PREDICTIVE ANALYSIS ON AIR QUALITY INDEX

Batch details	PGPDSE-FT Online FEB22
Team Members	Tarun Bansal, Gaurav Gandhi, Subham Kumar, Gitika Gupta, Ankit Patel
Domain of the Project	Environmental Analytics
Group No.	6
Mentor By	Mr Vikash Chandra

Date: **Signature of Mentor:**

greatlearning
Learning for Life

INDEX

OVERVIEW

Air is what keeps humans alive. Monitoring it and understanding its quality is of immense importance to our well-being. The presence of an unwanted thing, agent, or contaminant in the air leads to air pollution. Air Quality Index (AQI) is a tool to disseminate information on air quality in qualitative terms (e.g., good, satisfactory, poor) as well as its associated likely health impacts. The intended uses of AQI are to identify the poor air quality zones and public reporting for severity of exposure of poor air quality. This project attempts to present a review of all the major air quality indices developed in different states of India.

Business Problem Understanding

Particulate matter (PM2.5 and PM10) are the most harmful and life-threatening pollutants among the category of pollutants, according to several prior research on air quality. Every year, particulate matter is a factor in about 800,000 premature deaths.

ALSO READ

IPL 2021 auction: Full list of players retained, released by Delhi Capitals

IPL 2020 road to final: Delhi Capitals' journey to maiden IPL final

Air Quality Commission directs enforcement of dust control measure in Delhi

Air pollution costs Indian businesses $95 billion every year, or 3% of the country's Gross Domestic Product, and cuts annual consumer spending by $22 billion, according to a study released on Thursday.

Every year, higher pollution levels shave off $1.3 billion from India's IT sector in lost productivity, said the study by industry group CII; Clean Air Fund, a charity; and consulting firm Dalberg.

Source: *https://www.business-standard.com/article/current-affairs/air-pollution-costs-indian-businesses-95-billion-a-year-or-3-of-the-gdp-121042001539_1.html*

Air pollution costs Indian businesses $95 billion or 3% of India’s GDP every year. These are:

(a) the opportunity cost of lost worker productivity (e.g., due to higher absenteeism)

(b) revenue decrease/lost due to reduced consumer footfall

(c) the opportunity cost of premature mortality attributed to air pollution.

These costs differ in both the intensity and their directness of impact on businesses.

Approach

We develop to understand the underlying dataset using exploratory data analysis after taking care of null values or any discrepancy in data. After that we will look at how air quality has been affected by these twelve parameters (PM2.5, PM10, NO2, NH3, CO, CO, SO2, O3, Benzene, Toluene and Xylene) over the years. Then based on different locations of India analysis will be done. Comparisons will be made on which state is producing more pollutants and how it has affected those regions using graphs and plots. We will try to study which pollutant has which kind of impact using the data, and the analysis will be done on the same. After splitting our dataset into dependent and independent features we will perform machine learning regression, decision tree, random forest and classification models to study the impact of independent features on dependent ones. We will also perform time series analysis to predict the AQI for upcoming days. Our overall objective would be to detect local trends and relate the air quality changes to changes in environmental policy in India.

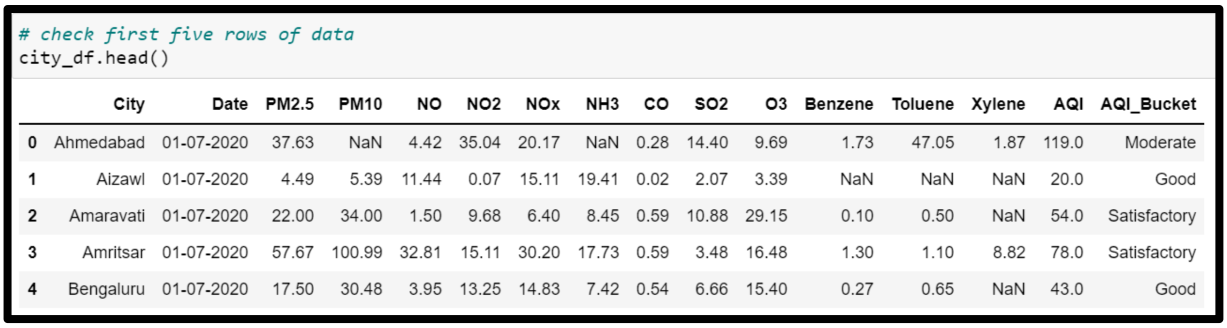
Data Report

Data has been taken from the Central Pollution Control Board, India. It consists of 3 files, one containing records of all the names of states and respective stations in those states, second one with city wise records of the number of pollutants of different dates and third one with records of pollutants in air respective to different stations in a city. The measure of AQI and a classification bucket has been defined for the same.

Now, as a first step, we will import all the necessary libraries or basic python packages that we think would require to perform the EDA (Exploratory Data Analysis).

Basic Data Exploration:

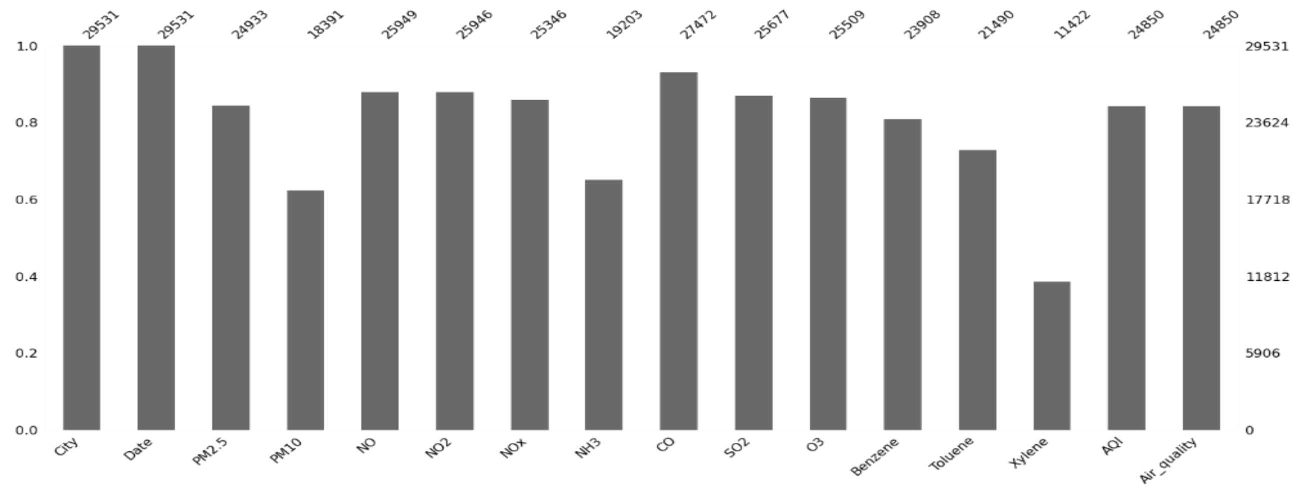
Finally, we will check if the data has been loaded, by using the head function to look at the first five initial rows.



```
# check first five rows of data
city_df.head()
```

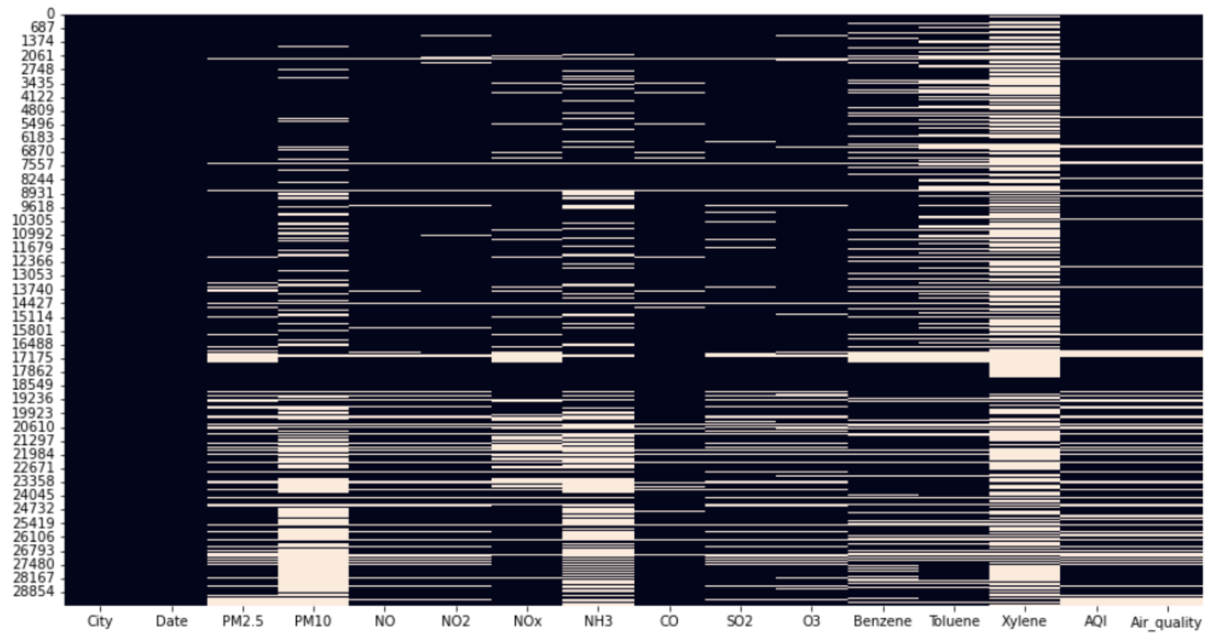
	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket
0	Ahmedabad	01-07-2020	37.63	NaN	4.42	35.04	20.17	NaN	0.28	14.40	9.69	1.73	47.05	1.87	119.0	Moderate
1	Aizawl	01-07-2020	4.49	5.39	11.44	0.07	15.11	19.41	0.02	2.07	3.39	NaN	NaN	NaN	20.0	Good
2	Amaravati	01-07-2020	22.00	34.00	1.50	9.68	6.40	8.45	0.59	10.88	29.15	0.10	0.50	NaN	54.0	Satisfactory
3	Amritsar	01-07-2020	57.67	100.99	32.81	15.11	30.20	17.73	0.59	3.48	16.48	1.30	1.10	8.82	78.0	Satisfactory
4	Bengaluru	01-07-2020	17.50	30.48	3.95	13.25	14.83	7.42	0.54	6.66	15.40	0.27	0.65	NaN	43.0	Good

Next, we will be plotting the bar plot to understand the behaviour of the data. It will show the values that are proportional to the non-missing data in the dataset. Along with that, the number of values missing is also shown.



The above bar graph represents the total count of each parameter(column). The maximum count of a column is 29531(City and Date) while the minimum is 11422(Xylene). Hence, Xylene has the most number of null values present.

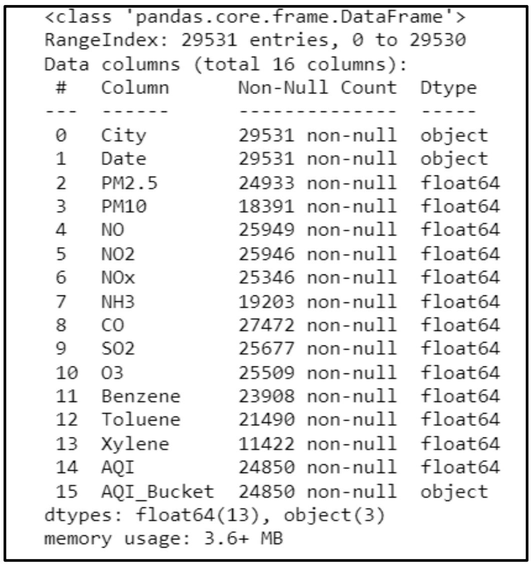
This Heat map represents the missing values in our dataset, where we can see 'Xylene' has most missing values.



<u>Visual inspection of data (rows, columns, descriptive details)</u>

- There are 29531 rows(observations) and 16 columns(variables) in the dataset.

# Summary of all the variables in the dataset

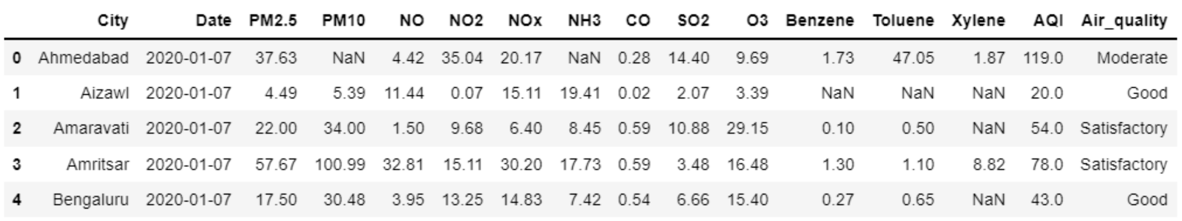

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29531 entries, 0 to 29530
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   City        29531 non-null  object
 1   Date        29531 non-null  object
 2   PM2.5       24933 non-null  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
```

From the summary we are able to understand the count and data type of a particular column. Also, it tells us the discrepancy in data representing null and not null values.

City, Date and AQI_Bucket are of the object datatype while PM2.5, PM10, NO, NO2, NOx, NH3, CO, SO2, O3, Benzene, Toulene, Xylene and AQI are float datatypes.

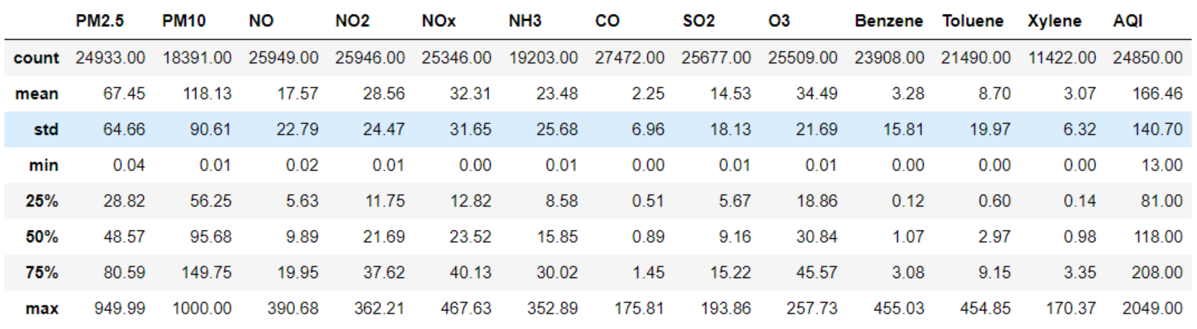
We will have to convert datatype of 'Date' column into Datetime format so as to perform EDA based on dates, also time series analysis using date column.

"Using pandas.to_datetime () function to convert string to datetime (YYY-MM-DD) format."



	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	Air_quality
0	Ahmedabad	2020-01-07	37.63	NaN	4.42	35.04	20.17	NaN	0.28	14.40	9.69	1.73	47.05	1.87	119.0	Moderate
1	Aizawl	2020-01-07	4.49	5.39	11.44	0.07	15.11	19.41	0.02	2.07	3.39	NaN	NaN	NaN	20.0	Good
2	Amaravati	2020-01-07	22.00	34.00	1.50	9.68	6.40	8.45	0.59	10.88	29.15	0.10	0.50	NaN	54.0	Satisfactory
3	Amritsar	2020-01-07	57.67	100.99	32.81	15.11	30.20	17.73	0.59	3.48	16.48	1.30	1.10	8.82	78.0	Satisfactory
4	Bengaluru	2020-01-07	17.50	30.48	3.95	13.25	14.83	7.42	0.54	6.66	15.40	0.27	0.65	NaN	43.0	Good

# Descriptive Statistics of all the numeric variables in the dataset:



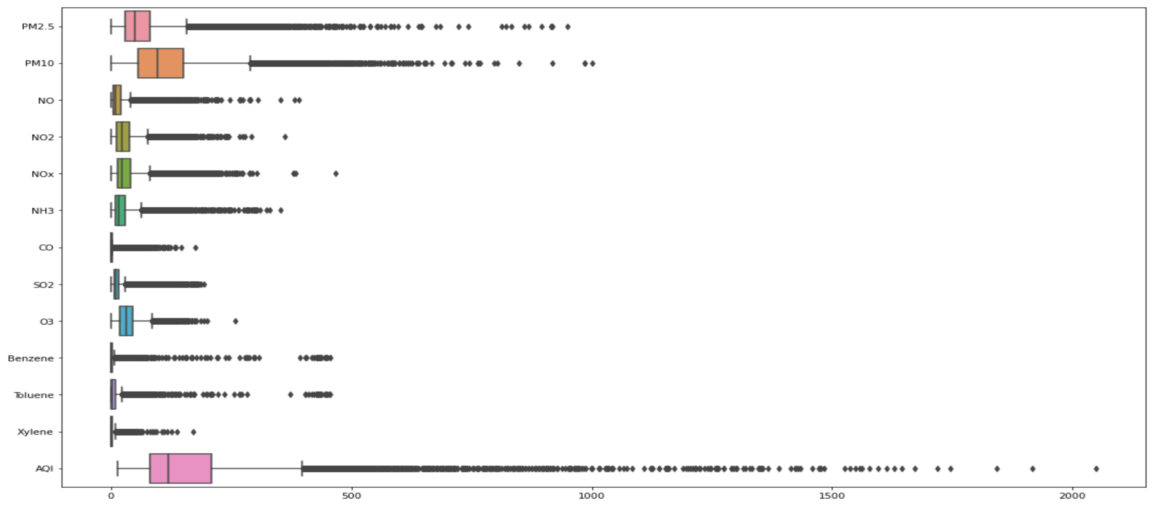
	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI
count	24933.00	18391.00	25949.00	25946.00	25346.00	19203.00	27472.00	25677.00	25509.00	23908.00	21490.00	11422.00	24850.00
mean	67.45	118.13	17.57	28.56	32.31	23.48	2.25	14.53	34.49	3.28	8.70	3.07	166.46
std	64.66	90.61	22.79	24.47	31.65	25.68	6.96	18.13	21.69	15.81	19.97	6.32	140.70
min	0.04	0.01	0.02	0.01	0.00	0.01	0.00	0.01	0.01	0.00	0.00	0.00	13.00
25%	28.82	56.25	5.63	11.75	12.82	8.58	0.51	5.67	18.86	0.12	0.60	0.14	81.00
50%	48.57	95.68	9.89	21.69	23.52	15.85	0.89	9.16	30.84	1.07	2.97	0.98	118.00
75%	80.59	149.75	19.95	37.62	40.13	30.02	1.45	15.22	45.57	3.08	9.15	3.35	208.00
max	949.99	1000.00	390.68	362.21	467.63	352.89	175.81	193.86	257.73	455.03	454.85	170.37	2049.00

Data summaries usually present the dataset's count, mean, median, and/or mode, standard deviation from mean or interquartile range and how it is distributed across the range of data. This helps us to find out the outliers and skewness in the dataset.

As seen in the data, there is a huge difference in the 75 percentile and the maximum value of each parameter, which shows extreme outliers present in the dataset, and the data is mostly right skewed.

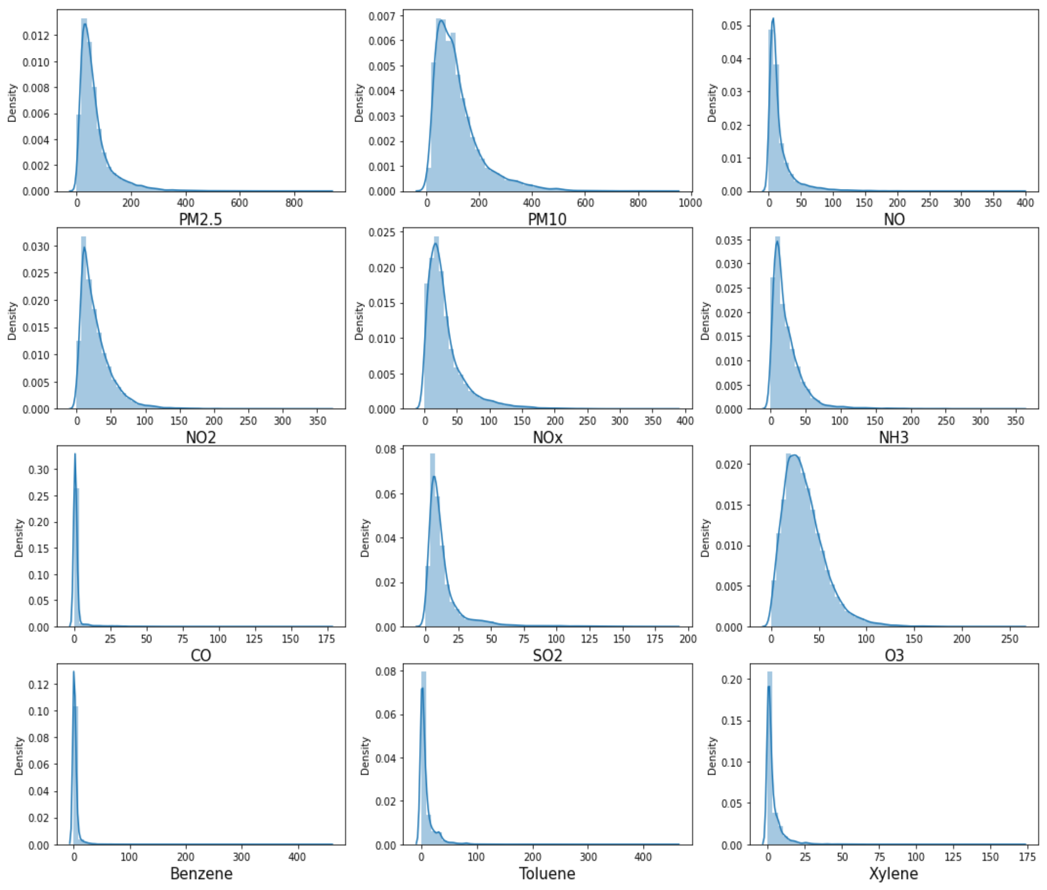
Exploratory Data Analysis:

# Plotting the Boxplot using boxplot() from seaborn for each variable.



A box plot is also used to identify outliers. The points beyond the whiskers are the outliers. Our dataset should mandatorily have outliers as in many of the cities AQI level is severe, the same is represented by our dataset. Yet for a better model we have removed the extreme outliers whose AQI value is more than 1478.

# Plotting the distplot for each variable to check the distribution and skewness of the variables



Inference: *The assumption we made earlier by 5-point summary that the data is mostly right skewed is also verified by the above distplot. The plot doesn't show a proper bell-shaped curve which means the data is not normally distributed.*

<u>Checking the null values/missing values:</u>

```
City              0
Date              0
PM2.5          4598
PM10          11140
NO             3582
NO2            3585
NOx            4185
NH3           10328
CO             2059
SO2            3854
O3             4022
Benzene        5623
Toluene        8041
Xylene        18109
AQI            4681
AQI_Bucket     4681
dtype: int64
```

Here we find out the missing / null values in the given data.

As seen, Xylene has the most number of null values present (18109), while City and Data have no null values.

As a part of our next step, we need to decide whether to drop the null values or to fill the same with the mean, median or forward/backward fill.

#<u>Replacing the null/missing values with 'Median':</u>

We have used the median of each city for each parameter to fill the null values as the mean of the data is affected by the presence of outliers. As the data is numeric, using mode would also not be a viable solution to the problem.

<u>#Checking the result:</u>

```
City           0
Date           0
PM2.5          0
PM10           0
NO             0
NO2            0
NOx            0
NH3            0
CO             0
SO2            0
O3             0
Benzene        0
Toluene        0
Xylene         0
AQI            0
Air_quality    0
dtype: int64
```

After imputing null values with the median values, we can see no null values are present now.

<u>#Statistical Testing</u>:

1. Pearson correlation: The Pearson correlation **measures the strength of the linear relationship between two variables**. It has a value between -1 to 1, with a value of -1 meaning a total negative linear correlation, 0 being no correlation, and + 1 meaning a total positive correlation.

```
PM2.5 has a clear Effect on Target!

PM10 has a clear Effect on Target!

NO has a clear Effect on Target!

NO2 has a clear Effect on Target!

NOx has a clear Effect on Target!

NH3 has a clear Effect on Target!

CO has a clear Effect on Target!

SO2 has a clear Effect on Target!

O3 has a clear Effect on Target!

Benzene has a clear Effect on Target!

Toluene has a clear Effect on Target!

Xylene has a clear Effect on Target!
```

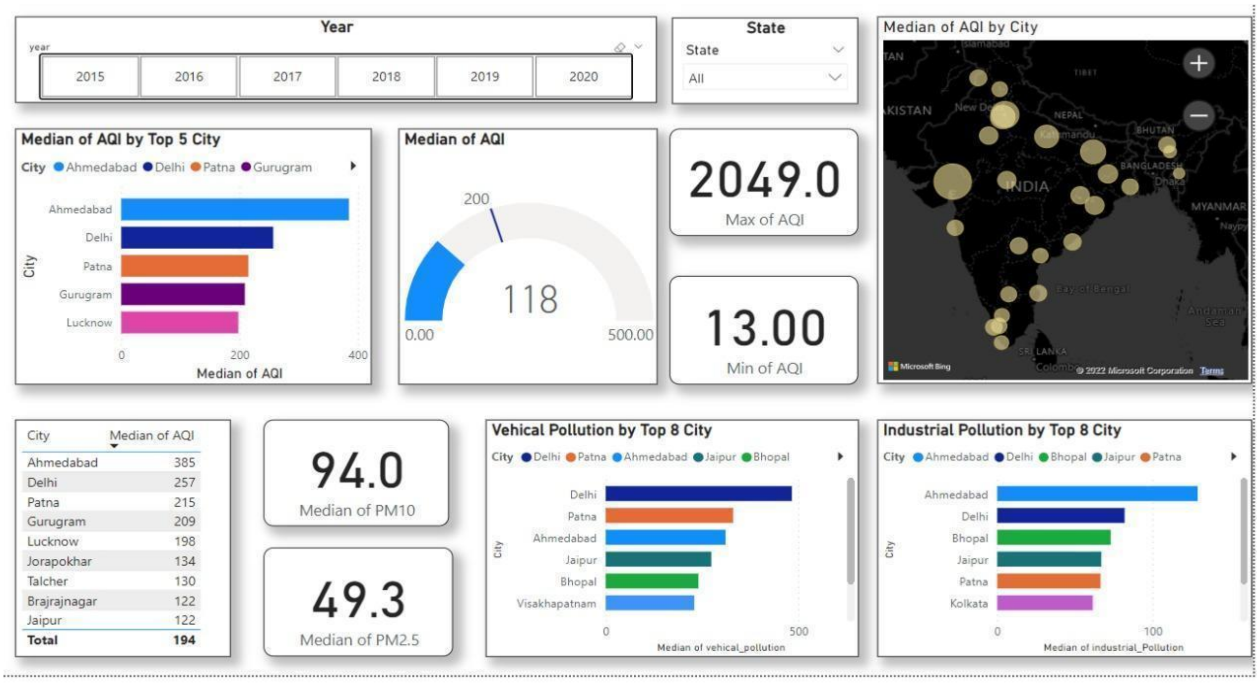
H0: Has No Effect on Target!

H1: Has a Clear Effect on

# Analysis of AQI - Pre Corona [2016 to 2020]:

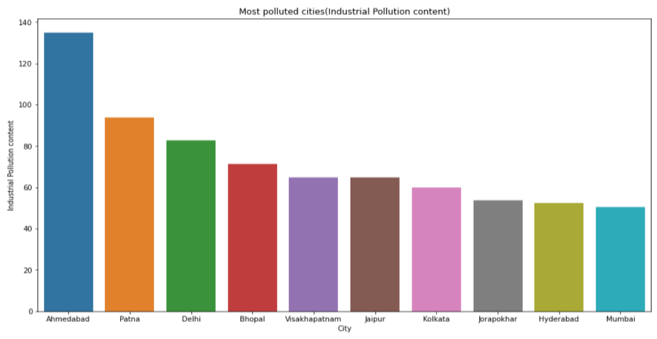
Here we divide the data set into two parts namely **Vehicular Pollution** content (PM2.5, PM10, NO2, NH3, CO,) and **Industrial Pollution content** (CO, SO2, O3, Benzene, Toluene, Xylene) and find how these contents correlated with AQI (air quality index).

# We have also built a power Bi dashboard for EDA:



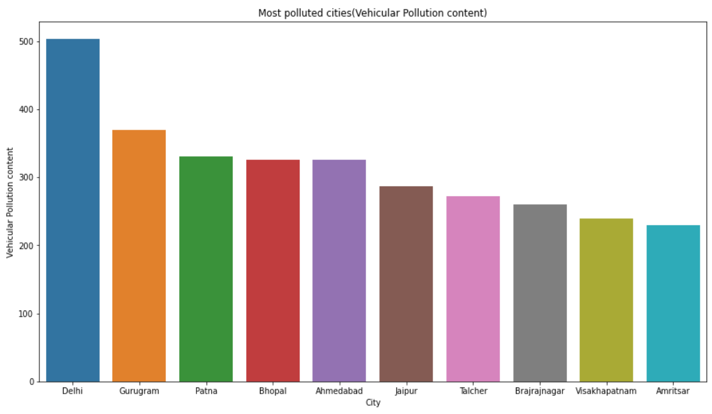
City	Median of AQI
Ahmedabad	385
Delhi	257
Patna	215
Gurugram	209
Lucknow	198
Jorapokhar	134
Talcher	130
Brajrajnagar	122
Jaipur	122
Total	**194**

# Checking Most polluted cities (Industrial Pollution content):



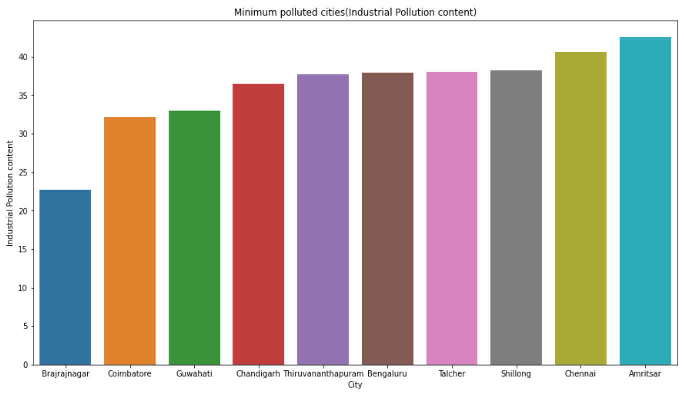
Inference: According to the given dataset Ahmedabad is the most polluted city considering Industrial Pollution Content.

# <u>Checking Most polluted cities (Vehicle Pollution content):</u>



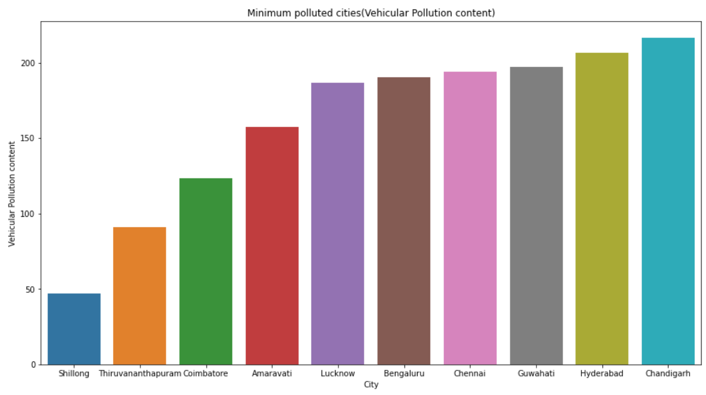
Inference: According to the given dataset Delhi is the most polluted city considering Vehicle Pollution Content.

# <u>Checking Least polluted cities (Industrial Pollution content):</u>



Inference: According to the given dataset Brajrajnagar is the least polluted city considering Industrial Pollution Content.

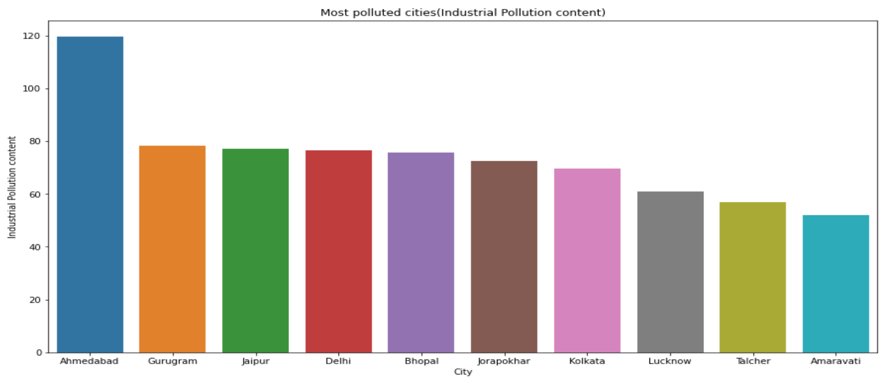
# <u>Checking Least polluted cities (Vehicle Pollution content):</u>



Inference: According to the given dataset Shillong is the least polluted city considering Industrial Pollution Content.

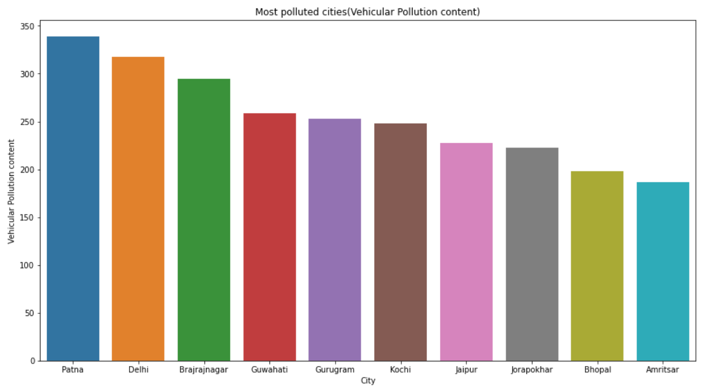
# Analysis of AQI - Post Corona [2020 >]

# Checking Most polluted cities (Industrial Pollution content):



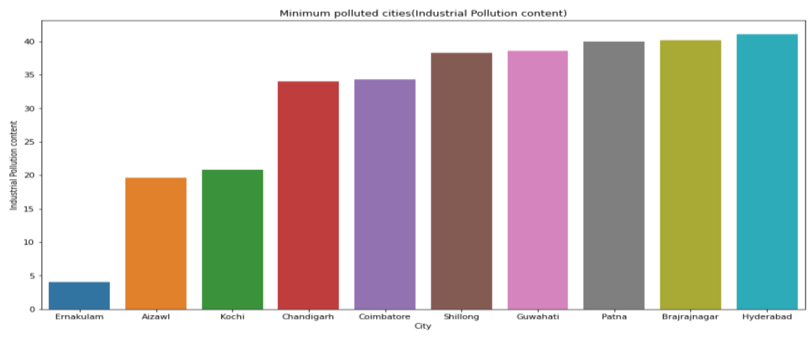
Inference: According to the given dataset Ahmedabad remains the most polluted city considering Industrial Pollution Content.

# Checking Most polluted cities (Vehicle Pollution content):



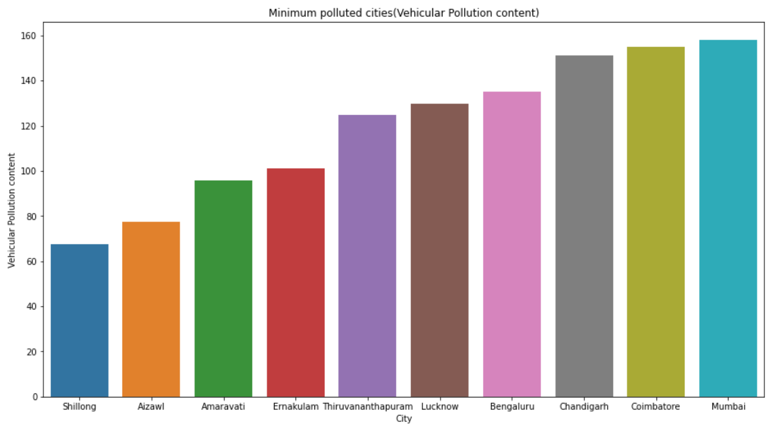
Inference: According to the given dataset post Corona, Patna overtakes Delhi as the most polluted city considering Vehicle Pollution Content.

# Checking Least polluted cities (Industrial Pollution content):



Inference: According to the given dataset post Corona, Ernakulam overtakes Brajrajnagar as the least polluted city considering Industrial Pollution Content.

# <u>Checking Least polluted cities (Vehicle Pollution content):</u>



Inference: According to the given dataset post Corona, Shillong remains the least polluted city taking into consideration Vehicle Pollution Content.

Model Building:

Multiple Linear Regression (OLS)

First, we create the data frame for independent and target variables for linear regression.

We have taken all the variables except AQI as the features for the model.

AQI is our target variable for the model.

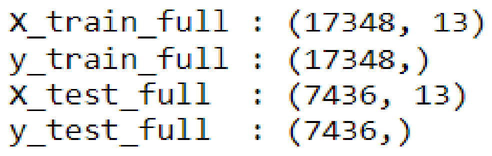
Scaling the data

As our first step, we have used StandardScaler to scale the data to get all the variables in the same range. With this, we can avoid the problem in which some features come to dominate solely because they tend to have larger values than others.

Splitting the data using Train-Test Split (70:30)

With this we have split the data into training and test sets.

X_train_full, y_train_full, X_test_full, y_test_full.



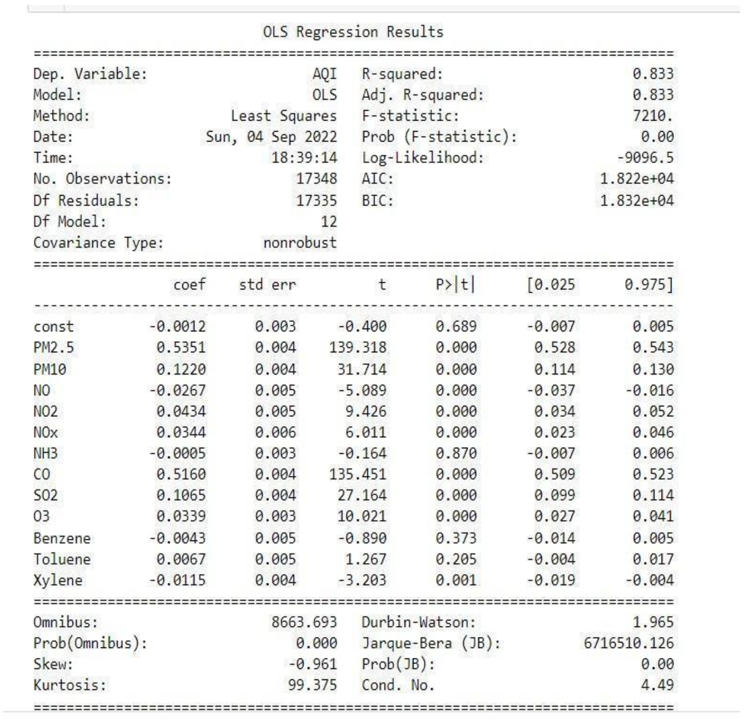
```
X_train_full : (17348, 13)
y_train_full : (17348,)
X_test_full  : (7436, 13)
y_test_full  : (7436,)
```

Out of 24,784 records, 17348 records have been put into the training dataset, and 7436 records into the test dataset, having a 70:30 split.

# Full Model (OLS):

1. We use OLS for estimating the unknown parameters in a linear regression model.

Then we built an MLR_FULL_MODEL and using fit() function; we fit the OLS model.



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    AQI   R-squared:                       0.833
Model:                            OLS   Adj. R-squared:                  0.833
Method:                 Least Squares   F-statistic:                     7210.
Date:                Sun, 04 Sep 2022   Prob (F-statistic):               0.00
Time:                        18:39:14   Log-Likelihood:                -9096.5
No. Observations:               17348   AIC:                         1.822e+04
Df Residuals:                   17335   BIC:                         1.832e+04
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0012      0.003     -0.400      0.689      -0.007       0.005
PM2.5          0.5351      0.004    139.318      0.000       0.528       0.543
PM10           0.1220      0.004     31.714      0.000       0.114       0.130
NO            -0.0267      0.005     -5.089      0.000      -0.037      -0.016
NO2            0.0434      0.005      9.426      0.000       0.034       0.052
NOx            0.0344      0.006      6.011      0.000       0.023       0.046
NH3           -0.0005      0.003     -0.164      0.870      -0.007       0.006
CO             0.5160      0.004    135.451      0.000       0.509       0.523
SO2            0.1065      0.004     27.164      0.000       0.099       0.114
O3             0.0339      0.003     10.021      0.000       0.027       0.041
Benzene       -0.0043      0.005     -0.890      0.373      -0.014       0.005
Toluene        0.0067      0.005      1.267      0.205      -0.004       0.017
Xylene        -0.0115      0.004     -3.203      0.001      -0.019      -0.004
==============================================================================
Omnibus:                     8663.693   Durbin-Watson:                   1.965
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          6716510.126
Skew:                          -0.961   Prob(JB):                         0.00
Kurtosis:                      99.375   Cond. No.                         4.49
==============================================================================
```

Inferences:

1. We find Root Mean Squared Error (RMSE) on training set to measure the average of the squares of the errors:

   ```
   Root Mean Squared Error (RMSE) on training set: 0.4088
   ```

2. Then find a Root Mean Squared Error (RMSE) on test set:

   ```
   Root Mean Squared Error (RMSE) on test set:  0.3891
   ```

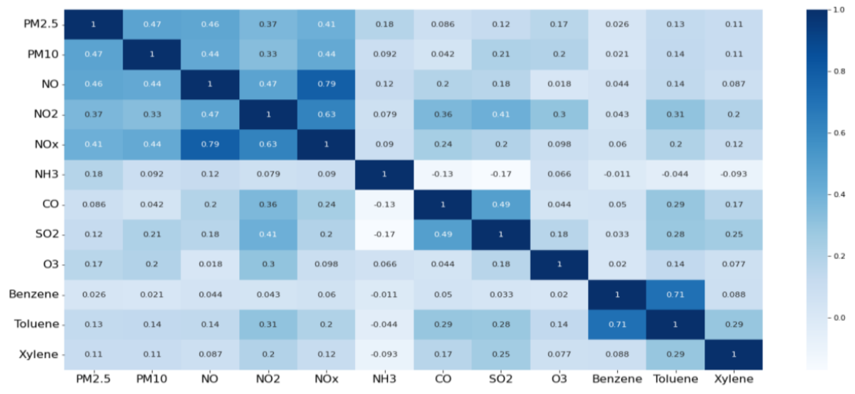
#MLR full model using Sklearn full model:

#Variance Inflation Factor (VIF) Model:

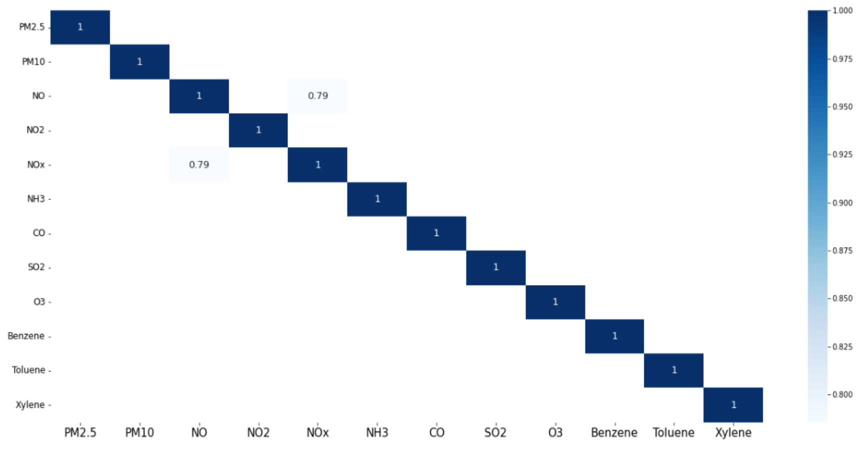
We will find Variance Inflation Factor (VIF) to detect the presence of multicollinearity between the features. The value of VIF equal to 1 indicates that no features are correlated. We calculate VIF of the numerical independent variables.

Correlation Matrix:

Plotting the heatmap to visualize the correlation matrix of the independent variables. The variables with a high correlation may induce multicollinearity in the data.



Then Identifying the variables with a correlation greater than 0.75 and less than -0.75.

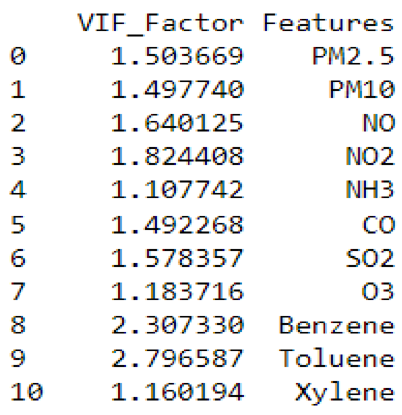


# **<u>Then we Calculate the VIF for each numeric variable</u>:**

The output shows that the variable 'NOx' has the highest VIF. Now, we use the for loop to find VIF and remove the variables with VIF greater than 3. We set the threshold to 3, as we wish to remove the variable for which the remaining variables explain more than 97% of the variation.

After calculating the VIF the variable 'NOx' has the highest VIF.

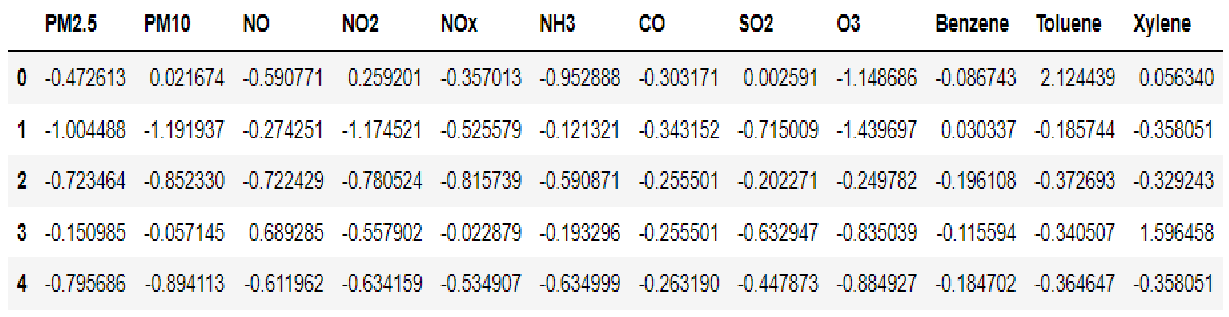
Now, we use the for loop to find VIF and remove the variables with VIF greater than 3. We set the threshold to 3, as we wish to remove the variable for which the remaining variables explain more than 97% of the variation:



```
     VIF_Factor  Features
0      1.503669     PM2.5
1      1.497740      PM10
2      1.640125        NO
3      1.824408       NO2
4      1.107742       NH3
5      1.492268        CO
6      1.578357       SO2
7      1.183716        O3
8      2.307330   Benzene
9      2.796587   Toluene
10     1.160194    Xylene
```

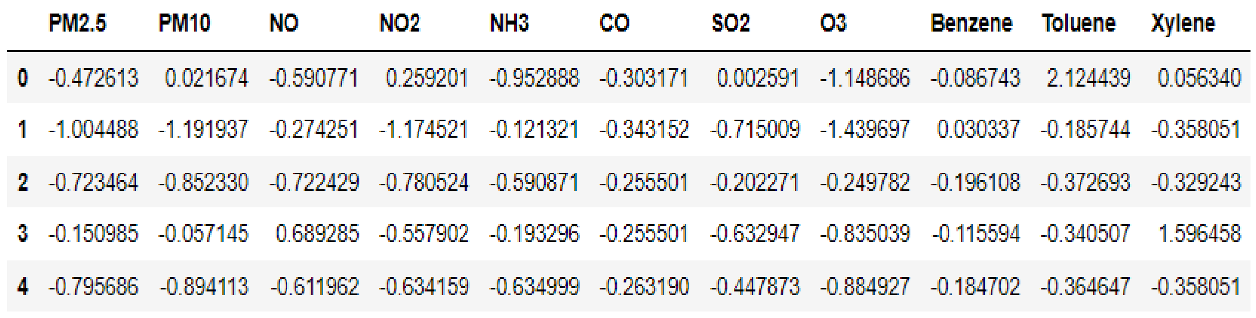
After calculating VIF we reduce the number of parameters.

# Then we check the first 5 observations of full features



	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene
0	-0.472613	0.021674	-0.590771	0.259201	-0.357013	-0.952888	-0.303171	0.002591	-1.148686	-0.086743	2.124439	0.056340
1	-1.004488	-1.191937	-0.274251	-1.174521	-0.525579	-0.121321	-0.343152	-0.715009	-1.439697	0.030337	-0.185744	-0.358051
2	-0.723464	-0.852330	-0.722429	-0.780524	-0.815739	-0.590871	-0.255501	-0.202271	-0.249782	-0.196108	-0.372693	-0.329243
3	-0.150985	-0.057145	0.689285	-0.557902	-0.022879	-0.193296	-0.255501	-0.632947	-0.835039	-0.115594	-0.340507	1.596458
4	-0.795686	-0.894113	-0.611962	-0.634159	-0.534907	-0.634999	-0.263190	-0.447873	-0.884927	-0.184702	-0.364647	-0.358051

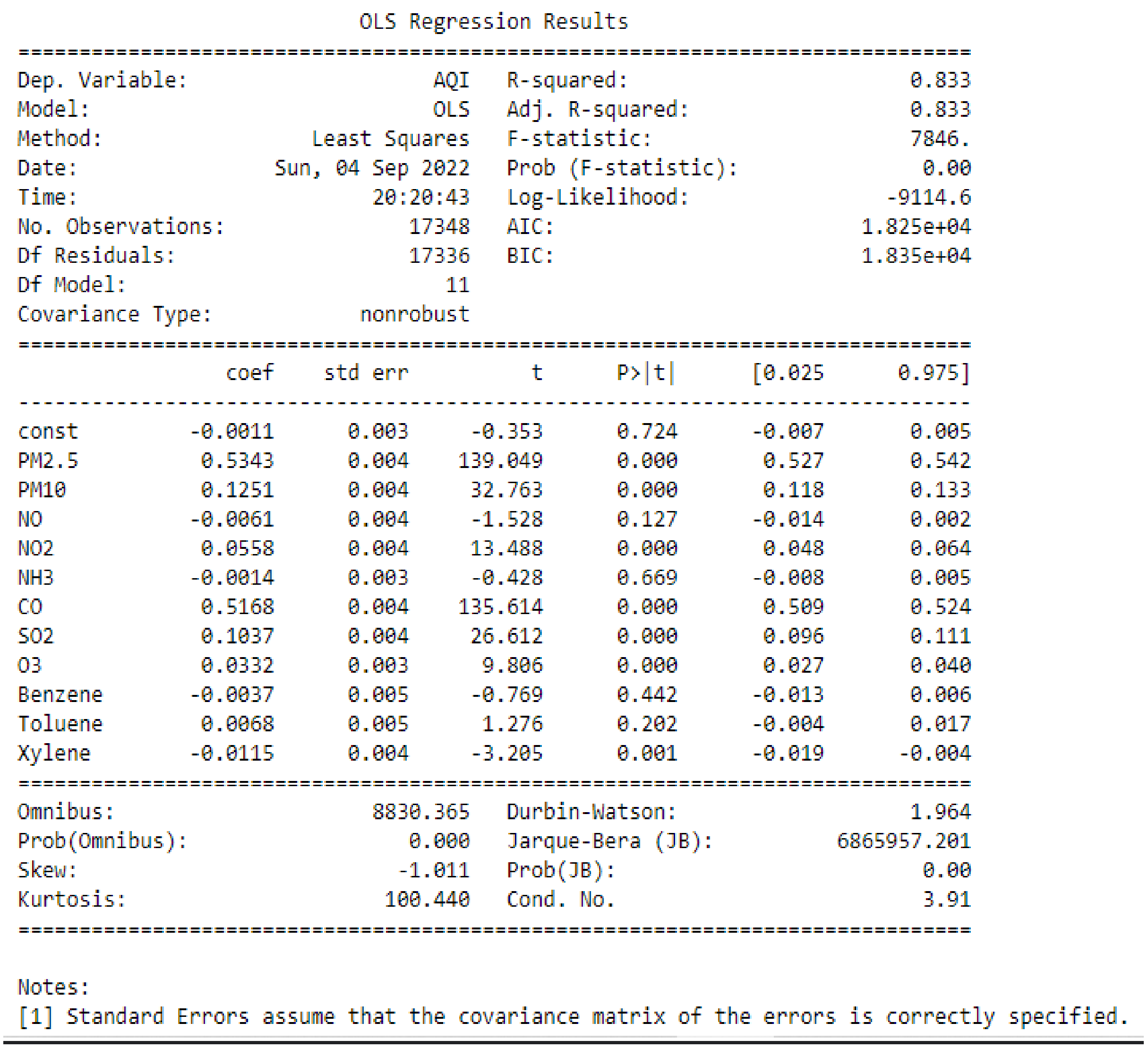
# We check the first 5 observations of VIF reduced features



	PM2.5	PM10	NO	NO2	NH3	CO	SO2	O3	Benzene	Toluene	Xylene
0	-0.472613	0.021674	-0.590771	0.259201	-0.952888	-0.303171	0.002591	-1.148686	-0.086743	2.124439	0.056340
1	-1.004488	-1.191937	-0.274251	-1.174521	-0.121321	-0.343152	-0.715009	-1.439697	0.030337	-0.185744	-0.358051
2	-0.723464	-0.852330	-0.722429	-0.780524	-0.590871	-0.255501	-0.202271	-0.249782	-0.196108	-0.372693	-0.329243
3	-0.150985	-0.057145	0.689285	-0.557902	-0.193296	-0.255501	-0.632947	-0.835039	-0.115594	-0.340507	1.596458
4	-0.795686	-0.894113	-0.611962	-0.634159	-0.634999	-0.263190	-0.447873	-0.884927	-0.184702	-0.364647	-0.358051

# New MLR_VIF_Model:

After calculating remaining features, create a new OLS model & we get reduction in Cond. no. from 4.49 to 3.91.



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    AQI   R-squared:                       0.833
Model:                            OLS   Adj. R-squared:                  0.833
Method:                 Least Squares   F-statistic:                     7846.
Date:                Sun, 04 Sep 2022   Prob (F-statistic):               0.00
Time:                        20:20:43   Log-Likelihood:                -9114.6
No. Observations:               17348   AIC:                         1.825e+04
Df Residuals:                   17336   BIC:                         1.835e+04
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0011      0.003     -0.353      0.724      -0.007       0.005
PM2.5          0.5343      0.004    139.049      0.000       0.527       0.542
PM10           0.1251      0.004     32.763      0.000       0.118       0.133
NO            -0.0061      0.004     -1.528      0.127      -0.014       0.002
NO2            0.0558      0.004     13.488      0.000       0.048       0.064
NH3           -0.0014      0.003     -0.428      0.669      -0.008       0.005
CO             0.5168      0.004    135.614      0.000       0.509       0.524
SO2            0.1037      0.004     26.612      0.000       0.096       0.111
O3             0.0332      0.003      9.806      0.000       0.027       0.040
Benzene       -0.0037      0.005     -0.769      0.442      -0.013       0.006
Toluene        0.0068      0.005      1.276      0.202      -0.004       0.017
Xylene        -0.0115      0.004     -3.205      0.001      -0.019      -0.004
==============================================================================
Omnibus:                     8830.365   Durbin-Watson:                   1.964
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          6865957.201
Skew:                          -1.011   Prob(JB):                         0.00
Kurtosis:                     100.440   Cond. No.                         3.91
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

After that we find Mean Squared Error (RMSE) on training set to measure the average of the squares of the errors:

```
Mean Squared Error (RMSE) on training set:  0.4092
```

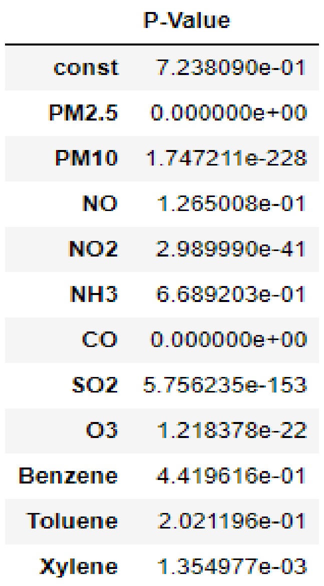
Then we find Mean Squared Error (RMSE) on test set:

```
Mean Squared Error (RMSE) on test set:  0.3889
```

Interpretation: *Although VIF_Factor for ['NOx"] is more than 3, dropping NOx columns did not impact on the model much.*

# Model with Significant P-value:

If we talk about significant features with P-value we got (PM2.5, PM10, NO2, CO, SO2, O3, Xylene) features.



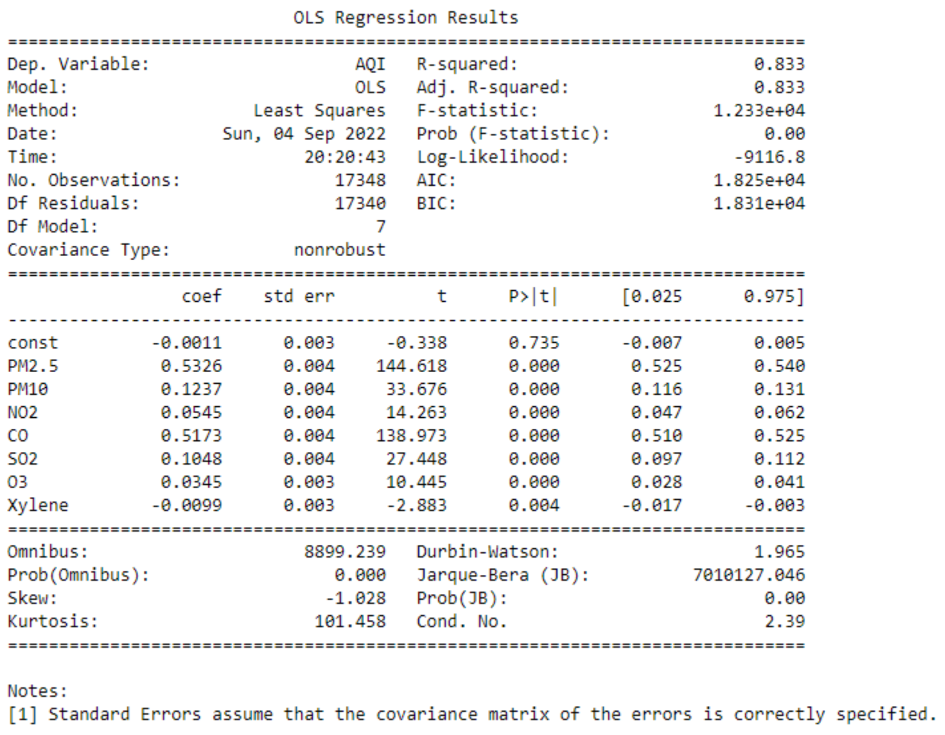
	P-Value
const	7.238090e-01
PM2.5	0.000000e+00
PM10	1.747211e-228
NO	1.265008e-01
NO2	2.989990e-41
NH3	6.689203e-01
CO	0.000000e+00
SO2	5.756235e-153
O3	1.218378e-22
Benzene	4.419616e-01
Toluene	2.021196e-01
Xylene	1.354977e-03

Then create a list of insignificant variables. After creating the insignificant variable, we have the list of insignificant variables:

```
['NO', 'NH3', 'Benzene',
'Toluene']
```

# OLS model with Significant Features:

Here, New OLS model has been created with significant Features & Cond. no. also improved from 3.91 to 2.39.



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    AQI   R-squared:                       0.833
Model:                            OLS   Adj. R-squared:                  0.833
Method:                 Least Squares   F-statistic:                 1.233e+04
Date:                Sun, 04 Sep 2022   Prob (F-statistic):               0.00
Time:                        20:20:43   Log-Likelihood:                -9116.8
No. Observations:               17348   AIC:                         1.825e+04
Df Residuals:                   17340   BIC:                         1.831e+04
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0011      0.003     -0.338      0.735      -0.007       0.005
PM2.5          0.5326      0.004    144.618      0.000       0.525       0.540
PM10           0.1237      0.004     33.676      0.000       0.116       0.131
NO2            0.0545      0.004     14.263      0.000       0.047       0.062
CO             0.5173      0.004    138.973      0.000       0.510       0.525
SO2            0.1048      0.004     27.448      0.000       0.097       0.112
O3             0.0345      0.003     10.445      0.000       0.028       0.041
Xylene        -0.0099      0.003     -2.883      0.004      -0.017      -0.003
==============================================================================
Omnibus:                     8899.239   Durbin-Watson:                   1.965
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          7010127.046
Skew:                          -1.028   Prob(JB):                         0.00
Kurtosis:                     101.458   Cond. No.                         2.39
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

After that we find Mean Squared Error (RMSE) on the training set to measure the average of the squares of the errors:

```
Mean Squared Error (MSE) on training set:  0.4093
```

Then find a Mean Squared Error (RMSE) on test set:

```
Mean Squared Error (MSE) on test set:  0.3887
```

***Interpretation:** Although p-value for ['NO', 'NH3', 'Benzene', 'Toluene'] is high, dropping those columns did not have much impact on the model.*

So mostly we should not drop them because they contribute to AQI. Dropping them would mean they have no impact on AQI, which is not correct.

# ***Stepwise Regression:***

1.Forward Selection:

This method considers the null model (model with no predictors) in the first step. In the next steps start adding one variable at each step until we run out of the independent variables or the stopping rule is achieved.

The variable is added based on its correlation with the target variable. Such a variable has the least p-value in the model.

Then Initiate linear regression model to use in feature selection.

Finally fit the step forward selection on training data using fit () to get the result are:

```
Features Selected using forward selection are:

('PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'CO', 'SO2', 'O3',
'Xylene').

 R-Squared:  0.8297376361141767.
```

2. Backward Elimination:

This method considers the full model (model with all the predictors) in the first step. In the next steps start removing one variable at each step until we run out of the independent variables or the stopping rule is achieved.

The least significant variable (with the highest p-value) is removed at each step.

Then initiate a linear regression model to use in feature selection.

After initiating model, we print the selected feature names when k_features = 'best' and print the R-squared value:

Features selected using backward elimination are:

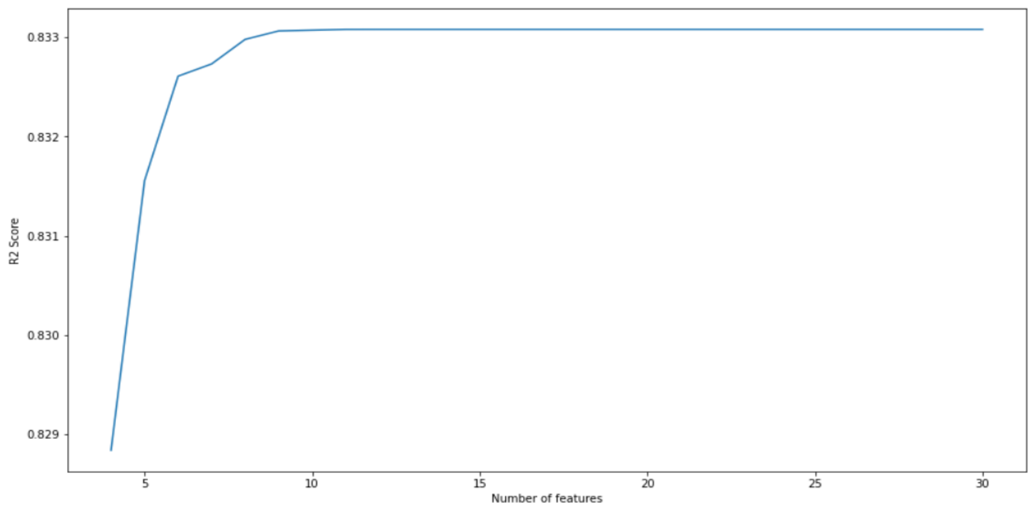
```
('PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'CO', 'SO2', 'O3', 'Xylene')
R-Squared:  0.8297376361141767
```

Recursive Feature Elimination (RFE):

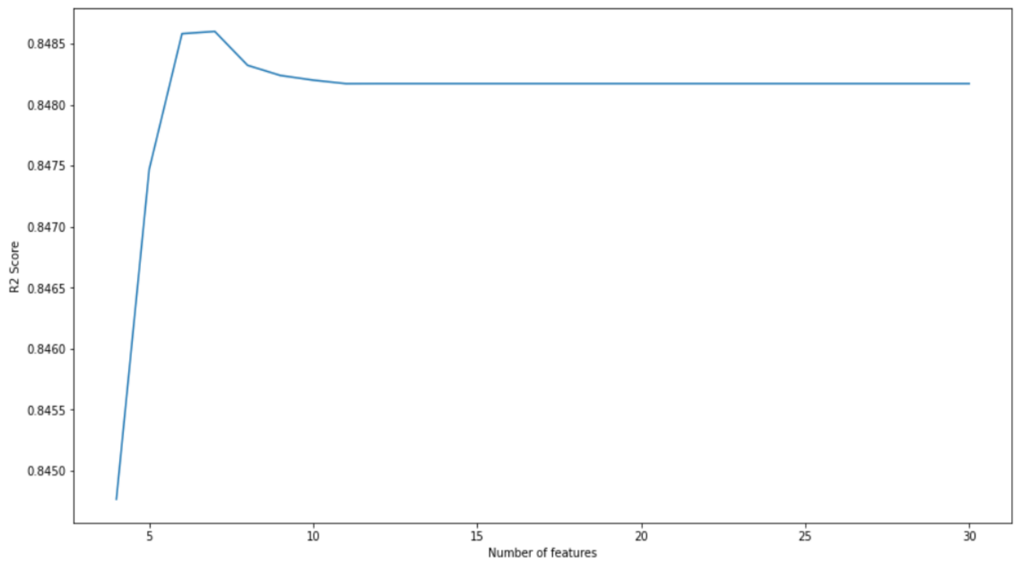
It is the process that returns the significant features in the dataset by recursively removing the less significant feature subsets.

Then We will try to find out what would be the significant number of features for RFE.

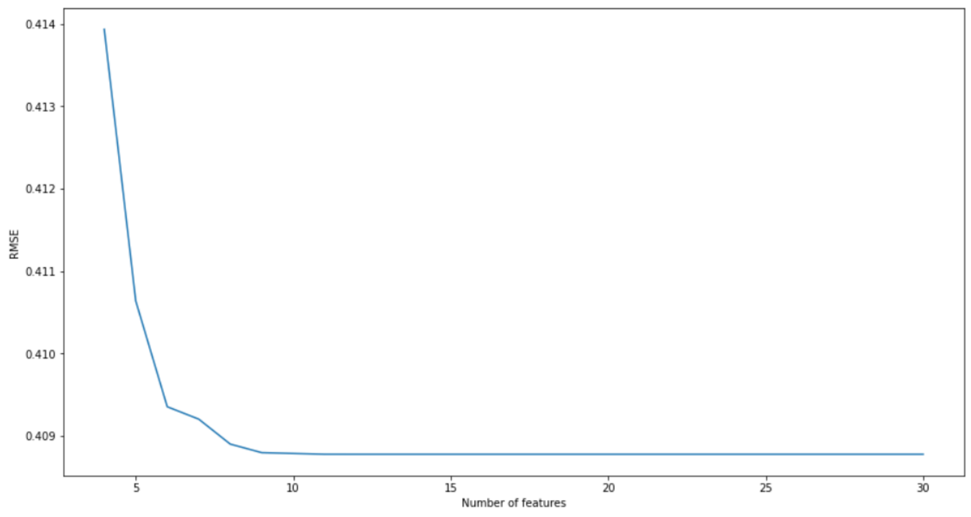
Draw the line plot for train scores with different numbers of features.



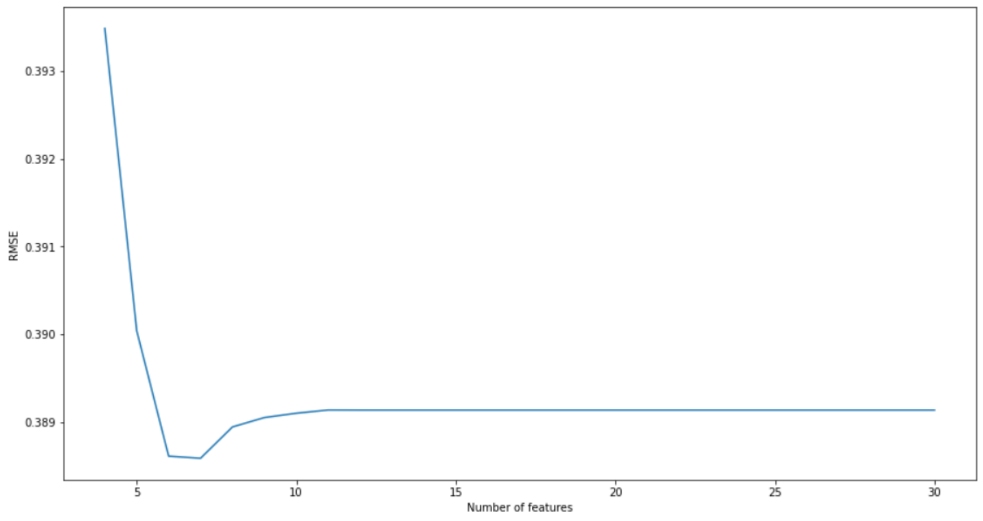
Draw the line plot for test scores with different numbers of features.



Draw the line plot for RMSE values of training dataset with different numbers of features.



Draw the line plot for RMSE values of the test dataset with different numbers of features.



From the above plots we see 7 features would be good to consider for RFE.

Making a model with 7 features.

After that we initiate a linear regression model to use in feature selection.

Then we get output as:

```
Index (['PM2.5', 'PM10', 'NO2', 'NOx', 'CO', 'SO2', 'O3'],
dtype='object').
```

Next, we create a linear regression model using the significant variables obtained after RFE

So, for that **store** the X_train with significant variables in new_X_train and new_X_test and then check the score with linreg.score: `0.832729343053249`

After that we print the RMSE for test and train set:

```
RMSE on train set: 0.4092

RMSE on test set: 0.3886
```

<u>Stochastic Gradient Descent (SGD)</u>:

The gradient descent method considers all the data points to calculate the values of the parameters at each step. For a very large dataset, this method becomes computationally expensive. To avoid this problem, we use Stochastic Gradient Descent (SGD) which considers a single data point (sample) to perform each iteration. Each sample is randomly selected for performing the iteration.

<u>Then we Build MLR model using SGD method:</u>

Instantiating the SGDRegressor and check the train and test score with linreg_with_SGD.score:

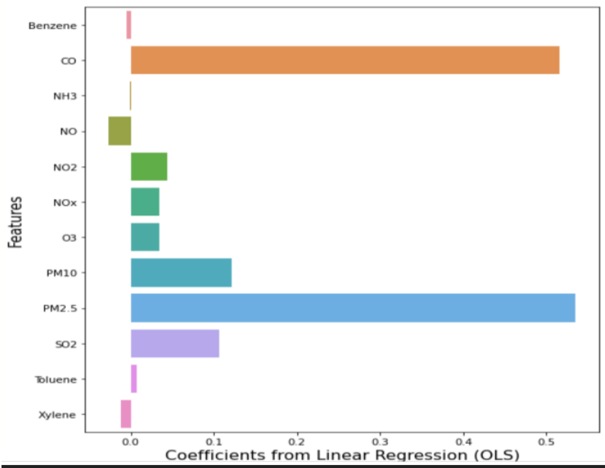
```
Train score: 0.8158685700899333

Test score: 0.8256891801245918
```

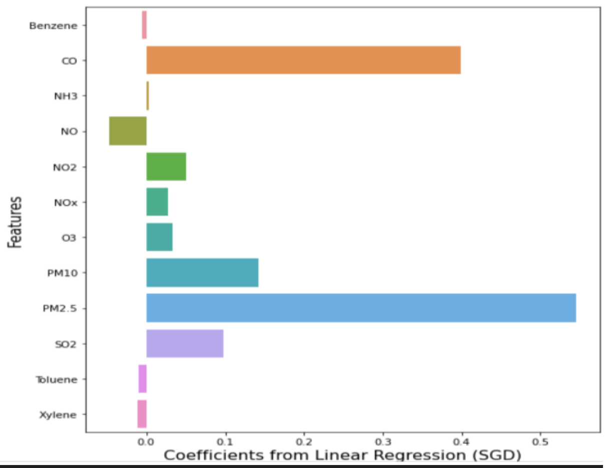
And print the value of RMSE of train and test set:

```
RMSE on train set: 0.4293

RMSE on test set: 0.417
```

<u>Visualize the change in values of coefficients obtained from MLR_model (using OLS) and linreg_with_SGD:</u>

Visualizing the coefficient of SGD model and MLR base model. We can see that there is a significant difference between the coefficients.





Regularization:

One way to deal with the overfitting problem is by adding the Regularization to the model. It is observed that inflation of the coefficients causes overfitting. To prevent overfitting, it is important to regulate the coefficients by penalizing possible coefficient inflations. Regularization imposes penalties on parameters if they inflate to large values to prevent them from being weighed too heavily. In this section, we will learn about the three regularization techniques:

1. Ridge Regression
2. Lasso Regression
3. Elastic Net Regression

Ridge Regression:

Most of the time our data can show multicollinearity in the variables. To analyze such data, we can use Ridge Regression. It uses the L2 norm for regularization.

Build a regression model using Ridge Regression for alpha = 1.

To build the model we use Ridge() to perform ridge regression and then we get

Score of the ridge model on training set: `0.8330774910843589`

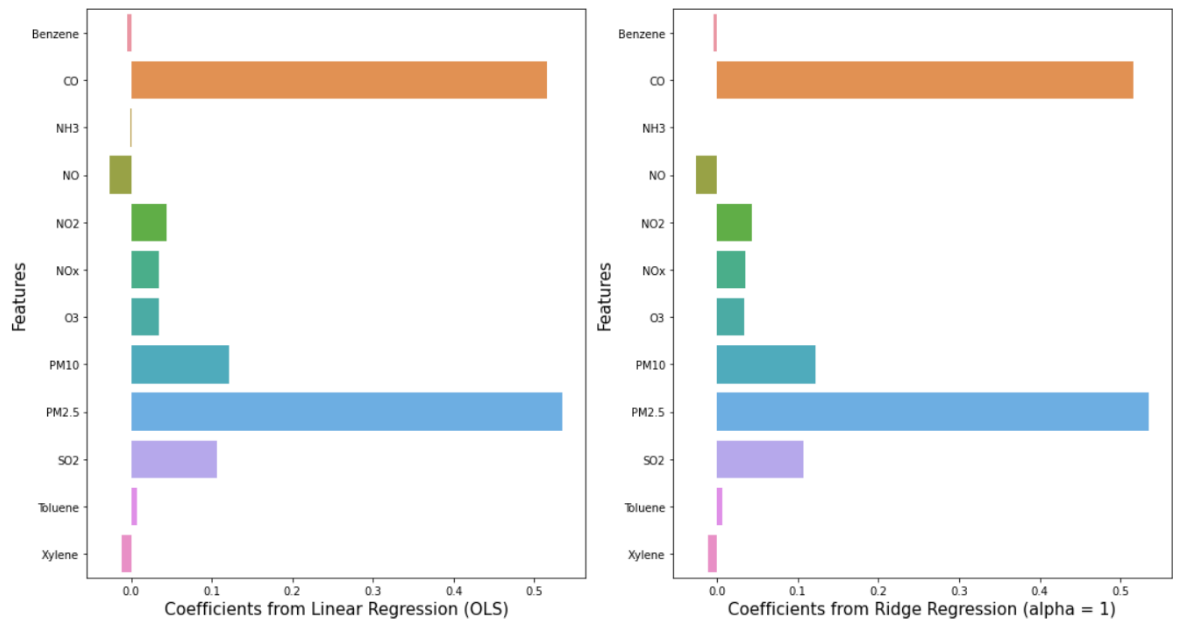
```
RMSE on train set: 0.4088

Score of the ridge model on test set 0.8481722865159047

RMSE on test set: 0.3891
```

After applying the ridge regression with alpha equal to one, we get 0.3891 as the RMSE value.

<u># Visualize the change in values of coefficients obtained from MLR_model (using OLS) and ridge_model:</u>

Then we use subplot to plot the coefficients from linear regression (OLS) and coefficients from ridge regression (alpha = 1).

Subplots method **provides a way to plot multiple plots on a single figure**. Given the number of rows and columns, it returns a tuple (fig, ax), giving a single figure with an array of axes. 4 subplots in a single figure.



The coefficients obtained from ridge regression have smaller values as compared to the coefficients obtained from linear regression using OLS.

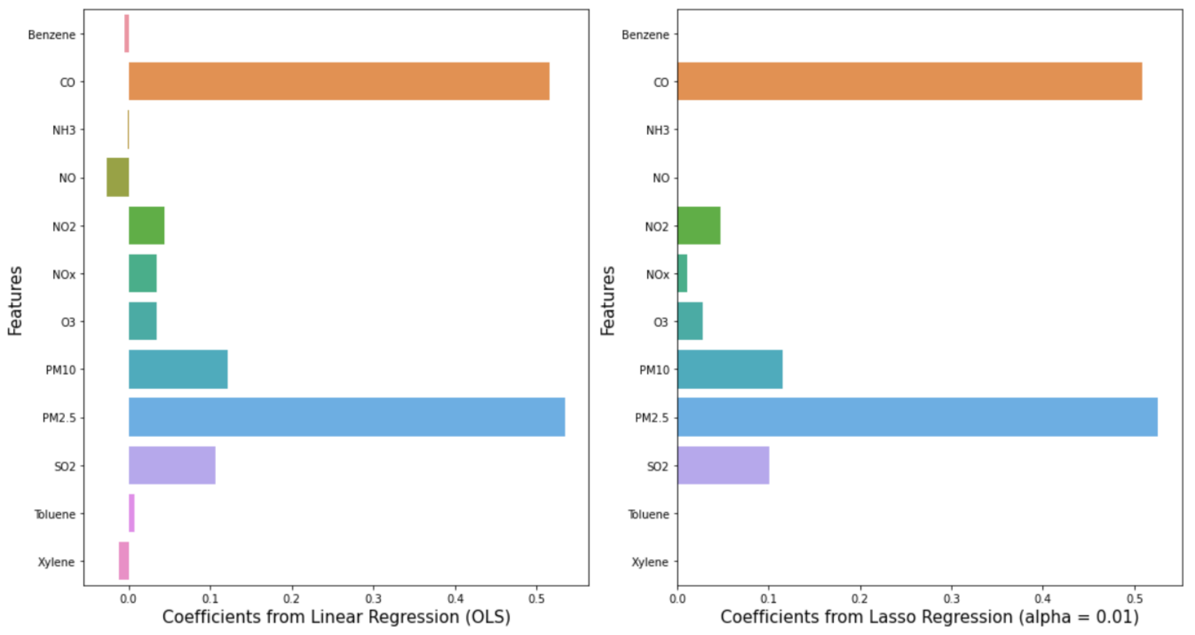
# <u>Lasso Regression</u>: Lasso regression shrinks the less important variable's coefficient to zero which makes this technique more useful when we are dealing with a large number of variables. It is a type of regularization technique that uses L1 norm for regularization.

We are using lasso () to perform lasso regression then, we print RMSE for train set and call the function 'get_train_rmse' as

```
Score of the lasso model on training set: 0.8324370463751743

RMSE on train set: 0.4096

Score of the lasso model on test set 0.8479104933045822

RMSE on test set: 0.3895
```

After applying the lasso regression with alpha equal to 0.01, the RMSE value is **0.3895.**

<u># Visualize the change in values of coefficients obtained from MLR_model (using OLS) and lasso_model:</u>



The second subplot (on the right) shows that the lasso regression has reduced the coefficients of some variables to zero.

Let us print the list of variables with zero coefficient and we get:

Insignificant variables obtained from Lasso Regression when alpha is **0.01**

['NO', 'NH3', 'Benzene', 'Toluene', 'Xylene']

# Elastic Net Regression:

Elastic net linear regression uses the penalties from both the lasso and ridge techniques to regularize regression models. The technique combines both the lasso and ridge regression methods by learning from their shortcomings to improve the regularization of statistical modelsElastic net linear regression uses the penalties from both the lasso and ridge techniques to regularize regression models. The technique combines both the lasso and ridge regression methods by learning from their shortcomings to improve the regularization of statistical models.

Using ElasticNet () to perform Elastic Net regression and then get the

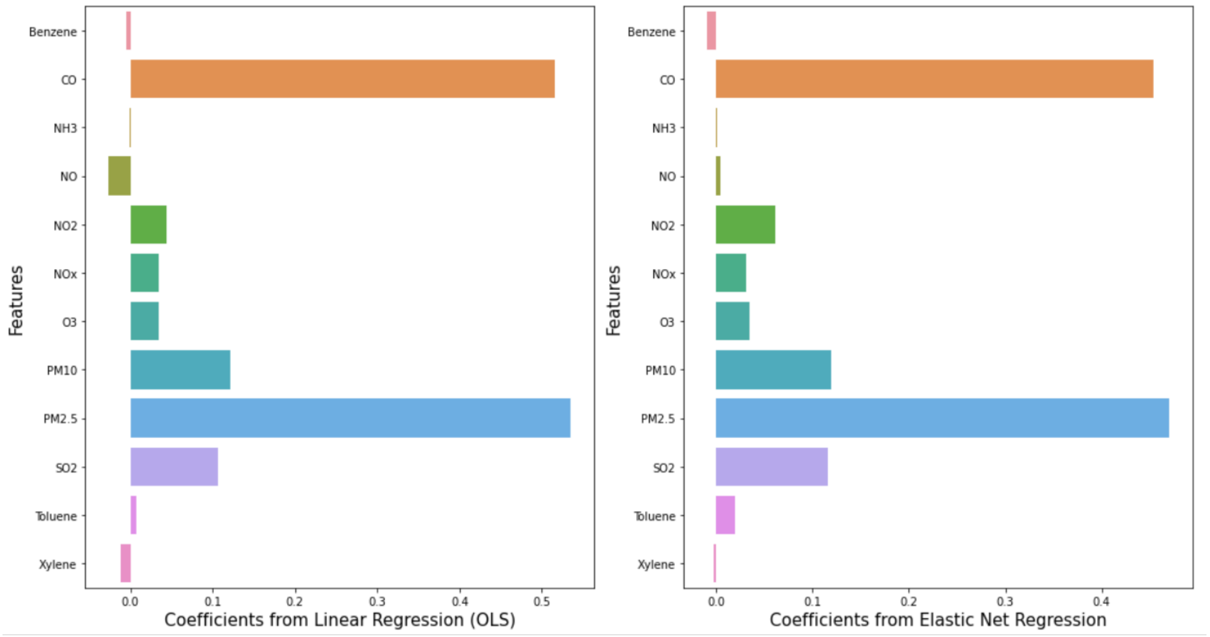
Score of the ElasticNet model on training set and also print RMSE for train set

```
Score of the ElasticNet model on training set: 0.827431127695652

RMSE on train set: 0.4156

and same for the test set we get

Score of the ElasticNet model on test set 0.8404756624518023

RMSE on test set: 0.3989
```

So, with the elastic-net regression, we get **0.3989** as the RMSE value.

# Visualize the change in values of coefficients obtained from MLR_model (using OLS) and Elastic Net regression:

Then we use subplot to plot the coefficients from linear regression (OLS) and coefficients from ridge regression (alpha = 1).



# GridSearchCV:

Hyperparameters are the parameters in the model that are present by the user. GridSearch considers all the combinations of hyperparameters and returns the best hyperparameter values. Following are some of the parameters that GridSearchCV takes:

1. estimator: pass the machine learning algorithm model
2. param_grid: takes a dictionary having parameter names as keys and list of parameters as values
3. cv: number of folds for k-fold cross validation

Then we Find optimal value of alpha for Ridge Regression and get the get the best parameters:

Best parameters for Ridge Regression: {'alpha': 100}.

And also print the RMSE for test and train set:

```
Score of the ridge_grid model on training set: 0.8330520411412479

RMSE on train set: 0.4088

Score of the ridge_grid model on test set 0.8480292945986232

RMSE on test set: 0.3893
```

Finally With the optimal value of alpha that we got from GridSearchCV, the RMSE of the test set is **0.3893.**

#Find optimal value of alpha & l1_ratio for Elastic Net Regression:

create a dictionary with hyperparameters and its values and get the best parameters for Elastic Net Regression: output

Best parameters for Elastic Net Regression: {'alpha': 0.001, 'l1_ratio': 1}

And also print the RMSE for test and train set:

```
Score of the enet_grid model on training set: 0.8330552265259593

RMSE on train set: 0.4088

Score of the enet_grid model on test set 0.8482933397855206

RMSE on test set: 0.389
```

Finally With the optimal value of alpha that we got from GridSearchCV, the RMSE of the test set is **0.389**.

# Decision Tree Regressor Model:

Decision Tree is a decision-making tool that uses a flowchart-like tree structure or is a model of decisions and all of their possible results, including outcomes, input costs, and utility.

Decision-tree algorithms fall under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables.

Then checking the dimensions of the train & test subset and we get:

X_train_dt: (17348, 12)

y_train_dt: (17348,)

X_test_dt: (7436, 12)

y_test_dt: (7436,)

Then we create the Decision tree as DTR_model and get the RMSE value as :

```
RMSE on train set: 0.2727

RMSE on test set: 0.383
```

#Then we create second model in decision tree as DTR_model_sig and get the RMSE value as:

```
RMSE on train set: 0.2727
RMSE on test set: 0.3827
```

Then fit the model on tuned parameters and get the best parameters:

```
Best parameters for decision tree classifier: {'criterion':
'friedman_mse', 'max_depth': 7}
```

#Then we have third model in Decision tree as dt_model_rfe and get the RMSE value as:

```
RMSE on train set: 0.2753

RMSE on test set: 0.3778
```

#Random Forest Regressor Model:

Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap and Aggregation, commonly known as bagging. Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap and Aggregation, commonly known as bagging.

Then we create the Random Forest mode as RFR_model and the get the RMSE value as:

```
RMSE on train set: 0.1177

RMSE on test set: 0.3139
```

After that we fit the model on tuned parameters and get the best parameters:

```
Best parameters for Random Forest Regressor: {'n_estimators': 140}
```

#Then we create the second model in Random Forest Regressor as RFR_model_sig and get the RMSE value as:

```
RMSE on train set: 0.1287

RMSE on test set: 0.3201
```

#Then we have third model in Random Forest Regressor as RFR_model_rfe and get the RMSE value as:

```
RMSE on train set: 0.1278

RMSE on test set: 0.3145
```

#Then we have forth model in Random Forest Regressor as RFR_grid and get the RMSE value as:

```
RMSE on train set: 0.2895

RMSE on test set: 0.3349
```

#AdaBoost Regressor:

An AdaBoost regressor is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction.

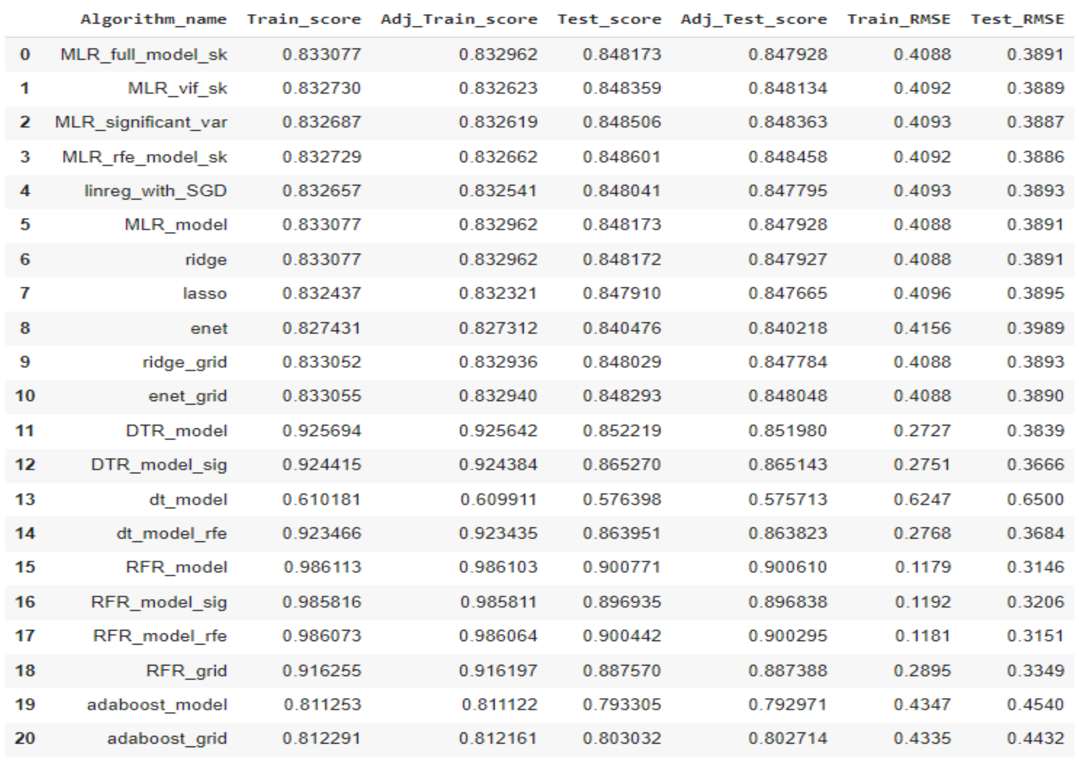
#Then we create a model with the `AdaBoostRegressor and get the RMSE as:`

```
RMSE on train set: 0.4347
RMSE on test set: 0.454
```

# After that we fit the model on tuned parameters and get the best parameters:

```
Best parameters for Ridge Regression: {'loss': 'exponential',
'n_estimators': 9}
```

# ALL MODELS COMPARISON:



	Algorithm_name	Train_score	Adj_Train_score	Test_score	Adj_Test_score	Train_RMSE	Test_RMSE
0	MLR_full_model_sk	0.833077	0.832962	0.848173	0.847928	0.4088	0.3891
1	MLR_vif_sk	0.832730	0.832623	0.848359	0.848134	0.4092	0.3889
2	MLR_significant_var	0.832687	0.832619	0.848506	0.848363	0.4093	0.3887
3	MLR_rfe_model_sk	0.832729	0.832662	0.848601	0.848458	0.4092	0.3886
4	linreg_with_SGD	0.832657	0.832541	0.848041	0.847795	0.4093	0.3893
5	MLR_model	0.833077	0.832962	0.848173	0.847928	0.4088	0.3891
6	ridge	0.833077	0.832962	0.848172	0.847927	0.4088	0.3891
7	lasso	0.832437	0.832321	0.847910	0.847665	0.4096	0.3895
8	enet	0.827431	0.827312	0.840476	0.840218	0.4156	0.3989
9	ridge_grid	0.833052	0.832936	0.848029	0.847784	0.4088	0.3893
10	enet_grid	0.833055	0.832940	0.848293	0.848048	0.4088	0.3890
11	DTR_model	0.925694	0.925642	0.852219	0.851980	0.2727	0.3839
12	DTR_model_sig	0.924415	0.924384	0.865270	0.865143	0.2751	0.3666
13	dt_model	0.610181	0.609911	0.576398	0.575713	0.6247	0.6500
14	dt_model_rfe	0.923466	0.923435	0.863951	0.863823	0.2768	0.3684
15	RFR_model	0.986113	0.986103	0.900771	0.900610	0.1179	0.3146
16	RFR_model_sig	0.985816	0.985811	0.896935	0.896838	0.1192	0.3206
17	RFR_model_rfe	0.986073	0.986064	0.900442	0.900295	0.1181	0.3151
18	RFR_grid	0.916255	0.916197	0.887570	0.887388	0.2895	0.3349
19	adaboost_model	0.811253	0.811122	0.793305	0.792971	0.4347	0.4540
20	adaboost_grid	0.812291	0.812161	0.803032	0.802714	0.4335	0.4432

#BASED ON ALL THE PARAMETERS, WE HAVE SELECTED TUNED RANDOM FOREST REGRESSOR MODEL (RFR_grid).

By comparing we see **RFR_grid** to be the most suitable model, as it is not much overfitted or underfitted.

Also RMSE values on test and train are less with close values.

We selected our final model to be as a **"Tuned Random Forest Regressor"**.

#Best Model deployed with Pickle :

- The pickle module keeps track of the objects it has already serialized, so that later references to the same object won't be serialized again, thus allowing for faster execution time.
- Allows saving models in very little time.
- Good For small models with fewer parameters like the one we used.

Then we save the model (RFR_grid) and load the data with pickle and get the pickle model score as :

```
(0.9151630765684426, 0.8878093191608822)
```

Due to the time complexity involved in training large models, saving is becoming a crucial part of the data-science realm. The process works on the concept of serialization (saving of data into its component form) and deserialization (restoring of data from the serialized chunks).

# CONCLUSION:

- The analytical procedure began with data cleaning and processing, missing/null records, detailed evaluation and in the end, model constructing and evaluation.
- Since our model is capable of predicting the current data with an 89% score, it will successfully predict the upcoming air quality index of any particular data within a given region. With this model we can forecast the AQI and alert the respected region of the country.
- From all the models built above, we have concluded Random Forest Regressor to be the best fitted model as we do not see any underfitting and overfitting condition.
- From the predicted values we can observe AQI for 2020 is predicted very high based on previous data, but actual values of AQI are very less in 2020. The reason for decrease might be the pandemic which leads to lockdown and hence decrease in pollutants.
- The range of RMSE values for train (0.2895) and test (0.3349) are very less which indicates minimal error while predicting the target variable.
- The target variable AQI is most affected by PM 2.5 and PM 10.
- Dataset shows Delhi is the most polluted city among others due to the presence of vehicles whereas Ahmedabad is polluted due to the presence of Industrial emission.

# CHALLENGES:

- Prediction of air quality is a challenging task because of the dynamic environment, unpredictability, and variability in space and time of pollutants. The grave consequences of air pollution on humans, animals, plants, monuments, climate, and environment call for consistent air quality monitoring and analysis, especially in developing countries.
- In this model the data is static. However, the government updates the data hourly. For better performance, real time data analysis using the cloud can produce better results.

# REFERENCES:

- https://www.ripublication.com/ijaerspl2019/ijaerv14n11spl_34.pdf
- https://www.geeksforgeeks.org/predicting-air-quality-index-using-python/
- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
- Python Machine Learning Decision Tree (w3schools.com)
- https://www.pranaair.com/blog/what-is-air-quality-index-aqi-and-its-calculation/
- https://cpcb.nic.in/
- https://en.wikipedia.org/wiki/Air_pollution_in_India
- https://app.cpcbccr.com/ccr_docs/FINAL-REPORT_AQI_.pdf
- https://airquality.cpcb.gov.in/AQI_India/