'],data = df_ch, ax = axes[11], alpha = 1)
```



We observe different pattern of density of gases from 2015 to 2020.

- There is rise of toluene
- Particulate matters, NOx NO2, have similar distributions.

Do explore further! In the next part we are upto statistical analysis.

CUNCLUTION AND FUTURE WORK (PHASE 3):

Project conclution:

In the phase 3 coclution, we wil summarize the key finding and insides from the advanced regression techniques. We will reiterate the impact of these techniques on improving the Air quality monitoring.

THE END...