DATS 6450 Final Term Project Report
Predicting Air Quality Index

Pranay Bhakthula
Professor : Reza Jafari
The George Washington University
15th December 2021

Table of Contents

Table of figures and tables

Abstract:

The focus of this project is to find the best time series model and predict the Air quality Index of 'Delhi' using various time series modelling methods. As the Air Quality Index (AQI) increases, it suggests that the pollution of the city is increasing. It is important for the Government to know the predictions of AQI and try to make policies so as to reduce the pollution of the city. In cities like 'Delhi', air pollution is a major concern and forecasts are required to warn residents to stay in homes, to reduce vehicle traffic and avoid the highways in order to curtail the accidents that happen due to smog/ fog in the months December - January.

Introduction:

In order to complete the objectives of this project we need to do data cleaning which includes filling the missing values with the mean of that column, then we find stationarity of the target variable using ACF plots, ADF, KPSS tests and if the target variables is non-stationary we differentiate the variable in order to make it stationary. We use base models like average method, naive method, drift method, SES method and Holt winter methods to model and predict the test values. These base models are like a lower bar set to compare the ARMA/ ARIMA models to say how much better these models are. We also use the OLS regression method to do feature reduction and also predict the test values. We use the GPAC table to find the na, nb values for the ARMA/ ARIMA models. We perform diagnostic tests on the ARMA/ARIMA models in order to check if the model is good enough in order to forecast. We then select the best model based on the mean, variance of residual and forecast error, plot the forecast values along the test values and see the difference between them.

The following are the theories behind the methods we used in this project.

Autocorrelation:
[source: DATS 6450 time series analysis lecture notes]
Auto correlation measures the linear relationship between lagged values of time series. It is the similarity between observations as a function of the time lag between them.
The notation used for the autocorrelation is Tk which gives the relationship between yt and yt-k.
Ry(t) is the estimated autocorrelation for stationary time series y(t).

$$\hat{R}_y(\tau) = \frac{\sum_{t=\tau+1}^{T}(y_t - \overline{y})(y_{t-\tau} - \overline{y})}{\sum_{t=1}^{T}(y_t - \overline{y})^2}$$

Average method:

[source: DATS 6450 time series analysis lecture notes]

The forecast of all future values are equal to the average of the historical data.

$$\hat{y}_{T+h|T} = \frac{y_1 + y_2 + \dots + y_T}{T}$$

The average method treats all the observations with equal importance. It gives equal weights while forecasting.

Naive method:

[source: DATS 6450 time series analysis lecture notes]

In the naive method the future value is equal to the last value. While forecast all the values are equal to the last value of the train data.

y|t+T = yT

Drift method:

[source: DATS 6450 time series analysis lecture notes]

In drift method the forecast values is equal to the last value plus the average change seen in the data. The forecast values graphs is just extrapolating the line connecting 1st and the last value of the train dataset.

$$\hat{y}_{T+h|T} = y_T + \frac{h}{T-1}\sum_{t=2}^{T}(y_t - y_{t-1}) = y_T + h\left(\frac{y_T - y_1}{T-1}\right)$$

Simple exponential smoothing (SES) method:

SES method is calculated using weighted averages, where least weight value is given the oldest observation from the current data.

$$\hat{y}_{T+1|T} = \sum_{j=0}^{T-1} \alpha(1-\alpha)^j y_{T-j} + (1-\alpha)^T \ell_0$$

Holt Winter method:

The Holt winter method uses three smoothing equations for the forecast equation so that it takes in level, trend and seasonality of the data into account while calculating the forecast values.

$$\begin{aligned}
\hat{y}_{t+h|t} &= \ell_t + hb_t + s_{t+h-m(k+1)} \\
\ell_t &= \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\
b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1} \\
s_t &= \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}
\end{aligned}$$

Where, lt - level, bt - trend, st - seasonality

ARMA model:

[Jiann-Fuh Chen, Wei-Ming Wang, Chao-Ming Huang,
Analysis of an adaptive time-series autoregressive moving-average (ARMA) model for short-term load forecasting,
Electric Power Systems Research,]

Autoregressive–moving-average (ARMA) models provide a description of a stationary stochastic process in terms of two polynomials, one for the autoregression (AR) and the second for the moving average (MA).

Regressing the variable on its own lagged (i.e., past) values is what the AR component entails. The MA portion entails modeling the error term as a linear combination of error terms that occur simultaneously and at different points in time in the past.

General form of ARMA:

$$\begin{aligned}
y(t) + a_1 y(t-1) + a_2 y(t-2) + \ldots + a_{n_a} y(t-n_a) &= \epsilon(t) + \\
& b_1\epsilon(t-1) + b_2\epsilon(t-2) + \ldots + b_{n_b}\epsilon(t-n_b)
\end{aligned}$$

ARIMA model:
[Schaffer, A.L., Dobbins, T.A. & Pearson, SA. Interrupted time series analysis using autoregressive integrated moving average (ARIMA) models: a guide for evaluating large-scale health interventions. *BMC Med Res Methodol* 21, 58 (2021). https://doi.org/10.1186/s12874-021-01235-8]

An autoregressive integrated moving average (ARIMA) model is a generalization of an autoregressive moving average (ARMA) model. Both of these models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting). ARIMA models are applied in some cases where data show evidence of non-stationarity in the sense of mean , where an initial differencing step which corresponds to the "integrated" part of the model, can be applied one or more times to eliminate the non-stationarity of the mean function.
General form of ARIMA:

$$(1+a_1 q^{-1}+\ldots+a_{n_a} q^{-n_a})(1-q^{-1})^d y(t) = (1+b_1 q^{-1}+\ldots+b_{n_b} q^{-n_b})\epsilon(t)$$

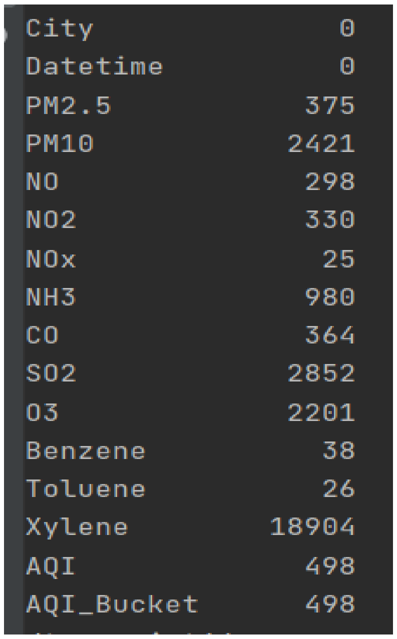
Description of the Dataset:

The dataset contains 707,875 observations which contains hourly data of Air quality Index and other pollutants for 26 cities in India in the period 2015-2020.

Out of these cities we have selected observations of ‘Delhi’ which has 48,192 observations. We are going to analyze and do the modelling on the Air Quality Index of Delhi.

a. Pre-processing dataset:

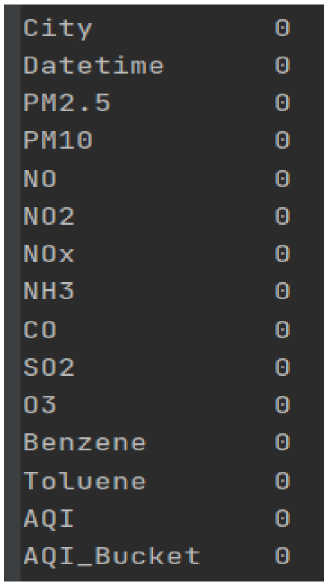
The columns of this dataset are:
'City', 'Datetime', 'PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3', 'CO',   'SO2', 'O3', 'Benzene', 'Toluene', 'Xylene', 'AQI', 'AQI_Bucket'.
The target variable is ‘AQI’ which refers to the Air Quality Index.
The missing values in the dataset are given below:



```
City              0
Datetime          0
PM2.5           375
PM10           2421
NO              298
NO2             330
NOx              25
NH3             980
CO              364
SO2            2852
O3             2201
Benzene          38
Toluene          26
Xylene        18904
AQI             498
AQI_Bucket      498
```

Figure:1 null values1

Since the missing values of ‘Xylene’ are too many, we have dropped that column.
For the rest of the numeric variables we have filled the missing values with the mean of respective columns.
For the categorical variables we have filled the missing values with the mode of the column.
The missing values in the dataset after data cleaning are given below:



```
City          0
Datetime      0
PM2.5         0
PM10          0
NO            0
NO2           0
NOx           0
NH3           0
CO            0
SO2           0
O3            0
Benzene       0
Toluene       0
AQI           0
AQI_Bucket    0
```

Figure:2 null values2

We have generated the Datetime variable from the date & time 2015-01-01 01:00:00 till 2020-07-01 00:00:00 and added it to the data frame since the given date time column is in string format and was difficult to convert to datatime format.

b. The plot of dependent variable versus the time:

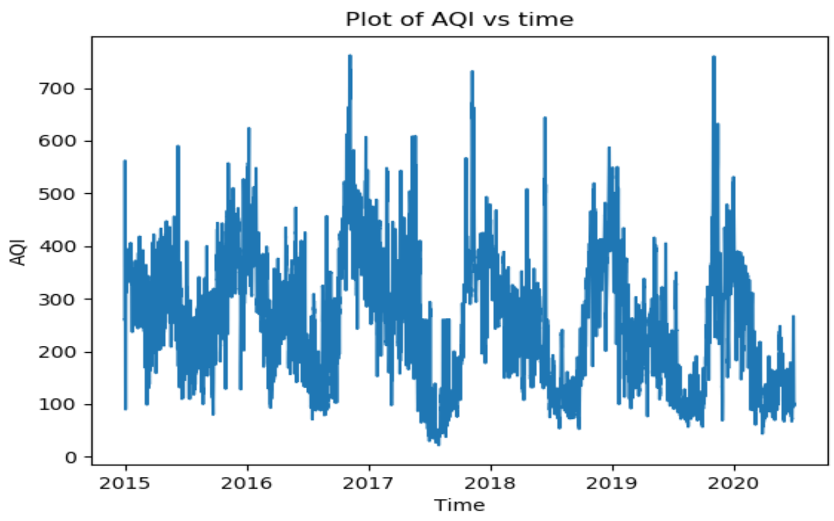


Figure:3 Plot of AQI vs Time

We can observe from the above graph that the AQI values are distributed within 22 - 762. The mean of AQI is 260.148. The standard deviation of AQi is 120.89. We can observe that AQI values are higher in the months of year ends than in the starting of the year.

c. ACF/PACF of the dependent variable:

The plot of ACF of the dependent variable is shown below:

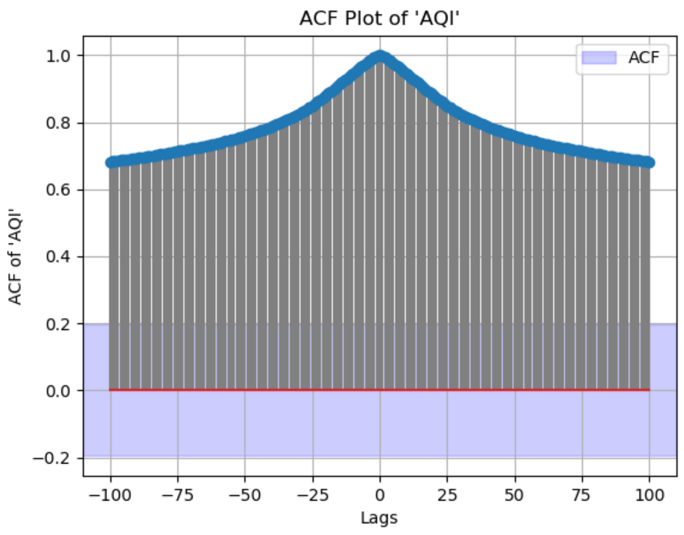


Figure:4 ACF plot of ‘AQI’

We can observe that the dependent variable does not tail-off to reach 0 value. So we can say that the variable is non-stationary.

The plot of PACF is shown below:

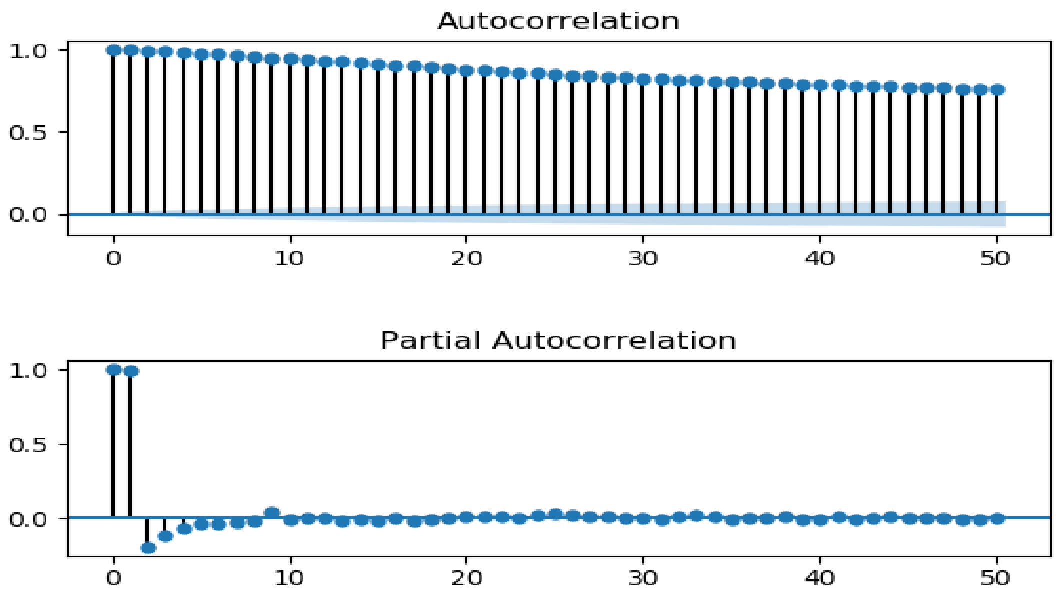


Figure:5 Autocorrelation and partial correlation

We can observe that the dependent variable cuts-off to reach in significant value for lag = 1 . So we can say that the nb value we can select as nb = 1.

d. Heatmap of pearson’s correlation coefficient:

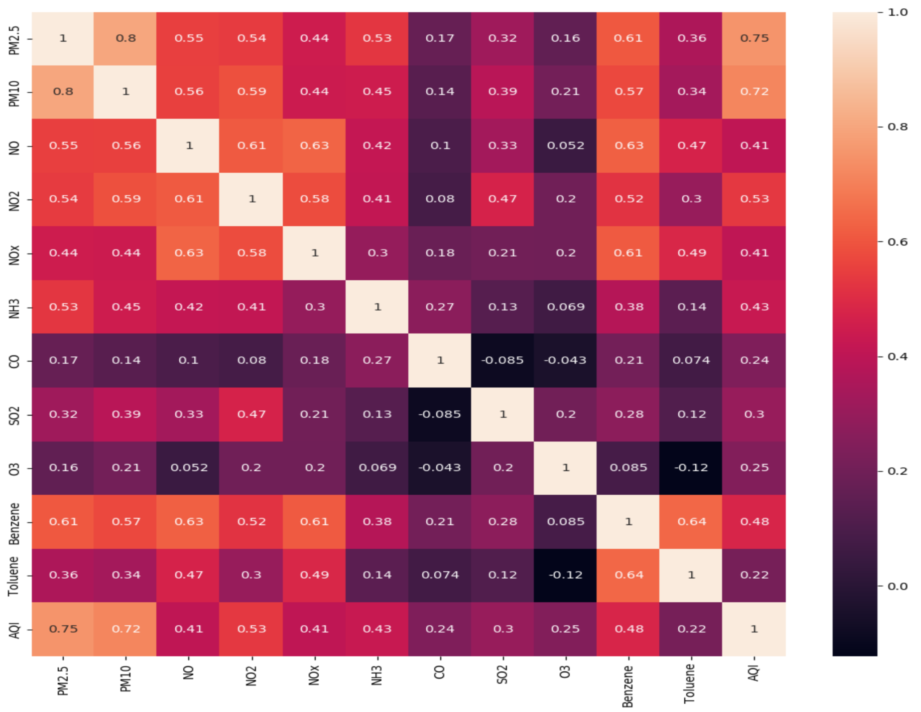
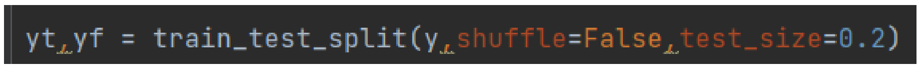


Figure: 6 Heatmap

We can observe that PM10, PM2.5 have high correlation with AQI. All the variables have positive correlation with the target variable.

e. Split the dataset into train and test values:

By using the following code line we can split the dataset into train and test datasets. Notice that shuffle=False since this is a time series dataset.



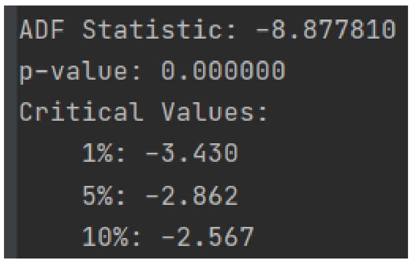
```
yt,yf = train_test_split(y,shuffle=False,test_size=0.2)
```

Figure:7 splitting data code

Stationarity check:

ADF test:

The results of the ADF tests on the raw data are shown below:



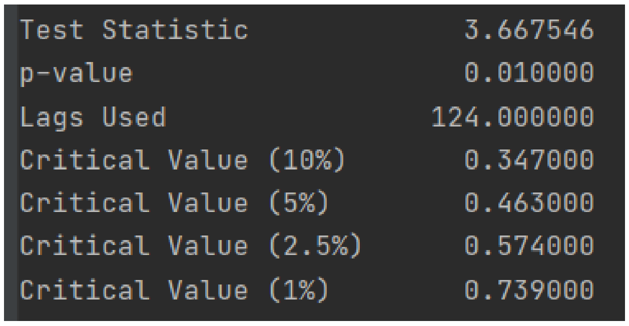
```
ADF Statistic: -8.877810
p-value: 0.000000
Critical Values:
    1%: -3.430
    5%: -2.862
    10%: -2.567
```

Figure:8 ADF statistic

From the ADF test results we can say that the variable is stationary since the p-value is less than 0.05.

KPSS test:

The results of the KPSS tests on the raw data are shown below:



```
Test Statistic              3.667546
p-value                     0.010000
Lags Used                 124.000000
Critical Value (10%)        0.347000
Critical Value (5%)         0.463000
Critical Value (2.5%)       0.574000
Critical Value (1%)         0.739000
```

Figure:9 Test Statistic

From the KPSS test results we can say that the variable is non - stationary since the p-value is less than 0.05.

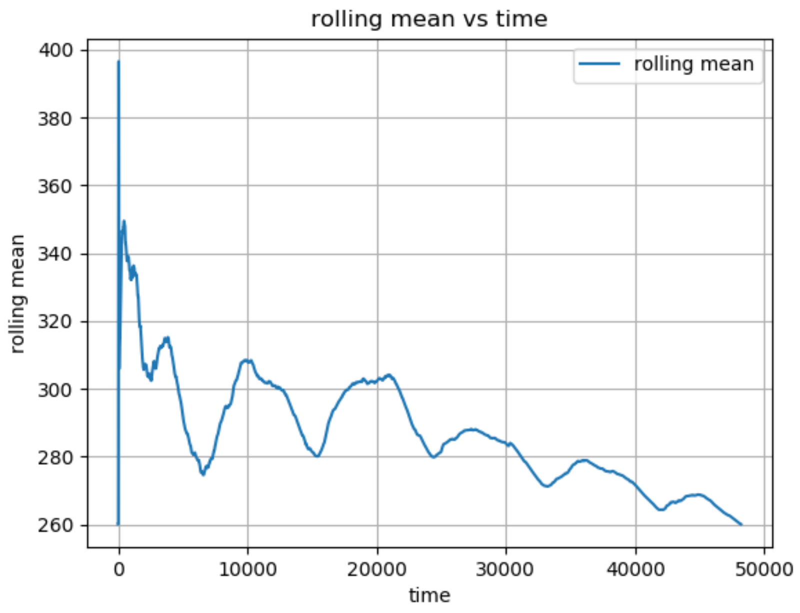


Figure:10 rolling mean vs time

From the rolling mean graph we can see that the trend is decreasing strongly and the values are not constant. So we can say that the variable is non-stationary.

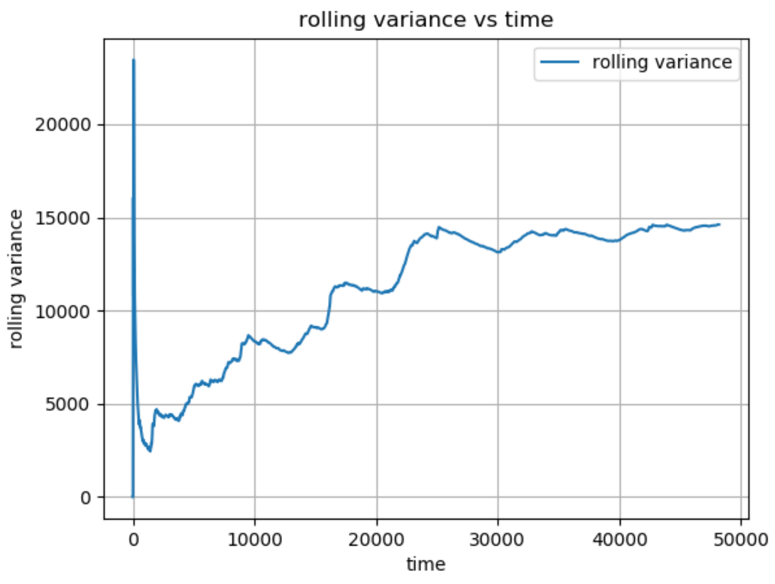


Figure:11 rolling variance vs time

From the rolling variance graph we can see that the value is increasing and the values are not constant. So we can say that the variable is non-stationary.

In order to make our target variable stationary we can use differencing of order 1. This is a commonly used method to make the variable stationary.

ACF plot for 1st order data:

Below is the ACF plot of the 1st order differenced data. We can observe that the variable is stationary since there is a tail off of the ACF values.

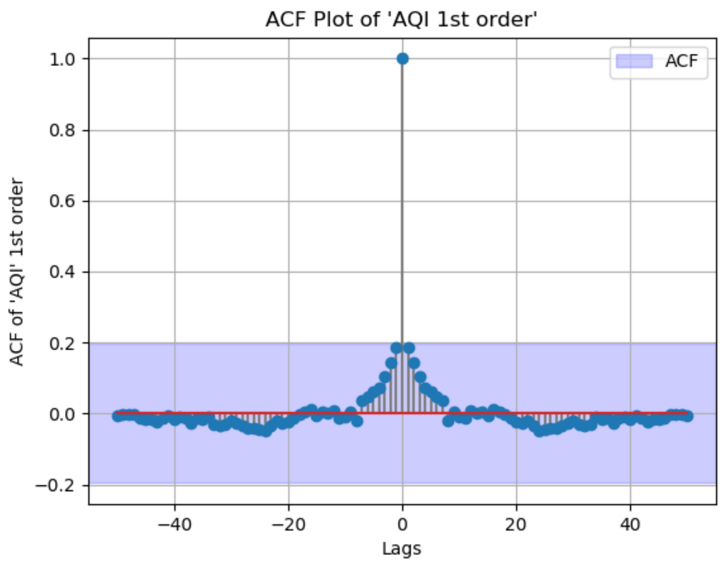
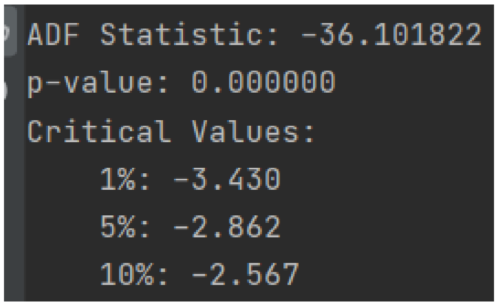


Figure: 12 ACF plot of ‘AQI 1st order’

ADF test on 1st order data:

The results of the ADF tests of the 1st order data are shown below:



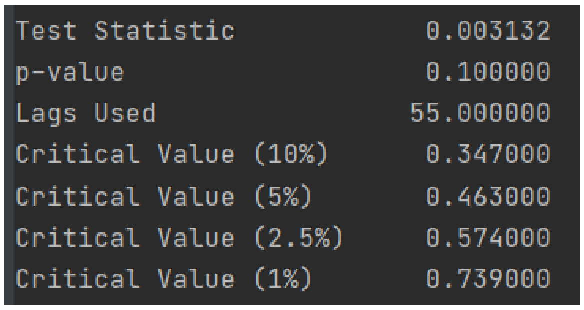
```
ADF Statistic: -36.101822
p-value: 0.000000
Critical Values:
    1%: -3.430
    5%: -2.862
    10%: -2.567
```

Figure:13 ADF test on 1st order data

From the ADF test results we can say that the variable is stationary, since the p-value is less than 0.05.

KPSS test on 1st order data:

The results of the KPSS tests on the raw data are shown below:



```
Test Statistic              0.003132
p-value                     0.100000
Lags Used                  55.000000
Critical Value (10%)        0.347000
Critical Value (5%)         0.463000
Critical Value (2.5%)       0.574000
Critical Value (1%)         0.739000
```

Figure:14 KPSS test on 1st order data

From the KPSS test results we can say that the variable is stationary, since the p-value is greater than 0.05.

Rolling mean and variance graphs of 1st order data:

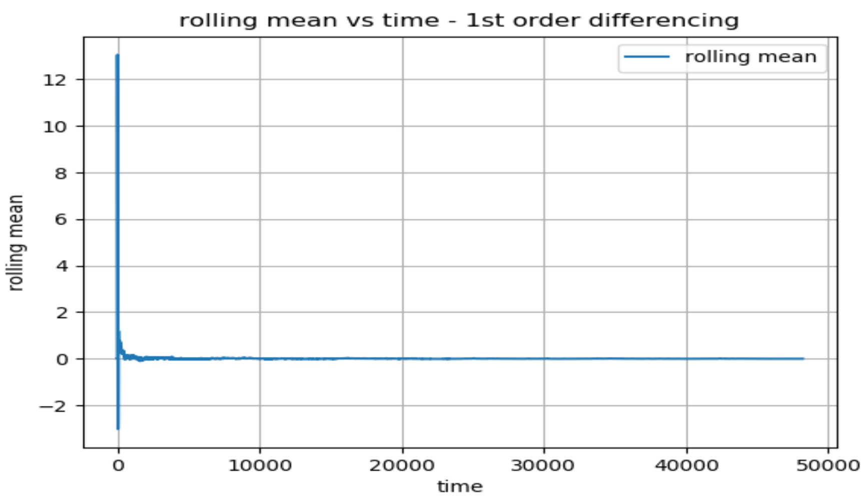


Figure:!5 rolling mean vs time -1st order differencing

From the rolling mean graph we can see that the values are constant and there is no trend. So we can say that the variable is stationary.

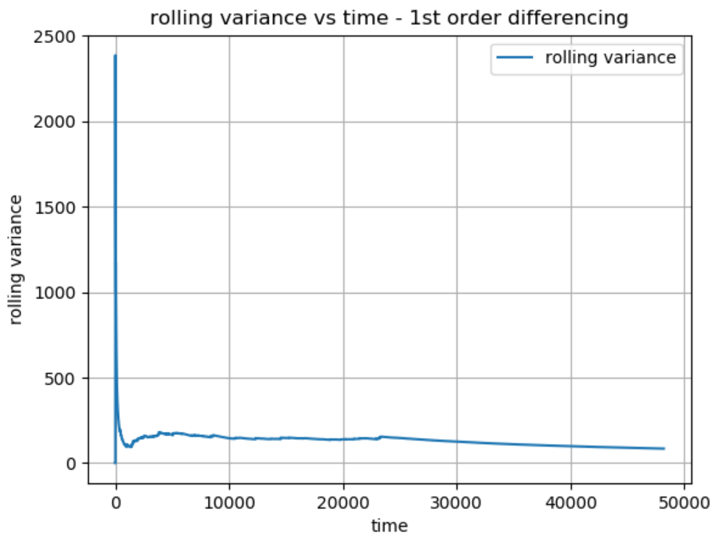
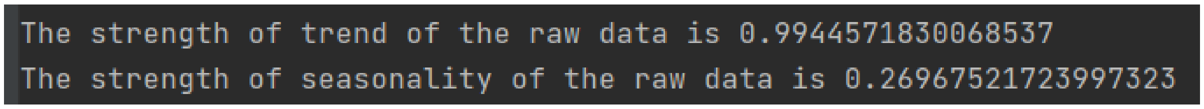


Figure:16 rolling variance vs time-1st order differencing

From the rolling variance graph we can see that the values are constant and there is no major trend. So we can say that the variable is stationary.

Time Series Decomposition:

We performed time series decomposition on the raw data and we get these results:



```
The strength of trend of the raw data is 0.9944571830068537
The strength of seasonality of the raw data is 0.26967521723997323
```

Figure:17 Time series Decomposition

We can observe that there is a very high trend in the raw data since the strength of the trend is very high at around ~ 0.99 and there is low seasonality in the raw data since the strength of the seasonality is low at around ~0.269.

Holt winter method:

We performed holt winter modelling on the raw data and we get these results:

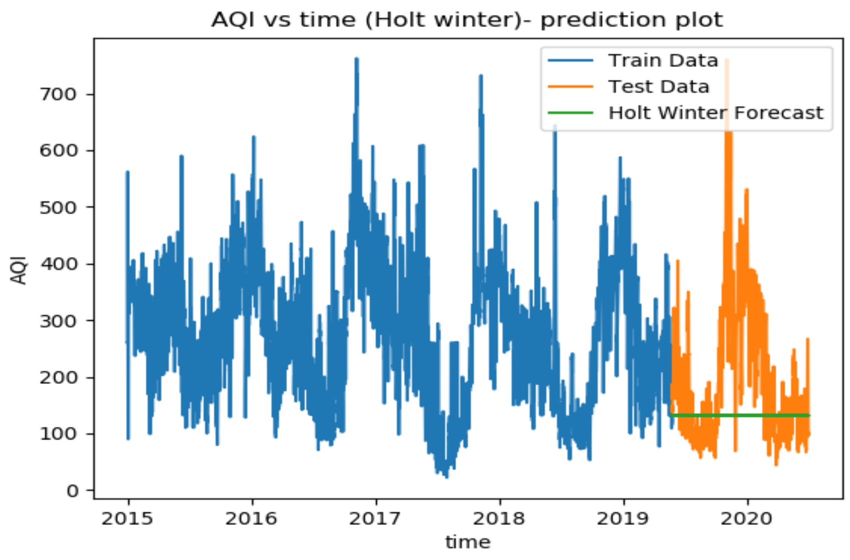


Figure:18 AQI vs time prediction plot

We can observe that the Holt winter method did not give good predictions as it is not following the test results.

Feature selection:

Shape of H is (12, 12)

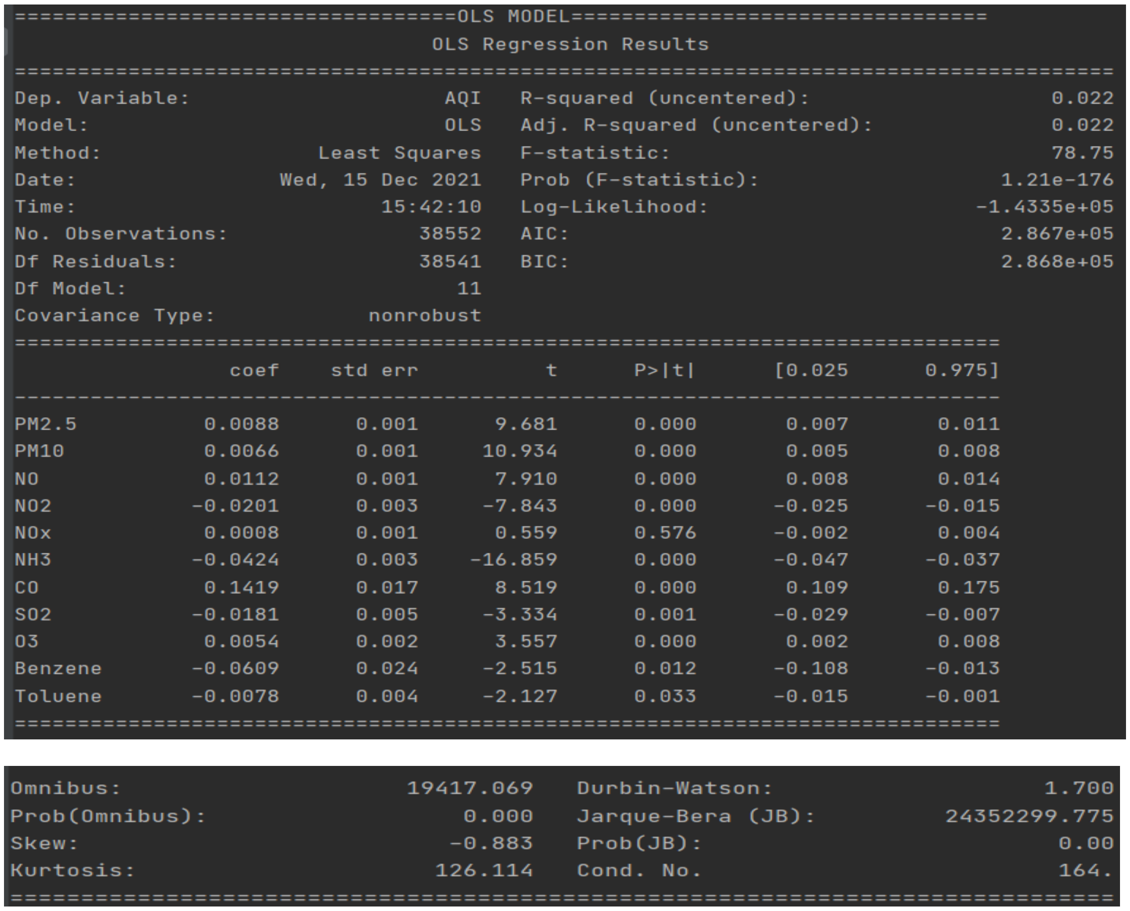
Singular Values [8.90020944e+09 2.69478145e+08 1.37153129e+08 1.25948535e+08
 5.70047990e+07 3.72612516e+07 2.14860955e+07 1.45793054e+07
 1.12590064e+07 3.45208219e+06 3.57062877e+05 1.73365033e+05]

The condition number is 226.5788457225573

Since condition number is greater than 100 and less than 1000, there is moderate degree of collinearity

unknown coefficients :
 [0.00876441 0.00661874 0.01115866 -0.02012788 0.00080782 -0.04241885
  0.14191928 -0.01810821 0.00537925 -0.0609149 -0.00782991]

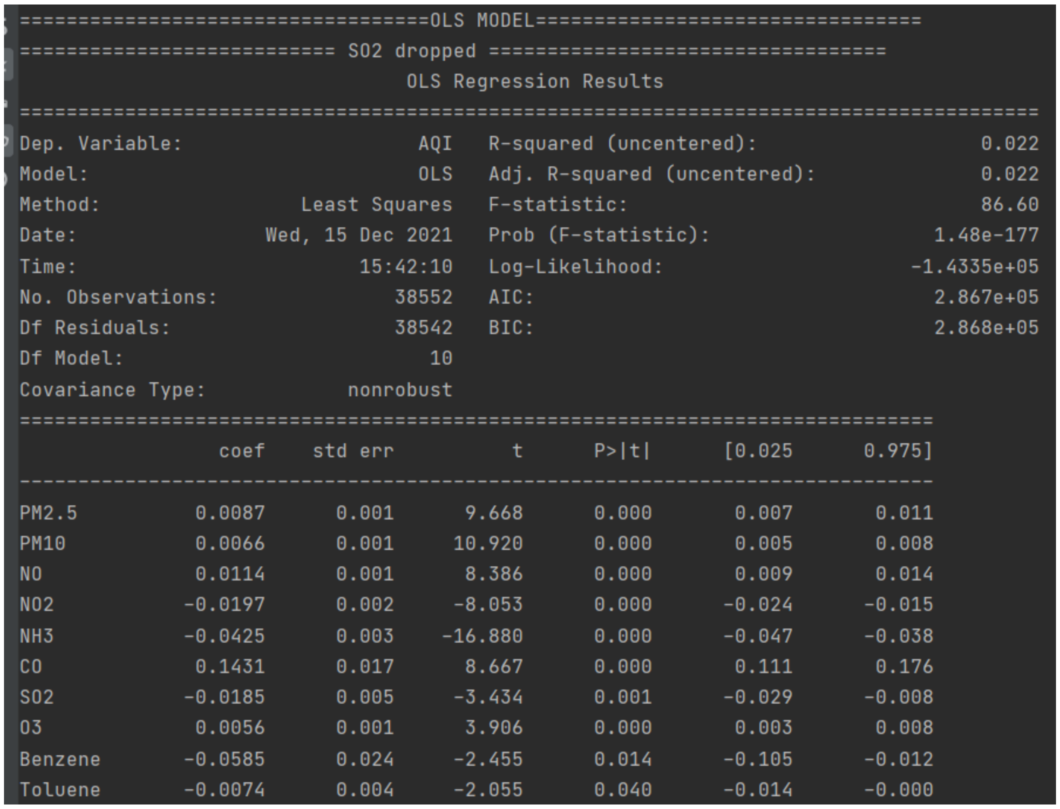
We perform feature selection of the dataset using the OLS regression. The results of the OLS regression are shown below:



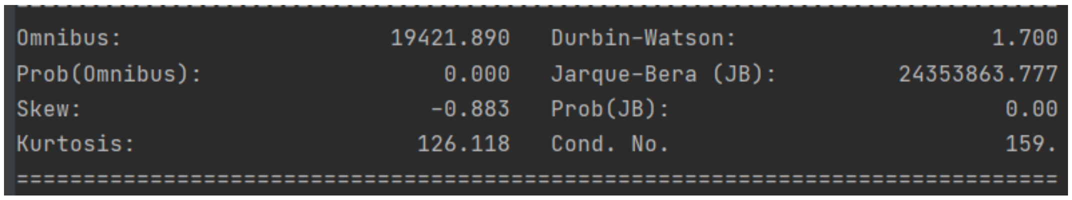
```
===================================OLS MODEL=================================
                                 OLS Regression Results
=======================================================================================
Dep. Variable:                    AQI   R-squared (uncentered):                   0.022
Model:                            OLS   Adj. R-squared (uncentered):              0.022
Method:                 Least Squares   F-statistic:                              78.75
Date:                Wed, 15 Dec 2021   Prob (F-statistic):                   1.21e-176
Time:                        15:42:10   Log-Likelihood:                     -1.4335e+05
No. Observations:               38552   AIC:                                  2.867e+05
Df Residuals:                   38541   BIC:                                  2.868e+05
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
PM2.5          0.0088      0.001      9.681      0.000       0.007       0.011
PM10           0.0066      0.001     10.934      0.000       0.005       0.008
NO             0.0112      0.001      7.910      0.000       0.008       0.014
NO2           -0.0201      0.003     -7.843      0.000      -0.025      -0.015
NOx            0.0008      0.001      0.559      0.576      -0.002       0.004
NH3           -0.0424      0.003    -16.859      0.000      -0.047      -0.037
CO             0.1419      0.017      8.519      0.000       0.109       0.175
SO2           -0.0181      0.005     -3.334      0.001      -0.029      -0.007
O3             0.0054      0.002      3.557      0.000       0.002       0.008
Benzene       -0.0609      0.024     -2.515      0.012      -0.108      -0.013
Toluene       -0.0078      0.004     -2.127      0.033      -0.015      -0.001
==============================================================================

Omnibus:                    19417.069   Durbin-Watson:                       1.700
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             24352299.775
Skew:                          -0.883   Prob(JB):                             0.00
Kurtosis:                     126.114   Cond. No.                             164.
==============================================================================
```

Figure:19 Feature selection 1

Here we can observe that NOx has a p-value of 0.576, which is high so we can remove this variable and run the regression again.



```
==================================OLS MODEL================================
=========================== SO2 dropped ===================================
                                 OLS Regression Results
=======================================================================================
Dep. Variable:                    AQI   R-squared (uncentered):                   0.022
Model:                            OLS   Adj. R-squared (uncentered):              0.022
Method:                 Least Squares   F-statistic:                              86.60
Date:                Wed, 15 Dec 2021   Prob (F-statistic):                   1.48e-177
Time:                        15:42:10   Log-Likelihood:                     -1.4335e+05
No. Observations:               38552   AIC:                                  2.867e+05
Df Residuals:                   38542   BIC:                                  2.868e+05
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
PM2.5          0.0087      0.001      9.668      0.000       0.007       0.011
PM10           0.0066      0.001     10.920      0.000       0.005       0.008
NO             0.0114      0.001      8.386      0.000       0.009       0.014
NO2           -0.0197      0.002     -8.053      0.000      -0.024      -0.015
NH3           -0.0425      0.003    -16.880      0.000      -0.047      -0.038
CO             0.1431      0.017      8.667      0.000       0.111       0.176
SO2           -0.0185      0.005     -3.434      0.001      -0.029      -0.008
O3             0.0056      0.001      3.906      0.000       0.003       0.008
Benzene       -0.0585      0.024     -2.455      0.014      -0.105      -0.012
Toluene       -0.0074      0.004     -2.055      0.040      -0.014      -0.000
```

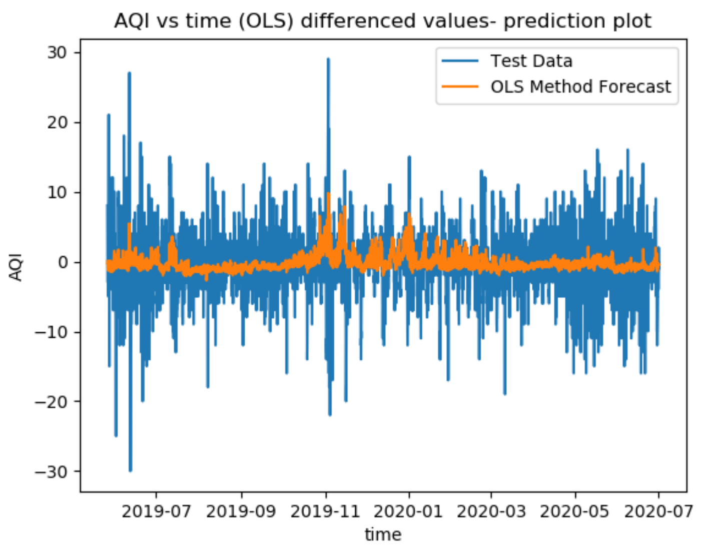


```
------------------------------------------------------------------------------
Omnibus:                    19421.890   Durbin-Watson:                   1.700
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         24353863.777
Skew:                          -0.883   Prob(JB):                         0.00
Kurtosis:                     126.118   Cond. No.                         159.
==============================================================================
```

Figure:20 Feature selection 2

Since none of the variables has a high p-value we don't need to drop any variable any more. We can perform predictions with this.

The predictions of the OLS model is:



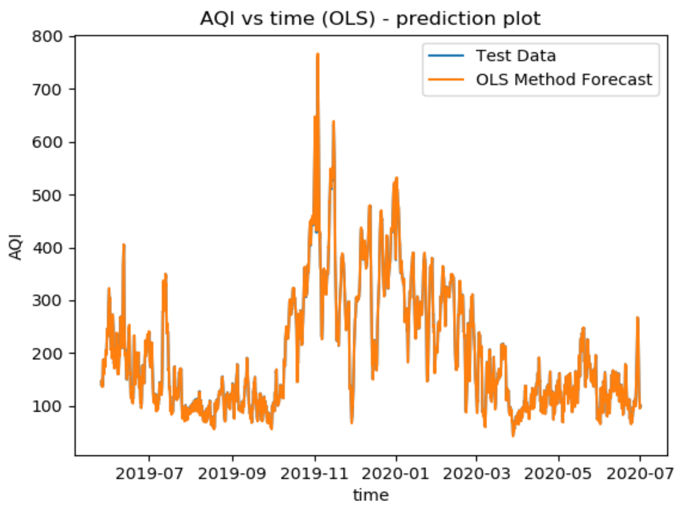
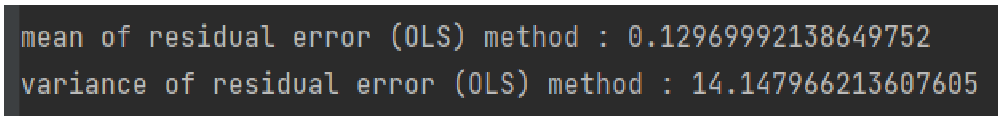


Figure:21 AQI vs time



```
mean of residual error (OLS) method : 0.12969992138649752
variance of residual error (OLS) method : 14.147966213607605
```

Figure:22 OLS method

We can observe that the OLS model is very good at predicting the AQI model. The mean of residual error is low.

Base models:

Average model:

The resultant graph of the average model is shown below:

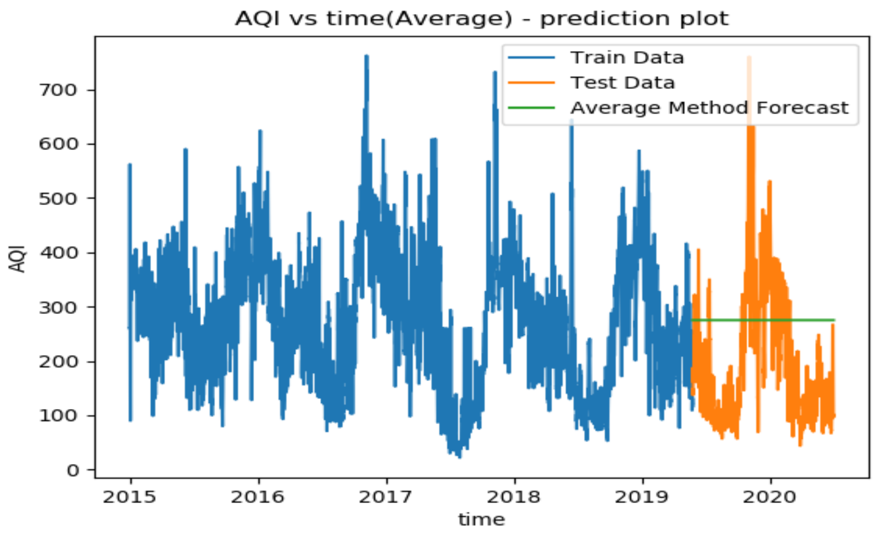


Figure:23 AQI vs time(Average)

From the graph we can see that the forecast of the average method for the length of the test dataset is the average of the train dataset.

Naive model:

The resultant graph of the naive model is shown below:

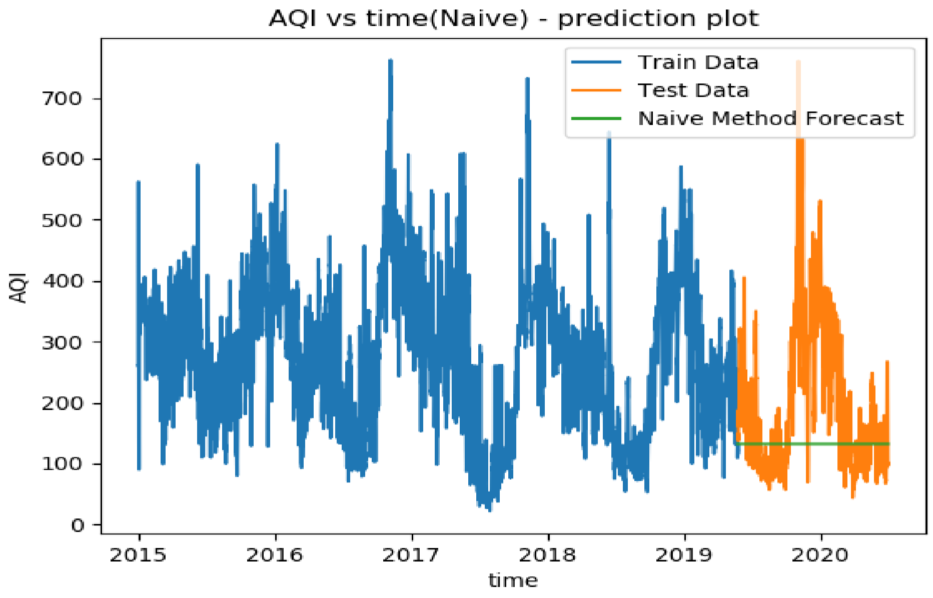


Figure:24 AQI vs time(Naive)

The graph we can see that the forecast of the naive method for the length of the test dataset is the last value of the train dataset.

Drift method:

The resultant graph of the drift model is shown below:

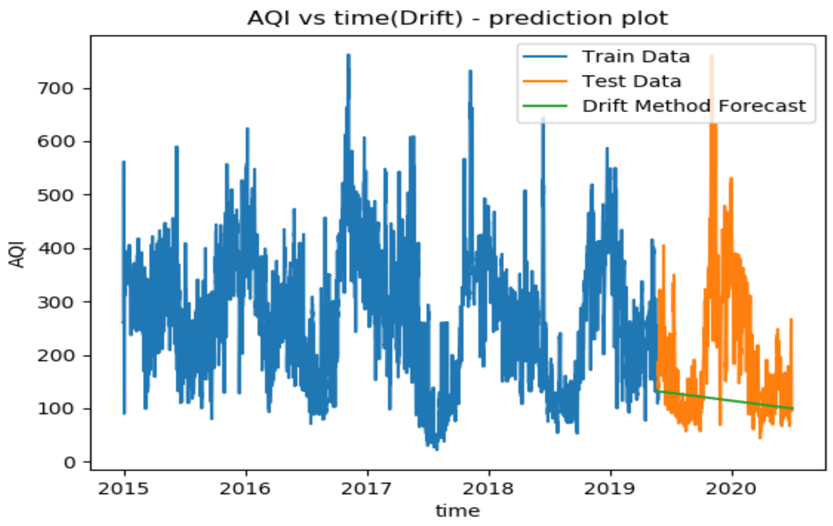


Figure:25 AQI vs time(Drift)

The graph we can see that the forecast of the drift method for the length of the test dataset is the extrapolating the line that connects the 1st and last value of the train dataset.

Simple Exponential Smoothing (SES) method:

The resultant graph of the SES model is shown below:

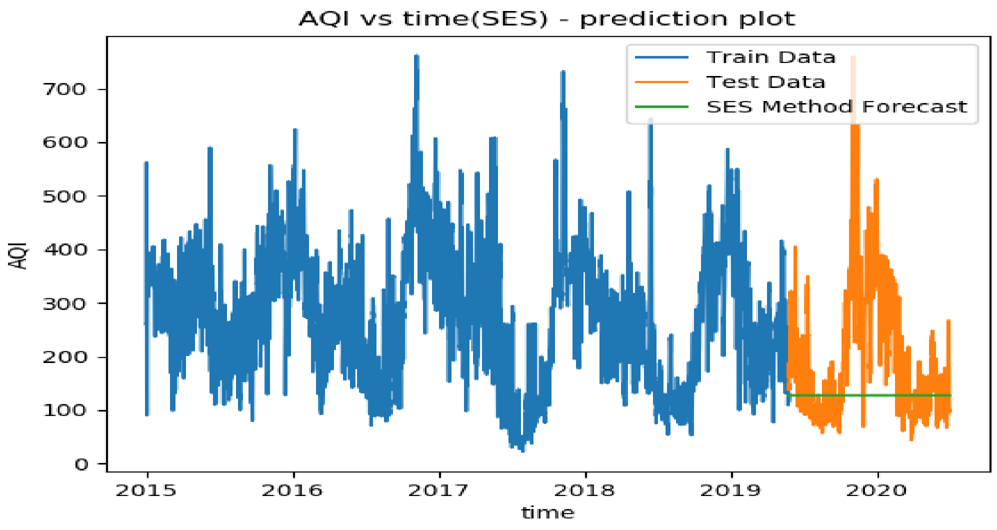
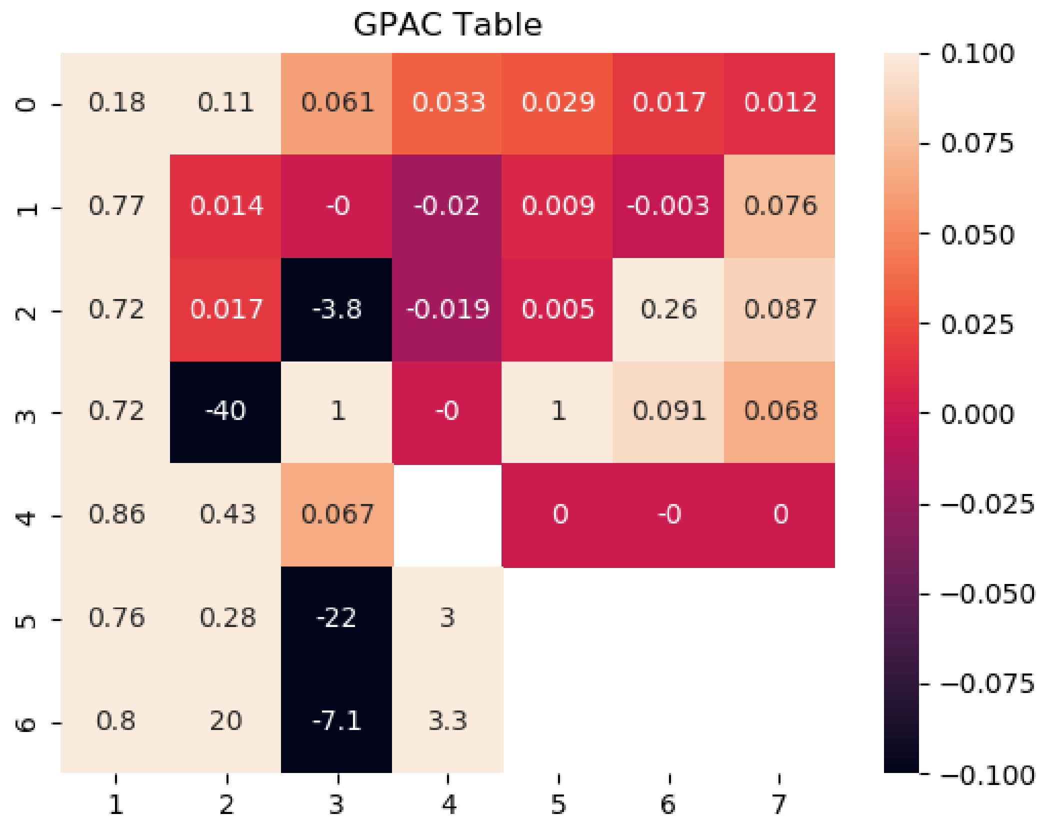


Figure: 26 AQI vs time(ses)

The graph we can see that the forecast of the SES method for the length of the test dataset follows the SES equation. The results are similar to the Holt winter model forecast.

ARMA model:

We perform the GPAC table of the ACF values of the 1st order differenced data. The following is the GPAC table that we obtain from it.



GPAC Table

	1	2	3	4	5	6	7
0	0.18	0.11	0.061	0.033	0.029	0.017	0.012
1	0.77	0.014	-0	-0.02	0.009	-0.003	0.076
2	0.72	0.017	-3.8	-0.019	0.005	0.26	0.087
3	0.72	-40	1	-0	1	0.091	0.068
4	0.86	0.43	0.067		0	-0	0
5	0.76	0.28	-22	3			
6	0.8	20	-7.1	3.3			

0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100

Figure:27 GPAC table

From the above GPAC table we can observe that the 1st column is the column of constants and its 1st row is the row of zeros, since the value of the 1st row is very close to zero.

So we can select the number of AR coefficients, na=1 and the number of MA coefficients, nb=1 for the ARMA and ARIMA models.

We can also select the 4th column as the column of constants and the 4th row is the row of zeros. So we can select the number of AR coefficients, na=4 and the number of MA coefficients nb=4 for the ARMA model.

In this project we performed the 2 ARMA models, each for ARMA(1,1) and ARMA (4,4).

ARMA (1,1) model:

```
ARMA Model Results
==============================================================================
======
Dep. Variable:                  AQI   No. Observations:                38552
Model:                   ARMA(1, 1)   Log Likelihood            -143001.861
Method:                     css-mle   S.D. of innovations               9.879
Date:              Wed, 15 Dec 2021   AIC                        286009.721
Time:                      14:50:05   BIC                   286035.401
Sample:                           0   HQIC                  286017.864

==============================================================================
======
               coef    std err          z      P>|z|      [0.025          0.975]
--------------------------------------------------------------------------
ar.L1.AQI      0.7172        0.014     52.598      0.000       0.690           0.744
ma.L1.AQI     -0.5756        0.016    -36.161      0.000  -0.607          -0.544
                                    Roots
==============================================================================
=====
                  Real          Imaginary           Modulus         Frequency
-------------------------------------------------------------------------
AR.1            1.3944        +0.0000j           1.3944           0.0000
MA.1            1.7373        +0.0000j           1.7373           0.0000
-------------------------------------------------------------------------
The AR Coefficient : a1 is:, 0.7171753492885212
The MA Coefficient : b1 is:, -0.5755961300211727
```

The AR coefficient is 0.71 and MA coefficient is -0.57.

The AIC and BIC values are large so we can say that this is a good model.

Diagnostic test:

The ACF plot of residual error is shown below:

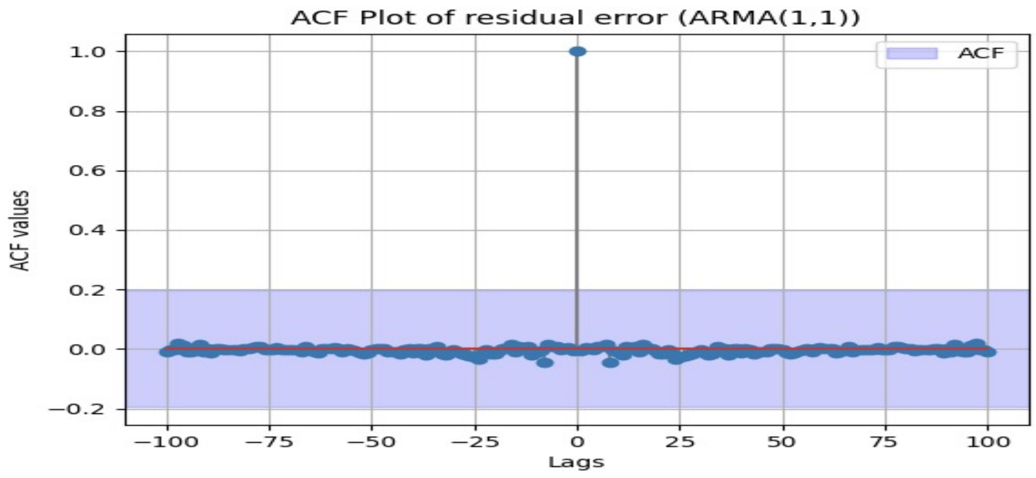


Figure:28 ACF plot of residual error

Chi Squared test results:

The residuals is white, chi squared value :2.835537718678142

From the above ACF plot we can observe that residual errors is white. We can confirm this by seeing the chi squared test.

```
confidence intervals of estimated parameters:                    0         1
ar.L1.AQI  0.690451  0.743900
ma.L1.AQI -0.606794 -0.544398
```

zero/cancellation:

zeros : [0.5755961300211727]

poles : [-0.7171753492885212]

variance of residual error : 97.59133695867716

variance of forecast error : 16.597283056833152

The variance of residual and forecast are relatively small so this is a good model for forecasting AQI.

The 1-step prediction of 1st 100 samples plot is shown below:

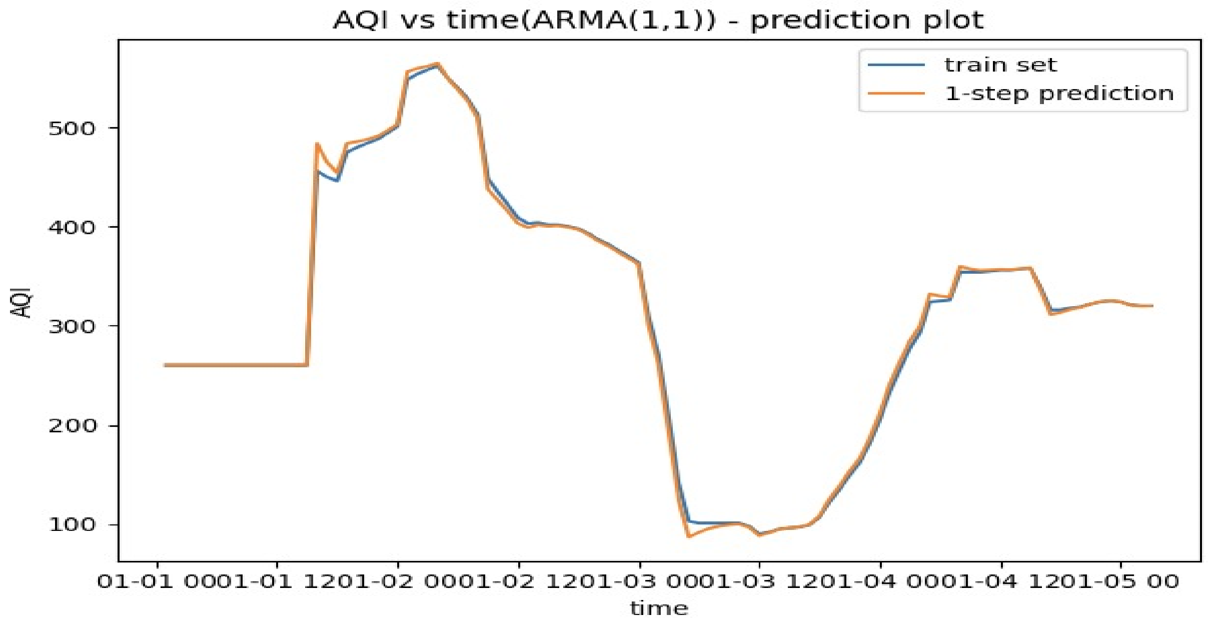


Figure:29 AQI vs time(ARMA(1,1))

The h-step forecast plot is shown below:

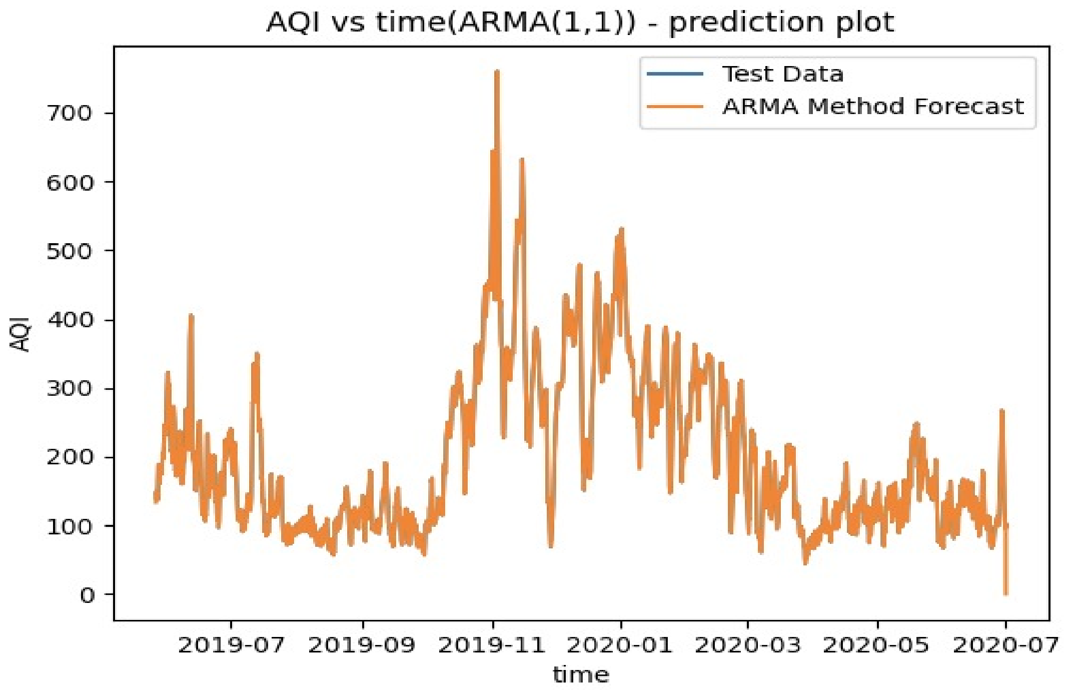


Figure:30 AQI vs time(ARMA(1,1))

From above 1-step prediction and h-step forecast plots we can see that the model is fitting the predictions very well. So we can say that this is a good model to predict AQI values.

ARMA (4,4) model:

ARMA Model Results

```
==============================================================================
======
Dep. Variable:                 AQI   No. Observations:              38552
Model:                ARMA(4, 4)   Log Likelihood          -142838.552
Method:               css-mle   S.D. of innovations              9.837
Date:           Wed, 15 Dec 2021   AIC                    285695.105
Time:                 14:50:39   BIC               285772.142
Sample:                      0   HQIC              285719.531

==============================================================================
======
                coef    std err          z        P>|z|   [0.025          0.975]
------------------------------------------------------------------------------
ar.L1.AQI       0.1435          0.031   4.624  0.000          0.083          0.204
ar.L2.AQI       1.1552          0.030   38.202         0.000          1.096          1.215
ar.L3.AQI       0.4000          0.036   10.966         0.000          0.328          0.471
ar.L4.AQI       -0.7201         0.027   -26.590        0.000  -0.773          -0.667
ma.L1.AQI       -0.0177         0.032   -0.544         0.586  -0.081          0.046
ma.L2.AQI       -1.0692         0.028   -38.184        0.000  -1.124          -1.014
ma.L3.AQI       -0.4981         0.036   -13.954        0.000  -0.568          -0.428
ma.L4.AQI       0.5915          0.026   22.701         0.000          0.540          0.643
                        Roots
==============================================================================
=====
                Real            Imaginary           Modulus        Frequency
------------------------------------------------------------------------------
AR.1            -0.8673         -0.5638j        1.0344         -0.4083
AR.2            -0.8673         +0.5638j        1.0344         0.4083
AR.3            1.0299          -0.0000j        1.0299         -0.0000
AR.4            1.2601          -0.0000j        1.2601         -0.0000
MA.1            -0.8658         -0.5681j        1.0355         -0.4076
MA.2            -0.8658         +0.5681j        1.0355         0.4076
MA.3            1.0051          -0.0000j        1.0051          -0.0000
MA.4            1.5685          -0.0000j        1.5685          -0.0000
-----------------------------------------------------------------------------
The AR Coefficient : a1 is:, 0.14351242375438633
The AR Coefficient : a2 is:, 1.1552483558701916
The AR Coefficient : a3 is:, 0.399960107818056
The AR Coefficient : a4 is:, -0.7200561297181469
The MA Coefficient : b1 is:, -0.017674312776005
The MA Coefficient : b2 is:, -1.0691590548106715
The MA Coefficient : b3 is:, -0.4980845511577915
The MA Coefficient : b4 is:, 0.591488024446287
```

The AIC and BIC values are large so we can say that this is a good model.

Diagnostic test:

The ACF plot of residual error is shown below:

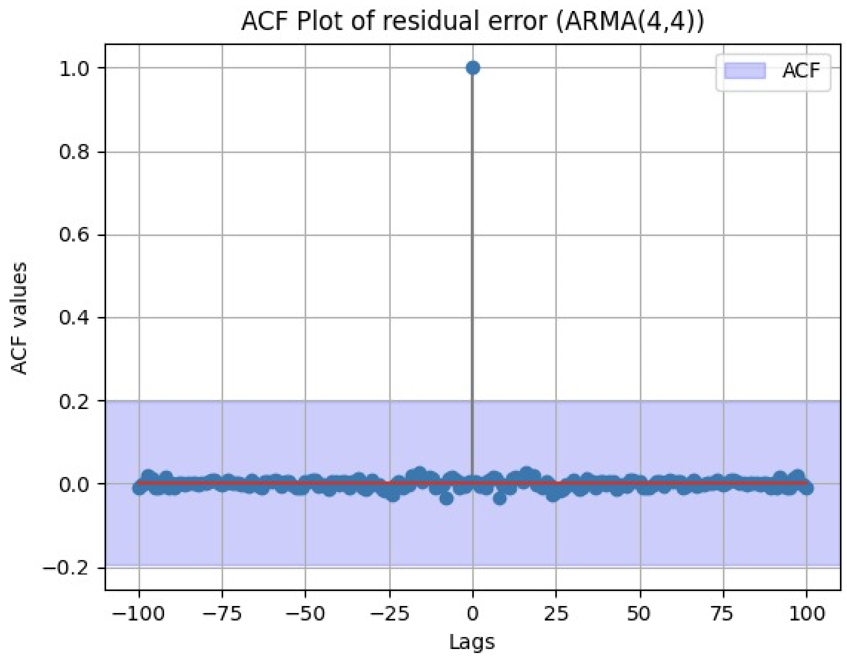


Figure:31 ACF plot of residual error(ARMA(4,4))

Chi Squared test results

The residuals is white, chi squared value :2.288682516508118

From the above ACF plot we can observe that residual errors is white. We can confirm this by seeing the chi squared test.

```
confidence intervals of estimated parameters:                    0          1
ar.L1.AQI  0.082686  0.204339
ar.L2.AQI  1.095978  1.214519
ar.L3.AQI  0.328476  0.471444
ar.L4.AQI -0.773132 -0.666980
ma.L1.AQI -0.081349  0.046000
ma.L2.AQI -1.124038 -1.014280
ma.L3.AQI -0.568043 -0.428126
ma.L4.AQI  0.540420  0.642556
```

From the confidence intervals we can see that only for one coefficient the interval is crossing 0, which is not good for the model.

```
zero/cancellation:
zeros : [0.01767431277600573, 1.0691590548106715, 0.4980845511577915,
-0.591488024446287]
poles : [-0.14351242375438633, -1.1552483558701916, -0.399960107818056,
0.7200561297181469]
variance of residual error : 96.77649614930512
variance of forecast error : 16.592978261645367
```

The 1-step prediction of 1st 100 samples plot is shown below:

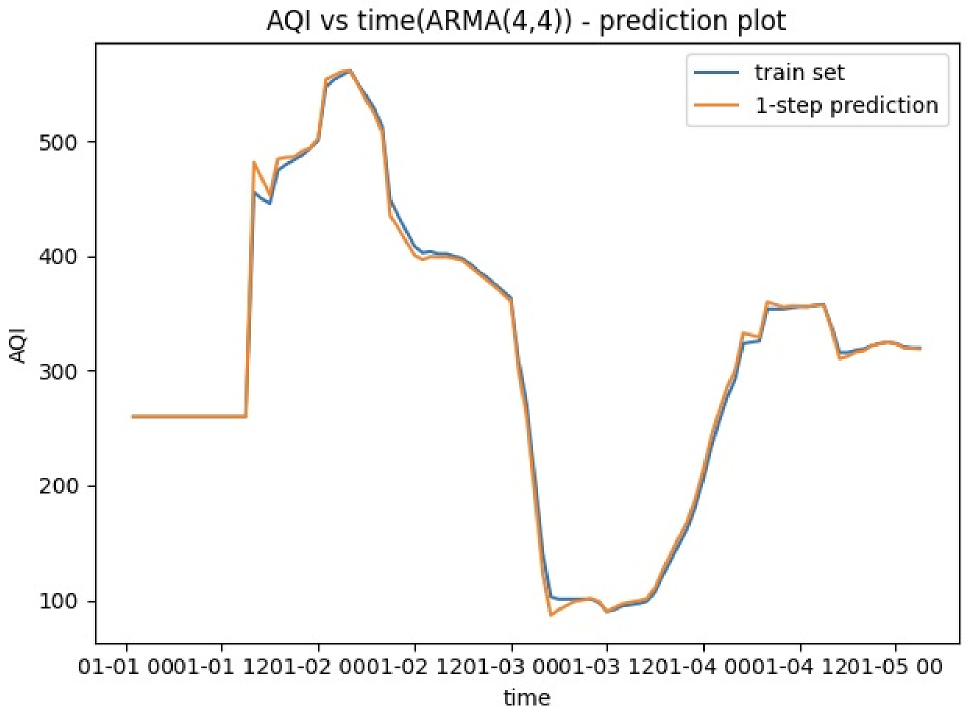


Figure:32 AQI vs time(ARMA(4,4))

The h-step forecast plot is shown below:

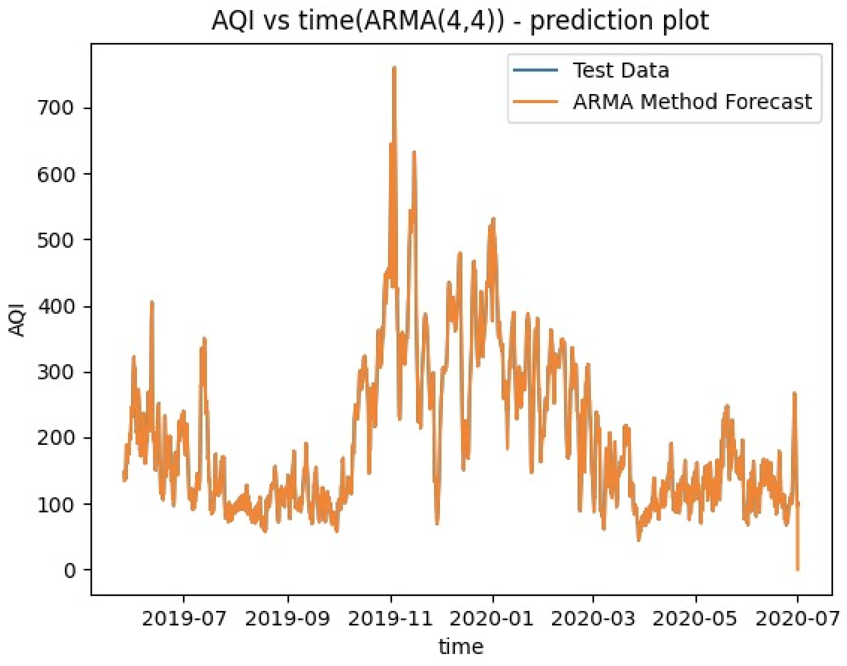


Figure: 33 AQI vs time(ARMA(4,4))

From above 1-step prediction and h-step forecast plots we can see that the model is fitting the predictions very well. So we can say that this is a good model to predict AQI values.

We can observe that the variance of the residual errors is slightly less than the ARMA (1,1). So we can say that the ARMA (4,4) model is better than the ARMA (1,1) model.

ARIMA (1,1,1) model:

```
ARIMA Model Results
==============================================================================
======
Dep. Variable:           D.AQI   No. Observations:          38552
Model:              ARIMA(1, 1, 1)   Log Likelihood          -143001.860
Method:             css-mle   S.D. of innovations            9.879
Date:        Wed, 15 Dec 2021   AIC                     286011.720
Time:               14:51:01   BIC              286045.959
Sample:                    1   HQIC             286022.576

==============================================================================
=======
             coef    std err        z      P>|z|   [0.025        0.975]
--------------------------------------------------------------------------------
const        -0.0032       0.075   -0.042       0.966   -0.151        0.145
ar.L1.D.AQI   0.7172       0.014   52.598       0.000         0.690         0.744
ma.L1.D.AQI  -0.5756       0.016   -36.161      0.000   -0.607        -0.544
                    Roots
==============================================================================
=====
             Real          Imaginary          Modulus      Frequency
-------------------------------------------------------------------------------
AR.1         1.3944        +0.0000j       1.3944        0.0000
MA.1         1.7373        +0.0000j       1.7373        0.0000
-------------------------------------------------------------------------------
The AR Coefficient : a1 is:, -0.003177624943659035
The MA Coefficient : b1 is:, 0.7171760560169066
```

The AIC and BIC values are large so we can say that this is a good model.

Diagnostic test:

The ACF plot of residual error is shown below:

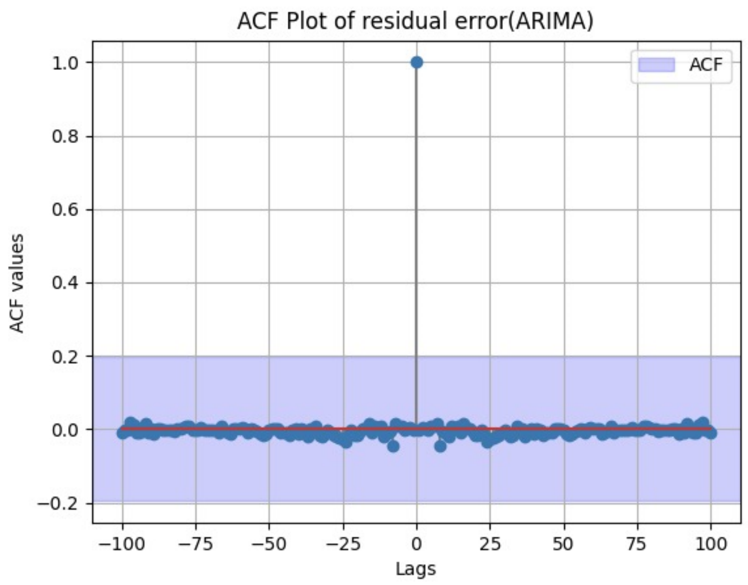


Figure: 34 ACF plot of residual error

Chi Squared test results

The residuals is white, chi squared value :2.8506147859063486

confidence intervals of estimated parameters: 0 1

const -0.151148 0.144793

ar.L1.D.AQI 0.690452 0.743900

ma.L1.D.AQI -0.606795 -0.544399

variance of residual error : 98.04075576373411

variance of forecast error : 13220.248652586179

The 1-step prediction of 1st 100 samples plot is shown below:

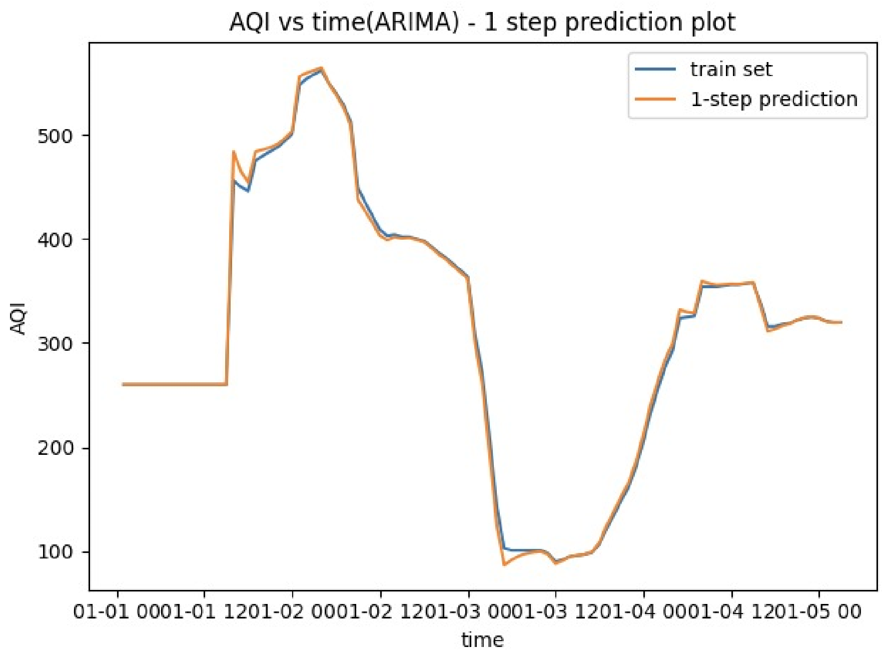


Figure:35 AQI vs time(ARIMA)

The h-step forecast plot is shown below:

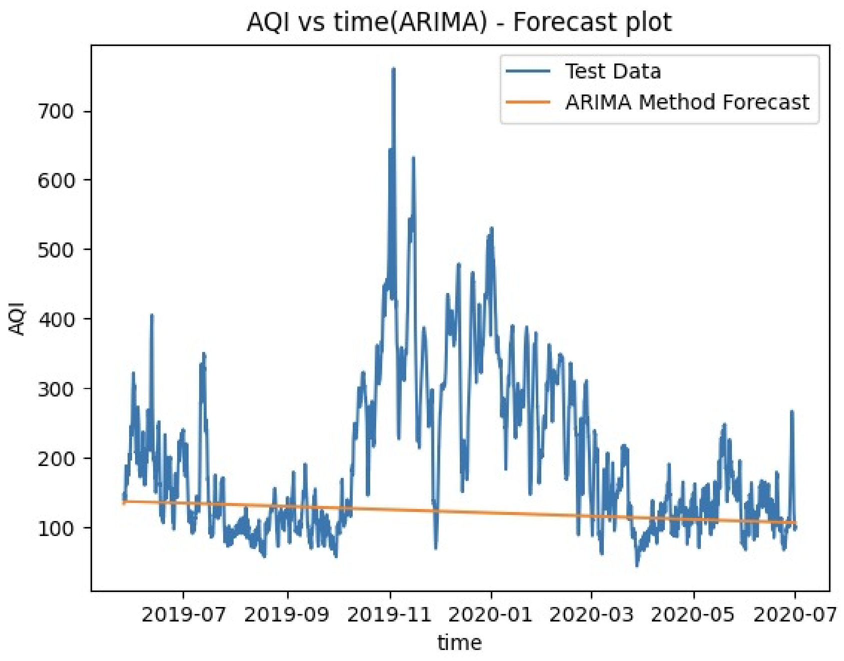


Figure:36 AQI vs time(ARIMA)

We can observe that the ARMA model performed better than the ARIMA model. Even though the 1-step prediction values were good the h-step forecast values are not following the values of test data. We can even observe this from the variance of forecast errors.

So our best model is ARMA (4,4).

Summary and conclusion:

We started off by cleaning the data to remove missing values and performed a stationarity test to check if the raw data is stationary. We found out that the raw data was not stationary so we performed 1st order differencing and checked once again with a stationarity test. We found out that 1st order differenced data is stationary.

We built some base models like average, naive, drift and simple exponential smoothing, Holt winter models to see how they perform for the given data. We have observed that all the base models did not perform exceptionally well.

Then we plotted the GPAC table and found out the auto regressive and moving average orders as na=1 and nb=1, we also found out the 2nd pair of na,nb=4,4.

We built the ARMA (1,1), ARMA(4,4) and ARIMA (1,1,1) models and compared the results of the prediction and forecast and found out that the ARMA(4,4) model is the best model as it has the lowest variance of residual error and variance of forecast error.

We did not perform SARIMA model as we found from the time series decomposition that our dataset did not have much seasonality.

Appendix:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy import signal
import statsmodels.api as sm
import math
import seaborn as sns
from statsmodels.graphics.tsaplots import plot_acf , plot_pacf
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.stattools import acf
import TS_Helper as helper
import statsmodels.tsa.holtwinters as ets
from sklearn.model_selection import train_test_split
from numpy import linalg as LA

np.random.seed(100)

df = pd.read_csv('city_hour.csv')

city1 = df['City'].unique()

for i in city1:
   z=df[df['City']==i]
   print(f'{i} = {len(z)}')

df = df[df['City']=='Delhi']

#==================================== Data Cleaning
==================================

print(df.isnull().sum())

df.drop(['Xylene'],axis=1,inplace=True)

cols = df.columns[2:14]
for i in cols:
   df.loc[:,i]=df[i].fillna(np.mean(df[i]))
df.loc[:,'AQI_Bucket'] = df['AQI_Bucket'].fillna(df['AQI_Bucket'].mode()[0])


print(df.isnull().sum())

df = df.reset_index()

start_date = '2015-01-01 01:00:00'
date = pd.date_range(start=start_date, periods=len(df), freq='H')
df['time_new'] = date

y = df['AQI']
time= df['time_new']

# ==================================== AQI vs time plot
==================================

plt.plot(time,y)
plt.title('Plot of AQI vs time')
plt.xlabel('Time')
# plt.xticks(rotation=90)
plt.ylabel('AQI')
plt.show()

#================================ Auto Correlation plot
==================================

lags=50

# acf3 = helper.auto_correlation_cal(y, lags)

acf3 = acf(y, nlags=lags)
plt.stem(np.arange(-lags, lags+1 ), np.hstack(((acf3[::-1])[:-1],acf3)),
linefmt='grey', markerfmt='o')
m = 1.96 / np.sqrt(100)
plt.axhspan(-m, m, alpha=.2, color='blue')
plt.title("ACF Plot of 'AQI'")
plt.xlabel("Lags")
plt.ylabel("ACF of 'AQI'")
plt.grid()
plt.legend(["ACF"], loc='upper right')
plt.show()

#================================= ACF/PACF plot
=================================

helper.ACF_PACF_Plot(y,50,'AQI')

#================================= Correlation plot
=================================

plt.figure(figsize=(10,10))
sns.heatmap((df.iloc[:,2:15]).corr(),annot=True)
plt.tight_layout()
plt.show()

#================================= Stationarity Check
=================================

# ADF plot
helper.ADF_Cal(y)

# KPSS plot
helper.kpss_test(y)

# rolling average
rol_m = helper.rolling_mean(y)

plt.plot(rol_m, label='rolling mean')
plt.title("rolling mean vs time")
plt.xlabel("time")
plt.ylabel("rolling mean")
plt.legend()
plt.grid()
plt.show()
```

```
# rolling variance
rol_v = helper.rolling_variance(y)

plt.plot(rol_v, label='rolling variance')
plt.title("rolling variance vs time")
plt.xlabel("time")
plt.ylabel("rolling variance")
plt.legend()
plt.grid()
plt.show()

# ================== 1st order differencing ======================
# y1 = helper.diff1(y1)
y1 = y.diff()
lags=50

# acf3 = helper.auto_correlation_cal(y1, lags)

acf3 = acf(y1[1:], nlags=lags)
plt.stem(np.arange(-lags, lags+1 ), np.hstack(((acf3[::-1])[:-1],acf3)),
linefmt='grey', markerfmt='o')
m = 1.96 / np.sqrt(100)
plt.axhspan(-m, m, alpha=.2, color='blue')
plt.title("ACF Plot of 'AQI 1st order'")
plt.xlabel("Lags")
plt.ylabel("ACF of 'AQI' 1st order")
plt.grid()
plt.legend(["ACF"], loc='upper right')
plt.show()



# rolling average of 1st order differencing
rol_m1 = helper.rolling_mean(y1[1:])

plt.plot(rol_m1, label='rolling mean')
plt.title("rolling mean vs time - 1st order differencing")
plt.xlabel("time")
plt.ylabel("rolling mean")
plt.legend()
plt.grid()
plt.show()

# rolling variance of 1st order differencing
rol_v1 = helper.rolling_variance(y1[1:])

plt.plot(rol_v1, label='rolling variance')
plt.title("rolling variance vs time - 1st order differencing")
plt.xlabel("time")
plt.ylabel("rolling variance")
plt.legend()
plt.grid()
plt.show()

# ADF test of 1st order differencing
helper.ADF_Cal(y1[1:])

# KPSS test of 1st order differencing
helper.kpss_test(y1[1:])

# ============================== Time Series Decomposition
======================
from statsmodels.tsa.seasonal import STL
STL=STL(y, period=12)
res = STL.fit()

st = res.seasonal
t = res.trend
rt = res.resid

Ft=1-(np.var(rt)/np.var(t+rt))
print(f'The strength of trend of the raw data is {Ft}')

Fs=1-(np.var(rt)/np.var(st+rt))
print(f'The strength of seasonality of the raw data is {Fs}')
```

```
#============================== Train Test split ============================

yt,yf = train_test_split(y,shuffle=False,test_size=0.2)
yt1,yf1 = train_test_split(y1[1:],shuffle=False,test_size=0.2)
dtrain,dtest = train_test_split(time,shuffle=False,test_size=0.2)


#============================== Holt Winter Method ============================

holtt=ets.ExponentialSmoothing(yt,trend='mul',damped_trend=True,seasonal='mul',
seasonal_periods=12).fit()
holtf=holtt.forecast(steps=len(yf))
holtf=pd.DataFrame(holtf).set_index(yf.index)

plt.plot(dtrain,yt, label='Train Data')
plt.plot(dtest,yf, label='Test Data')
plt.plot(dtest,holtf.values.flatten(), label='Holt Winter Forecast')
plt.title('AQI vs time (Holt winter)- prediction plot')
plt.xlabel('time')
plt.ylabel('AQI')
plt.legend()
plt.show()

# ============================ Feature Selection
================================

x = df[['PM2.5', 'PM10', 'NO', 'NO2', 'NOx', 'NH3', 'CO',
       'SO2', 'O3', 'Benzene', 'Toluene',]]

x = x.iloc[1:,:]
y_f = y1[1:]

X_train, X_test, y_train, y_test = train_test_split(x, y_f, test_size=0.2,
random_state=100, shuffle=False)

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

```python
H = np.matmul(svd_X.T, svd_X)
print(f'Shape of H is {H.shape}')
s, d, v = np.linalg.svd(H)
print(f'Singular Values {d}')
print(f'The condition number is {LA.cond(svd_X)}')
print('unknown coefficients :',helper.LSE(X_train, y_train))

print("====================================OLS
MODEL==================================")

# X = sm.add_constant(X_train)
model_1 = sm.OLS(y_train, X_train)
results_1 = model_1.fit()

print(results_1.summary())

print("====================================OLS
MODEL==================================")
print("============================ SO2 dropped
====================================")

X_train_new=X_train.drop(['NOx'],axis=1)
results_1=sm.OLS(y_train,X_train_new).fit()
print(results_1.summary())

X_test_new = X_test.drop(['NOx'],axis=1)
y_pred_ols = results_1.predict(X_test_new)

y_pred_ols_inv =
helper.inverse_diff(y[len(yt1)+1:].values,np.array(y_pred_ols),1)

# def inverse_diff_forecast(y_last,z_hat,interval=1):
#     y_new = np.zeros(len(z_hat))
#     y_new[0] = y_last
#     for i in range(1,len(z_hat)):
#         y_new[i] = z_hat[i-interval] + y_new[i-interval]
#     # y_new = y_new[1:]
#     return y_new
# y_pred_ols_inv = inverse_diff_forecast(yt.values[-1],np.array(y_pred_ols),1)

plt.plot(dtest, y[len(yt1)+1:].values.flatten(), label='Test Data')
plt.plot(dtest[:-1], y_pred_ols_inv, label='OLS Method Forecast')
plt.title('AQI vs time (OLS) - prediction plot')
plt.xlabel('time')
plt.ylabel('AQI')
plt.legend()
plt.show()
print("\n")



plt.plot(dtest, y_test.values.flatten(), label='Test Data')
plt.plot(dtest, y_pred_ols.values.flatten(), label='OLS Method Forecast')
plt.title('AQI vs time (OLS) differenced values- prediction plot')
plt.xlabel('time')
plt.ylabel('AQI')
plt.legend()
plt.show()
print("\n")



res_ols_err = y_test-y_pred_ols

lags=100

acf_ols_err = helper.auto_correlation_cal(res_ols_err, lags)

plt.stem(np.arange(-lags+1, lags ),
np.hstack(((acf_ols_err[::-1])[:-1],acf_ols_err)), linefmt='grey',
markerfmt='o')
m = 1.96 / np.sqrt(100)
plt.axhspan(-m, m, alpha=.2, color='blue')
plt.title("ACF Plot of residual error (OLS)")
plt.xlabel("Lags")
plt.ylabel("ACF values")
plt.grid()
plt.legend(["ACF"], loc='upper right')
plt.show()



# diagnostic testing

print(f'mean of residual error (OLS) method : {np.mean(res_ols_err)}')

print(f'variance of residual error (OLS) method : {np.var(res_ols_err)}')

# ================================== Base-Models
================================

# ================================= average method
==============================

df1 = pd.DataFrame()
y_hat=[]
for i in range(len(yt)):
    if i==0:
            y_hat.append(np.nan)
    else:
            y_hat.append(round((pd.Series(yt)).head(i).mean(),3))

df1['yt']=yt
df1['average']=y_hat

y_avg_pred = []
for j in yf.index:
    y_avg_pred.append(round(df1['yt'].mean(),3))

y_avg_pred = pd.Series(y_avg_pred,index=yf.index)

# plt.plot(yt, label='Train Data')
# plt.plot(yf, label='Test Data')
# plt.plot(y_avg_pred, label='Average Method Forecast')
plt.plot(dtrain,yt, label='Train Data')
plt.plot(dtest,yf, label='Test Data')
plt.plot(dtest,y_avg_pred, label='Average Method Forecast')
plt.title('AQI vs time(Average) - prediction plot')
plt.xlabel('time')
plt.ylabel('AQI')
plt.legend()
plt.show()



# ================================= naive method
=================================

y_hat1=[]
for i in range(len(df1)):
    if i==0:
        y_hat1.append(np.nan)
    else:
        y_hat1.append(df1.loc[i-1,'yt'])

df1['naive']=y_hat1

y_naive_pred = []
for j in yf.index:
    y_naive_pred.append(round(df1.loc[len(df1)-1,'yt']))

y_naive_pred = pd.Series(y_naive_pred,index=yf.index)

# plt.plot(yt, label='Train Data')
# plt.plot(yf, label='Test Data')
# plt.plot(y_naive_pred, label='Naive Method Forecast')
plt.plot(dtrain,yt, label='Train Data')
plt.plot(dtest,yf, label='Test Data')
plt.plot(dtest,y_naive_pred, label='Naive Method Forecast')
plt.title('AQI vs time(Naive) - prediction plot')
plt.xlabel('time')
plt.ylabel('AQI')
plt.legend()
plt.show()
```

```
#  ================================== drift method
==================================

y_hat2=[]
for i in range(len(df1)):
   if i<=1:
       y_hat2.append(np.nan)
   else:

y_hat2.append(df1.loc[i-1,'yt']+1*((df1.loc[i-1,'yt']-df1.loc[0,'yt'])/((df1.in
dex[i-1]+1)-1) ))

df1['drift']=y_hat2

y_drift_pred = []
h=1
for i in range(len(yf)):

y_drift_pred.append(df1.loc[len(df1)-1,'yt']+h*((df1.loc[len(df1)-1,'yt']-df1.l
oc[0,'yt'])/(len(df1)-1) ))
   h+=1

y_drift_pred = pd.Series(y_drift_pred,index=yf.index)

# plt.plot(yt, label='Train Data')
# plt.plot(yf, label='Test Data')
# plt.plot(y_drift_pred, label='Drift Method Forecast')
plt.plot(dtrain,yt, label='Train Data')
plt.plot(dtest,yf, label='Test Data')
plt.plot(dtest,y_drift_pred, label='Drift Method Forecast')
plt.title('AQI vs time(Drift) - prediction plot')
plt.xlabel('time')
plt.ylabel('AQI')
plt.legend()
plt.show()

#  ================================== SES method
==================================
y_hat3=[]
alpha=0.5
for i in range(len(df1)):
    if i==0:
           y_hat3.append(np.nan)
    elif i==1:
           y_hat3.append(alpha*df1.loc[i-1,'yt']+(1-alpha)*df1.loc[0,'yt']) # since
initial condition is 1st sample
    else:
           y_hat3.append(alpha*df1.loc[i-1,'yt']+(1-alpha)*y_hat3[i-1])

df1['ses']=y_hat3

y_ses_pred = []

for i in range(len(yf)):
    y_ses_pred.append(alpha*df1.loc[len(df1)-1,'yt']+(1-alpha)*y_hat3[-1])

y_ses_pred = pd.Series(y_ses_pred,index=yf.index)

# plt.plot(yt, label='Train Data')
# plt.plot(yf, label='Test Data')
# plt.plot(y_ses_pred, label='SES Method Forecast')
plt.plot(dtrain,yt, label='Train Data')
plt.plot(dtest,yf, label='Test Data')
plt.plot(dtest,y_ses_pred, label='SES Method Forecast')
plt.title('AQI vs time(SES) - prediction plot')
plt.xlabel('time')
plt.ylabel('AQI')
plt.legend()
plt.show()

#  ================================== ARMA Process
==================================

print('========================= ARMA (1,1) model
==================================')
```

```
acf_y2 = acf(y1[1:],100)
helper.Cal_GPAC(acf_y2,7,7)

na = 1
nb = 1

model = sm.tsa.ARMA(yt1,(na,nb)).fit(trend='nc',disp=0)
print(model.summary())

for i in range(na):
    print(f'The AR Coefficient : a{i+1} is:, {model.params[i]}')

for i in range(nb):
    print(f'The MA Coefficient : b{i+1} is:, {model.params[i+na]}')

# =========== 1 step prediction =================
def inverse_diff(y20,z_hat,interval=1):
    y_new = np.zeros(len(y20))
    for i in range(1,len(z_hat)):
        y_new[i] = z_hat[i-interval] + y20[i-interval]
    y_new = y_new[1:]
    return y_new

model_hat = model.predict(start=0,end=len(yt1)-1)

# res_arma_error = np.array(yt) - np.array(model_hat)

y_hat = helper.inverse_diff(y[:len(yt1)].values,np.array(model_hat),1)

res_arma_error = y[1:len(yt1)] - y_hat

lags=100

helper.ACF_PACF_Plot(res_arma_error,lags,'residual error (ARMA(1,1))')
# acf_res = helper.auto_correlation_cal(res_arma_error, lags)
acf_res = acf(res_arma_error, nlags= lags)
```

```
plt.stem(np.arange(-lags, lags+1 ), np.hstack(((acf_res[::-1])[:-1],acf_res)),
linefmt='grey', markerfmt='o')
m = 1.96 / np.sqrt(100)
plt.axhspan(-m, m, alpha=.2, color='blue')
plt.title("ACF Plot of residual error (ARMA(1,1))")
plt.xlabel("Lags")
plt.ylabel("ACF values")
plt.grid()
plt.legend(["ACF"], loc='upper right')
plt.show()



plt.plot(time[:99],y[:99], label = 'train set')
plt.plot(time[:99],y_hat[:99], label = '1-step prediction')
plt.title('AQI vs time(ARMA(1,1)) - prediction plot')
plt.xlabel('time')
plt.ylabel('AQI')
plt.legend()
plt.tight_layout()
plt.show()

# diagnostic testing
from scipy.stats import chi2

print('confidence intervals of estimated parameters:',model.conf_int())

poles = []
for i in range(na):
    poles.append(-(model.params[i]))

print('zero/cancellation:')
zeros = []
for i in range(nb):
    zeros.append(-(model.params[i+na]))

print(f'zeros : {zeros}')
print(f'poles : {poles}')
```

```
Q = len(yt)*np.sum(np.square(acf_res[lags:]))

DOF = lags-na-nb

alfa = 0.01

chi_critical = chi2.ppf(1-alfa, DOF)

print('Chi Squared test results')

if Q<chi_critical:
    print(f'The residuals is white, chi squared value :{Q}')
else:
    print(f'The residual is NOT white, chi squared value :{Q}')

# =========== h step prediction ================
forecast = model.forecast(steps=len(yf1))

y_arma_pred = pd.Series(forecast[0], index=yf1.index)

y_hat_fore = inverse_diff(y[len(yt1):].values,np.array(y_arma_pred),1)

# h step prediction

plt.plot(dtest,y[len(yt1)+1:].values.flatten(), label='Test Data')
plt.plot(dtest,y_hat_fore, label='ARMA Method Forecast')
plt.title('AQI vs time(ARMA(1,1)) - prediction plot')
plt.xlabel('time')
plt.ylabel('AQI')
plt.legend()
plt.show()

res_arma_forecast = y[len(yt1)+1:] - y_hat_fore

print(f'variance of residual error : {np.var(res_arma_error)}')

print(f'variance of forecast error : {np.var(res_arma_forecast)}')
```

```
# ========================== 2nd ARMA model na=4, nb=4
===========================
print('========================= ARMA (4,4) model
==================================')

na = 4
nb = 4

model1 = sm.tsa.ARMA(yt1,(na,nb)).fit(trend='nc',disp=0)
print(model1.summary())

for i in range(na):
   print(f'The AR Coefficient : a{i+1} is:, {model1.params[i]}')

for i in range(nb):
   print(f'The MA Coefficient : b{i+1} is:, {model1.params[i+na]}')



#================= 1 step prediction =====================
model_hat1 = model1.predict(start=0,end=len(yt1)-1)

# res_arma_error = np.array(yt) - np.array(model_hat)

y_hat1 = helper.inverse_diff(y[:len(yt1)].values,np.array(model_hat1),1)

res_arma_error1 = y[1:len(yt1)] - y_hat1

lags=100



helper.ACF_PACF_Plot(res_arma_error1,lags,'ARMA(4,4) residual error')
# acf_res = helper.auto_correlation_cal(res_arma_error, lags)
acf_res = acf(res_arma_error1, nlags= lags)
plt.stem(np.arange(-lags, lags+1 ), np.hstack(((acf_res[::-1])[:-1],acf_res)),
linefmt='grey', markerfmt='o')
m = 1.96 / np.sqrt(100)
plt.axhspan(-m, m, alpha=.2, color='blue')
plt.title("ACF Plot of residual error (ARMA(4,4))")
plt.xlabel("Lags")
plt.ylabel("ACF values")
plt.grid()
plt.legend(["ACF"], loc='upper right')
plt.show()


plt.plot(time[:99],y[:99], label = 'train set')
plt.plot(time[:99],y_hat1[:99], label = '1-step prediction')
plt.title('AQI vs time(ARMA(4,4)) - prediction plot')
plt.xlabel('time')
plt.ylabel('AQI')
plt.legend()
plt.tight_layout()
plt.show()

# diagnostic testing
from scipy.stats import chi2

print('confidence intervals of estimated parameters:',model1.conf_int())

poles = []
for i in range(na):
    poles.append(-(model1.params[i]))

print('zero/cancellation:')
zeros = []
for i in range(nb):
    zeros.append(-(model1.params[i+na]))

print(f'zeros : {zeros}')
print(f'poles : {poles}')

Q = len(yt1)*np.sum(np.square(acf_res[lags:]))

DOF = lags-na-nb

alfa = 0.01
```

```
chi_critical = chi2.ppf(1-alfa, DOF)

print('Chi Squared test results')

if Q<chi_critical:
    print(f'The residuals is white, chi squared value :{Q}')
else:
    print(f'The residual is NOT white, chi squared value :{Q}')

# =========== h step prediction ================

forecast1 = model1.forecast(steps=len(yf1))

y_arma_pred1 = pd.Series(forecast1[0], index=yf1.index)

y_hat_fore1 = inverse_diff(y[len(yt1):].values,np.array(y_arma_pred1),1)

# h step prediction

plt.plot(dtest,y[len(yt1)+1:].values.flatten(), label='Test Data')
plt.plot(dtest,y_hat_fore1, label='ARMA Method Forecast')
plt.title('AQI vs time(ARMA(4,4)) - prediction plot')
plt.xlabel('time')
plt.ylabel('AQI')
plt.legend()
plt.show()

res_arma_forecast1 = y[len(yt1)+1:] - y_hat_fore1

print(f'variance of residual error : {np.var(res_arma_error1)}')

print(f'variance of forecast error : {np.var(res_arma_forecast1)}')
#  ================================= ARIMA Process
=================================

print('========================= ARIMA (1,1,1) model
=================================')
```

```
na = 1
d=1
nb = 1

# yt,yf=train_test_split(y, shuffle=False, test_size=0.2)

model2 = sm.tsa.ARIMA(endog=yt,order=(na,d,nb)).fit()
print(model2.summary())
for i in range(na):
    print(f'The AR Coefficient : a{i+1} is:, {model2.params[i]}')

for i in range(nb):
    print(f'The MA Coefficient : b{i+1} is:, {model2.params[i+na]}')

print('confidence intervals of estimated parameters:',model2.conf_int())



# ================== 1 step prediction ========================
model_hat2 = model2.predict(start=1,end=len(yt)-1)

y_hat3 = inverse_diff(y[:len(yt)].values,np.array(model_hat2),1)

res_arima_error = y[1:len(yt)] - y_hat3

lags=100

acf_res = acf(res_arima_error, nlags=lags)

plt.stem(np.arange(-lags, lags+1), np.hstack(((acf_res[::-1])[:-1],acf_res)),
linefmt='grey', markerfmt='o')
m = 1.96 / np.sqrt(100)
plt.axhspan(-m, m, alpha=.2, color='blue')
plt.title("ACF Plot of residual error(ARIMA)")
plt.xlabel("Lags")
plt.ylabel("ACF values")
plt.grid()
plt.legend(["ACF"], loc='upper right')
plt.show()

plt.plot(time[:99],y[:99], label = 'train set')
plt.plot(time[:99],y_hat3[:99], label = '1-step prediction')
plt.title('AQI vs time(ARIMA) - 1 step prediction plot')
plt.xlabel('time')
plt.ylabel('AQI')
plt.legend()
plt.tight_layout()
plt.show()

# diagnostic testing
from scipy.stats import chi2

print('confidence intervals of estimated parameters:',model2.conf_int())

Q = len(yt)*np.sum(np.square(acf_res[lags:]))

DOF = lags-na-nb

alfa = 0.01

chi_critical = chi2.ppf(1-alfa, DOF)

print('Chi Squared test results')

if Q<chi_critical:
   print(f'The residuals is white, chi squared value :{Q}')
else:
   print(f'The residual is NOT white, chi squared value :{Q}')

# ================== h step prediction ========================
forecast2 = model2.forecast(steps=len(yf))

y_arima_pred = pd.Series(forecast2[0], index=yf.index)

# plt.plot(dtrain,yt, label='Train Data')
plt.plot(dtest,yf.values.flatten(), label='Test Data')
plt.plot(dtest,y_arima_pred.values.flatten(), label='ARIMA Method Forecast')
plt.title('AQI vs time(ARIMA) - Forecast plot')
plt.xlabel('time')
plt.ylabel('AQI')
plt.legend()
plt.show()

res_arima_forecast = np.array(yf) - forecast[0]

print(f'variance of residual error : {np.var(res_arima_error)}')

print(f'variance of forecast error : {np.var(res_arima_forecast)}')
```

References:

https://www.kaggle.com/rohanrao/air-quality-data-in-india

https://plotly.com/

https://dash.plotly.com/dash-core-components

https://dash.plotly.com/dash-html-components

Class lecture videos and lecture material

Labs and assignment code