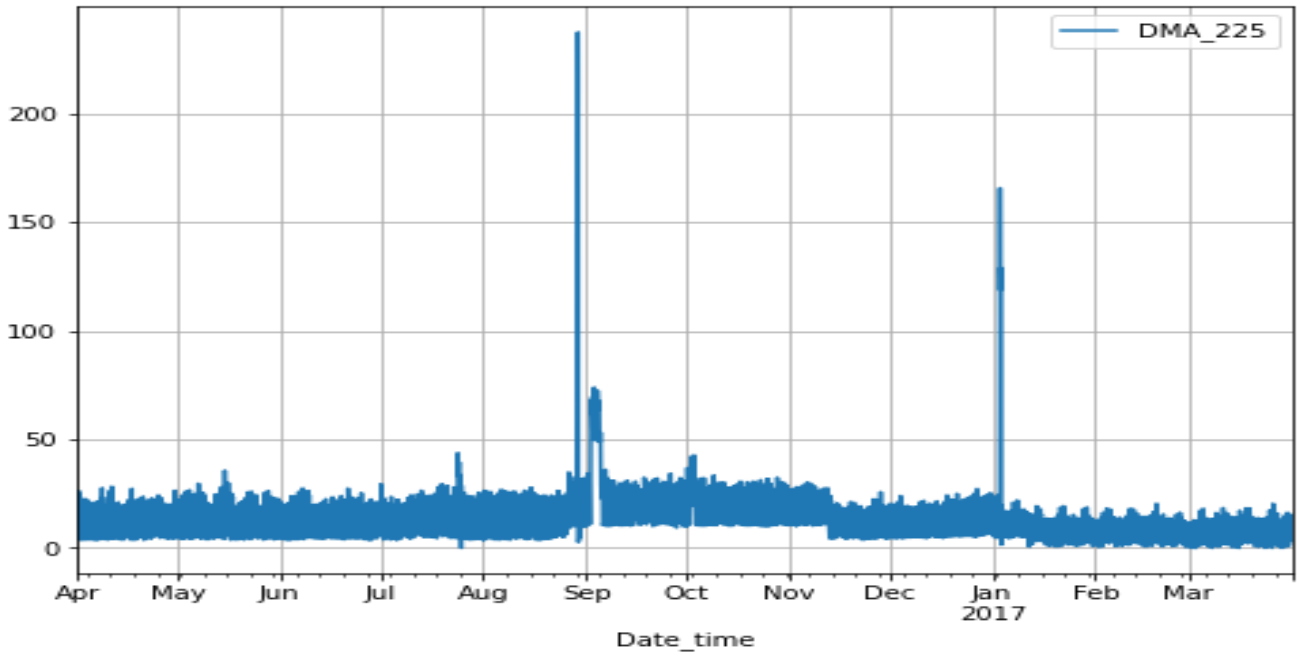
Univariate time series analysis – DMA 225

Dataset Observation

Dataset

The dataset is of 1 year from April 2016 – March 2017 with 15 minutes duration. Its shape is around 35040 rows and other DMA's columns. So hence considered only two columns DMA_225 and the data-time (with hour and data details). After that, created index of date-time to perform analysis on DMA_225. The data is transformed into hourly basis, with the shape of (8760,1). The aim of the project is to calculate 24 hours prediction, based on the past values.

Performed models: one step ahead, sarimax



Stationarity Check

The dataset is checked for its stationary check with "Augmented Dickey-fuller test" and Kwiatkowski–Phillips–Schmidt–Shin tests to test null hypothesis.

ADF:

```
ADF Statistic: -7.013940
p-value: 0.000000
Critical Values:
        1%: -3.431
        5%: -2.862
        10%: -2.567
```

Observation :

1.The ADF statistic value of -7. The more negative this statistic, the more likely we are to reject the null hypothesis (we have a stationary dataset).

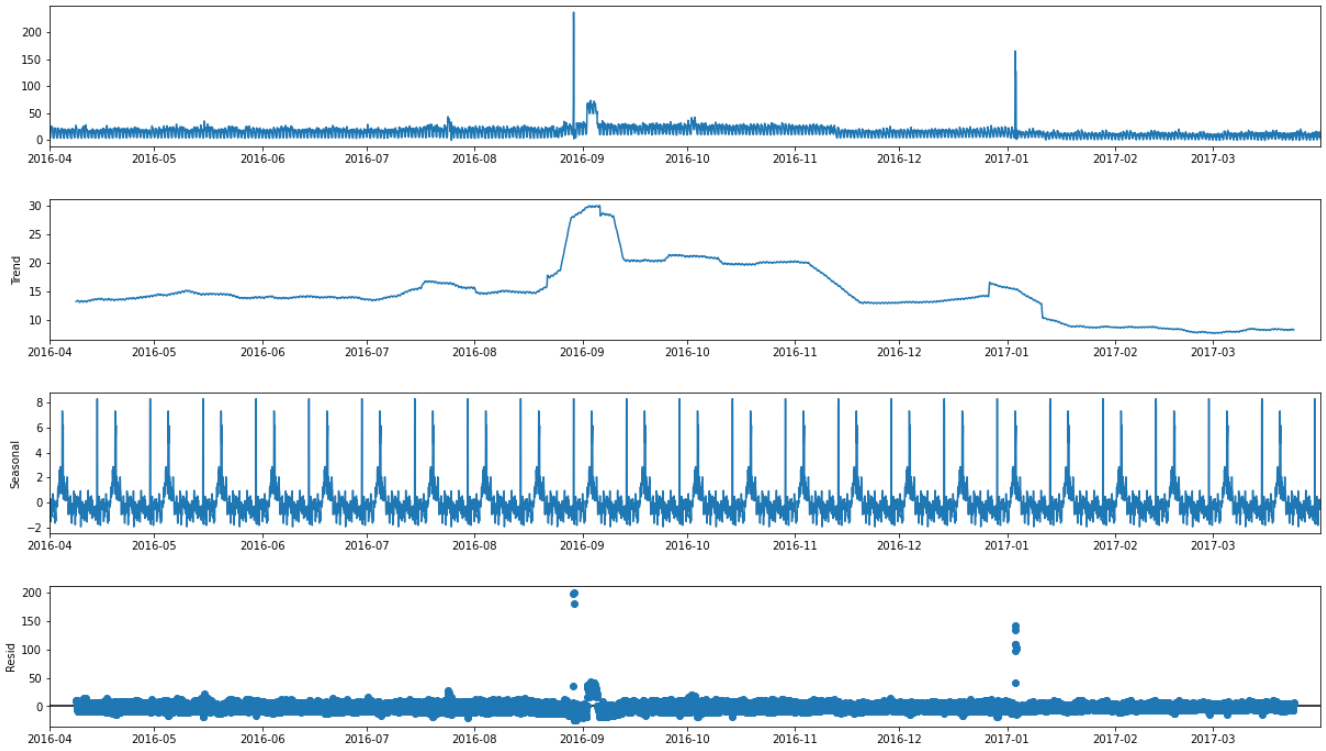
2.p value - 0.000058 < 0.05 ; Data is stationary

KPSS:

```
(4.31149360491682,
 0.01,
 37,
 {'1%': 0.739, '10%': 0.347, '2.5%': 0.574, '5%': 0.463})
```

Observation : 1.The ADF statistic value of 4.3. The positive value is statistic, and hence we have a stationary dataset.

Time series decomposition



Baseline Model

One step ahead prediction

One step ahead baseline model by shifting one row as below.

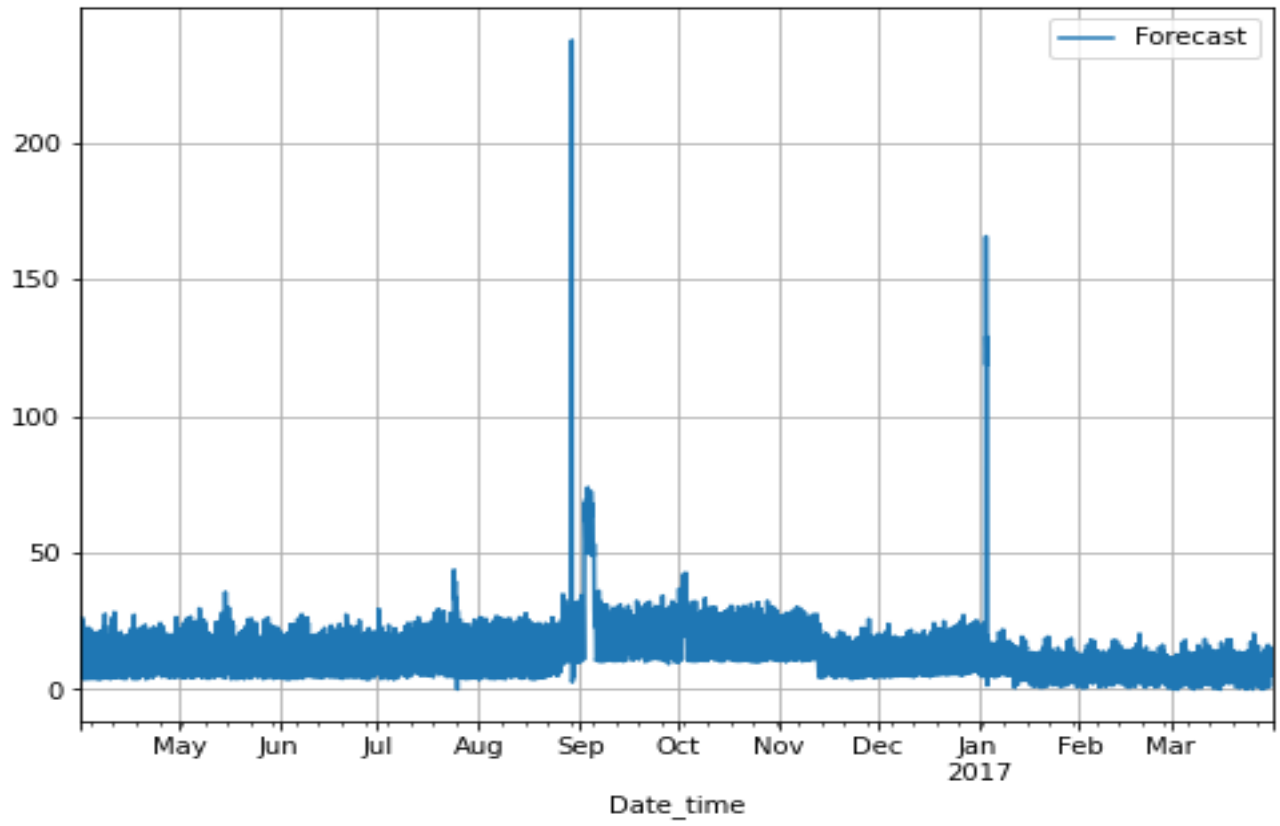
```
# Making a df
Original = hourly_data["DMA_225"].to_frame().rename(columns = {"DMA_225": "Original" })
Forecast  = hourly_data["DMA_225"].to_frame().shift(1).rename(columns = {"DMA_225": "Forecast" })
baseline = pd.concat([Original,Forecast],axis=1)
```

With baseline model, the RMSE value between actual and predicted value is 4.425.

```
# Calculate the RMSE
rmse = np.sqrt(mean_squared_error(final.Original, final.Forecast))
rmse = round(rmse, 3)
print (" The root mean square value on dataset: ",rmse)
```

```
The root mean square value on dataset:  4.425
```

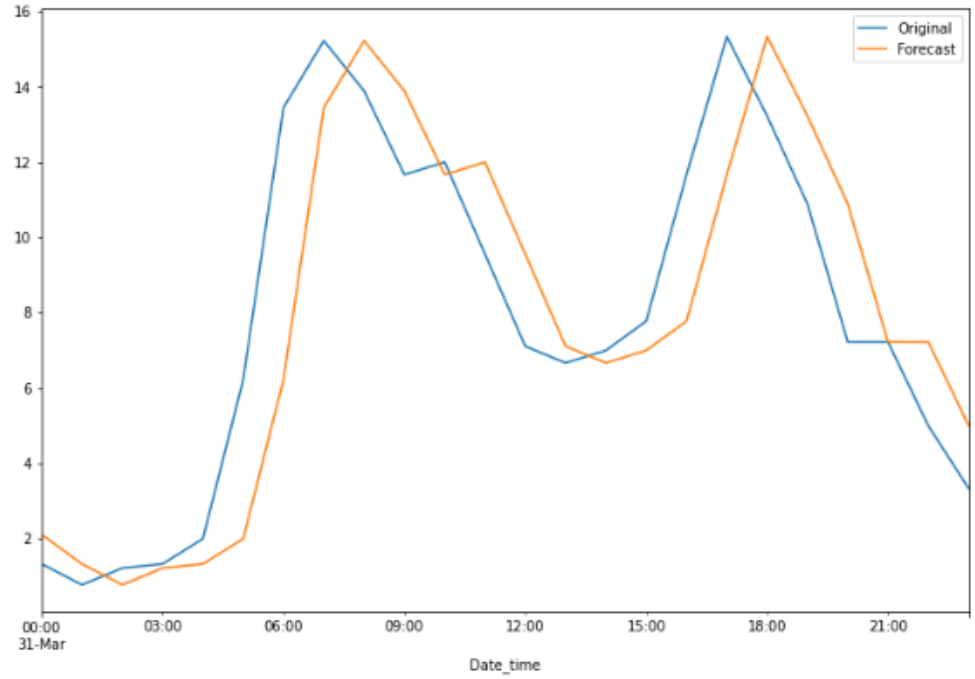
Below is the forecast graph achieved by one step ahead prediction:



Predicted 24 hours using one step baseline model as below with achieved RMSE 2.533

```
rmse1 = np.sqrt(mean_squared_error(baseline_graph.Original, baseline_graph.Forecast))
rmse1 = round(rmse1, 3)
print (" The root mean square value on dataset: ",rmse1)
baseline_graph.plot(figsize=(12,8))
```

```
 The root mean square value on dataset:  2.533
<matplotlib.axes._subplots.AxesSubplot at 0x7fe97c8e7150>
```



SARIMAX

1. The data is seasonal, and hence performed sarimax and obtained the best order with AIC value.
2. Grid search hyper parameter tuning is performed on the entire dataset to get best p, q, d order along with AIC values.

```
# Getting best arima model
p = d = q = range(0, 2)
pdq_cal = list(itertools.product(p, d, q))
# generation of different combinations of seasonal p, q and q triplets
pdq_seasonal = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]
aic_bestvalue = np.inf
pdq_bestvalue = None
pdq_seasonal_best = None
tempmodel = None
for params in pdq_cal:
    for seasonal_params in pdq_seasonal:
        tempmodel = SARIMAX(hourly_data['DMA_225'],order=params,seasonal_order = seasonal_params,enforce_invertibility=False,enforce_stationarity=False)
        results = tempmodel.fit(disp=False)
        if results.aic < aic_bestvalue:
            aic_bestvalue = results.aic
            pdq_bestvalue = params
            pdq_seasonal_best = seasonal_params
print("Best ARIMA with seasonality (SARIMAX) {} x {} model - AIC:{}".format(pdq_bestvalue,pdq_seasonal_best,aic_bestvalue))
```

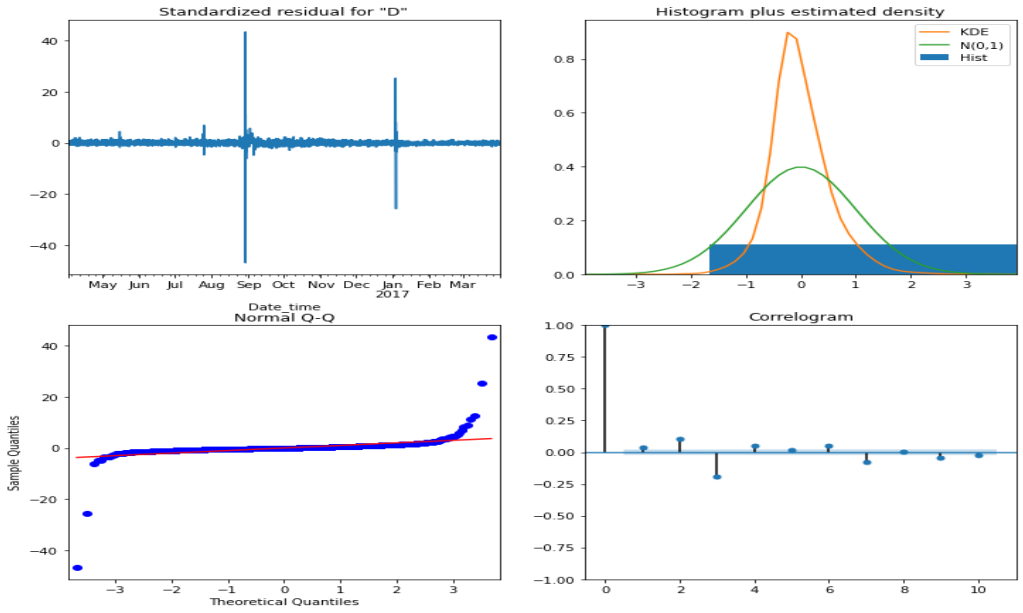
3. Best Seasonal arima model with AIC value is obtained as below:

```
Best ARIMA with seasonality (SARIMAX) (1, 0, 1) x (1, 1, 1, 12) model - AIC:48654.26552054069
```

4. Predicting 24 hours value:

```
# Training the model on the full dataset and predict last 24 hours output
model =  SARIMAX(hourly_data['DMA_225'],
                         order = (1, 0, 1),
                         seasonal_order =(1, 1, 1, 12),enforce_invertibility=False,
                              enforce_stationarity=False)
sarimax = model.fit()
```

5. Below is the plot diagnostics of the model

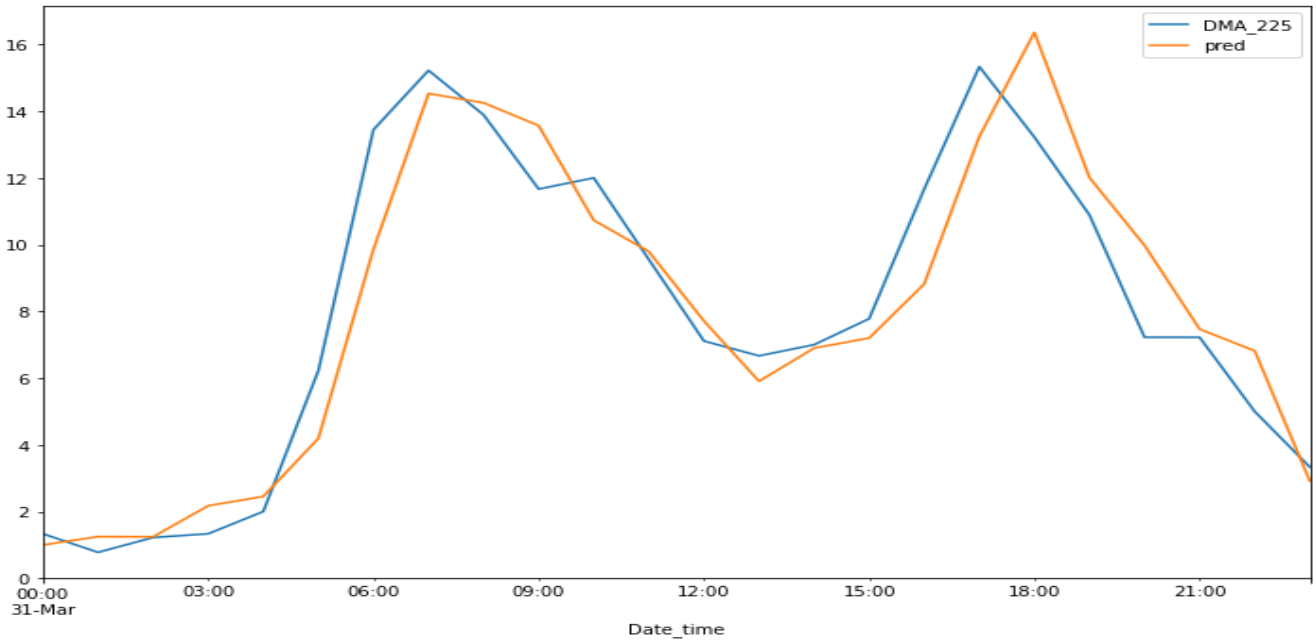


6. RMSE value achieved by sarimax is 1.582

```
rmse2 = np.sqrt(mean_squared_error(cv.DMA_225, cv.pred))
rmse2 = round(rmse2, 3)
print (" The root mean square value on dataset using sarimax: ",rmse2)
```

```
 The root mean square value on dataset using sarimax:  1.582
```

7. Below is the Original vs forecast graph using sarimax for 24-hour prediction



Conclusion

For 24-hour prediction below are the RMSE values:

1. **Baseline model - 2.533**
2. **Sarimax – 1.582**