Phase 3:AIR QUALITY ANALYSIS AND PREDICTION

Using datascience for loading and preprocessing of datasets

Air Quality Data-Loading and Preprocessing:

- About 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. The dataset is obtained from Kaggle here. The data has been made publicly available by the Central Pollution Control Board: https://cpcb.nic.in/ which is the official portal of Government of India.

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_m
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_
from sklearn.metrics import accuracy_score
from sklearn. preprocessing import Standar
from sklearn.ensemble import RandomForestC
from sklearn.linear_model import LogisticR
from sklearn.naive_bayes import GaussianNB
```

```
from sklearn. preprocessing import Standa
from sklearn.ensemble import RandomForest
from sklearn.linear_model import Logistic
from sklearn.naive_bayes import GaussianN
from sklearn.naive_bayes import Bernoulli
from sklearn.tree import DecisionTreeClas
from sklearn.neighbors import KNeighborsC
from xgboost import XGBClassifier
from collections import Counter
from sklearn.model_selection import GridS
from sklearn.metrics import classificatio
from sklearn.metrics import confusion_mat
from sklearn import metrics
from sklearn.model_selection import cross
from sklearn.svm import SVC
```

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

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.

```
path = '/content/drive/MyDrive/Projects

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),
```

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
df_c.to_csv(fp1, index=False)
```

```
df_c = pd.read_csv('/content/drive/MyDr
```

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()
basic_exploration(steps)
```

Output:

```
(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
```

```
df_c['AQI_Bucket'].unique()
```

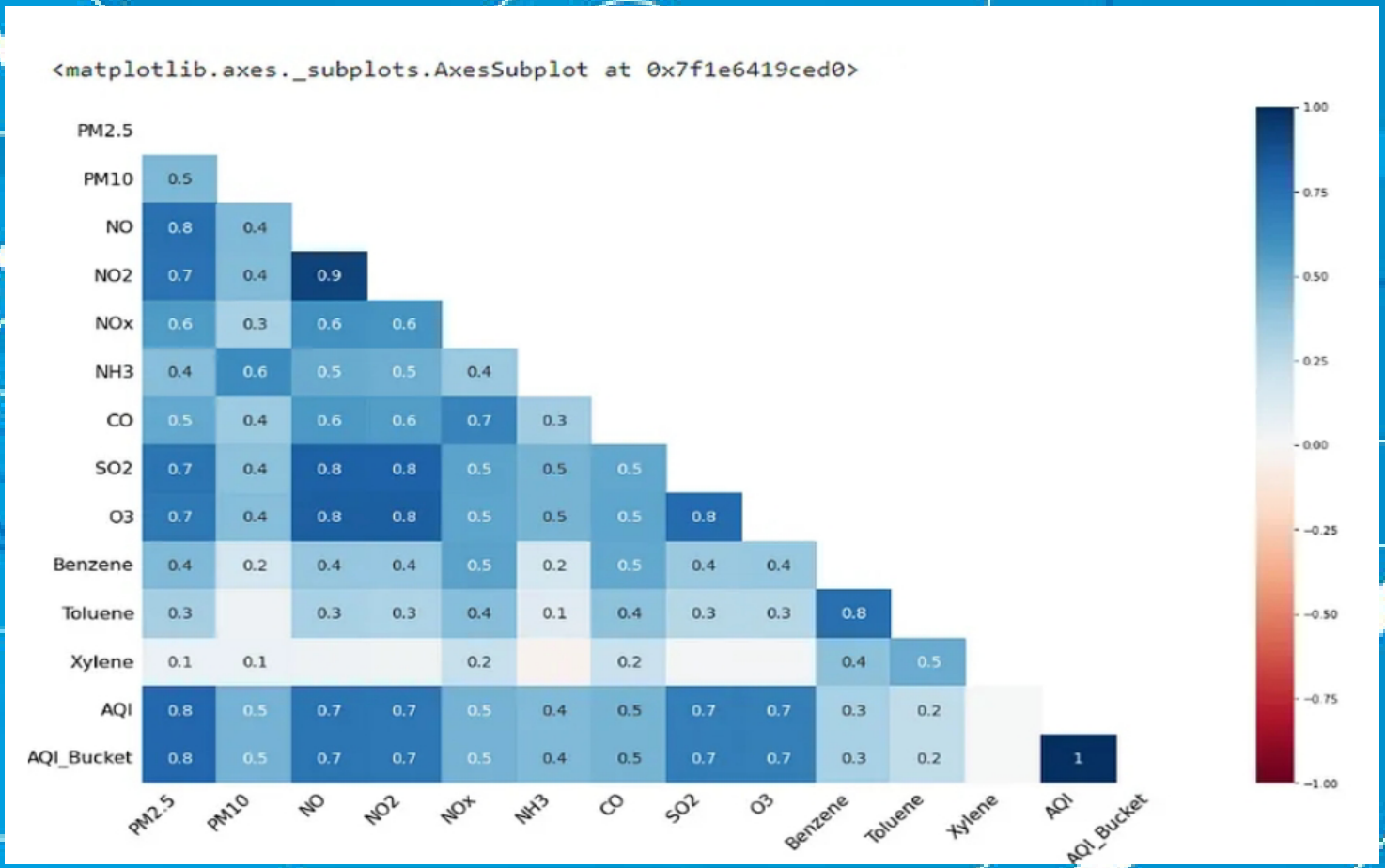
Label encode the ‘AQI_Bucket’ which is the target column for our classification model. Label Encoder assign integers to the classes.

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

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)
```



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.

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 {
```

Check for Null and remove null

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

```
df_c_2['PM2.5'] = df_c_2['PM2.5'].fill
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
df_c_2.isnull().sum()
```

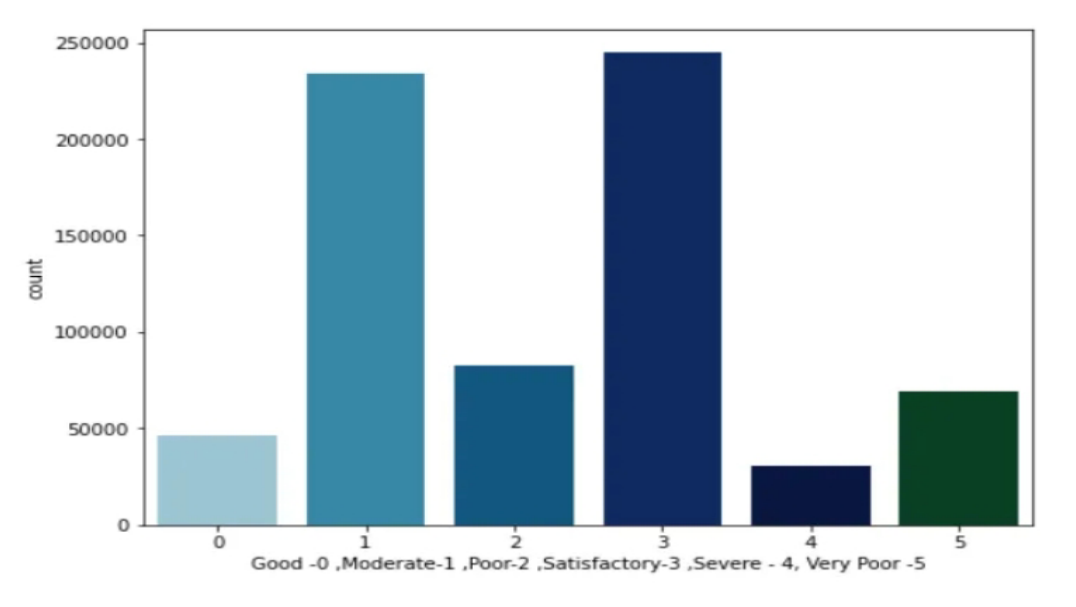
There is rise of toluene
Particulate matters, NOx NO2, have similar distributions.

Exploratory Data Analysis:

Distribution of classes in the Data

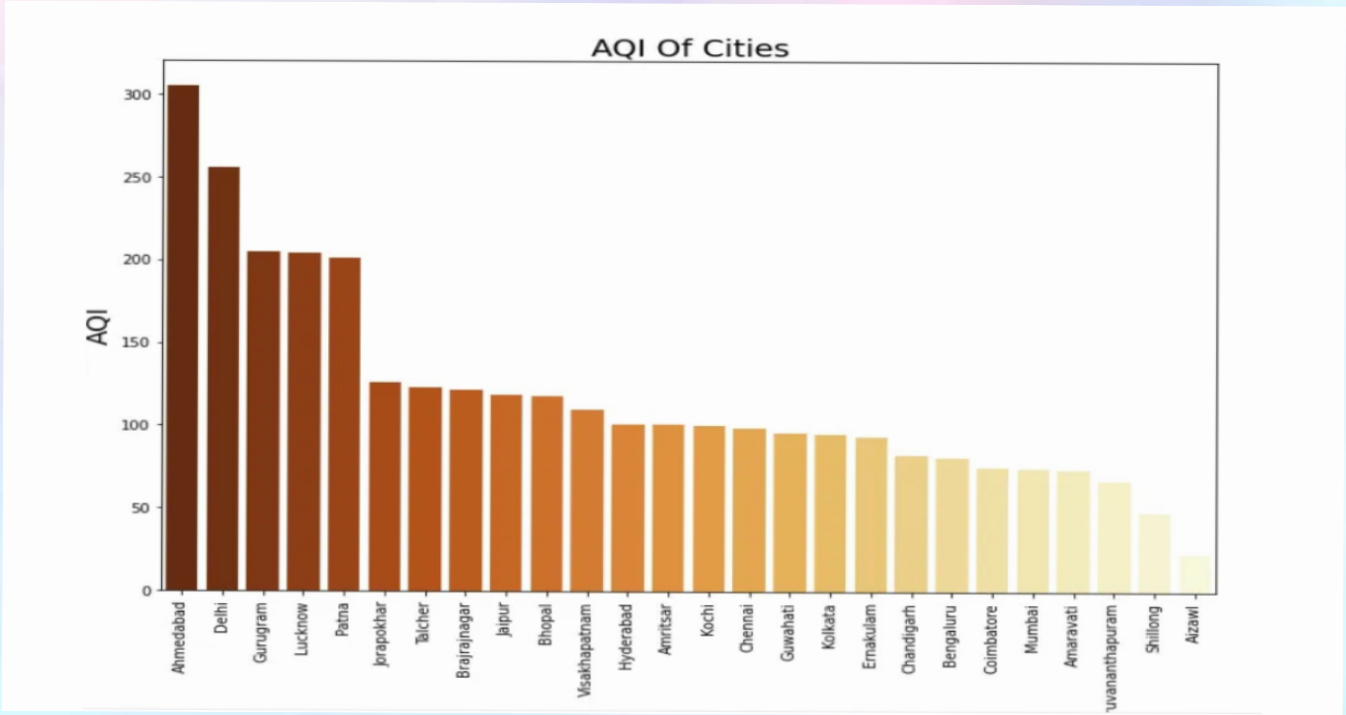
Countplot is a univariate plot ie. single variable under consideration, showing the number of data points representing the categorical variables.

```
def countplot(x_val):
  plt.figure(figsize = (8, 6))
  sns.countplot(x = x_val, palette = '
  plt.xlabel("Good -0 ,Moderate-1 ,Poo
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.

```
"""Select city, AQI from df_ch ORDER BY
# do average of every city aqi value,
city_vs_AQI = df_ch[['City', 'AQI']].gro

def barplot(a,b, title, x_label, y_label
   plt.figure(figsize=(12,8))
   sns.barplot(x = a, y = b, palette =pal
   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
```



Gases and top 3 cities with maximum of that particular gas

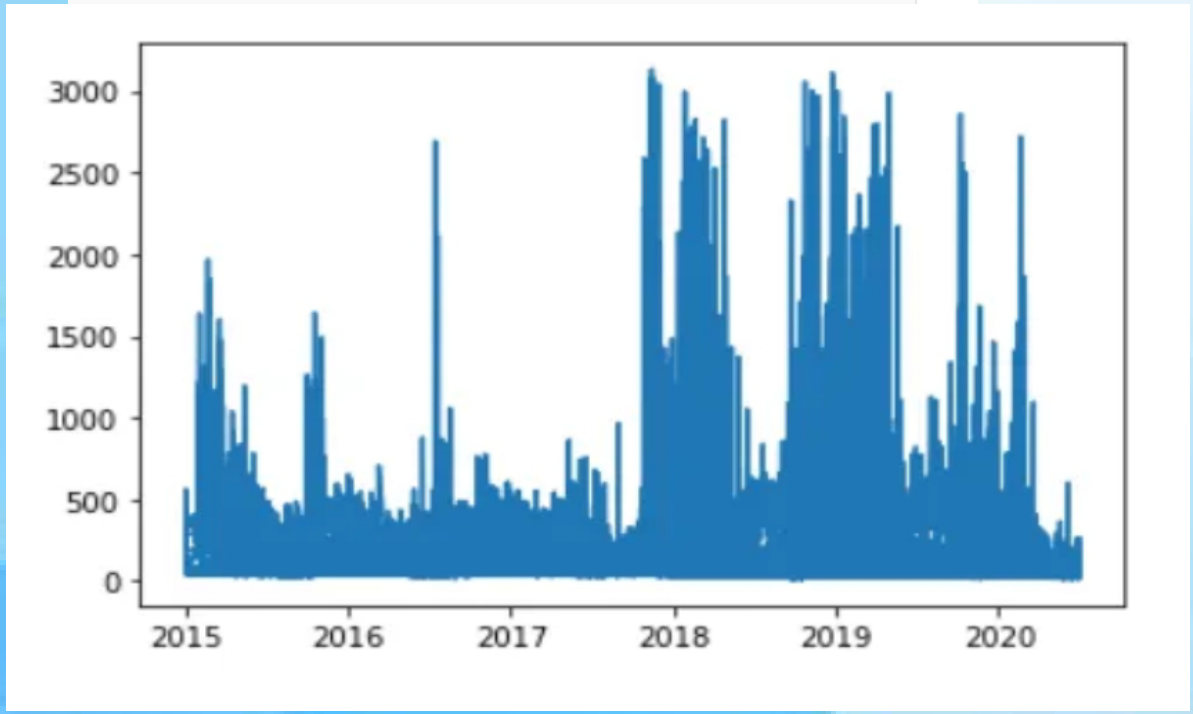
Different cities are extracted which has maximum amount of gas type. Different gases have different cities in top 3.

```
cols = ['PM2.5','PM10','NO','NO2', 'NOx',

# Analyze the states with highest particu
dff = []
for i in cols:
  data = df_ch[[i, 'City']].groupby(['Cit
  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.


THANK YOU