PROJECT

16:198:512:03

REGRESSION & TIME SERIES

AQI Trends: Patterns, Anomalies, and Environmental Insights

AUTHORS	**NetID**
Adish Golechha	ag2384
FNU Vasureddy	fv121
Pradhyumna Kasula	pk732
Shreyash Shreedhar Kalal	ssk241
Taniya Satheesh Kulkarni	tk740

Thursday 14th December, 2023

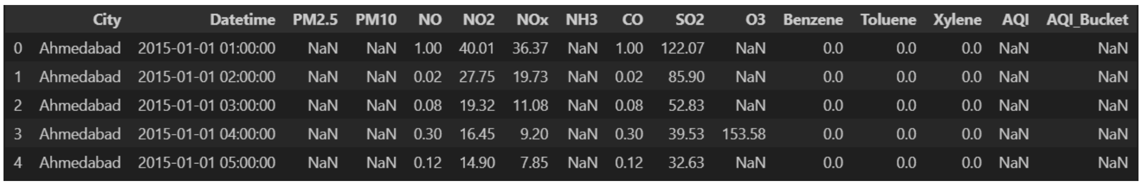
TABLE OF CONTENTS

1 INTRODUCTION

1.1 About Data-set

This dataset purports to give extensive air quality information for numerous cities in India, comprising daily data. It contains the AQI (Air Quality Index), a critical indication of air pollution levels. This information might be useful for assessing air quality trends and locating pollution hot spots.
We have extracted this data from the Central Pollution Control Board (CPCB) [1] for the years 2015-2020.

1.1.1 Data Overview



	City	Datetime	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket
0	Ahmedabad	2015-01-01 01:00:00	NaN	NaN	1.00	40.01	36.37	NaN	1.00	122.07	NaN	0.0	0.0	0.0	NaN	NaN
1	Ahmedabad	2015-01-01 02:00:00	NaN	NaN	0.02	27.75	19.73	NaN	0.02	85.90	NaN	0.0	0.0	0.0	NaN	NaN
2	Ahmedabad	2015-01-01 03:00:00	NaN	NaN	0.08	19.32	11.08	NaN	0.08	52.83	NaN	0.0	0.0	0.0	NaN	NaN
3	Ahmedabad	2015-01-01 04:00:00	NaN	NaN	0.30	16.45	9.20	NaN	0.30	39.53	153.58	0.0	0.0	0.0	NaN	NaN
4	Ahmedabad	2015-01-01 05:00:00	NaN	NaN	0.12	14.90	7.85	NaN	0.12	32.63	NaN	0.0	0.0	0.0	NaN	NaN

Figure 1: Data Overview

REGRESSORS

1. Cities: 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
2. Date: Includes all dates from 2015 to 2020
3. PM2.5: Particulate Matter 2.5-micrometer in μg / m^3
4. PM10: Particulate Matter 10-micrometer in μg / m^3
5. NO: Nitric Oxide in μg / m^3
6. NO2: Nitric Dioxide in μg / m^3
7. NOx: Any Nitric x-oxide in ppb
8. NH3: Ammonia in μg / m^3
9. CO: Carbon Monoxide in μg / m^3
10. SO2: Sulphur Dioxide in μg / m^3
11. O3: Ozone in μg / m^3
12. Benzene: Benzene in μg / m^3
13. Toulene: Toluene in μg / m^3
14. Xylene: Xylene in μg / m^3

PREDICTED VALUES

Air Quality Index (AQI)

How is the AQI calculated?

The AQI primarily makes use of seven key measures: PM2.5, PM10, SO2, NOx, NH3, CO, and O3.

- Variables:
  - PM2.5, PM10, SO2, NOx, NH3
    The average value for 24 hours is calculated and used (provided at least 16 values are available)
  - CO and O3
    A maximum value of 8 hours is used
- Sub-Index Calculation:
  The measures are converted into sub-indexes based on pre-defined groups
- Final AQI Determination:
  - The final AQI is the maximum sub-index
  - Minimum requirement: at least one of PM2.5 and PM10 should be available, and a minimum of three out of the seven measures should be available overall.
  - Other pollutants such as Toluene, Benzene, Xylene, etc., are used to evaluate based on their presence in specific locations.

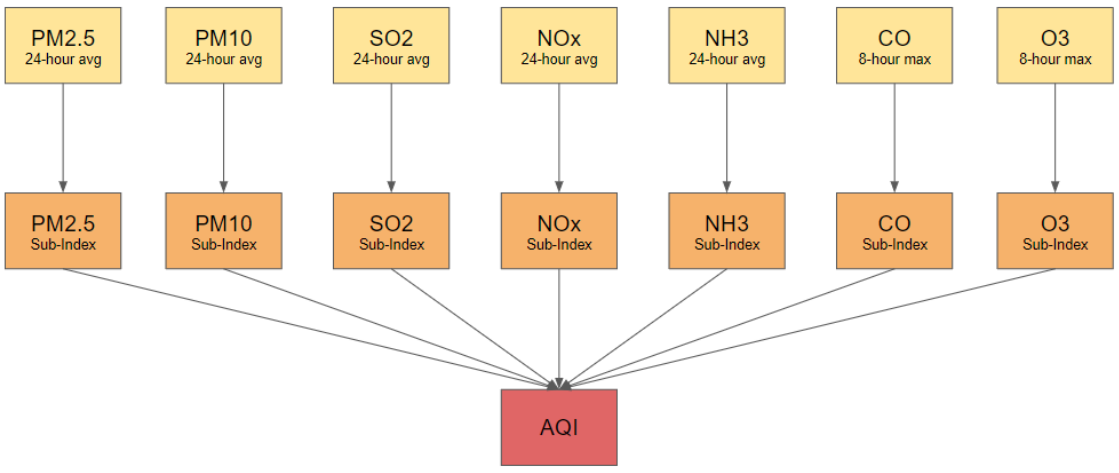


Figure 2: Visual depiction of how AQI is calculated

1.2 Objective

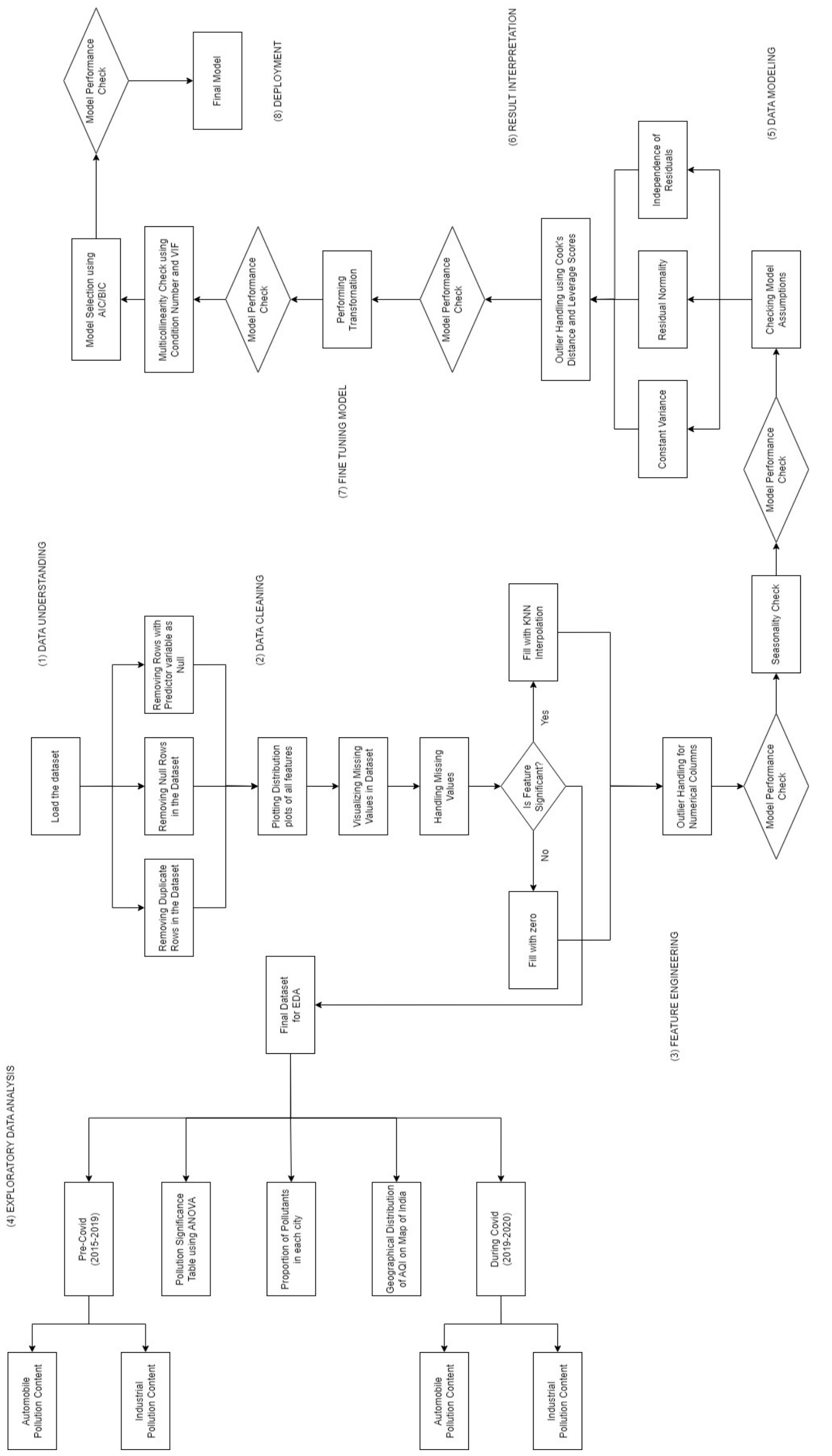

(1) DATA UNDERSTANDING
Load the dataset
Removing Duplicate Rows in the Dataset
Removing Null Rows in the Dataset
Removing Rows with Predictor variable as Null
(2) DATA CLEANING
Plotting Distribution plots of all features
Visualizing Missing Values in Dataset
Handling Missing Values
Is Feature Significant?
Yes
No
Fill with KNN Interpolation
Fill with zero
(3) FEATURE ENGINEERING
Outlier Handling for Numerical Columns
Model Performance Check
Final Dataset for EDA
(4) EXPLORATORY DATA ANALYSIS
Pre-Covid (2015-2019)
Pollution Significance Table using ANOVA
Proportion of Pollutants in each city
Geographical Distribution of AQI on Map of India
During Covid (2019-2020)
Automobile Pollution Content
Industrial Pollution Content
Automobile Pollution Content
Industrial Pollution Content
(5) DATA MODELING
Seasonality Check
Model Performance Check
Checking Model Assumptions
Constant Variance
Residual Normality
Independence of Residuals
(6) RESULT INTERPRETATION
Outlier Handling using Cook's Distance and Leverage Scores
Model Performance Check
(7) FINE TUNING MODEL
Performing Transformation
Model Performance Check
Multicollinearity Check using Condition Number and VIF
Model Selection using AIC/BIC
(8) DEPLOYMENT
Model Performance Check
Final Model

Figure 3: Workflow

2 MISSING VALUE ANALYSIS

2.1 EDA on Original Data

For the given data we have generated a heat map and the matrix to find the missing values in the data using the missingno[2] package

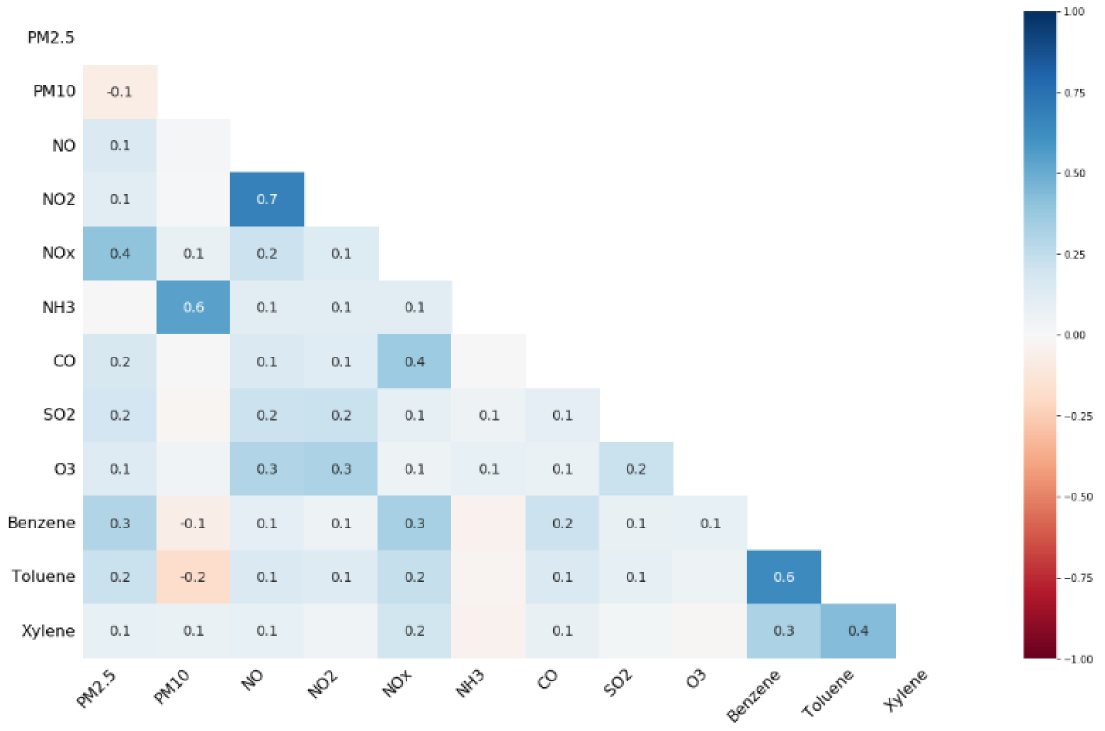


Figure 4: Heat-Map

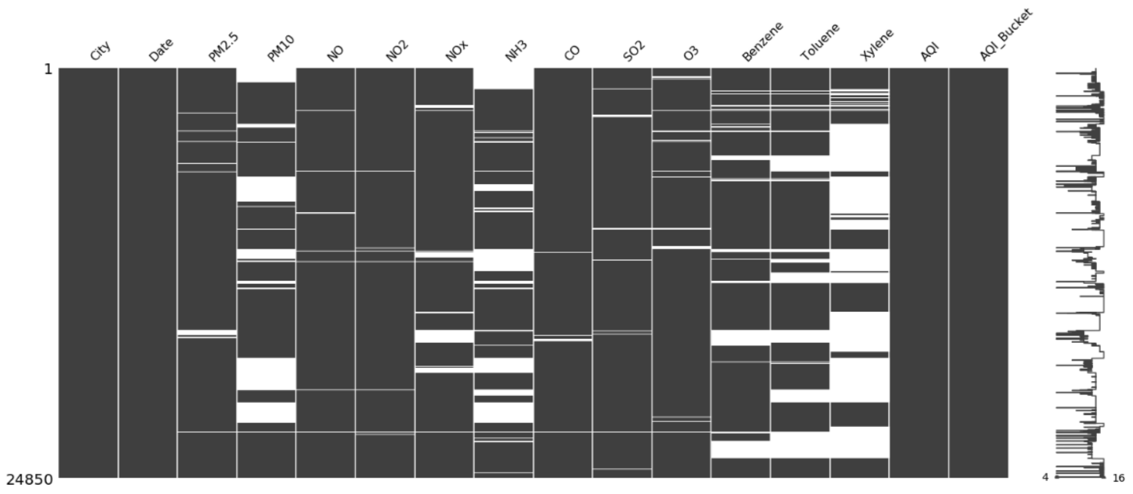
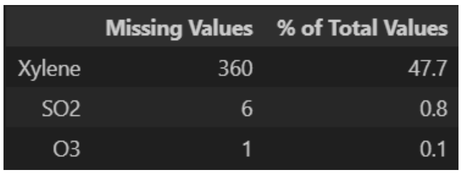


Figure 5: Matrix

2.2 Treating Missing Values

1. Drop rows with all null values
2. Drop all duplicate rows
3. Drop rows where AQI is NaN
4. Split data frame based on cities
5. For each city do the following:
   - Generate a missing values table



	Missing Values	% of Total Values
Xylene	360	47.7
SO2	6	0.8
O3	1	0.1

Figure 6: Missing values table for Kolkata

   - If a column has 100% missing values, we drop the column, concluding that the variable does not have any impact on the AQI
   - We eliminate all rows from the city data frame where the hypothesised variable is null
   - Upon dropping, we then train the model using linear regression on this data
   - Once this is complete, we proceed to perform hypothesis testing to see the significance of that feature on the model.
   - If the hypothesis testing results in a p-value < 0.05, we reject the null hypothesis i.e., the parameter has some significance.
     Otherwise, we fail to reject the null hypothesis.
   - If the parameter has some significance, we perform KNN imputation[3] on the missing values.
     If not, we set the NaN values to 0 in the column.

Reason: We do the above steps to see if the parameter holds any significance in calculating the AQI for a given city.
To streamline and simplify the process, we've tried to automate the above process across all cities[4].

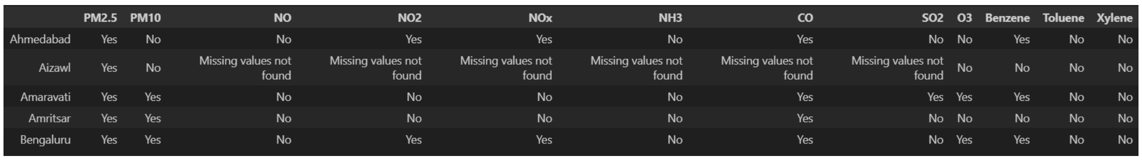
```
    def missing_values_table(df):
        # Total missing values
        mis_val = df.isnull().sum()

        # Percentage of missing values
        mis_val_percent = 100 * df.isnull().sum() / len(df)

        # Make a table with the results
        mis_val_table = pd.concat([mis_val, mis_val_percent], axis=1)

        # Rename the columns
        mis_val_table_ren_columns = mis_val_table.rename(
```

```python
        columns = {0 : 'Missing Values', 1 : '% of Total Values'})

        # Sort the table by percentage of missing descending
        mis_val_table_ren_columns = mis_val_table_ren_columns[
            mis_val_table_ren_columns.iloc[:,1] != 0].sort_values(
        '% of Total Values', ascending=False).round(1)

        # Print some summary information
        print ("Your selected dataframe has " + str(df.shape[1]) + " columns.\n"
            "There are " + str(mis_val_table_ren_columns.shape[0]) +
              " columns that have missing values.")

        # Return the dataframe with missing information
        return mis_val_table_ren_columns
```

Listing 1: Function to generate missing values table

```python
#Master Function
city_list = list(df['City'].unique())
column_list = list(df.columns)
grouped_df = df.groupby('City')
category_dfs = {name: group for name, group in grouped_df}

for i in city_list:
    city_df = category_dfs[i]
    iter = missing_values_table(city_df)
    missing_value_table_index_list = list(iter.index)

    useless_columns=[]
    for k in range(len(iter['% of Total Values'])):
        if iter['% of Total Values'][k]==100:
            useless_columns.append(iter.index[k])

    if useless_columns:
        for l in useless_columns:
            city_df = city_df.drop(columns=[l])
            significant_df[l][i]='No'
    else:
        pass

    for j in missing_value_table_index_list:
        if j not in useless_columns:
            city_column_df = city_df.dropna(subset=[j])
            city_column_df = city_column_df.drop(columns=['City', 'Date', 'AQI_Bucket'])
            city_column_df = city_column_df.fillna(0)

            X = city_column_df.drop(columns=['AQI'])
            Y = city_column_df['AQI']

            X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

            # Create a linear regression model
            model = LinearRegression()

            # Fit the model on the training data
            model.fit(X_train, y_train)

            X_train_with_const = sm.add_constant(X_train)

            # Fit a linear regression model using statsmodels
            model_statsmodels = sm.OLS(y_train, X_train_with_const).fit()

            #Perform the Hypothesis Test
            # Define the hypothesis matrix
            hypothesis_matrix = j+' = 0'

            # Perform the Wald test
            wald_test_result = model_statsmodels.wald_test(hypothesis_matrix)
            p_value = wald_test_result.pvalue.item()

            if p_value <= 0.05:
                significant_df[j][i]='Yes'
            else:
                significant_df[j][i]='No'
```

Listing 2: Function to generate significance matrix for each combination of city and feature



	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene
Ahmedabad	Yes	No	No	Yes	Yes	No	Yes	No	No	Yes	No	No
Aizawl	Yes	No	Missing values not found	Missing values not found	Missing values not found	Missing values not found	Missing values not found	Missing values not found	No	No	No	No
Amaravati	Yes	Yes	No	No	No	No	Yes	Yes	Yes	Yes	No	No
Amritsar	Yes	Yes	No	No	No	No	Yes	No	No	No	No	No
Bengaluru	Yes	Yes	No	Yes	Yes	No	Yes	No	Yes	Yes	No	No

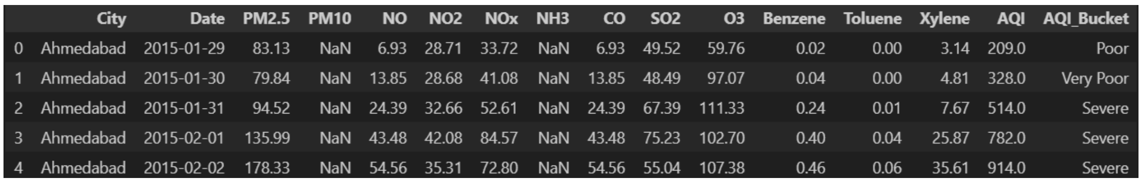
Figure 7: Generated significance table

CODE TO FILL MISSING VALUES

```python
def fill_insignificant(df, significant_matrix):
    for city in city_list:
        y_features = []
        n_features = []

        for i in significant_matrix.columns:
            if significant_matrix[i][city] == 'Yes':
                y_features.append(i)
            elif significant_matrix[i][city] == 'No':
                n_features.append(i)

        df.loc[df['City'] == city, n_features] = df.loc[df['City'] == city, n_features].fillna(0)

        city_data = df[df['City'] == city][['Date'] + y_features].copy()
        city_data.set_index('Date', inplace=True)

        knn_imputer = KNNImputer(n_neighbors=5)

        city_data_interpolated = knn_imputer.fit_transform(city_data)
        df.loc[df['City'] == city, y_features] = city_data_interpolated

    return df
```

Listing 3: Code to fill missing values

Here is the cleaned data:



	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket
0	Ahmedabad	2015-01-29	83.13	NaN	6.93	28.71	33.72	NaN	6.93	49.52	59.76	0.02	0.00	3.14	209.0	Poor
1	Ahmedabad	2015-01-30	79.84	NaN	13.85	28.68	41.08	NaN	13.85	48.49	97.07	0.04	0.00	4.81	328.0	Very Poor
2	Ahmedabad	2015-01-31	94.52	NaN	24.39	32.66	52.61	NaN	24.39	67.39	111.33	0.24	0.01	7.67	514.0	Severe
3	Ahmedabad	2015-02-01	135.99	NaN	43.48	42.08	84.57	NaN	43.48	75.23	102.70	0.40	0.04	25.87	782.0	Severe
4	Ahmedabad	2015-02-02	178.33	NaN	54.56	35.31	72.80	NaN	54.56	55.04	107.38	0.46	0.06	35.61	914.0	Severe

Figure 8: Cleaned data

3 EXPLORATORY DATA ANALYSIS (EDA)

We have considered two categories:

- Vehicular Pollution Content
  This is a combination of PM2.5, PM10, NO, NO2, NOx, NH3, and CO.
- Industrial Pollution Content
  This is a combination of SO2, O3, Toluene, Benzene and Xylene.

3.1 Distribution of variables

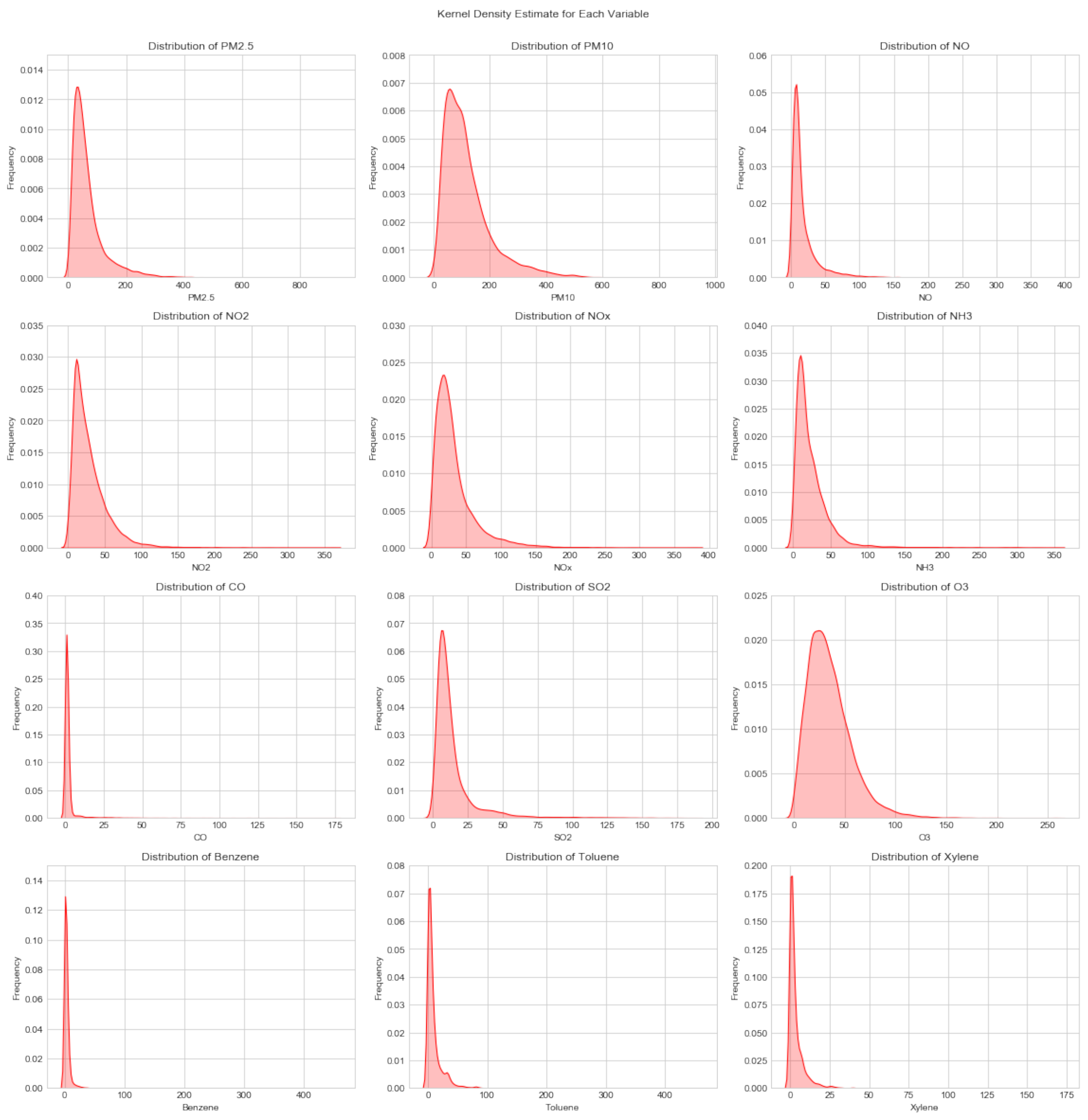
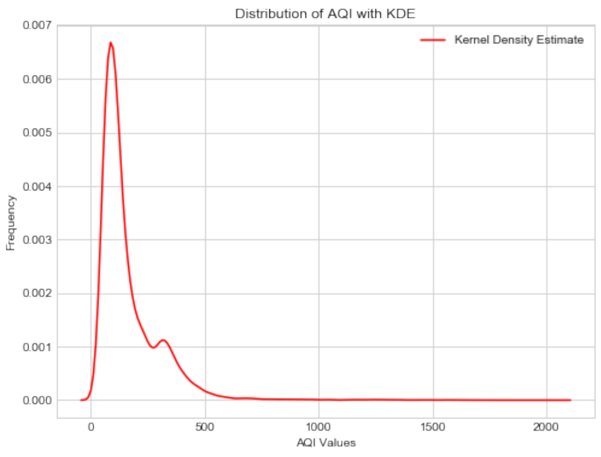
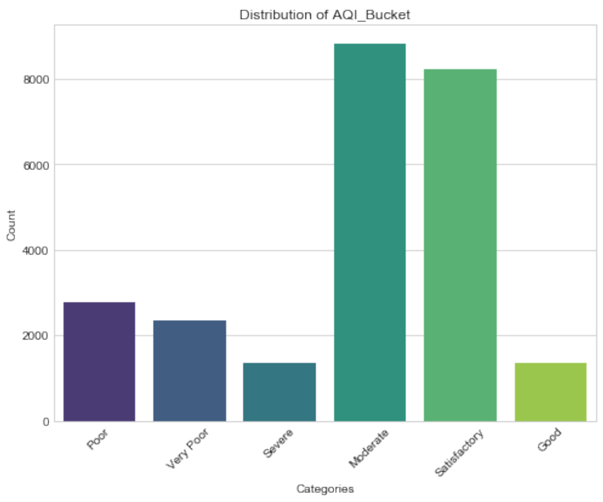


Figure 9: Distribution of all regressors[5]



(a) AQI Distribution



(b) AQI Bucket Distribution

Figure 10: Distribution of AQI and AQI bucket

Our understanding from the above distributions:

1. **PM2.5:** Ranges from 0 to 400 μg / m^3, but a majority of the values lie between 0 to 200 μg / m^3.

2. **PM10:** Ranges from 0 to 600 μg / m^3, but a majority of all the values lie between 0 to 200 μg / m^3. (similar to PM2.5). We also do have a few values that are slightly more than 200 μg / m^3.

3. **NO:** Ranges from 0 to 100 μg / m^3, with a majority of the values lying between 0 and 50 μg / m^3.

4. **NO2:** Similar to NO, this also ranges between 0 to 100 μg / m^3 and peaks between 0 to 50 μg / m^3. We also have a few values over 50 μg / m^3.

5. **NOx:** The NOx has values between 0 to 150 ppb, with the majority of values lying between 0 to 50 ppb. Like NO2, we have a good percentage of values over 50 ppb.

6. **NH3:** This also ranges between 0 to 100 μg / m^3, witch values peaking between 0 to 50 μg / m^3.

7. **CO:** Unlike all the other pollutants, CO seems to be in very limited quantities in the air as it ranges between 0 to 25 μg / m^3. However, the majority of the values are closer to 0 μg / m^3.

8. **SO2:** This ranges mostly between 0 to 50 μg / m^3, with the peak lying between 0 to 25 μg / m^3.

9. **O3:** This has values between 0 to 100 μg / m^3, and peaks between 0 to 50 μg / m^3. There is a good chunk of values between 50 to 100 μg / m^3.

10. **Benzene:** This pollutant has a very small distribution range, with a majority of the values close to 0 μg / m^3.

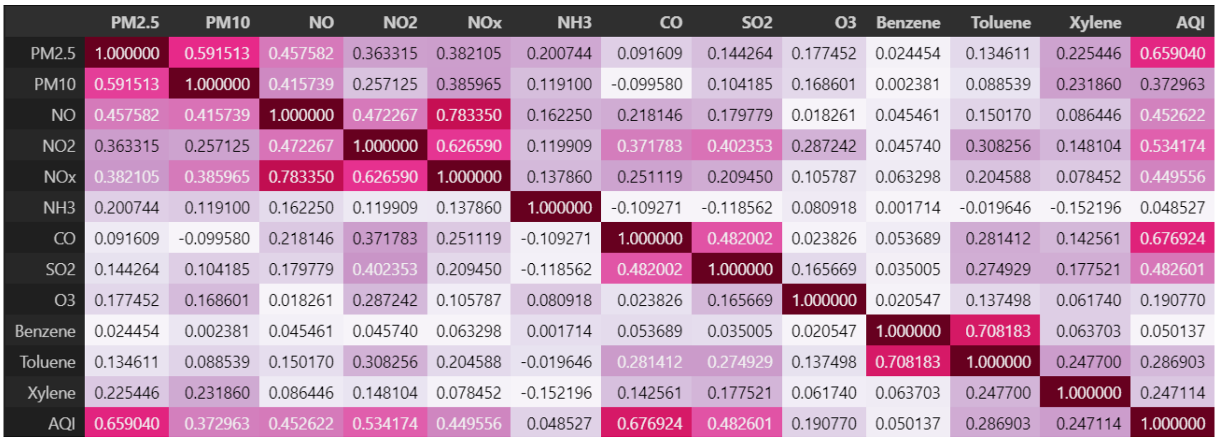
11. **Toluene:** Similar to Benzene, this ranges between 0 to 100 μg / m^3 but peaks with values close to 0 μg / m^3.

12. **Xylene:** This ranges between 0 to 25 μg / m^3, and like Benzene, Toluene and CO, the majority of the values lie close to 0 μg / m^3.

13. **AQI:** The AQI ranges between 0 to 500.

14. **AQI bucket:** The dataset predominantly consists of AQI values categorized as 'Moderate,' totalling more than 8000 values, closely followed by the 'Satisfactory' category.

The concentrations of pollutants such as **Xylene, Toluene and Benzene** in the air are notably low, likely indicating their predominant presence as industrial waste (and are extremely harmful). Certain cities might have 0 μg / m^3 of these pollutants, which we will be checking for in the next section.



	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI
PM2.5	1.000000	0.591513	0.457582	0.363315	0.382105	0.200744	0.091609	0.144264	0.177452	0.024454	0.134611	0.225446	0.659040
PM10	0.591513	1.000000	0.415739	0.257125	0.385965	0.119100	-0.099580	0.104185	0.168601	0.002381	0.088539	0.231860	0.372963
NO	0.457582	0.415739	1.000000	0.472267	0.783350	0.162250	0.218146	0.179779	0.018261	0.045461	0.150170	0.086446	0.452622
NO2	0.363315	0.257125	0.472267	1.000000	0.626590	0.119909	0.371783	0.402353	0.287242	0.045740	0.308256	0.148104	0.534174
NOx	0.382105	0.385965	0.783350	0.626590	1.000000	0.137860	0.251119	0.209450	0.105787	0.063298	0.204588	0.078452	0.449556
NH3	0.200744	0.119100	0.162250	0.119909	0.137860	1.000000	-0.109271	-0.118562	0.080918	0.001714	-0.019646	-0.152196	0.048527
CO	0.091609	-0.099580	0.218146	0.371783	0.251119	-0.109271	1.000000	0.482002	0.023826	0.053689	0.281412	0.142561	0.676924
SO2	0.144264	0.104185	0.179779	0.402353	0.209450	-0.118562	0.482002	1.000000	0.165669	0.035005	0.274929	0.177521	0.482601
O3	0.177452	0.168601	0.018261	0.287242	0.105787	0.080918	0.023826	0.165669	1.000000	0.020547	0.137498	0.061740	0.190770
Benzene	0.024454	0.002381	0.045461	0.045740	0.063298	0.001714	0.053689	0.035005	0.020547	1.000000	0.708183	0.063703	0.050137
Toluene	0.134611	0.088539	0.150170	0.308256	0.204588	-0.019646	0.281412	0.274929	0.137498	0.708183	1.000000	0.247700	0.286903
Xylene	0.225446	0.231860	0.086446	0.148104	0.078452	-0.152196	0.142561	0.177521	0.061740	0.063703	0.247700	1.000000	0.247114
AQI	0.659040	0.372963	0.452622	0.534174	0.449556	0.048527	0.676924	0.482601	0.190770	0.050137	0.286903	0.247114	1.000000

Figure 11: Correlation heat-map for all variables

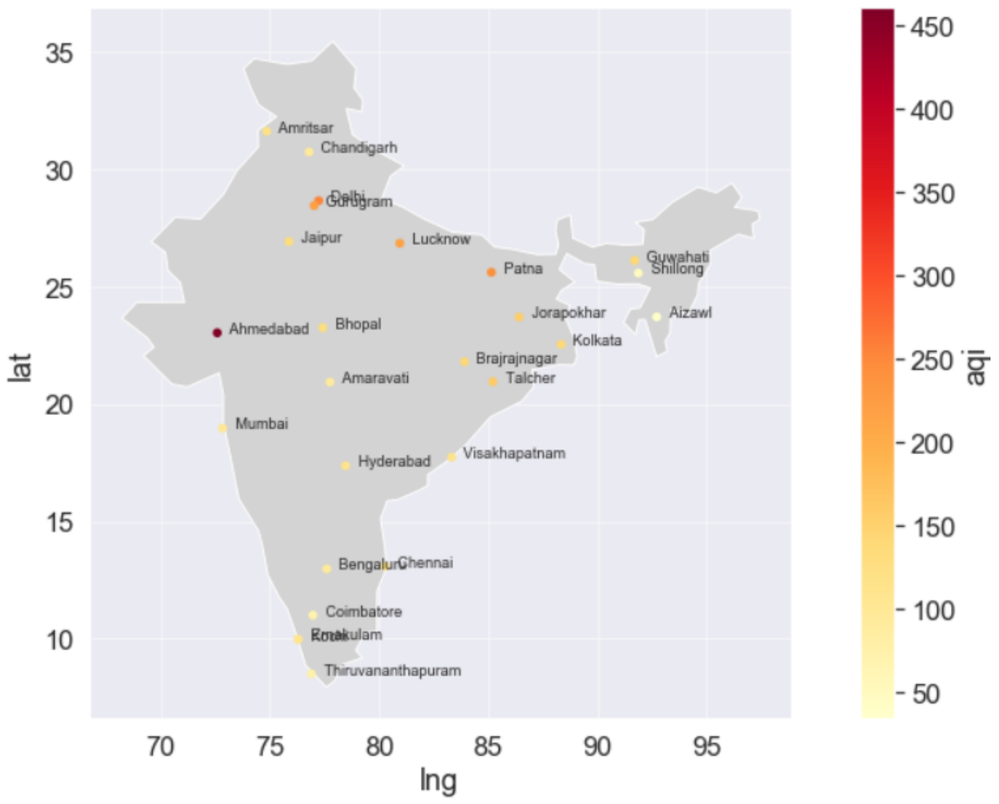


Figure 12: AQI on Indian Map[6]

The following is what we observe from the above image:

- According to the image, Ahmedabad has the highest average Air Quality Index (AQI), whereas Aizawl has the lowest average AQI.
- The northern parts of India contain the cities with higher AQI relative to the southern parts of India
  - Highest AQI based on Region:
    - West India: Ahmedabad
    - East India: Talcher
    - North India: Delhi and Patna
    - South India: Vishakhapatnam
  - Lowest AQI based on Region:
    - West India: Mumbai
    - East India: Aizawl
    - North India: Chandigarh
    - South India: Coimbatore

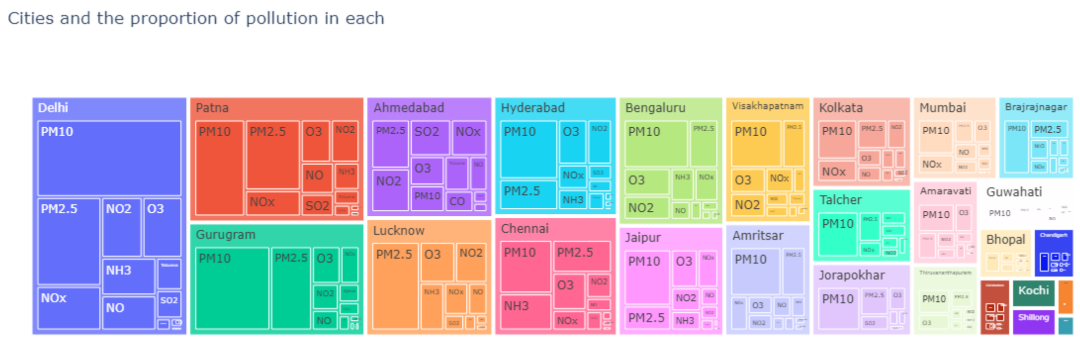


Figure 13: Proportion of pollutants in each city[7]

We see a difference in data set sizes, with Delhi having the most values and Aizawl having the fewest. Furthermore, the data distribution suggests that PM10 contributes the most in majority of the cities, whereas Xylene contributes the least.

3.2 Pre-Covid (2015-2019)

3.2.1 Automobile Pollution Content

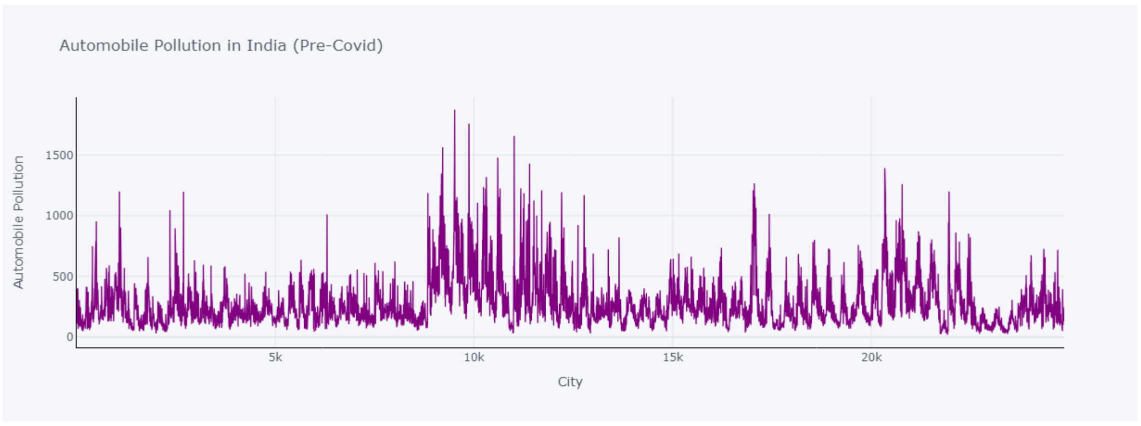


Figure 14: Automobile Pollution Content Distribution

This graph depicts the relationship between automobile pollution levels and different cities in India during the pre-COVID period from 2015 to 2019.

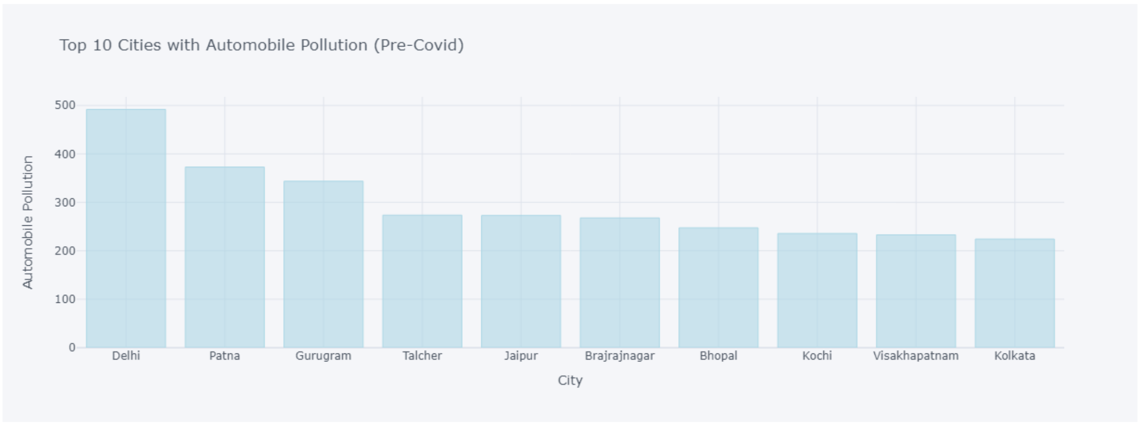


Figure 15: Top Cities with Highest Automobile Pollution

Among all the cities, Delhi experienced the highest levels of automobile pollution before the COVID era, with Patna and Gurugram following closely behind.

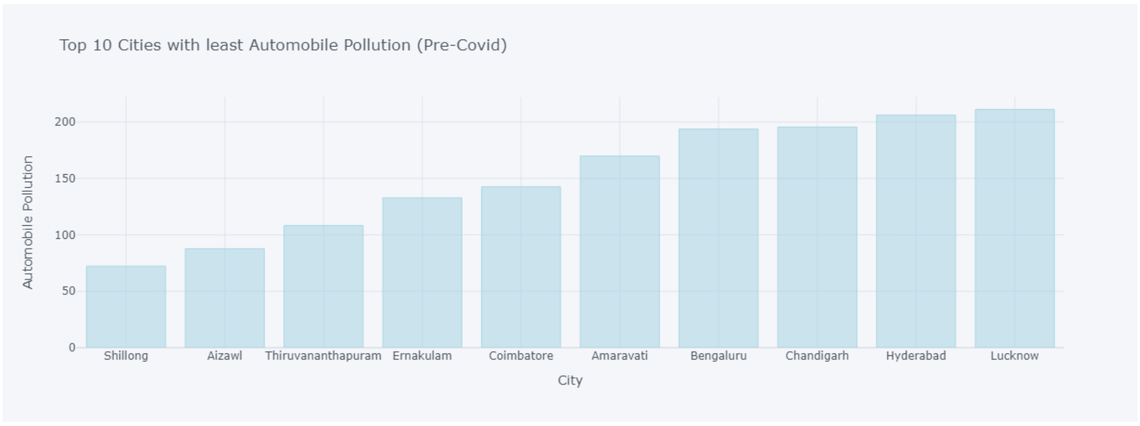


Figure 16: Top Cities with Lowest Automobile Pollution

Shillong possesses the lowest automobile pollution content among all cities.

3.2.2 Industrial Pollution Content

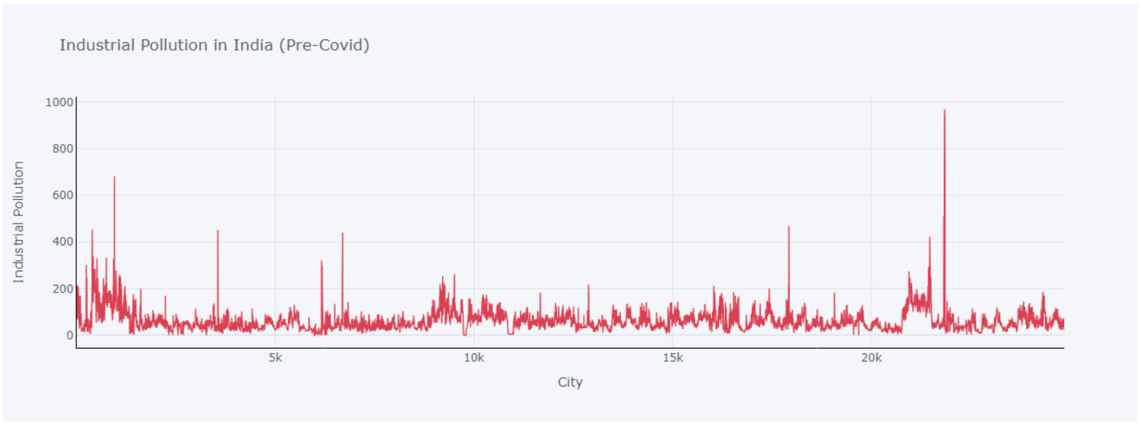


Figure 17: Industrial Pollution Content Distribution

This graph depicts the connection between industrial pollution levels across different cities in India during the pre-COVID period from 2015 to 2019.

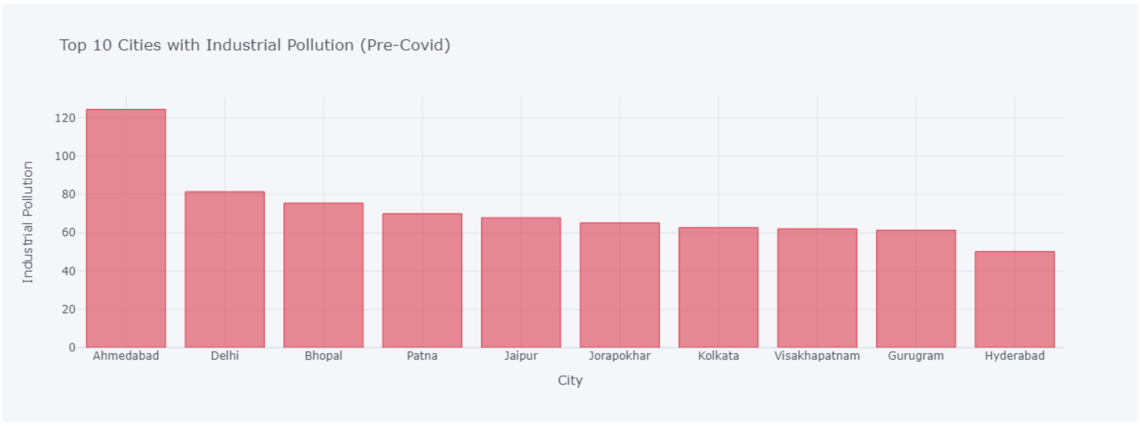


Figure 18: Top Cities with Highest Industrial Pollution

In the top 10 cities, Ahmedabad registered the highest industrial pollution levels before the COVID era, with Delhi and Bhopal closely trailing behind.

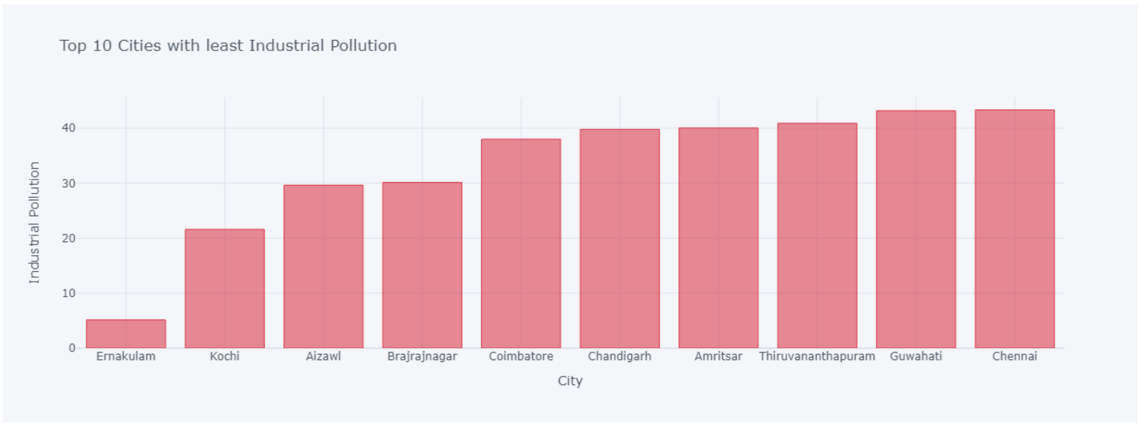


Figure 19: Top Cities with Lowest Industrial Pollution

In the list of the top 10 cities with the least industrial pollution before the COVID era, Ernakulam, Kochi, and Aizawl have secured the top positions.

3.3 During Covid (2019-2020)

3.3.1 Automobile Pollution Content

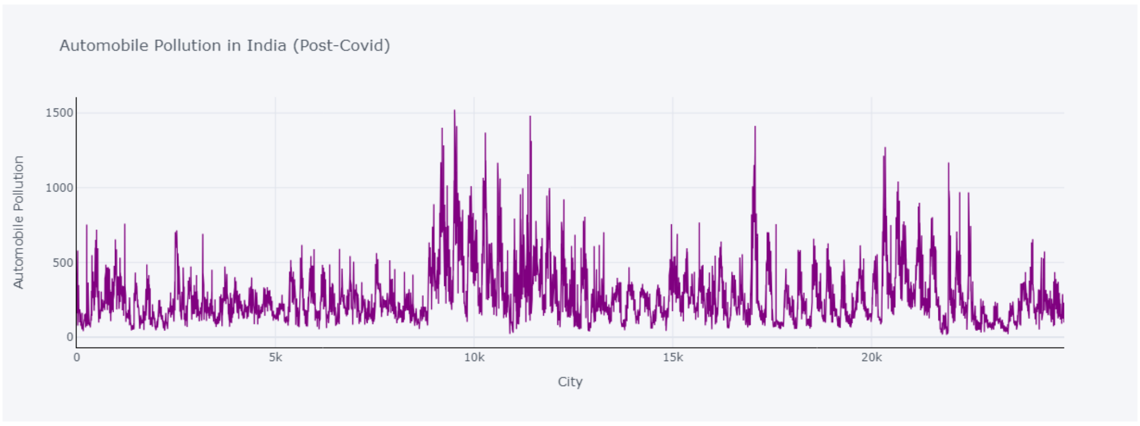


Figure 20: Automobile Pollution Content Distribution

The graph illustrates the distribution of automobile pollution across Indian cities after the onset of COVID-19 in 2020.

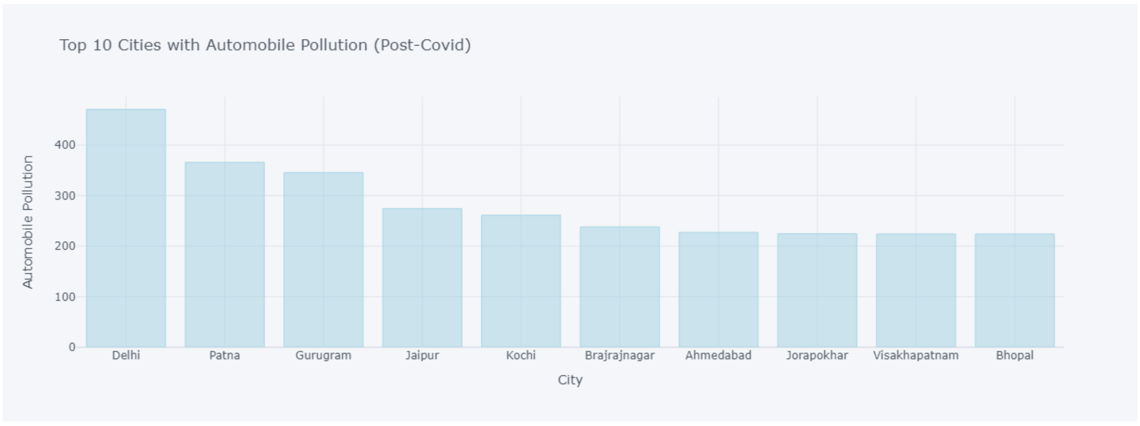


Figure 21: Top Cities with Highest Automobile Pollution

Even after a slight improvement in the Air Quality Index (AQI), Delhi, Patna, and Gurugram continue to hold the leading positions for automobile pollution post-COVID.

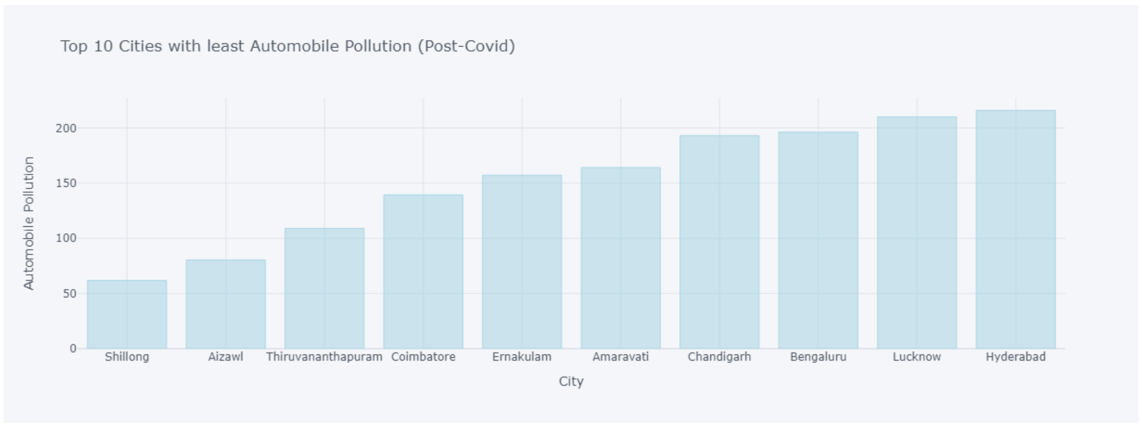


Figure 22: Top Cities with Lowest Automobile Pollution

Ranking first among the ten cities with the least amount of vehicle pollution before the start of COVID-19 are Shillong and Aizawl. Hyderabad has moved up to the tenth spot on this list, taking Lucknow's place.

3.3.2 Industrial Pollution Content

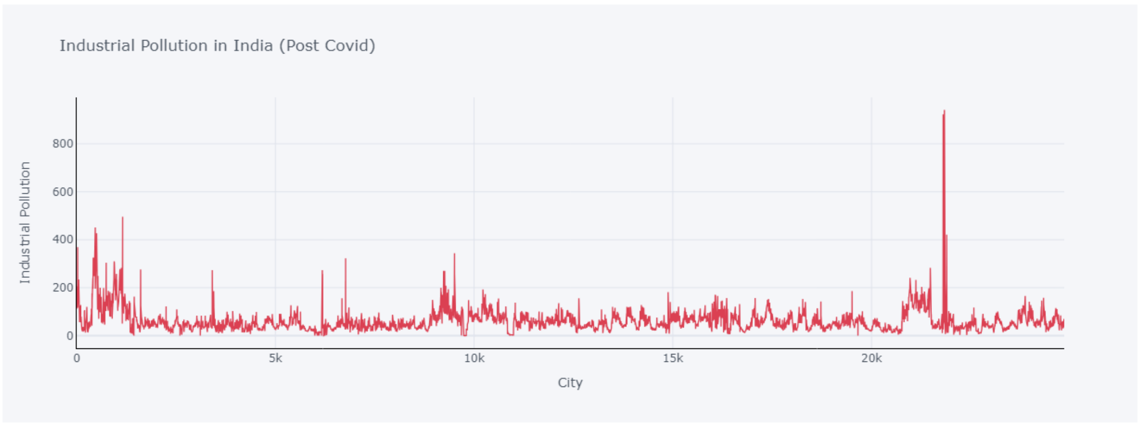


Figure 23: Industrial Pollution Content Distribution

The graph shows how industrial pollution was distributed among cities following the onset of COVID-19 in 2020.

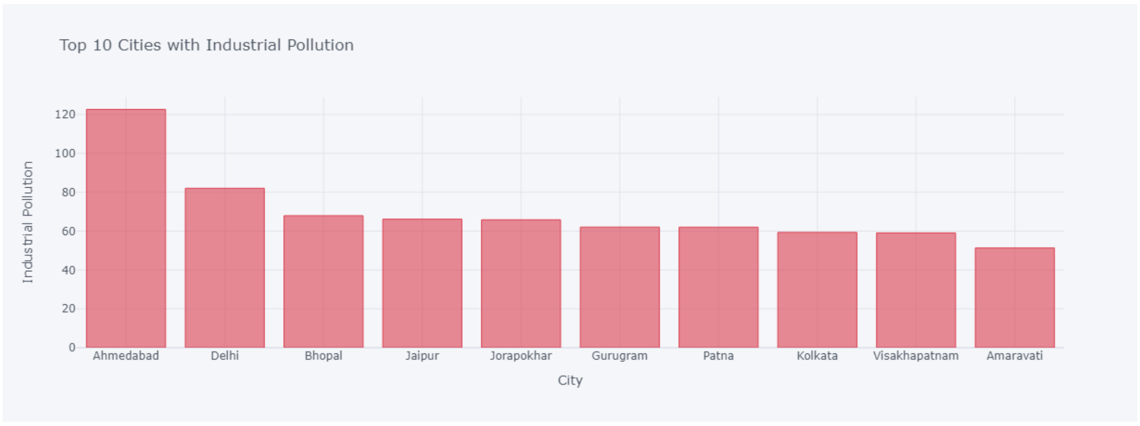


Figure 24: Top Cities with Highest Industrial Pollution

The order of the top ten cities with the highest levels of industrial pollution following the COVID-19 pandemic is in line with the pre-pandemic ranking, with Ahmedabad, Delhi, and Bhopal at the top of the list.

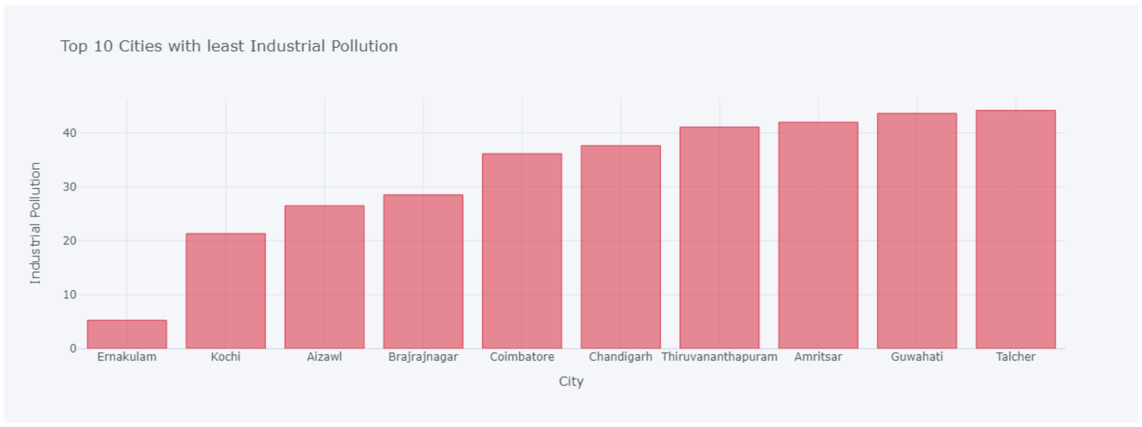


Figure 25: Top Cities with Lowest Industrial Pollution

Ernakulam, Kochi, and Aizawl top the list of the top ten cities with the least amount of industrial pollution following the COVID period, in the same order as they did before the COVID era.

3.3.3 Overall Observations

- Highest Automobile Pollution Content:
  - Even though Delhi remains the leading city with the highest automobile pollutants, there is a decrease in the amount by 4%
  - Patna and Gurugram's pollutant content did not reduce
  - Talcher reduced had the highest amount of decrease by 19% post covid
  - Kolkata's pollutant content reduced by 14%
  - Jaipur, Ahmedabad, Kochi decreased by 2% post covid
- Lowest Automobile Pollution Content:
  - Although Shillong continued to be the city with the least pollutant content, there was still a decrease of nearly 15%
  - Aizawl's pollutant content reduced by 8%
  - The other cities did not see much decrease in the pollutant content %
- Highest Industrial Pollution Content:
  - Ahmedabad remained the highest during both periods
  - Patna decreased its pollutant content by nearly 10
  - There was a negligible decrease in the case of Jaipur and the rest of the cities
- Lowest Industrial Pollution Content:
  - There was not much change in the pollutant content

1. Although there was quite a significant change in automobile pollutant content, we do not see the same in the case of industrial. This could be because of the overall decrease in vehicular traffic during COVID-19, whereas the industries continued to run even during COVID-19 to some extent.

2. The cities with lower automobile and industrial pollution content didn't see much change. One probable reason for the lack of change in industrial pollution content is that there was no major industrial plant present in these locations.

3.4 Significance of Industrial & Automobile Pollution Content

In this section, we will be performing ANOVA[8] to determine the significance of the variables that contribute to Industrial and Automobile pollution, towards calculating AQI for each city.

Industrial Pollution:

$$H_0 : Benzene = Toluene = Xylene = O3 = SO2 = 0$$

$$H_1 : \text{At least one of them is non-zero}$$

Automobile Pollution:

$$H_0 : PM2.5 = PM10 = NO = NO2 = NOx = NH3 = CO = 0$$

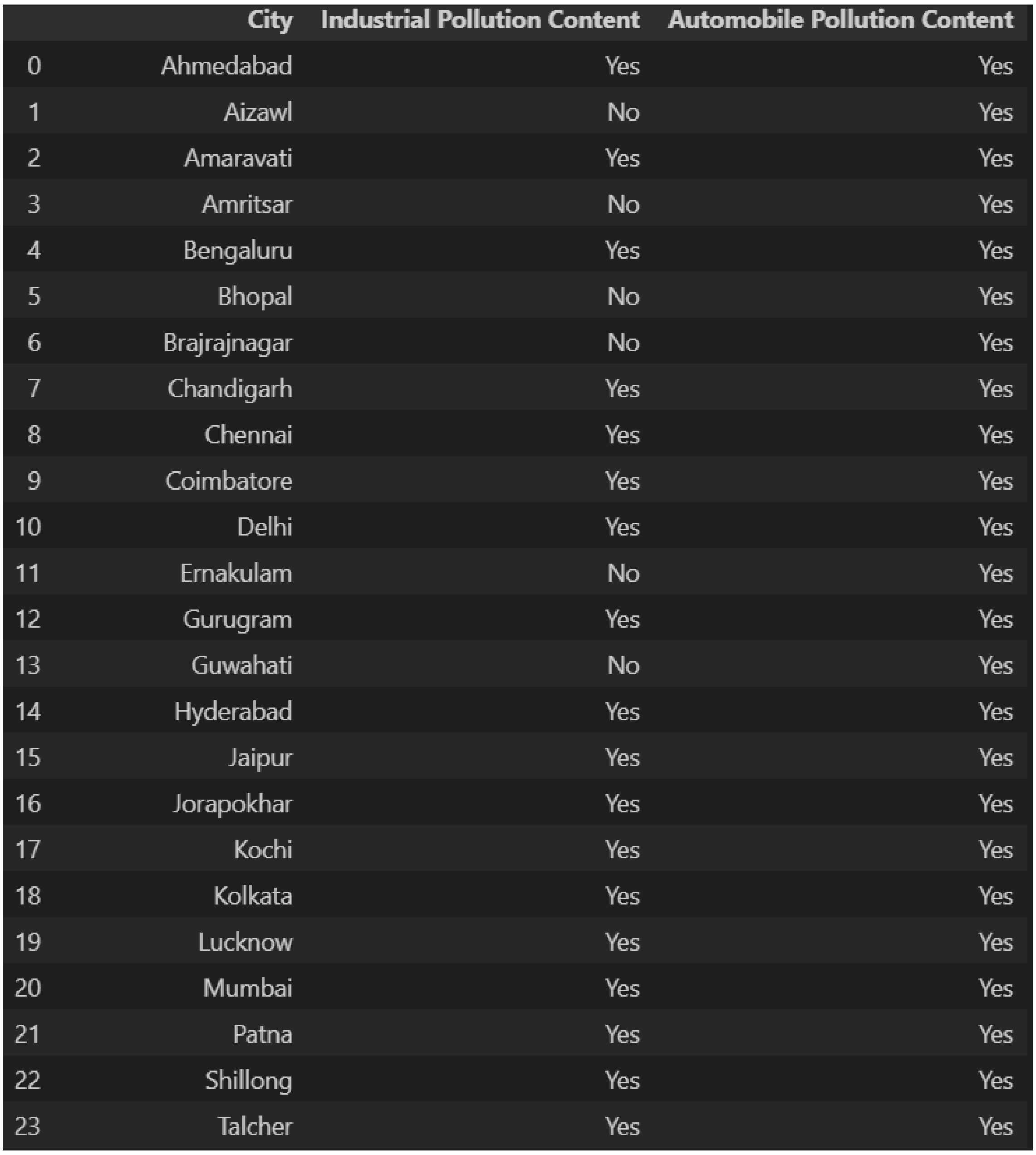
$$H_1 : \text{At least one of them is non-zero}$$

We've compiled a table containing information on the significance of Industrial and Automobile pollution content for each city[9]. Here is a snippet from the table:

It appears that Automobile Pollution Content is significant in all cities. This is due to the significance of PM2.5 and PM10 in calculating the Air Quality Index (AQI) for each city.

CODE:

```python
def test_lin_reg(df, target_column='AQI', hypothesis_matrix='Xylene = 0'):
    df = df.drop(columns=['Date', 'AQI_Bucket', 'City'])
    X = df.drop(columns=[target_column])
    Y = df[target_column]

    X_train, X_test, y_train, y_test = train_test_split(X, Y, random_state
=42)

    model = LinearRegression()

    model.fit(X_train, y_train)

    X_train_with_const = sm.add_constant(X_train)

    model_statsmodels = sm.OLS(y_train, X_train_with_const).fit()

    # Perform the Wald test if hypothesis_matrix is provided
    if hypothesis_matrix:
        # Perform the Wald test
        wald_test_result = model_statsmodels.wald_test(hypothesis_matrix,
scalar=False)

```



	City	Industrial Pollution Content	Automobile Pollution Content
0	Ahmedabad	Yes	Yes
1	Aizawl	No	Yes
2	Amaravati	Yes	Yes
3	Amritsar	No	Yes
4	Bengaluru	Yes	Yes
5	Bhopal	No	Yes
6	Brajrajnagar	No	Yes
7	Chandigarh	Yes	Yes
8	Chennai	Yes	Yes
9	Coimbatore	Yes	Yes
10	Delhi	Yes	Yes
11	Ernakulam	No	Yes
12	Gurugram	Yes	Yes
13	Guwahati	No	Yes
14	Hyderabad	Yes	Yes
15	Jaipur	Yes	Yes
16	Jorapokhar	Yes	Yes
17	Kochi	Yes	Yes
18	Kolkata	Yes	Yes
19	Lucknow	Yes	Yes
20	Mumbai	Yes	Yes
21	Patna	Yes	Yes
22	Shillong	Yes	Yes
23	Talcher	Yes	Yes

Figure 26: Pollution significance table for all cities

```
        # Return the p-value for further evaluation
        return wald_test_result.pvalue

# Iterate through unique cities
unique_cities = df_main['City'].unique()
significance_data = []

for city_name in unique_cities:
    city_data = df_main[df_main['City'] == city_name].copy()  # Copy to
   avoid modifying the original DataFrame

    Indus_hyp = 'Benzene = Toluene = Xylene = O3 = SO2 = 0'
    Auto_hyp = 'PM2.5 = PM10 = NO = NO2 = NOx = NH3 = CO = 0'
    # Test significance for Industrial Pollution content
    p_value_indus = test_lin_reg(city_data, target_column='AQI',
    hypothesis_matrix=Indus_hyp)
    p_value_auto = test_lin_reg(city_data, target_column='AQI',
    hypothesis_matrix=Auto_hyp)
    # Determine significance based on p-value
    if p_value_indus  < 0.05:
        is_significant_indus = 'Yes'
    else:
        is_significant_indus = 'No'
    if p_value_auto  < 0.05:
        is_significant_auto = 'Yes'
    else:
        is_significant_auto = 'No'

    # Append data to the significance DataFrame
    significance_data.append({'City': city_name, 'Industrial Pollution
    Content': is_significant_indus, 'Automobile Pollution Content':
    is_significant_auto})
    #significance_data.append({'City': city_name,  'Automobile Pollution
    Content': is_significant_auto,  'Automobile Pollution p-value':
    p_value_auto})

# Create the significance DataFrame
significance_df = pd.DataFrame.from_dict(significance_data)

significance_df.head()
```

Listing 4: Code to generate significance table

4 OUTLIER HANDLING - NUMERICAL COLUMNS

We will be analyzing the impact of outliers in each of the numerical columns on the adjusted R-squared of the model.
We will be considering values outside the 3-sigma limits as outliers.

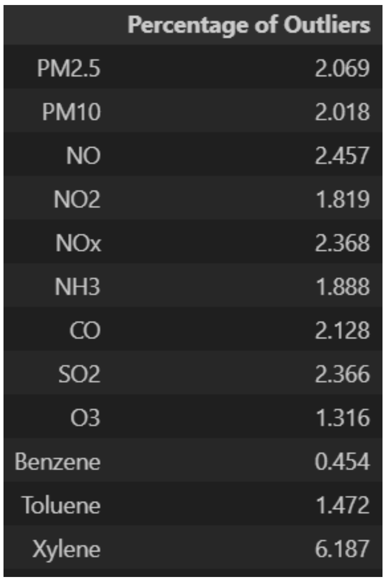
$$lowerbound = \mu - 3\sigma$$

$$upperbound = \mu + 3\sigma$$

The following are the steps:

1. Determine the percentage of outliers in each numerical column

```
    # Function to calculate the percentage of outliers
    def percentage_of_outliers(column):
        lower_bound = column.mean() - 3*column.std()
        upper_bound = column.mean() + 3*column.std()
        outliers = column[(column < lower_bound) | (column > upper_bound
)]
        non_zero_count = (column != 0).sum()
        percentage = len(outliers) / non_zero_count * 100
        return percentage

```



	Percentage of Outliers
PM2.5	2.069
PM10	2.018
NO	2.457
NO2	1.819
NOx	2.368
NH3	1.888
CO	2.128
SO2	2.366
O3	1.316
Benzene	0.454
Toluene	1.472
Xylene	6.187

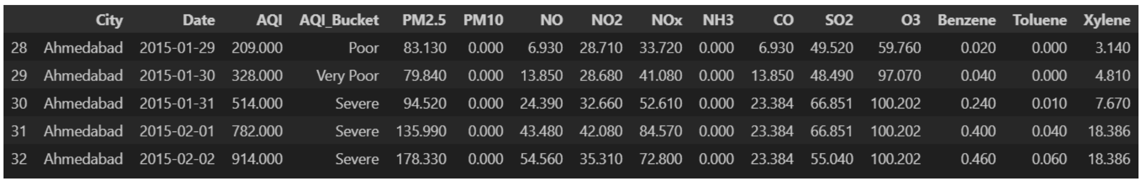
Figure 27: Percentage of outliers

2. Truncate the outliers with the upper and lower bound respectively (any value lower than the lower bound will be equated to the lower bound and the same would be applied for the upper bound as well)

```
    def truncate_numerical_column(df, numerical_column):
    """
    Truncate or cap extreme values in a numerical column of a DataFrame.

    Parameters:
    - df: DataFrame
        Input DataFrame.
    - numerical_column: str
```

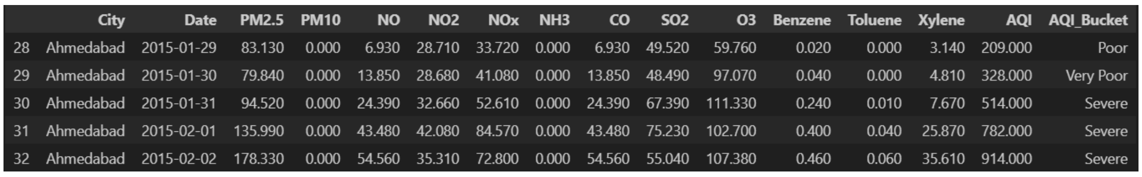
```python
            Name of the numerical column to be truncated.
        - lower_percentile: float, optional (default: 0)
            Lower percentile for truncation.
        - upper_percentile: float, optional (default: 95)
            Upper percentile for truncation.

        Returns:
        - DataFrame
            New DataFrame with the specified numerical column truncated.
        """
        # Calculate the threshold values based on percentile
        lower_threshold = df[numerical_column].mean() - 3*df[numerical_column].std()
        upper_threshold = df[numerical_column].mean() + 3*df[numerical_column].std()

        # Truncate extreme values to the specified thresholds
        truncated_column_name = numerical_column + '_truncated'
        df[truncated_column_name] = df[numerical_column].clip(lower=lower_threshold, upper=upper_threshold)

        return df

```

3. Prepare two separate data frames: one with outliers and one without outliers



	City	Date	AQI	AQI_Bucket	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene
28	Ahmedabad	2015-01-29	209.000	Poor	83.130	0.000	6.930	28.710	33.720	0.000	6.930	49.520	59.760	0.020	0.000	3.140
29	Ahmedabad	2015-01-30	328.000	Very Poor	79.840	0.000	13.850	28.680	41.080	0.000	13.850	48.490	97.070	0.040	0.000	4.810
30	Ahmedabad	2015-01-31	514.000	Severe	94.520	0.000	24.390	32.660	52.610	0.000	23.384	66.851	100.202	0.240	0.010	7.670
31	Ahmedabad	2015-02-01	782.000	Severe	135.990	0.000	43.480	42.080	84.570	0.000	23.384	66.851	100.202	0.400	0.040	18.386
32	Ahmedabad	2015-02-02	914.000	Severe	178.330	0.000	54.560	35.310	72.800	0.000	23.384	55.040	100.202	0.460	0.060	18.386

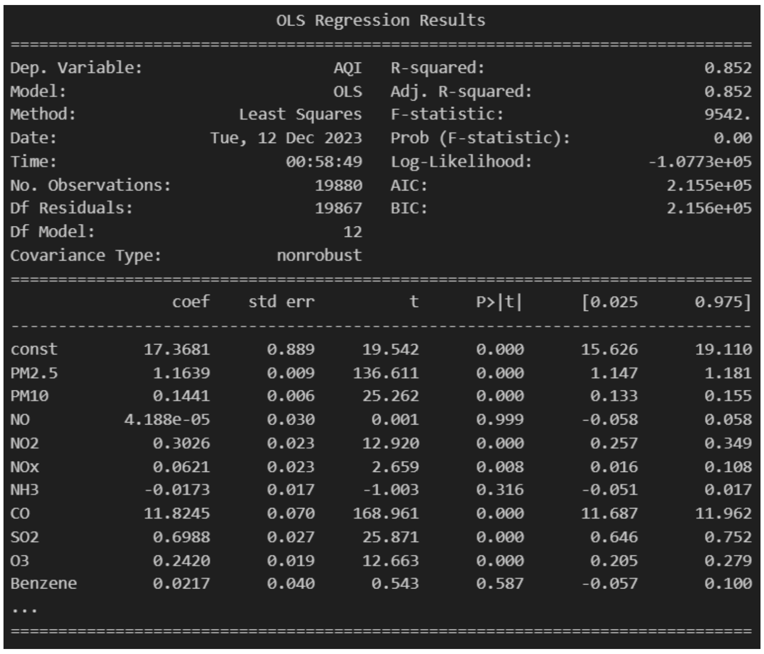
Figure 28: Without Outliers



	City	Date	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	AQI_Bucket
28	Ahmedabad	2015-01-29	83.130	0.000	6.930	28.710	33.720	0.000	6.930	49.520	59.760	0.020	0.000	3.140	209.000	Poor
29	Ahmedabad	2015-01-30	79.840	0.000	13.850	28.680	41.080	0.000	13.850	48.490	97.070	0.040	0.000	4.810	328.000	Very Poor
30	Ahmedabad	2015-01-31	94.520	0.000	24.390	32.660	52.610	0.000	24.390	67.390	111.330	0.240	0.010	7.670	514.000	Severe
31	Ahmedabad	2015-02-01	135.990	0.000	43.480	42.080	84.570	0.000	43.480	75.230	102.700	0.400	0.040	25.870	782.000	Severe
32	Ahmedabad	2015-02-02	178.330	0.000	54.560	35.310	72.800	0.000	54.560	55.040	107.380	0.460	0.060	35.610	914.000	Severe

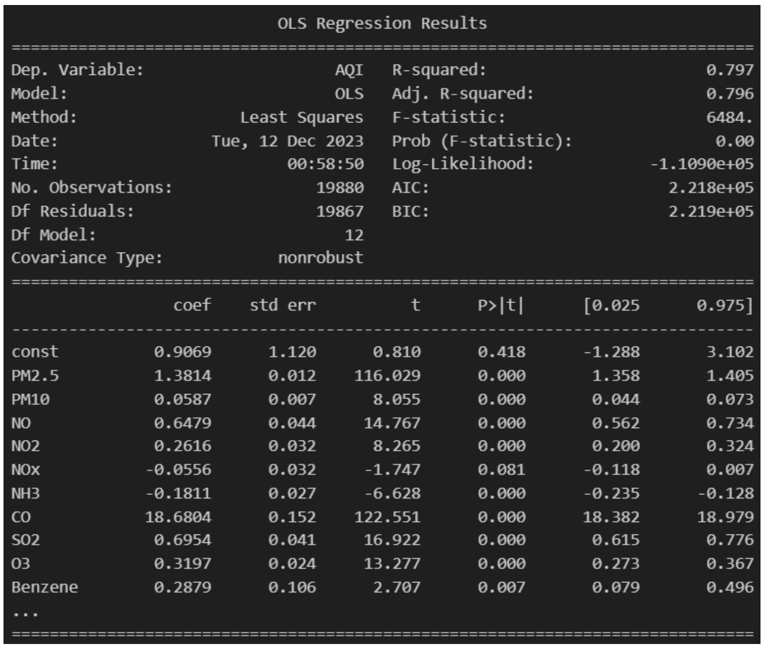
Figure 29: With Outliers

4. Run the linear regression models[10][11] on both data frames and compare the adjusted R-squared values.



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    AQI   R-squared:                       0.852
Model:                            OLS   Adj. R-squared:                  0.852
Method:                 Least Squares   F-statistic:                     9542.
Date:                Tue, 12 Dec 2023   Prob (F-statistic):               0.00
Time:                        00:58:49   Log-Likelihood:            -1.0773e+05
No. Observations:               19880   AIC:                         2.155e+05
Df Residuals:                   19867   BIC:                         2.156e+05
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         17.3681      0.889     19.542      0.000      15.626      19.110
PM2.5          1.1639      0.009    136.611      0.000       1.147       1.181
PM10           0.1441      0.006     25.262      0.000       0.133       0.155
NO          4.188e-05      0.030      0.001      0.999      -0.058       0.058
NO2            0.3026      0.023     12.920      0.000       0.257       0.349
NOx            0.0621      0.023      2.659      0.008       0.016       0.108
NH3           -0.0173      0.017     -1.003      0.316      -0.051       0.017
CO            11.8245      0.070    168.961      0.000      11.687      11.962
SO2            0.6988      0.027     25.871      0.000       0.646       0.752
O3             0.2420      0.019     12.663      0.000       0.205       0.279
Benzene        0.0217      0.040      0.543      0.587      -0.057       0.100
...
==============================================================================
```

Figure 30: Model summary With Outliers



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    AQI   R-squared:                       0.797
Model:                            OLS   Adj. R-squared:                  0.796
Method:                 Least Squares   F-statistic:                     6484.
Date:                Tue, 12 Dec 2023   Prob (F-statistic):               0.00
Time:                        00:58:50   Log-Likelihood:            -1.1090e+05
No. Observations:               19880   AIC:                         2.218e+05
Df Residuals:                   19867   BIC:                         2.219e+05
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.9069      1.120      0.810      0.418      -1.288       3.102
PM2.5          1.3814      0.012    116.029      0.000       1.358       1.405
PM10           0.0587      0.007      8.055      0.000       0.044       0.073
NO             0.6479      0.044     14.767      0.000       0.562       0.734
NO2            0.2616      0.032      8.265      0.000       0.200       0.324
NOx           -0.0556      0.032     -1.747      0.081      -0.118       0.007
NH3           -0.1811      0.027     -6.628      0.000      -0.235      -0.128
CO            18.6804      0.152    122.551      0.000      18.382      18.979
SO2            0.6954      0.041     16.922      0.000       0.615       0.776
O3             0.3197      0.024     13.277      0.000       0.273       0.367
Benzene        0.2879      0.106      2.707      0.007       0.079       0.496
...
==============================================================================
```

Figure 31: Model summary Without Outliers

5. Consider the data set with the higher adjusted R-squared values.
   We see that the adjusted R-squared value for the data with the outliers (0.852) is higher than the data without the outliers (0.796). Hence, we do not remove the outliers from the numerical columns.

5 SEASONALITY CHECK

5.1 Seasonality trends

5.1.1 Over the years

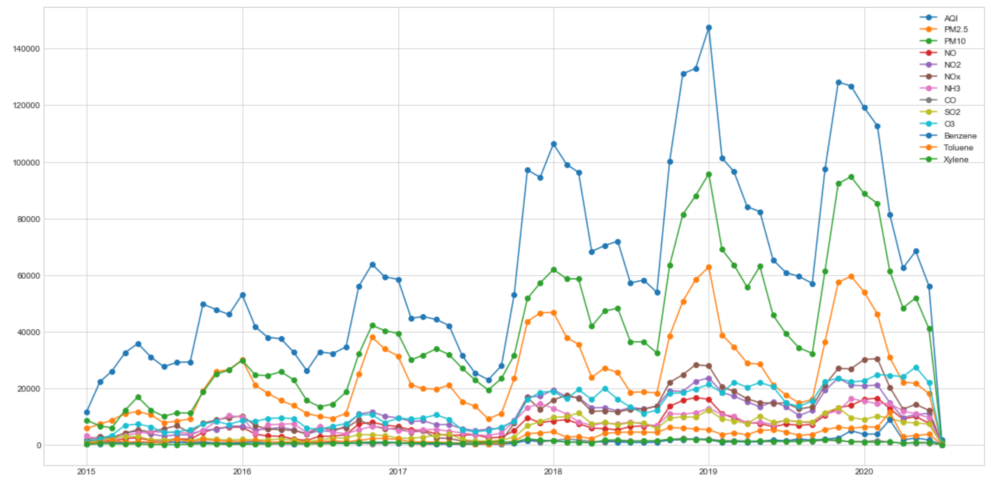


Figure 32: Seasonality trends

From the above figure, we see that the pollutants decrease from May to September every year.
Reason: This could be because a lot of the cities in our data set experience monsoon (rain) during these months which plays a major contributing factor in decreasing the content of these pollutants in the air.

5.1.2 One year

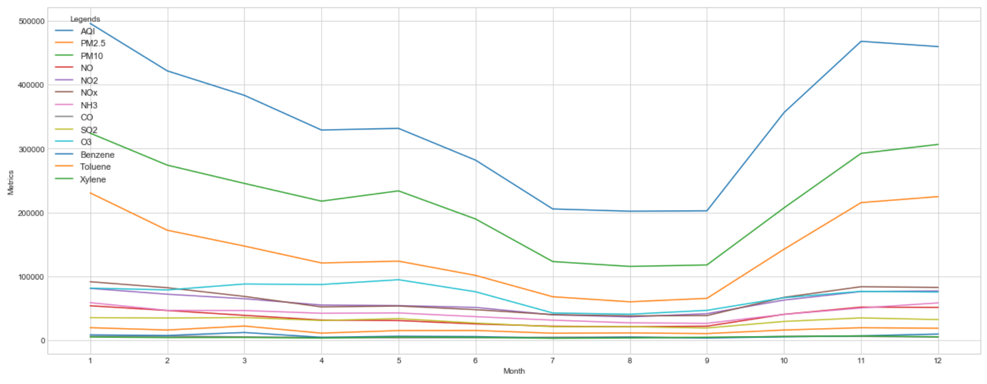
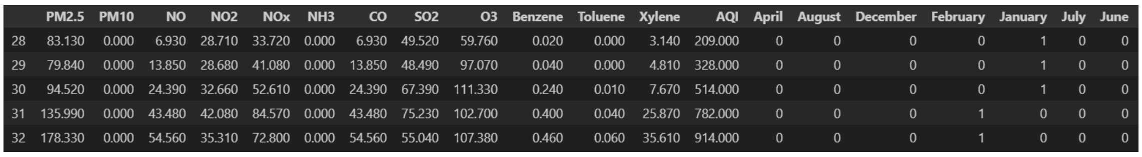


Figure 33: Seasonality trends for a year

Now that we have observed a monthly seasonality, we will investigate the effect of including a "Month" column in the data set.

5.2 One-Hot Encoding

We will be including a ”Month” column in the data set by extracting the months from the ”Date” column.
As the ”Month” data is not ordinal, we will have to one-hot[12] encode it.



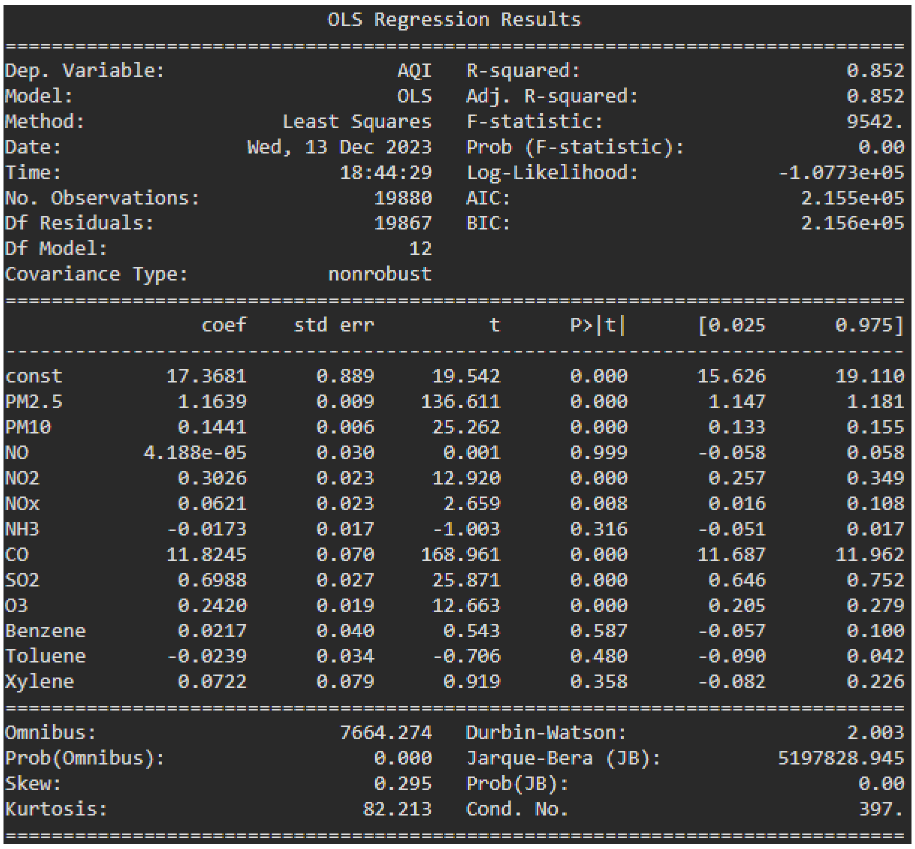
	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI	April	August	December	February	January	July	June
28	83.130	0.000	6.930	28.710	33.720	0.000	6.930	49.520	59.760	0.020	0.000	3.140	209.000	0	0	0	0	1	0	0
29	79.840	0.000	13.850	28.680	41.080	0.000	13.850	48.490	97.070	0.040	0.000	4.810	328.000	0	0	0	0	1	0	0
30	94.520	0.000	24.390	32.660	52.610	0.000	24.390	67.390	111.330	0.240	0.010	7.670	514.000	0	0	0	0	1	0	0
31	135.990	0.000	43.480	42.080	84.570	0.000	43.480	75.230	102.700	0.400	0.040	25.870	782.000	0	0	0	1	0	0	0
32	178.330	0.000	54.560	35.310	72.800	0.000	54.560	55.040	107.380	0.460	0.060	35.610	914.000	0	0	0	1	0	0	0

Figure 34: One-hot encoded data

We will compare the adjusted R-squared of the data set without the ”Month” column and the One-Hot[12] encoded data with the months.

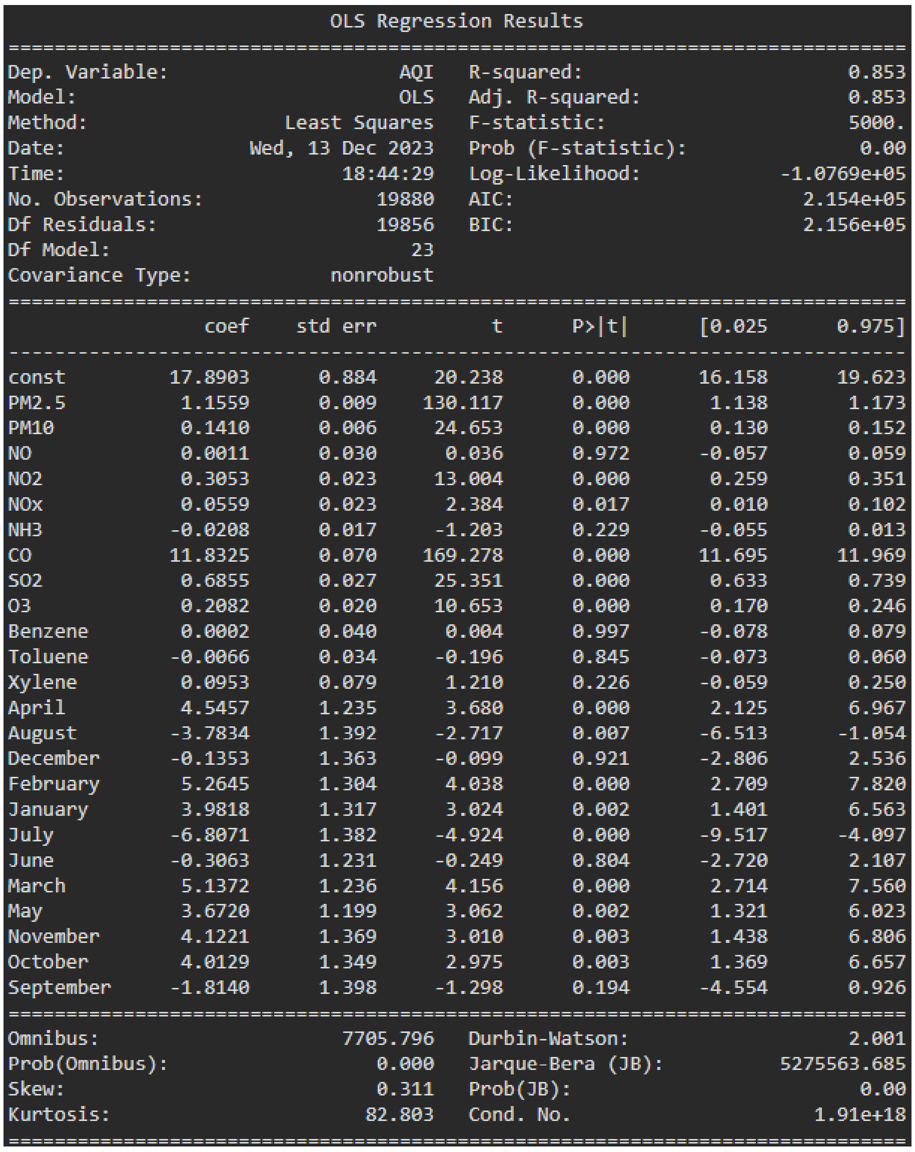
5.3 Comparison of Models

5.3.1 Without one-hot encoding



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    AQI   R-squared:                       0.852
Model:                            OLS   Adj. R-squared:                  0.852
Method:                 Least Squares   F-statistic:                     9542.
Date:                Wed, 13 Dec 2023   Prob (F-statistic):               0.00
Time:                        18:44:29   Log-Likelihood:            -1.0773e+05
No. Observations:               19880   AIC:                         2.155e+05
Df Residuals:                   19867   BIC:                         2.156e+05
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         17.3681      0.889     19.542      0.000      15.626      19.110
PM2.5          1.1639      0.009    136.611      0.000       1.147       1.181
PM10           0.1441      0.006     25.262      0.000       0.133       0.155
NO          4.188e-05      0.030      0.001      0.999      -0.058       0.058
NO2            0.3026      0.023     12.920      0.000       0.257       0.349
NOx            0.0621      0.023      2.659      0.008       0.016       0.108
NH3           -0.0173      0.017     -1.003      0.316      -0.051       0.017
CO            11.8245      0.070    168.961      0.000      11.687      11.962
SO2            0.6988      0.027     25.871      0.000       0.646       0.752
O3             0.2420      0.019     12.663      0.000       0.205       0.279
Benzene        0.0217      0.040      0.543      0.587      -0.057       0.100
Toluene       -0.0239      0.034     -0.706      0.480      -0.090       0.042
Xylene         0.0722      0.079      0.919      0.358      -0.082       0.226
==============================================================================
Omnibus:                     7664.274   Durbin-Watson:                   2.003
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          5197828.945
Skew:                           0.295   Prob(JB):                         0.00
Kurtosis:                      82.213   Cond. No.                         397.
==============================================================================
```

5.3.2 With one-hot encoding



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    AQI   R-squared:                       0.853
Model:                            OLS   Adj. R-squared:                  0.853
Method:                 Least Squares   F-statistic:                     5000.
Date:                Wed, 13 Dec 2023   Prob (F-statistic):               0.00
Time:                        18:44:29   Log-Likelihood:            -1.0769e+05
No. Observations:               19880   AIC:                         2.154e+05
Df Residuals:                   19856   BIC:                         2.156e+05
Df Model:                          23
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         17.8903      0.884     20.238      0.000      16.158      19.623
PM2.5          1.1559      0.009    130.117      0.000       1.138       1.173
PM10           0.1410      0.006     24.653      0.000       0.130       0.152
NO             0.0011      0.030      0.036      0.972      -0.057       0.059
NO2            0.3053      0.023     13.004      0.000       0.259       0.351
NOx            0.0559      0.023      2.384      0.017       0.010       0.102
NH3           -0.0208      0.017     -1.203      0.229      -0.055       0.013
CO            11.8325      0.070    169.278      0.000      11.695      11.969
SO2            0.6855      0.027     25.351      0.000       0.633       0.739
O3             0.2082      0.020     10.653      0.000       0.170       0.246
Benzene        0.0002      0.040      0.004      0.997      -0.078       0.079
Toluene       -0.0066      0.034     -0.196      0.845      -0.073       0.060
Xylene         0.0953      0.079      1.210      0.226      -0.059       0.250
April          4.5457      1.235      3.680      0.000       2.125       6.967
August        -3.7834      1.392     -2.717      0.007      -6.513      -1.054
December      -0.1353      1.363     -0.099      0.921      -2.806       2.536
February       5.2645      1.304      4.038      0.000       2.709       7.820
January        3.9818      1.317      3.024      0.002       1.401       6.563
July          -6.8071      1.382     -4.924      0.000      -9.517      -4.097
June          -0.3063      1.231     -0.249      0.804      -2.720       2.107
March          5.1372      1.236      4.156      0.000       2.714       7.560
May            3.6720      1.199      3.062      0.002       1.321       6.023
November       4.1221      1.369      3.010      0.003       1.438       6.806
October        4.0129      1.349      2.975      0.003       1.369       6.657
September     -1.8140      1.398     -1.298      0.194      -4.554       0.926
==============================================================================
Omnibus:                     7705.796   Durbin-Watson:                   2.001
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          5275563.685
Skew:                           0.311   Prob(JB):                         0.00
Kurtosis:                      82.803   Cond. No.                     1.91e+18
==============================================================================
```

From the above images, we can see that both models have similar adjusted R-squared values (0.853). As it is not a significant increase in the value, we stick to the data without the "Month" column.

6 CHECK FOR MODEL ACCURACY & MODEL ASSUMPTIONS

MODEL ASSUMPTIONS:

1. $\epsilon_{i=1}^{n}$ are independent.
2. $\epsilon_{i=1}^{n}$ have mean 0 and variance σ^2
3. $\epsilon_{i=1}^{n}$ are often (but not necessarily) assumed to follow a normal distribution.

$\epsilon_1, \epsilon_2, \epsilon_3, ..., \epsilon_n$ are iid $N(0, \sigma^2)$

FOR REGRESSION DIAGNOSTICS WE DO THE FOLLOWING:

1. Residuals v/s Fitted Values plot – To check if the residuals have a constant variance across all levels of independent variables, and they are scattered with a mean around 0.
2. Normal Q-Q Plot – To check how close the residuals are to follow a normal distribution.
3. Shaprio-Wilk Test[13] – Hypothesis test to check if the residuals are normally distributed.
4. Durbin- Watson Test[14] – To check if residuals are independent, or the autocorrelation amongst residuals.
5. Residuals v/s Leverage Plot – To assess the influence of individual data points on the estimated coefficients of the model.
6. Cook's Distance vs. Half Normal Quantiles Plot - plot is particularly useful in identifying influential points that might have a significant impact on the model's fit and performance.

6.1 Original Data Frame

6.1.1 Cross-Validation

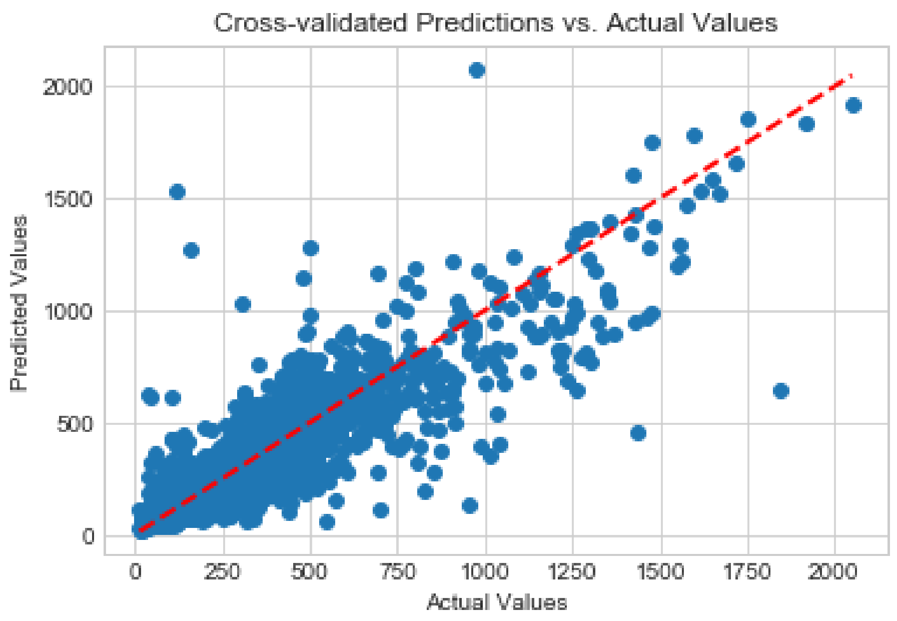
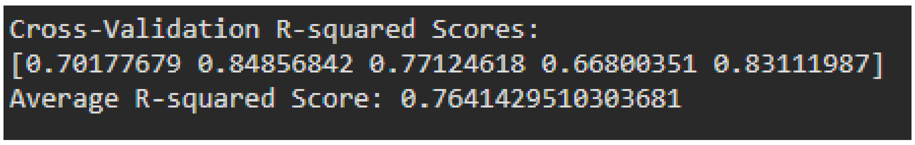
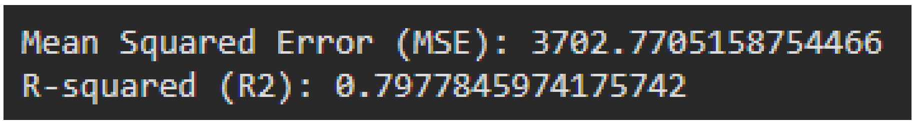


Figure 35: Cross validation for original data frame



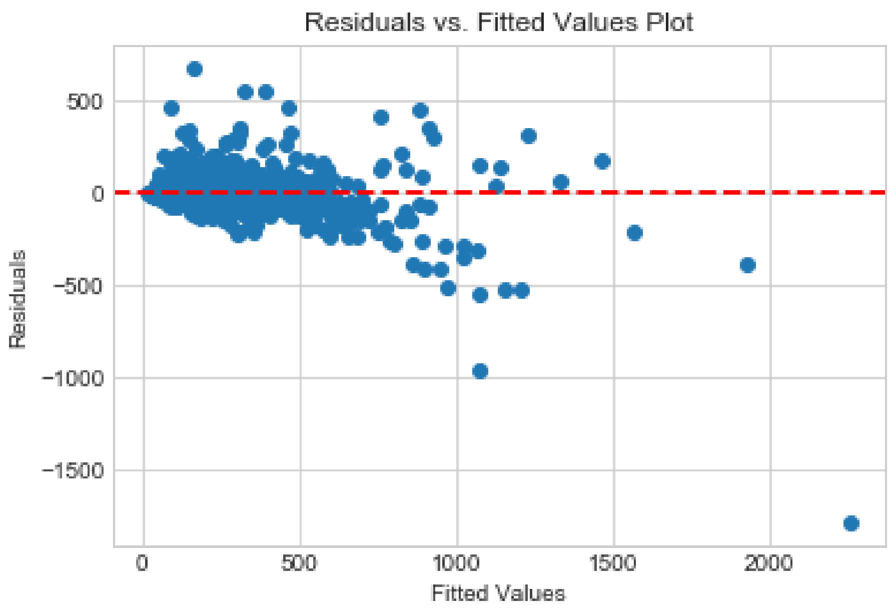
```
Cross-Validation R-squared Scores:
[0.70177679 0.84856842 0.77124618 0.66800351 0.83111987]
Average R-squared Score: 0.7641429510303681
```



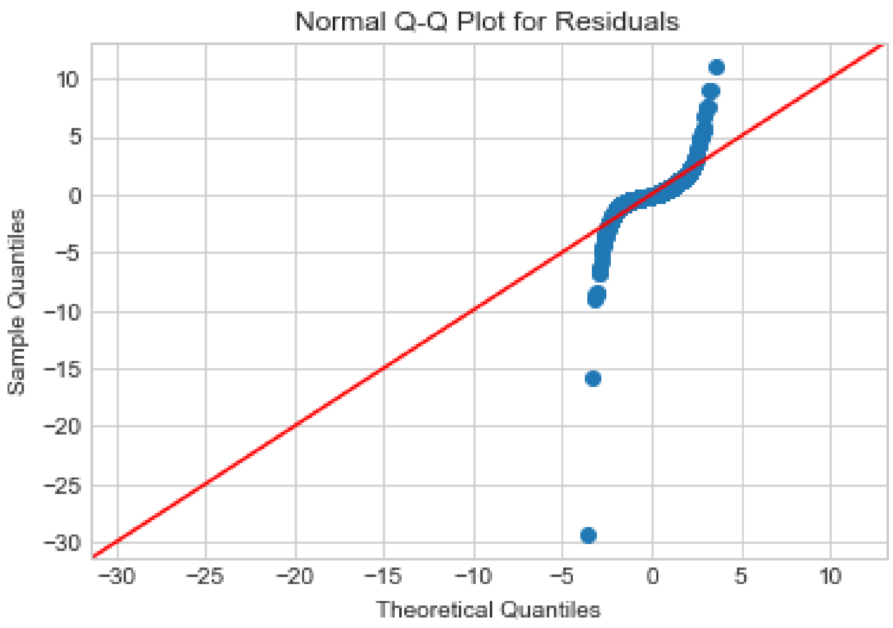
```
Mean Squared Error (MSE): 3702.7705158754466
R-squared (R2): 0.7977845974175742
```

6.1.2 Check for model assumptions

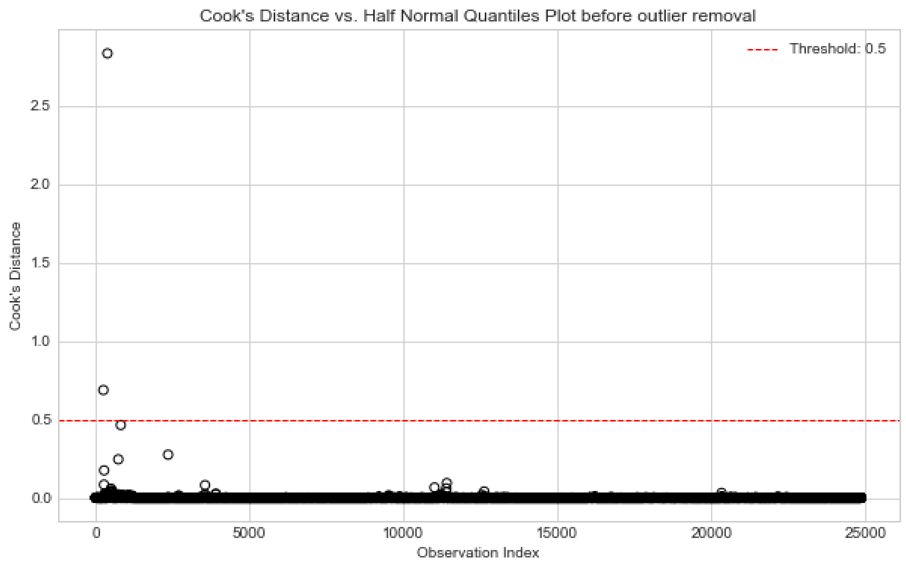
1. Residual vs Fitted Values plot



2. Normal Q-Q plot for residuals



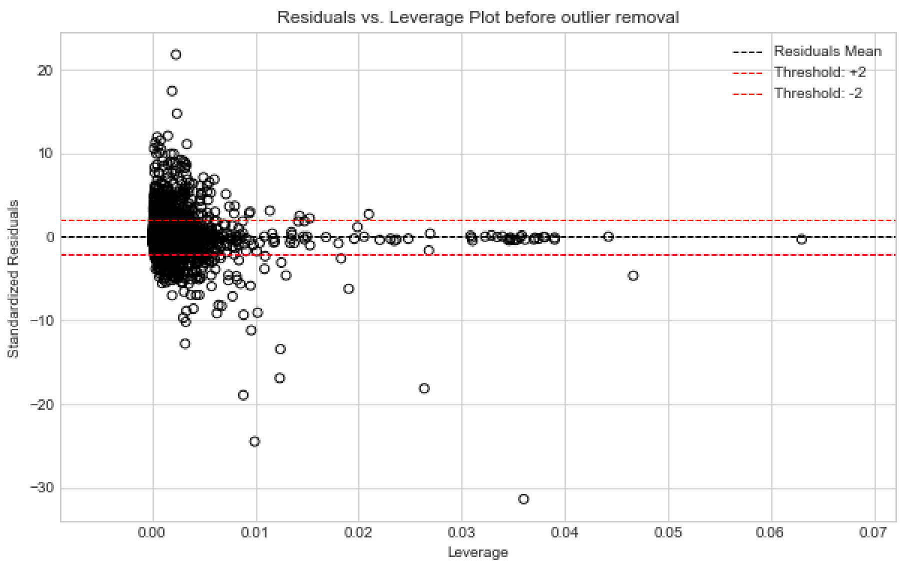
3. Cook's Distance vs. half normal quantiles plot before outlier removal



Observations:

- We see that there are just two points outside the threshold and most of the points are within the threshold.
- Hence, we can deduce the following:
  - The fact that most points are within the threshold suggests that the overall model is relatively stable, and no single observation disproportionately affects the model parameters.
  - If the majority of points fall within an acceptable range, it indicates that the model adequately captures the data patterns, and influential outliers are not unduly influencing the results.
  - In such a scenario, you can interpret the plot as reassuring evidence of the robustness of your regression model. The absence of outliers or influential points allows for greater confidence in the estimated coefficients.

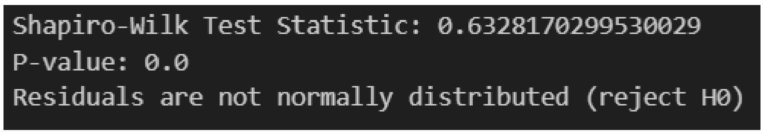
4. Residual vs Leverage plot before outlier removal



Observations:

- We can see that there are no observations in the top right or bottom right corner, as these would indicate outliers which have a significant impact on the model, which would need to be investigated individually.
- Most of the observations with high residuals, have low leverages.
- So, we can predict that getting rid of them would have inversely affected the performance of the model.

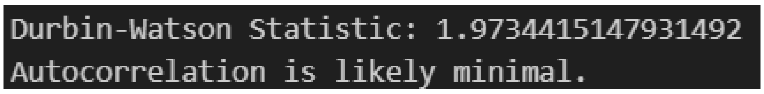
5. Shapiro-Wilk test



```
Shapiro-Wilk Test Statistic: 0.6328170299530029
P-value: 0.0
Residuals are not normally distributed (reject H0)
```

We perform the Wilk Shapiro test to check if the residuals are normally distributed. As the hypothesis is rejected we conclude that the Residuals are not normally distributed.

6. Durbin-Watson test



```
Durbin-Watson Statistic: 1.9734415147931492
Autocorrelation is likely minimal.
```

Observations:

- We see that, in the Normal Q-Q Plot, the plot is distorted showing a Fat Tail distribution, and the plot does not trace the normal line with major deviations at the ends of the plot.
- Additionally, the residuals also fail the Shapiro-Wilk test.
- As a next step, we will try to check if the distortion might be because of the existence of observations in the data set that might be outliers.

6.2 Outlier Handling

6.2.1 Observations

1. We will be identifying the observations that are outliers by calculating the Cook's Distance(D_i)[15] and Leverage score(h_{ii}) for each observation.

2. The observation is considered influential if:

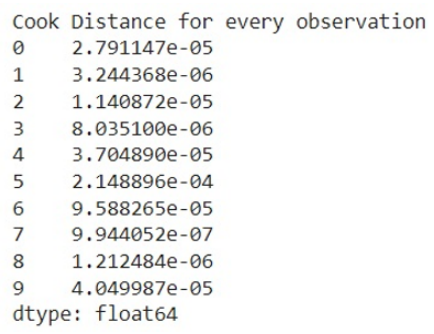
$$D > 4/n - p - 1$$

where,
D_i – Cook's Distance of the observation
n – Number of Observations
p – Number of Independent Variables

```python
import pandas as pd
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import OLSInfluence

X = df.drop(columns=['AQI'])
y = df['AQI']

X = sm.add_constant(X)
model = sm.OLS(y, X).fit()

# Get the Cook's distance and other influence statistics
influence = OLSInfluence(model)
cooks_distance = influence.cooks_distance
cooks_distance_values = cooks_distance[0]
print("Cook Distance for every observation")
display(cooks_distance_values.head(10))

# Leverage score for each observation
leverage_influence = model.get_influence()
leverage_values = leverage_influence.hat_matrix_diag
leverage_matrix = pd.DataFrame({'Leverage': leverage_values})
display(leverage_matrix.head(10))

# Set a threshold to identify influential observations
threshold = 4 / (len(y) - 12 - 1)
influential_obs = cooks_distance_values > threshold

# Print or do something with the influential observations
print("Influential observations:", display(df[influential_obs]))
df_without_outliers = df[~influential_obs].copy()

# Print or do something with the new dataset without outliers
print("Original dataset shape:", df.shape)
print("Dataset shape without outliers:", df_without_outliers.shape)
```

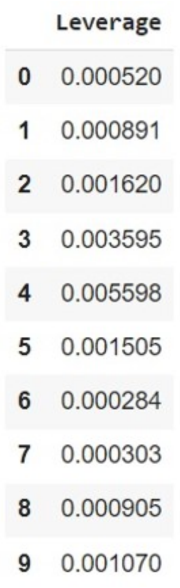
Output:

- Cook Distance for every observation



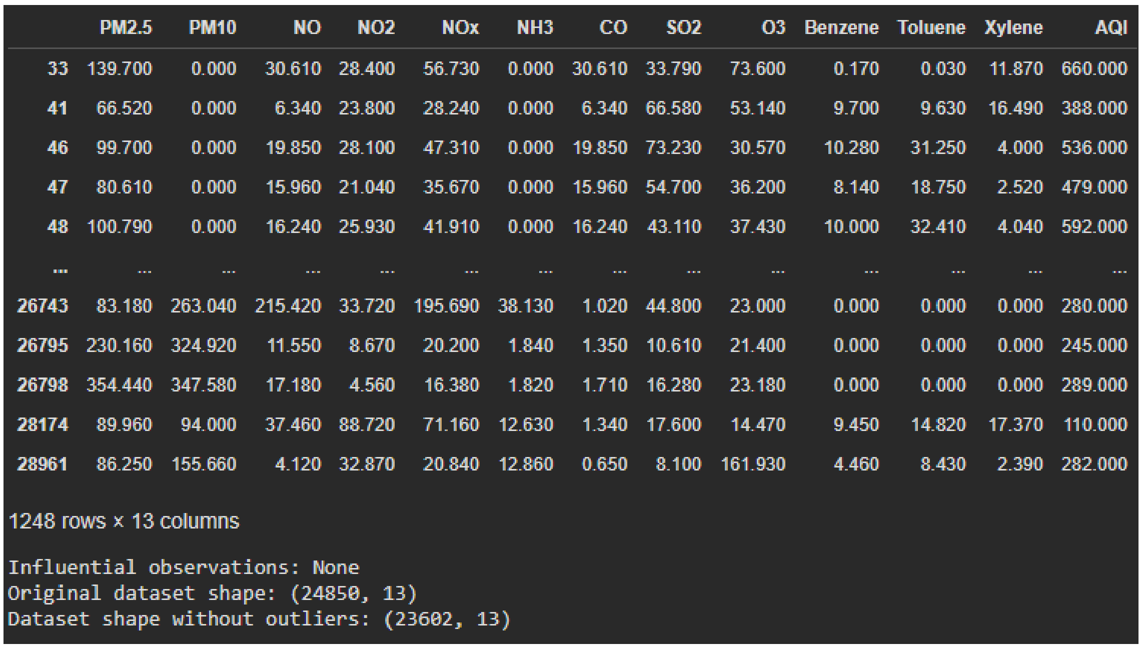
```
Cook Distance for every observation
0    2.791147e-05
1    3.244368e-06
2    1.140872e-05
3    8.035100e-06
4    3.704890e-05
5    2.148896e-04
6    9.588265e-05
7    9.944052e-07
8    1.212484e-06
9    4.049987e-05
dtype: float64
```

- Leverage



	Leverage
0	0.000520
1	0.000891
2	0.001620
3	0.003595
4	0.005598
5	0.001505
6	0.000284
7	0.000303
8	0.000905
9	0.001070

- Outliers: We can see that 1248 observations are outliers.

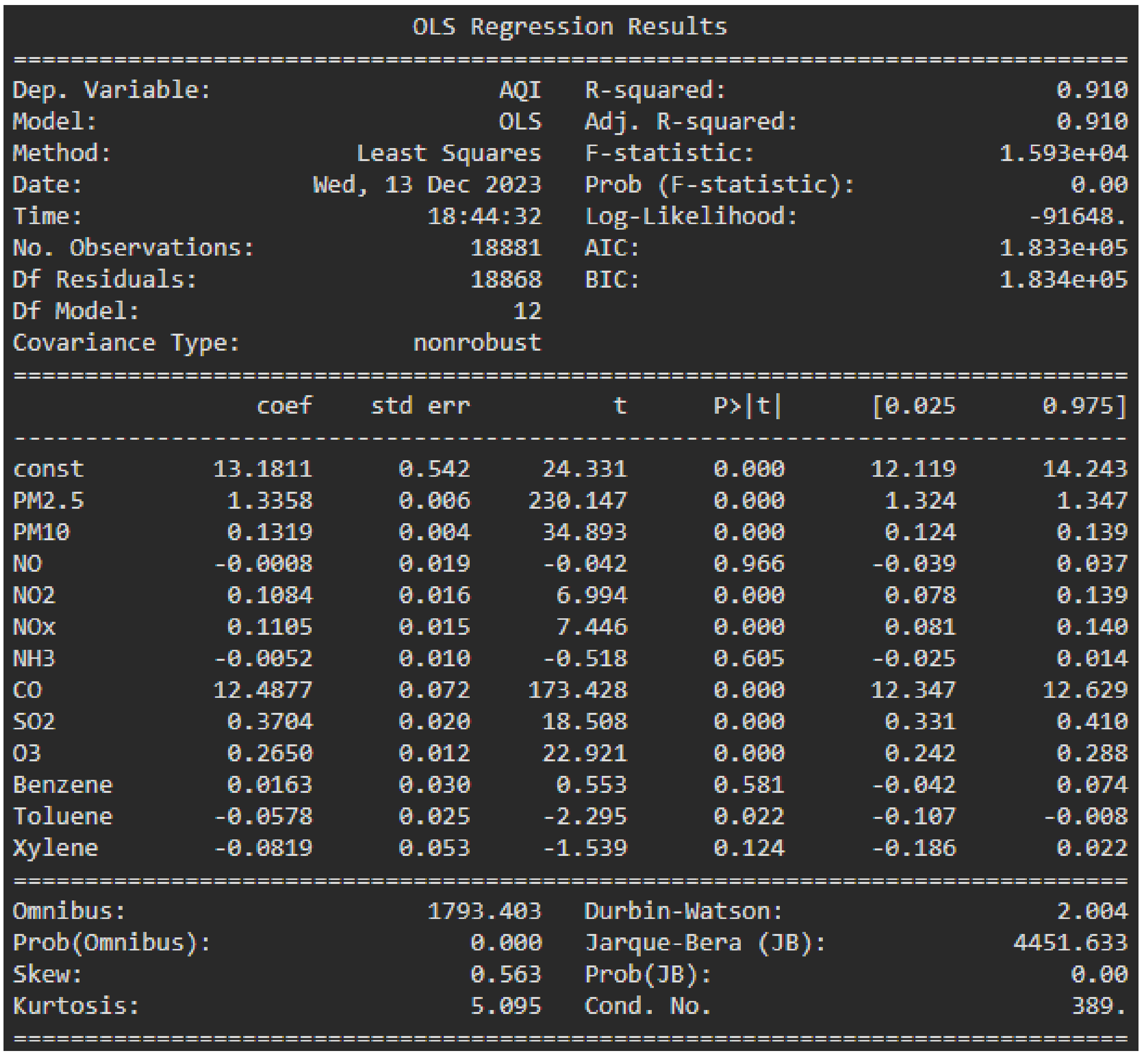


	PM2.5	PM10	NO	NO2	NOx	NH3	CO	SO2	O3	Benzene	Toluene	Xylene	AQI
33	139.700	0.000	30.610	28.400	56.730	0.000	30.610	33.790	73.600	0.170	0.030	11.870	660.000
41	66.520	0.000	6.340	23.800	28.240	0.000	6.340	66.580	53.140	9.700	9.630	16.490	388.000
46	99.700	0.000	19.850	28.100	47.310	0.000	19.850	73.230	30.570	10.280	31.250	4.000	536.000
47	80.610	0.000	15.960	21.040	35.670	0.000	15.960	54.700	36.200	8.140	18.750	2.520	479.000
48	100.790	0.000	16.240	25.930	41.910	0.000	16.240	43.110	37.430	10.000	32.410	4.040	592.000
...	...	...	...	...	...	...	...	...	...	...	...	...	...
26743	83.180	263.040	215.420	33.720	195.690	38.130	1.020	44.800	23.000	0.000	0.000	0.000	280.000
26795	230.160	324.920	11.550	8.670	20.200	1.840	1.350	10.610	21.400	0.000	0.000	0.000	245.000
26798	354.440	347.580	17.180	4.560	16.380	1.820	1.710	16.280	23.180	0.000	0.000	0.000	289.000
28174	89.960	94.000	37.460	88.720	71.160	12.630	1.340	17.600	14.470	9.450	14.820	17.370	110.000
28961	86.250	155.660	4.120	32.870	20.840	12.860	0.650	8.100	161.930	4.460	8.430	2.390	282.000

1248 rows × 13 columns

```
Influential observations: None
Original dataset shape: (24850, 13)
Dataset shape without outliers: (23602, 13)
```

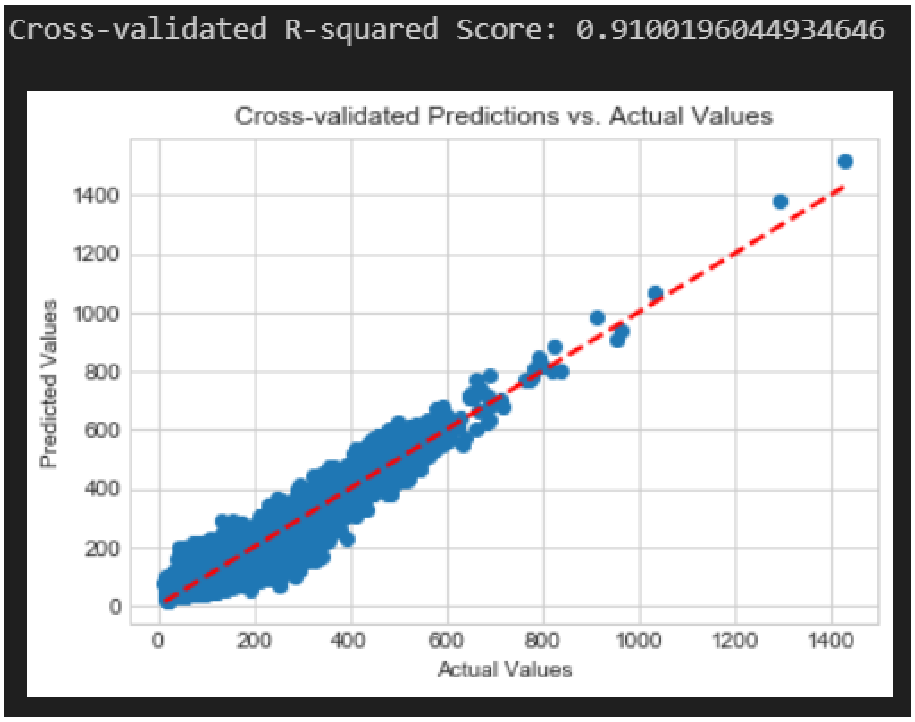
6.2.2 OLS Model for Data Frame without outliers



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    AQI   R-squared:                       0.910
Model:                            OLS   Adj. R-squared:                  0.910
Method:                 Least Squares   F-statistic:                 1.593e+04
Date:                Wed, 13 Dec 2023   Prob (F-statistic):               0.00
Time:                        18:44:32   Log-Likelihood:                -91648.
No. Observations:               18881   AIC:                         1.833e+05
Df Residuals:                   18868   BIC:                         1.834e+05
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         13.1811      0.542     24.331      0.000      12.119      14.243
PM2.5          1.3358      0.006    230.147      0.000       1.324       1.347
PM10           0.1319      0.004     34.893      0.000       0.124       0.139
NO            -0.0008      0.019     -0.042      0.966      -0.039       0.037
NO2            0.1084      0.016      6.994      0.000       0.078       0.139
NOx            0.1105      0.015      7.446      0.000       0.081       0.140
NH3           -0.0052      0.010     -0.518      0.605      -0.025       0.014
CO            12.4877      0.072    173.428      0.000      12.347      12.629
SO2            0.3704      0.020     18.508      0.000       0.331       0.410
O3             0.2650      0.012     22.921      0.000       0.242       0.288
Benzene        0.0163      0.030      0.553      0.581      -0.042       0.074
Toluene       -0.0578      0.025     -2.295      0.022      -0.107      -0.008
Xylene        -0.0819      0.053     -1.539      0.124      -0.186       0.022
==============================================================================
Omnibus:                     1793.403   Durbin-Watson:                   2.004
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4451.633
Skew:                           0.563   Prob(JB):                         0.00
Kurtosis:                       5.095   Cond. No.                         389.
==============================================================================
```

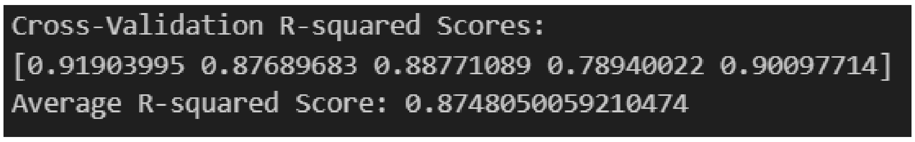
We observe that the Adjusted R-squared value of the data set without outliers is better than that of the data set with outliers.
Therefore, now we will use the data set without outliers.

6.2.3 Cross-Validation

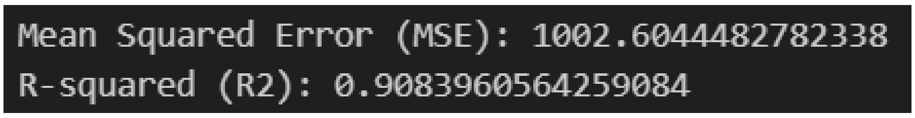


```
Cross-validated R-squared Score: 0.9100196044934646
```

Figure 36: Cross validation for data frame after outlier removal



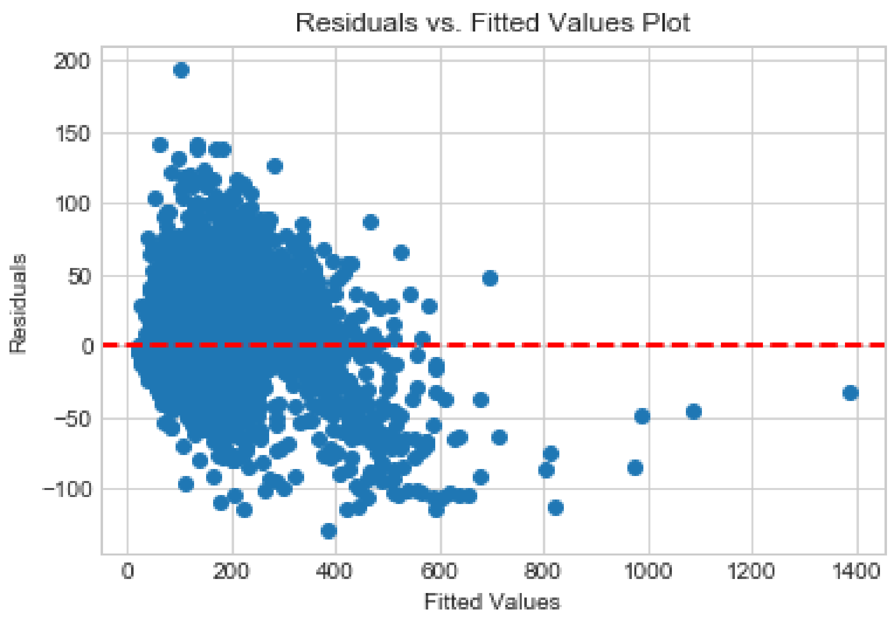
```
Cross-Validation R-squared Scores:
[0.91903995 0.87689683 0.88771089 0.78940022 0.90097714]
Average R-squared Score: 0.8748050059210474
```



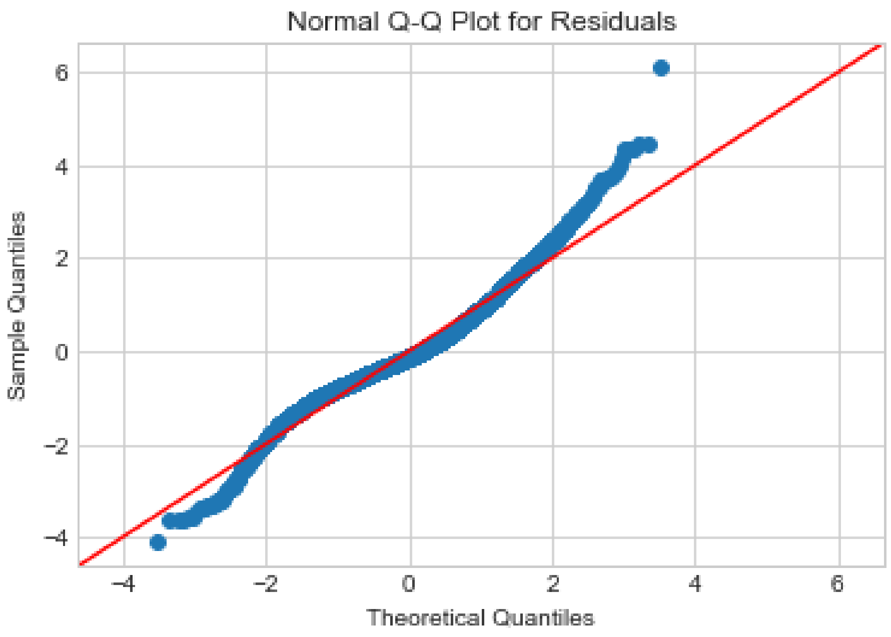
```
Mean Squared Error (MSE): 1002.6044482782338
R-squared (R2): 0.9083960564259084
```

6.2.4 Check for model assumptions

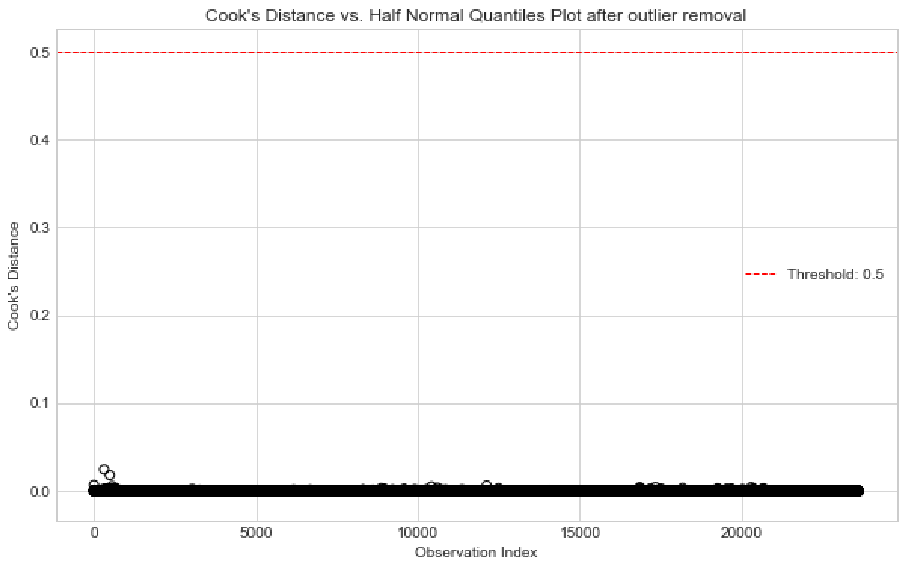
1. Residual vs Fitted Values plot



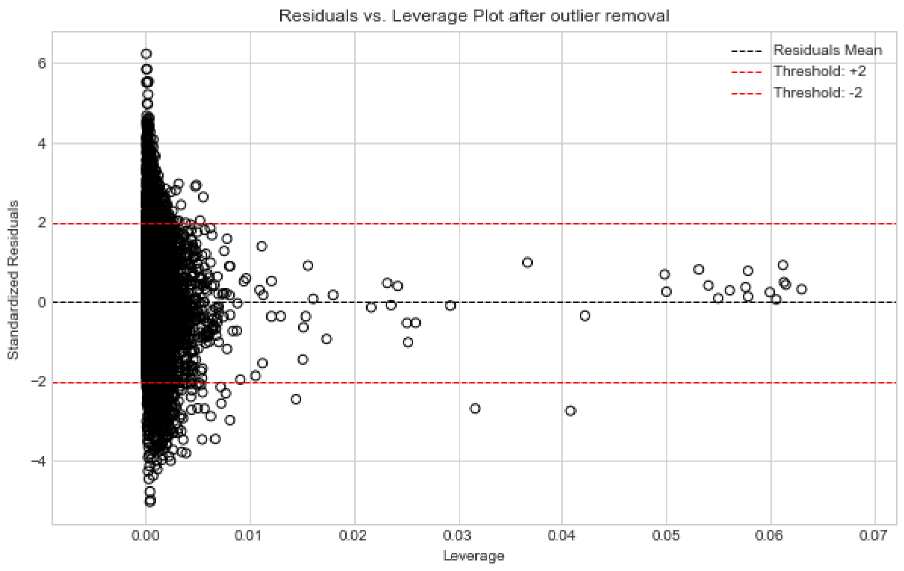
2. Normal Q-Q plot for residuals



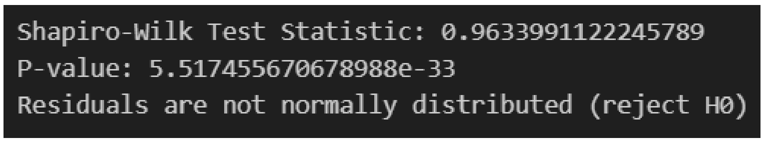
3. Cook's Distance vs. half normal quantiles plot before outlier removal



4. Residual vs Leverage plot before outlier removal



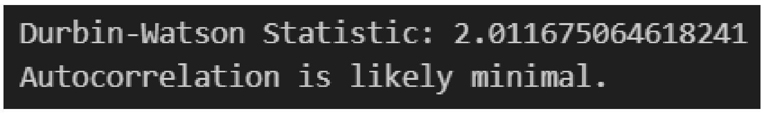
5. Shapiro-Wilk test



```
Shapiro-Wilk Test Statistic: 0.9633991122245789
P-value: 5.517455670678988e-33
Residuals are not normally distributed (reject H0)
```

As the hypothesis is rejected we conclude that the Residuals are not normally distributed.

6. Durbin-Watson test



```
Durbin-Watson Statistic: 2.011675064618241
Autocorrelation is likely minimal.
```

Observations:

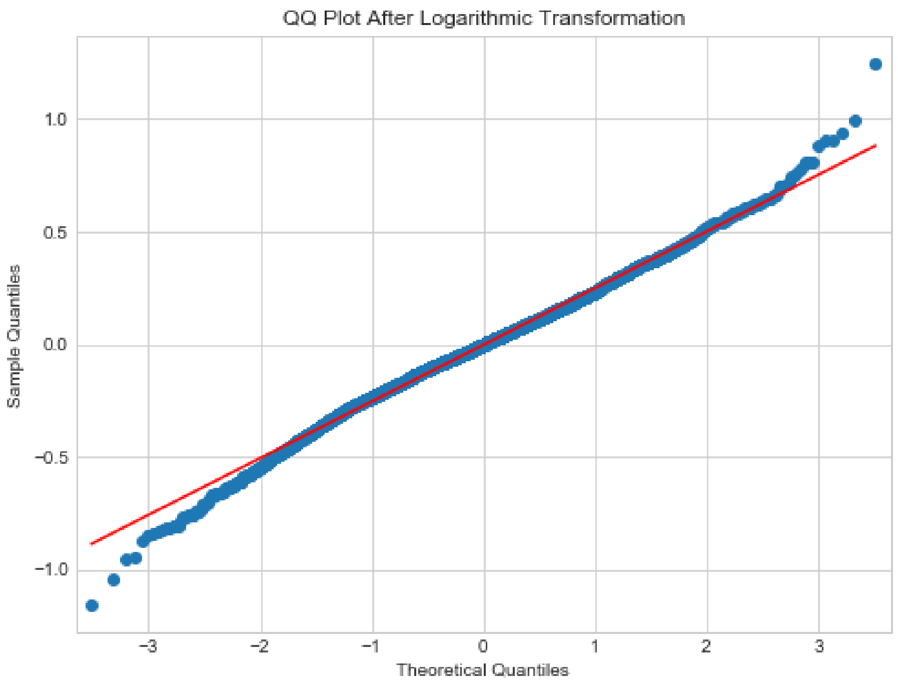
- We see that even though the residuals still fail the Shapiro-Wilk Normality test, there is a significant improvement in how the Normal Q-Q plot traces the normal line, still indicating a Fat Tail distribution.
  Now we'll be exploring various transformation techniques to improve the Normal Q-Q Plot of residuals.
  The two transformations we'll be testing are Log and Box-Cox transformations.

6.3 Transformation of Data Frame

6.3.1 Log Transformation

```python
X = df_without_outliers.drop('AQI', axis=1)
Y = df_without_outliers['AQI']

# Add a constant term to the predictor variables
X = sm.add_constant(X)

# Apply a logarithmic transformation to the response variable
Y_transformed = np.log1p(Y)

#Splitting into test and train data
#X_train, X_test, y_train, y_test = train_test_split(X, Y_transformed,
    test_size=0.2, random_state=42)

# Fit the OLS model with the transformed response variable
model_transformed = sm.OLS(Y_transformed, X).fit()

# Get the Cook's distance and other influence statistics
influence = OLSInfluence(model_transformed)
cooks_distance = influence.cooks_distance
cooks_distance_values = cooks_distance[0]

# Set a threshold to identify influential observations
threshold = 4 / len(y)
influential_obs = cooks_distance_values > threshold

# Print or do something with the influential observations
# print("Influential observations:", display(data_without_outliers[
    influential_obs]))
df_without_outliers_log = df_without_outliers[~influential_obs].copy()

X_log = df_without_outliers_log.drop('AQI', axis=1)
Y_log = df_without_outliers_log['AQI']

Y_transformed_log = np.log1p(Y_log)

X_log = sm.add_constant(X_log)

#Splitting into test and train data
X_train_log, X_test_log, y_train_log, y_test_log = train_test_split(X_log,
    Y_transformed_log, test_size=0.2, random_state=42)

# Fit the OLS model with the transformed response variable
model_log = LinearRegression()
model_log.fit(X_train_log, y_train_log)
model_transformed_new_log = sm.OLS(y_train_log, X_train_log).fit()

# Make predictions on the test data
y_pred_transformed_log = model_transformed_new_log.predict(X_test_log)
```

Q-Q PLOT:

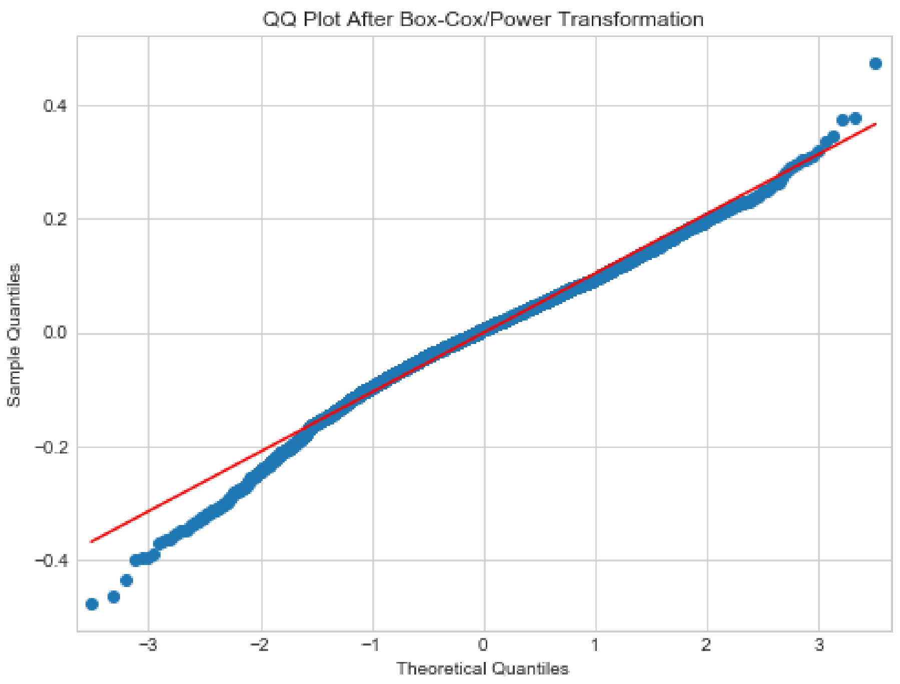


6.3.2 Box-Cox Transformation

```
X = df_without_outliers.drop('AQI', axis=1)
Y = df_without_outliers['AQI']

# Add a constant term to the predictor variables
X = sm.add_constant(X)

# Apply a logarithmic transformation to the response variable
Y_transformed, lambda_value = boxcox(Y)

# Fit the OLS model with the transformed response variable
model_transformed = sm.OLS(Y_transformed, X).fit()

# Get the Cook's distance and other influence statistics
influence = OLSInfluence(model_transformed)
cooks_distance = influence.cooks_distance
cooks_distance_values = cooks_distance[0]

# Set a threshold to identify influential observations
threshold = 4 / len(y)
influential_obs = cooks_distance_values > threshold

# Print or do something with the influential observations
# print("Influential observations:", display(data_without_outliers[
    influential_obs]))
df_without_outliers_power = df_without_outliers[~influential_obs].copy()

X_power = df_without_outliers_power.drop('AQI', axis=1)
Y_power = df_without_outliers_power['AQI']

Y_transformed_power, lambda_value = boxcox(Y_power)
```

```

X_power = sm.add_constant(X_power)

#Splitting into test and train data
X_train_power, X_test_power, y_train_power, y_test_power = train_test_split(
    X_power, Y_transformed_power, test_size=0.2, random_state=42)

# Fit the OLS model with the transformed response variable
model_transformed_new_power = sm.OLS(y_train_power, X_train_power).fit()

# Make predictions on the test data
y_pred_transformed_power = model_transformed_new_power.predict(X_test_power)
```

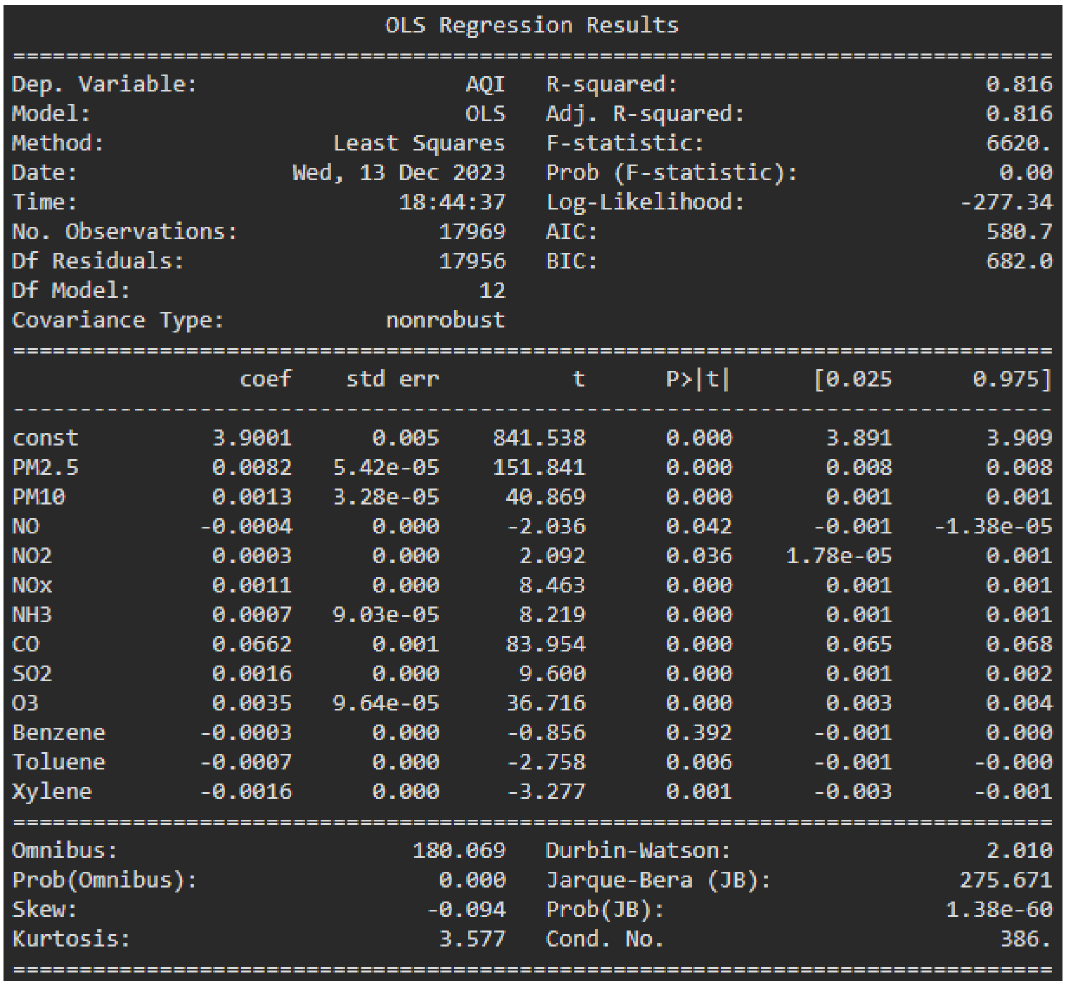
Q-Q PLOT: Observation:



We can see that the Normal Q-Q Plot for the residuals after log transformation is better than that for the box-cox/power transformation.
So we will use the data set after the log transformation.

6.4 Log Transformed data frame

6.4.1 OLS model for log-transformed data

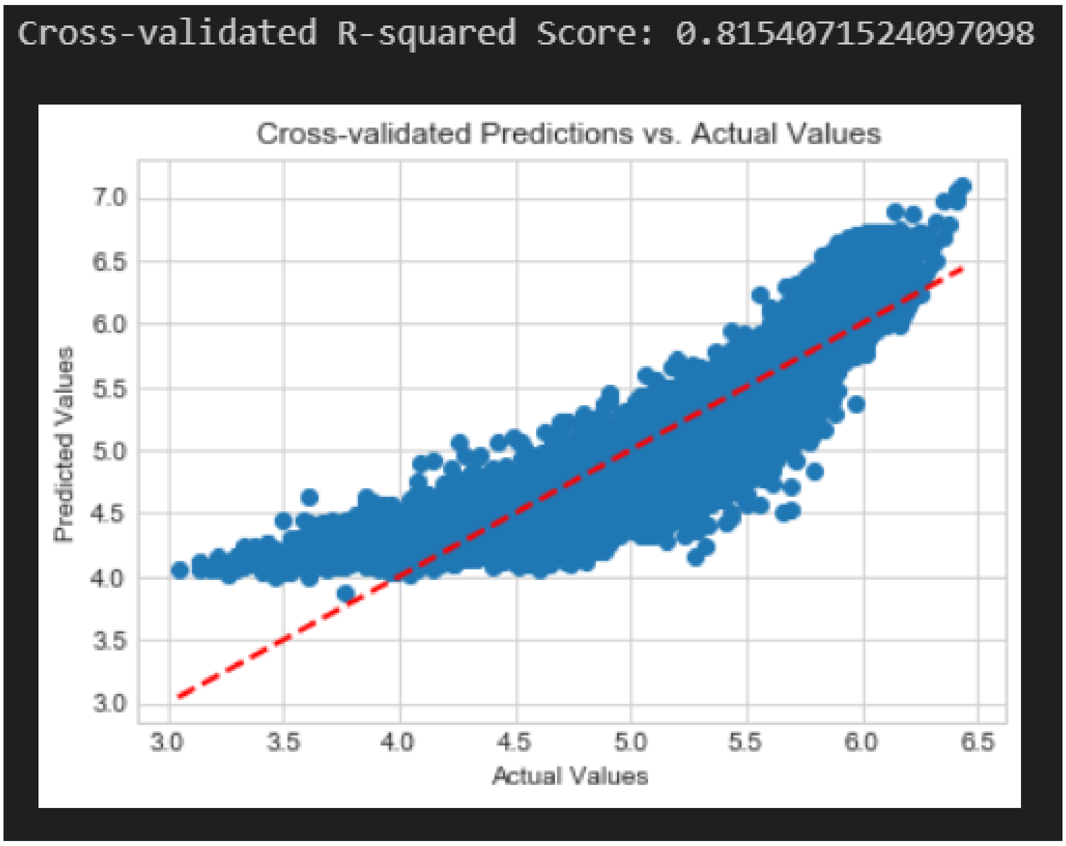


```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    AQI   R-squared:                       0.816
Model:                            OLS   Adj. R-squared:                  0.816
Method:                 Least Squares   F-statistic:                     6620.
Date:                Wed, 13 Dec 2023   Prob (F-statistic):               0.00
Time:                        18:44:37   Log-Likelihood:                -277.34
No. Observations:               17969   AIC:                             580.7
Df Residuals:                   17956   BIC:                             682.0
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          3.9001      0.005    841.538      0.000       3.891       3.909
PM2.5          0.0082   5.42e-05    151.841      0.000       0.008       0.008
PM10           0.0013   3.28e-05     40.869      0.000       0.001       0.001
NO            -0.0004      0.000     -2.036      0.042      -0.001   -1.38e-05
NO2            0.0003      0.000      2.092      0.036    1.78e-05       0.001
NOx            0.0011      0.000      8.463      0.000       0.001       0.001
NH3            0.0007   9.03e-05      8.219      0.000       0.001       0.001
CO             0.0662      0.001     83.954      0.000       0.065       0.068
SO2            0.0016      0.000      9.600      0.000       0.001       0.002
O3             0.0035   9.64e-05     36.716      0.000       0.003       0.004
Benzene       -0.0003      0.000     -0.856      0.392      -0.001       0.000
Toluene       -0.0007      0.000     -2.758      0.006      -0.001      -0.000
Xylene        -0.0016      0.000     -3.277      0.001      -0.003      -0.001
==============================================================================
Omnibus:                      180.069   Durbin-Watson:                   2.010
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              275.671
Skew:                          -0.094   Prob(JB):                     1.38e-60
Kurtosis:                       3.577   Cond. No.                         386.
==============================================================================
```

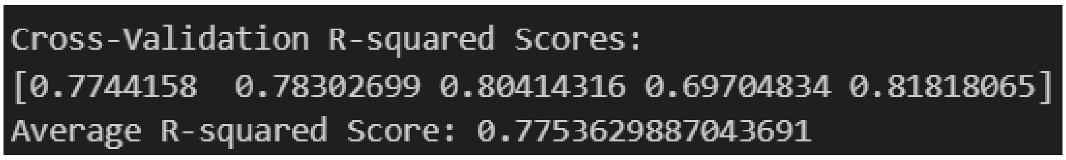
Observation:

We see that there is a decrease in the adjusted R-squared values from 0.910 to 0.816, which has been traded for an improvement in the trace of the QQ plot w.r.t the normal line.

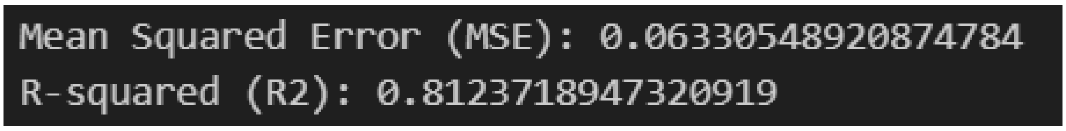
6.4.2 Cross-Validation



```
Cross-validated R-squared Score: 0.8154071524097098
```



```
Cross-Validation R-squared Scores:
[0.7744158  0.78302699 0.80414316 0.69704834 0.81818065]
Average R-squared Score: 0.7753629887043691
```



```
Mean Squared Error (MSE): 0.06330548920874784
R-squared (R2): 0.8123718947320919
```

7 CHECK FOR MULTI-COLLINEARITY

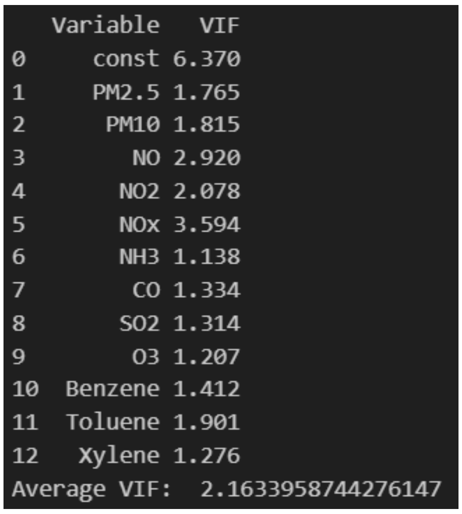
When two or more predictors in a model are correlated to each other, this is referred to as multi-collinearity. This would cause problems with unstable estimates and variances, resulting in poor hypothesis testing.

Here, we check for multi-collinearity on the log-transformed data set.
This can be done using two methods:

1. Variance Inflation Factor (VIF):

   - Calculate the VIF and its average for each feature.

```python
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Separate predictor variables (X) and dependent variable (y)
X = df_without_outliers_log.drop('AQI', axis=1)
y = df_without_outliers_log['AQI']

correlation_matrix = X.corr()

# Add a constant term to the predictor variables
X = sm.add_constant(X)

# Calculate VIF for each predictor
vif_data = pd.DataFrame()
vif_data['Variable'] = X.columns
vif_data['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
average_vif = vif_data['VIF'].mean()

# Display the VIF values
print(vif_data)
print("Average VIF: ", average_vif)

```



```
   Variable   VIF
0     const 6.370
1     PM2.5 1.765
2      PM10 1.815
3        NO 2.920
4       NO2 2.078
5       NOx 3.594
6       NH3 1.138
7        CO 1.334
8       SO2 1.314
9        O3 1.207
10  Benzene 1.412
11  Toluene 1.901
12   Xylene 1.276
Average VIF:  2.1633958744276147
```

Figure 37: VIF for each feature

- Observations:
  - The VIF values indicate that multi-collinearity is relatively low in our model. The average VIF is around 2.16, which is generally considered low to moderate. This means that, on average, the variance of the estimated coefficients is inflated by a factor of 2.16 due to multi-collinearity.
  - All individual VIF values are below 5, with the constant variable having a slightly higher VIF of 6.37. It's worth noting that a higher VIF for the constant term is common and generally not a concern.
  - The highest individual VIF is for the NOx variable at 3.59, which is still below the commonly used threshold of 5. Overall, these VIF values suggest that multi-collinearity is not a severe issue in our model.
  - As we evaluate the regression model, we'll also consider other diagnostic measures and contextual factors. If our model is providing stable and meaningful results, the current level of multi-collinearity appears to be acceptable. It's important to remember that these are guidelines, and the interpretation of VIF values depends on the specific context of our analysis.

Conslusion: We do this by calculating the Variance Inflation Factor for each feature and calculating its average. None of the features has VIF greater than 10 and the average VIF is not much greater than 1. We can conclude that there is not Multicollinearity in the data.

2. Correlation Matrix and Condition Number:

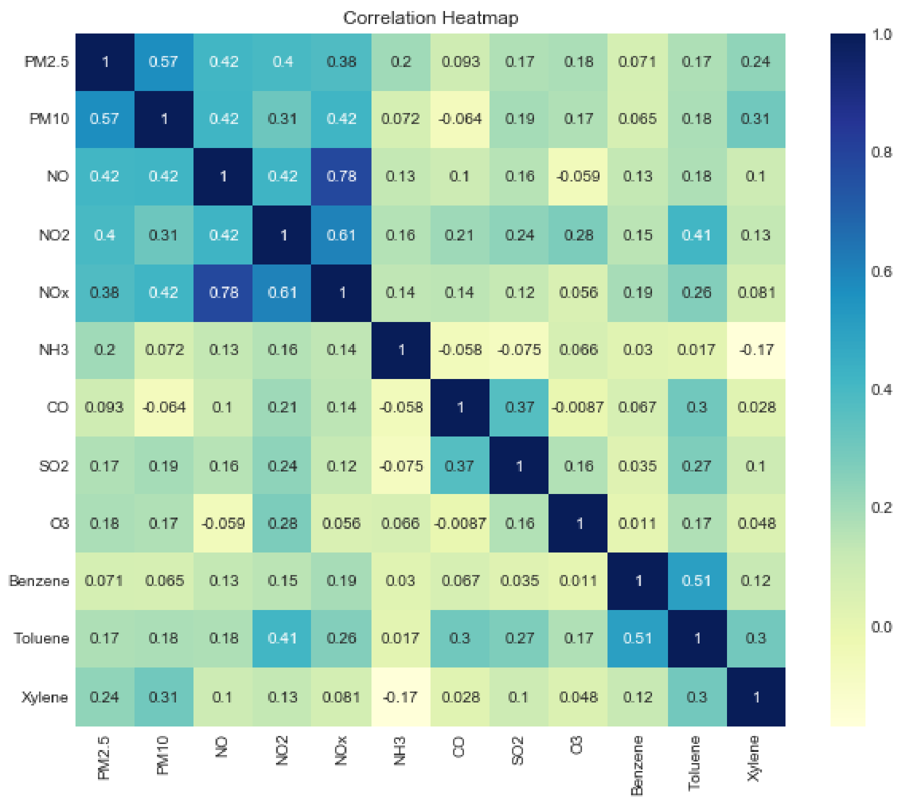
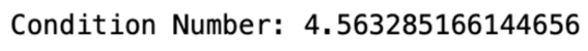


Figure 38: Correlation Matrix

```python
eigenvalues = np.linalg.eigvals(correlation_matrix)
condition_number = np.sqrt(np.max(eigenvalues) / np.min(eigenvalues))

# Display the condition number
print("Condition Number:", condition_number)

```



```
Condition Number: 4.563285166144656
```

Figure 39: Condition number generated

- Observation:
  - We observe that the condition number of 4.56 is relatively low. Lower condition numbers are indicative of better numerical stability, and a value below 30 is often considered satisfactory.
  - Given this condition number, it suggests that the numerical stability of our regression coefficients is not a major concern. This implies that the coefficients are not highly sensitive to small changes in the data due to multi-collinearity.
  - The correlation matrix also does not show any pair of independent variables that might be highly correlated.
  - Therefore, we can conclude that no significant correlation exists between any of the independent variables.
    Hence, there is no change in the data set after the log transformation.

8 MODEL SELECTION USING AIC/BIC

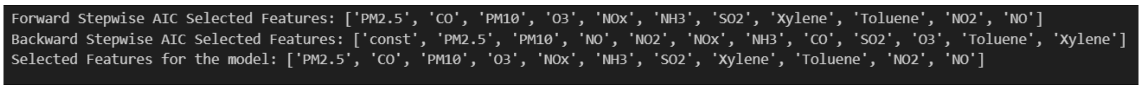
To select a model we perform the following steps:

1. Fit the model and calculate AIC:

    - We check for backward step-wise AIC elimination
    - We check for forward step-wise AIC elimination

```python
# Function to fit a model and calculate AIC
def calculate_aic(X, y):
    X = sm.add_constant(X)
    model = sm.OLS(y, X).fit()
    aic = model.aic
    return aic

# Forward Stepwise AIC
def forward_stepwise_aic(X, y):
    best_features = []
    remaining_features = list(X.columns)
    current_aic = float('inf')

    while remaining_features:
        best_aic = float('inf')
        best_feature = None

        for feature in remaining_features:
            candidate_features = best_features + [feature]
            aic = calculate_aic(X[candidate_features], y)

            if aic < best_aic:
                best_aic = aic
                best_feature = feature

        if best_aic < current_aic:
            current_aic = best_aic
            best_features.append(best_feature)
            remaining_features.remove(best_feature)
        else:
            break

    return best_features

# Backward Stepwise AIC
def backward_stepwise_aic(X, y):
    features = list(X.columns)
    best_features = features.copy()
    current_aic = calculate_aic(X[best_features], y)

    while len(features) > 1:
        best_aic = float('inf')
        worst_feature = None

        for feature in features:
            candidate_features = [f for f in best_features if f !=
   feature]
            aic = calculate_aic(X[candidate_features], y)

            if aic < best_aic:
                best_aic = aic
                worst_feature = feature

        if best_aic < current_aic:
            current_aic = best_aic
            best_features.remove(worst_feature)
        else:
            break

    return best_features

# Forward Stepwise AIC
forward_selected_features = forward_stepwise_aic(X_train_log,
    y_train_log)
print("Forward Stepwise AIC Selected Features:",
    forward_selected_features)

# Backward Stepwise AIC
backward_selected_features = backward_stepwise_aic(X_train_log,
    y_train_log)
print("Backward Stepwise AIC Selected Features:",
    backward_selected_features)

print("Selected Features for the model:", forward_selected_features)
```

Output:



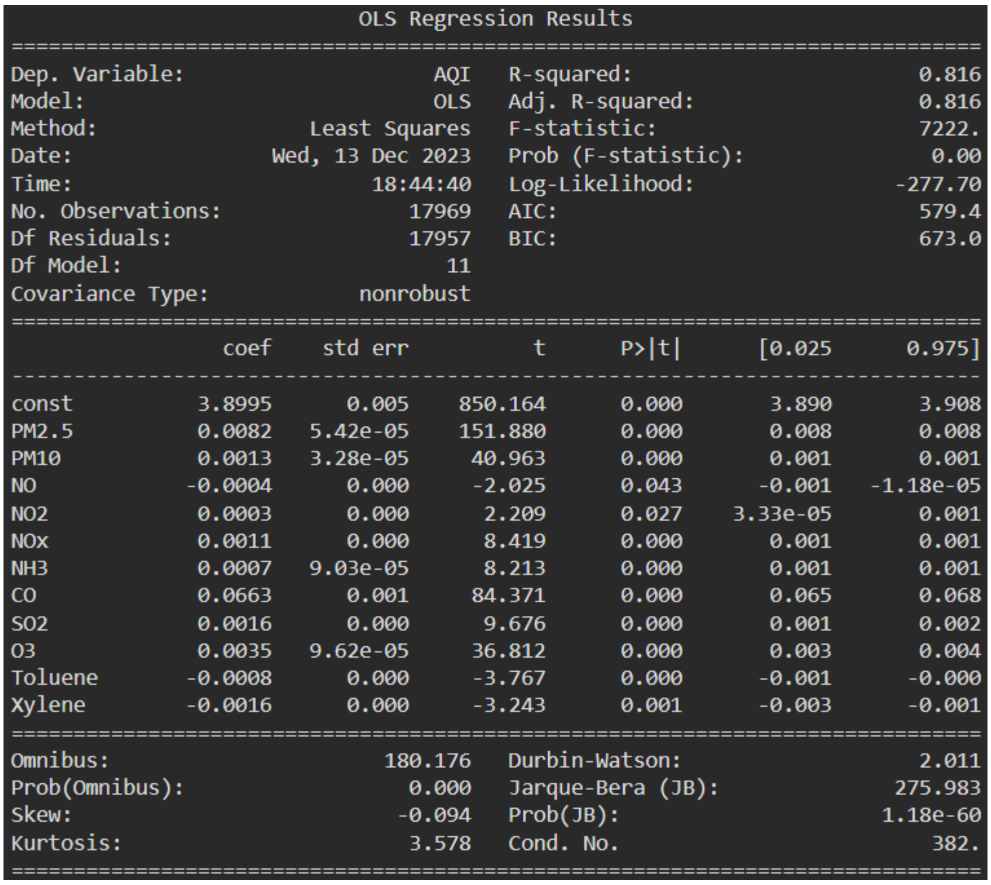
```
Forward Stepwise AIC Selected Features: ['PM2.5', 'CO', 'PM10', 'O3', 'NOx', 'NH3', 'SO2', 'Xylene', 'Toluene', 'NO2', 'NO']
Backward Stepwise AIC Selected Features: ['const', 'PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3', 'CO', 'SO2', 'O3', 'Toluene', 'Xylene']
Selected Features for the model: ['PM2.5', 'CO', 'PM10', 'O3', 'NOx', 'NH3', 'SO2', 'Xylene', 'Toluene', 'NO2', 'NO']
```

Figure 40: AIC elimination

From the above, we choose forward elimination and observe that Benzene has been eliminated as it does not have much significance in the model.

2. Drop the eliminated variable from the transformed data frame.

3. Perform OLS on this data frame.



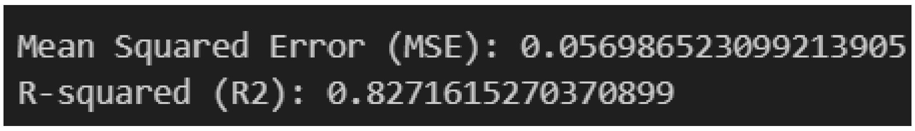
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    AQI   R-squared:                       0.816
Model:                            OLS   Adj. R-squared:                  0.816
Method:                 Least Squares   F-statistic:                     7222.
Date:                Wed, 13 Dec 2023   Prob (F-statistic):               0.00
Time:                        18:44:40   Log-Likelihood:                -277.70
No. Observations:               17969   AIC:                             579.4
Df Residuals:                   17957   BIC:                             673.0
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          3.8995      0.005    850.164      0.000       3.890       3.908
PM2.5          0.0082   5.42e-05    151.880      0.000       0.008       0.008
PM10           0.0013   3.28e-05     40.963      0.000       0.001       0.001
NO            -0.0004      0.000     -2.025      0.043      -0.001   -1.18e-05
NO2            0.0003      0.000      2.209      0.027    3.33e-05       0.001
NOx            0.0011      0.000      8.419      0.000       0.001       0.001
NH3            0.0007   9.03e-05      8.213      0.000       0.001       0.001
CO             0.0663      0.001     84.371      0.000       0.065       0.068
SO2            0.0016      0.000      9.676      0.000       0.001       0.002
O3             0.0035   9.62e-05     36.812      0.000       0.003       0.004
Toluene       -0.0008      0.000     -3.767      0.000      -0.001      -0.000
Xylene        -0.0016      0.000     -3.243      0.001      -0.003      -0.001
==============================================================================
Omnibus:                      180.176   Durbin-Watson:                   2.011
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              275.983
Skew:                          -0.094   Prob(JB):                     1.18e-60
Kurtosis:                       3.578   Cond. No.                         382.
==============================================================================
```

Figure 41: Model using transformed data

Final Regression Equation:

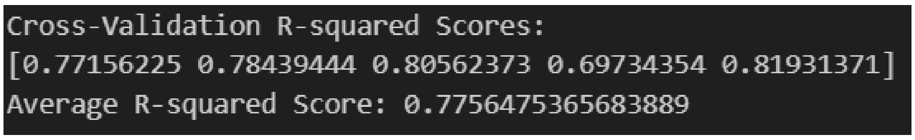
$$AQI = 3.8995 + 0.0082(PM2.5) + 0.0013(PM10) - 0.0004(NO) + 0.0003(NO2) + 0.0011(NOx)$$
$$+ 0.0007(NH3) + 0.0663(CO) + 0.0016(SO2) + 0.0035(O3) - 0.0008(Toluene) - 0.0016(Xylene)$$

4. Extract R-squared and MSE value.



```
Mean Squared Error (MSE): 0.056986523099213905
R-squared (R2): 0.8271615270370899
```

5. Calculate cross-validation and average R-squared



```
Cross-Validation R-squared Scores:
[0.77156225 0.78439444 0.80562373 0.69734354 0.81931371]
Average R-squared Score: 0.7756475365683889
```

6. Perform cross-validated predictions and evaluate the R-squared score on the training set.

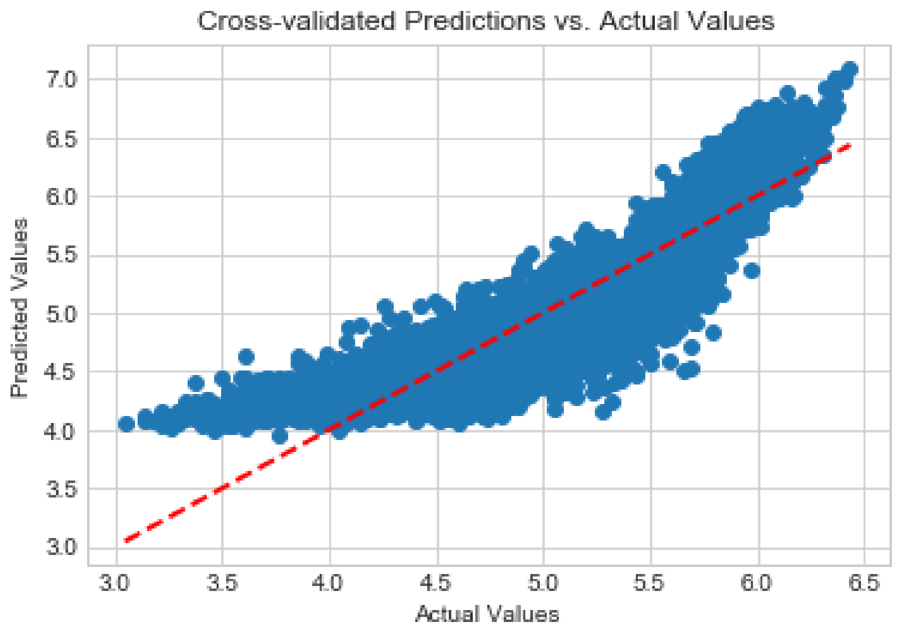
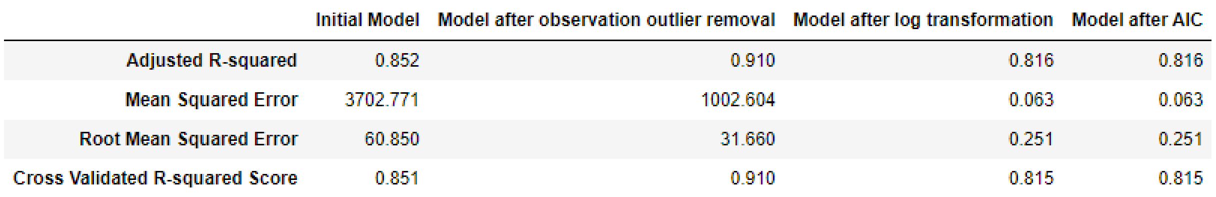


Figure 42: Plot of the actual vs predicted values

The cross-validated for this model is R-squared Score: 0.8131676123921064

9 CONCLUSION



	Initial Model	Model after observation outlier removal	Model after log transformation	Model after AIC
Adjusted R-squared	0.852	0.910	0.816	0.816
Mean Squared Error	3702.771	1002.604	0.063	0.063
Root Mean Squared Error	60.850	31.660	0.251	0.251
Cross Validated R-squared Score	0.851	0.910	0.815	0.815

Figure 43: Trace of model evaluations

1. **FINAL REGRESSION EQUATION:**

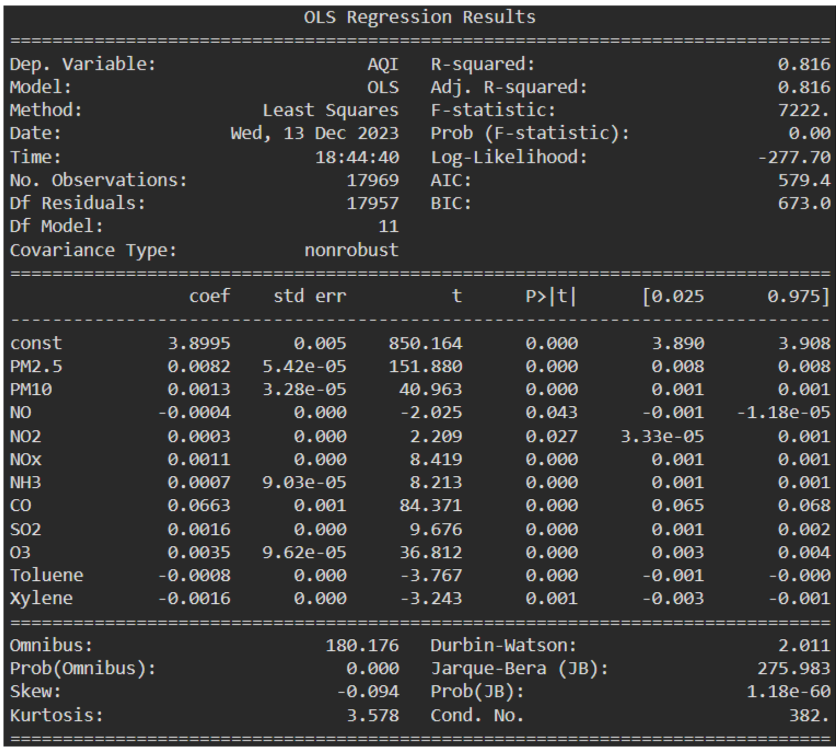
$$AQI = 3.8995 + 0.0082(PM2.5) + 0.0013(PM10) – 0.0004(NO) + 0.0003(NO2)$$

$$+0.0011(NOx) + 0.0007(NH3) + 0.0663(CO) + 0.0016(SO2) + 0.0035(O3) –$$

$$0.0008(Toluene) - 0.0016(Xylene)$$

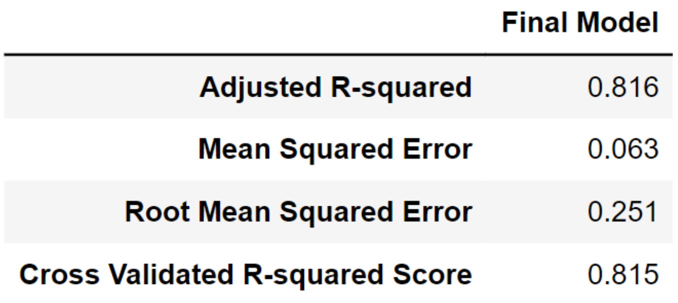
2. **Adjusted R-squared - 0.816**
   Hence, we can imply that 81.6 % of the variance in the AQI is explained by the regressor variables.

3. Each of the regressors has a p-value greater than 0.05, which implies that each of them is significant in calculating AQI.



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    AQI   R-squared:                       0.816
Model:                            OLS   Adj. R-squared:                  0.816
Method:                 Least Squares   F-statistic:                     7222.
Date:                Wed, 13 Dec 2023   Prob (F-statistic):               0.00
Time:                        18:44:40   Log-Likelihood:                -277.70
No. Observations:               17969   AIC:                             579.4
Df Residuals:                   17957   BIC:                             673.0
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          3.8995      0.005    850.164      0.000       3.890       3.908
PM2.5          0.0082   5.42e-05    151.880      0.000       0.008       0.008
PM10           0.0013   3.28e-05     40.963      0.000       0.001       0.001
NO            -0.0004      0.000     -2.025      0.043      -0.001   -1.18e-05
NO2            0.0003      0.000      2.209      0.027    3.33e-05       0.001
NOx            0.0011      0.000      8.419      0.000       0.001       0.001
NH3            0.0007   9.03e-05      8.213      0.000       0.001       0.001
CO             0.0663      0.001     84.371      0.000       0.065       0.068
SO2            0.0016      0.000      9.676      0.000       0.001       0.002
O3             0.0035   9.62e-05     36.812      0.000       0.003       0.004
Toluene       -0.0008      0.000     -3.767      0.000      -0.001      -0.000
Xylene        -0.0016      0.000     -3.243      0.001      -0.003      -0.001
==============================================================================
Omnibus:                      180.176   Durbin-Watson:                   2.011
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              275.983
Skew:                          -0.094   Prob(JB):                     1.18e-60
Kurtosis:                       3.578   Cond. No.                         382.
==============================================================================
```

Figure 44: Model using transformed data



	Final Model
Adjusted R-squared	0.816
Mean Squared Error	0.063
Root Mean Squared Error	0.251
Cross Validated R-squared Score	0.815

Figure 45: Performance metrics of the final model

References

[1] [Online]. Available: `https://cpcb.nic.in/`.

[2] A. Bilogur. [Online]. Available: `https://github.com/ResidentMario/missingno/tree/master`.

[3] R. J. Little and D. B. Rubin, "Statistical analysis with missing data," 2019.

[4] W. McKinney, "Data structures for statistical computing in python," *Proceedings of the 9th Python in Science Conference*, 2010.

[5] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science Engineering*, vol. 9, 2007. DOI: `10.1109/MCSE.2007.55`.

[6] K. Jordahl, "Geopandas: Python tools for geographic data," *Journal of Open Source Software*, vol. 2, no. 15, p. 352, 2017. DOI: `10.21105/joss.00201`. [Online]. Available: `https://joss.theoj.org/papers/10.21105/joss.00201`.

[7] M. Waskom *et al.*, *Mwaskom/seaborn: V0.8.1 (september 2017)*, Sep. 2017. DOI: `10.5281/zenodo.883859`. [Online]. Available: `https://doi.org/10.5281/zenodo.883859`.

[8] D. C. Montgomery, *Design and Analysis of Experiments*. Wiley, 2017.

[9] A. Wald, "A test of the equality of dependent correlation coefficients," *Annals of Mathematical Statistics*, vol. 11, no. 2, pp. 324–337, 1940. DOI: `10.1214/aoms/1177731828`. [Online]. Available: `https://projecteuclid.org/euclid.aoms/1177731828`.

[10] C. F. Gauss, "Ordinary least squares regression," 1809, Historical reference, first introduced by Carl Friedrich Gauss.

[11] S. Seabold and J. Perktold, "Statsmodels: Econometric and statistical modeling with python," *Proceedings of the 9th Python in Science Conference*, vol. 57, pp. 61–66, 2010. [Online]. Available: `http://conference.scipy.org/proceedings/scipy2010/seabold.html`.

[12] C. Mencía and A. García-Serrano, "One-hot encoding for categorical data: An overview," *Data Science Journal*, vol. 18, no. 0, pp. 1–13, 2019. DOI: `10.5334/dsj-2019-001`. [Online]. Available: `https://datascience.codata.org/articles/10.5334/dsj-2019-001/`.

[13] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, no. 3-4, pp. 591–611, 1965. DOI: `10.2307/2333709`. [Online]. Available: `https://www.jstor.org/stable/2333709`.

[14] J. Durbin and G. S. Watson, "Testing for serial correlation in least squares regression. i," *Biometrika*, vol. 37, no. 3/4, pp. 409–428, 1950. DOI: `10.2307/2332391`. [Online]. Available: `https://www.jstor.org/stable/2332391`.

[15] R. D. Cook, "Detection of influential observation in linear regression," *Technometrics*, vol. 19, no. 1, pp. 15–18, 1977. DOI: `10.2307/1268249`. [Online]. Available: `https://www.jstor.org/stable/1268249`.