**Machine Learning Engineer Nanodegree**

**Capstone Project**

Ashish kumar

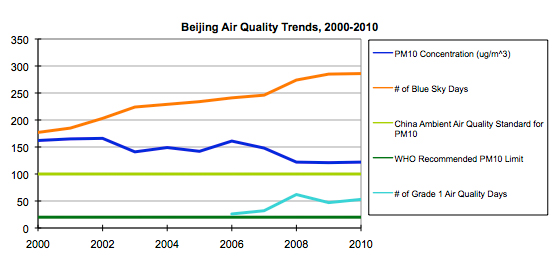
26/02/2020

**Time Series Analysis of Air Quality Data**

**1 Definition**

**1.1 Project Overview**

In recent years Air Quality has become a major health concern. The increase in amount of particulate matters like Pm-2.5, Pm-10, SO2, NO2 causes severe damage to lungs causing respiratory diseases and cancers. Peoples who suffer from seasonal allergies or have a weak immune system are more prone to the degradation in air quality. A time series analysis for the air quality can determine the increased risks associated with time so that necessary steps can be taken in advance like the availability of air masks, air purifiers etc.



The above picture**[2]** shows that the air quality trends of Beijing’s air quality from 2000 to 2010 does not meet China’s own air quality standard, and is six times worse than the recommended particulate matter target set by the WHO.

The aimed to analyze and forecast the Air Pollutants using Statistical models of Time Series Analysis. My personal motivation for working on this Project is my interest to work on Time Series Analysis. The Time-Series case studies at Udacity gave me enough confidence to pick the Time Series Analysis as my Capstone Project.

**1.2 Problem Statement**

The problem is to analyse the concentrations of various pollutants in air like PM2.5, PM10, SO2, NO2, O3 and CO using time-series analysis. As the data is hourly formatted, So we have a good amount of data to work with. The Project is aimed at fitting different statistical models of Time-Series Analysis like SARIMA, Prophet and Holt-Winters.

The goal of the project is to compare the different models performances and see how they perform with the selected dataset. Their performances is compared with LSTM (Long Short Term Memory) Recurrent Neural Network.

**1.3 Evaluation Matrix:**

The Time Series models are evaluated based on the following measures:

**1. Mean Absolute Error(MAE)**

The mean absolute error, or MAE, is calculated as the average of the forecast error values, where all of the forecast values are forced to be positive.

**2. Mean Square Error(MSE)**

The mean squared error, or MSE, is calculated as the average of the squared forecast error values. Squaring the forecast error values forces them to be positive; it also has the effect of putting more weight on large errors.

**3. Root Mean Square Error(RMSE)**

Root Mean Square Error (RMSE) is the standard deviation of the prediction errors. RMSE is a measure of how spread out these residuals are. In other words, it tells us how concentrated the data is around the line of best fit.

Root mean square error is commonly used in climatology, forecasting, and regression analysis to verify experimental results.

Among these three, RMSE and MAE are used for comparing models evaluation plots.

**2 Analysis**

**2.1 Data Exploration**

The Dataset was taken from Kaggle site. It contains the measurement of six major air pollutants PM2.5, PM10, SO2, NO2, CO and O3 at multiple sites in Beijing. The data set includes hourly air pollutants data from 12 nationally-controlled air-quality monitoring sites

The data set includes hourly air pollutants data from 12 nationally-controlled air-quality monitoring sites.

**2.1.1 Selecting the Best Dataset**

The best dataset was selected by counting the null values. Dataset with least null value was taken.

**2.1.2 Features of the Dataset**

The dataset contains a total of 18 columns and 35064 rows. Except No, year, month, day, hour and station, all columns contain null values. The information about the dataset is given in the following figure.

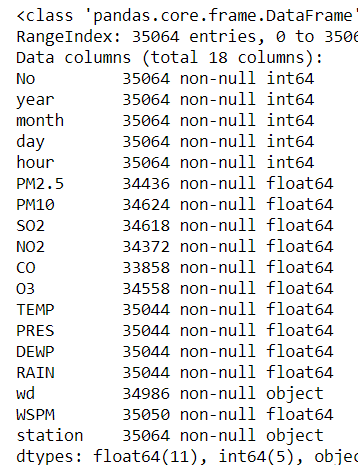


Fig 1: Shows the features of the data.

Six Time-Series will be made based of six major pollutants PM2.5, PM10, SO2, NO2, CO, O3.

Columns like year, month, day, hour will be clubbed to make Date-Time Index.

**2.2 Exploratory Visualization:**

The below graph shows the hourly, daily and monthly time series indication the concentration level of PM2.5 pollutant.

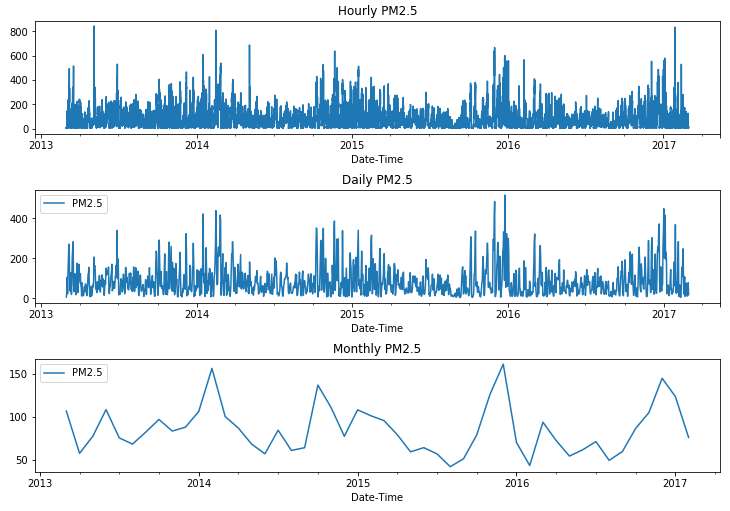


Fig 2: Shows the hourly,daily and monthly data

**2.2.1 Test for Stationarity**

By visualizing the plots it’s hard to determine that the series is stationary or not. We gain slight idea that the trend is not available, but there can be seasonality present.

The two-stationarity test’s **ADF (Augmented Dickey Fuller) Test** and **KPSS (Kwiatkowski-Phillips-Schmidt-Shin)** Test were applied to test the stationarity of the series.

For both the tests, Test statistic should be lower than the critical value.

KPSS test showed not-stationary behavior in SO2. Contrary, ADF test showed static behavior for SO2. Thus, there may be slight seasonality or trend available. But we can ignore this, since SO2 is the only exception among all the pollutants.

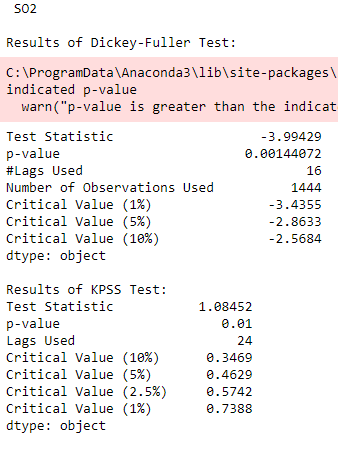


Fig3: Shows the test for stationarity for SO2

**2.3 Algorithms and Techniques:**

There are total four models used in the entire project to fit the monthly and daily data. Firstly, I tried all the models on daily data. Then I compared the fit with monthly data.

**2.3.1 Seasonal ARIMA (Benchmark model)**

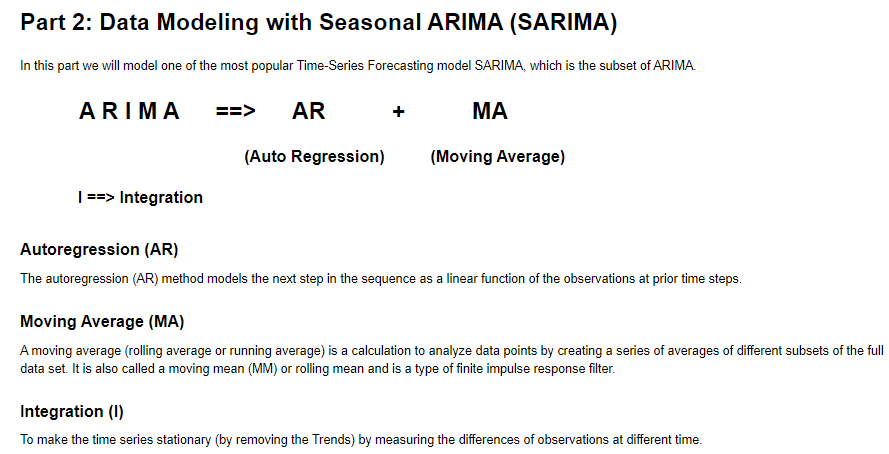


Fig 4: Shows the snapshot of my Part 2 Notebook on modeling with SARIMA

Seasonal ARIMA is one of the widely used model for Time Series Analysis. SARIMA model can handle data with trends and seasonality. Since, most of the data in real world is non-stationarity so this model is a good choice. Moreover, Air Quality data is tends to show seasonal behavior over time, So this model is a good choice for our data. Because of its popularity in Time-Series Analysis and its trends and seasonality handling capabilities I took this model as my **Benchmark model.**

**2.3.2 FBProphet (Facebook Prophet)**

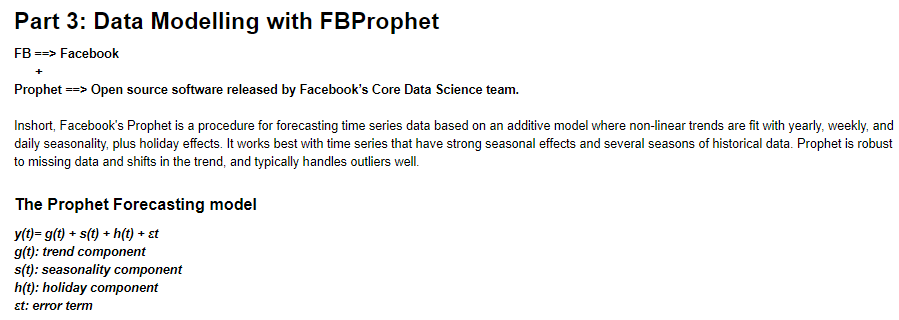


Fig 5 : shows the brief overview of FBProphet from the 3rd part of the notebook.

FBProphet is best suitable for data with strong seasonal behavior with effect of holiday’s. Daily data of PM2.5 was trained with the intention to keeping periodicity to 365 days with frequency as ‘D’ (Daily). Monthly data of all pollutants were trained with the intention of keeping periodicity 12 (for yearly periodicity) and freq as ‘MS’

**2.3.3 Holt-Winter**

Holt-Winter is simplest and popular Time-Series forecasting Algorithm which works on the principle of Single and multi-level Exponential Smoothening**.** Like FBProphet seasonal period was kept at 365 for daily data and at 12 for monthly data with the intention to apply yearly seasonality.

**2.3.4 Long Short Term Memory(LSTM) Recurrent Neural Network**

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the exploding and vanishing gradient problems that can be encountered when training traditional RNNs.

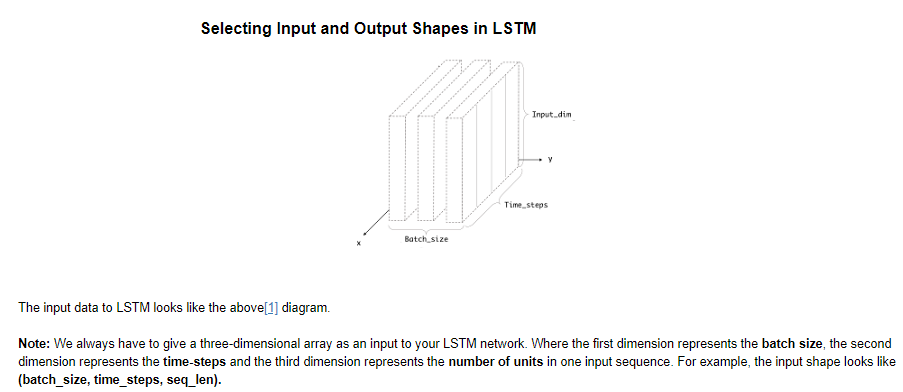


Fig 6: Gives a brief overview of selecting Input and Output shapes for LSTM.

Like other models, LSTM was first trained on daily data of PM2.5 pollutant and then on rest of the monthly pollutants.

2.4 **Benchmark Model**

Benchmark model was SARIMA. The Benchmark score of SARIMA obtained with respect different pollutants are:

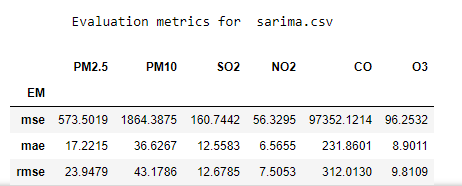


Fig 7: Shows the evaluation metrics for SARIMA.

**3 Methodology**

**3.1 Data Preprocessing**

**3.1.1 Indexing data with Date-Time**

The Time-Series index was created using pandas DateTimeIndex function with start date-time as 2013-03-01 00:00:00 and end date-time as 2017-02-28 23:00:00, freq as 'H' which stands for Hour.

**3.1.2 Removing unwanted columns**

Except major pollutants PM2.5, PM10, SO2, NO2, CO, O3, rest of the columns were removed. This was done as I decided to split these six major pollutants into six individual Time-Series for Time-Series Analysis.

**3.1.3 Handling Null values**

Among various methods like dropping null values, imputing with mean, median etc. I choose linear Interpolate method of pandas to fill null values with linear transition.

**3.1.4 Splitting data with respect to pollutant**

The data was splitted into six individual time series of pollutants. Each Time-Series is used for separate analysis.

**3.1.5 Resampling Data**

The hourly data was resampled to daily and monthly data. Daily Resampling was done by taking average value of pollutants all over the day. For monthly resampling, first the maximum concentration of pollutants all over the day was calculated. Then the data is resampled to monthly data by taking average of the maximum concentration of pollutants throughout the day.

**3.1.6 Train–Test split**

**3.1.6.1 For Daily data**

Sliced daily data from March-1-2013 to February-29-2016(note 2016 is a leap year) to Train set So that the remaining one year data can be used for testing.

**3.1.6.2 For Monthly data**

Sliced monthly data from March-2013 to February-2016 to Train-set and remaining one year data for Test-set.

Proportion of our train-test split was 0.33. Both monthly and daily Train-Test data were saved locally in ‘dataset/monthly’ and ‘dataset/daily’ in separate Train-Test folders.

**3.2 Implementation**

**3.2.1 SARIMA model**

**3.2.1.1 Searching for Parameters (p,d,q)(P,D,Q,S)**

In order to best fit SARIMA model into our daily data, we need to find best trend parameters(p, d, q) and Seasonal parameter (P,D,Q,S). Since we have very large daily data, it is not feasible to apply grid search for all possible parameters. So we'll use visual inspection of ACF (Autocorrelation) and PACF(Partial Autocorrelation) plots to find hyper parameters for our SARIMA model.

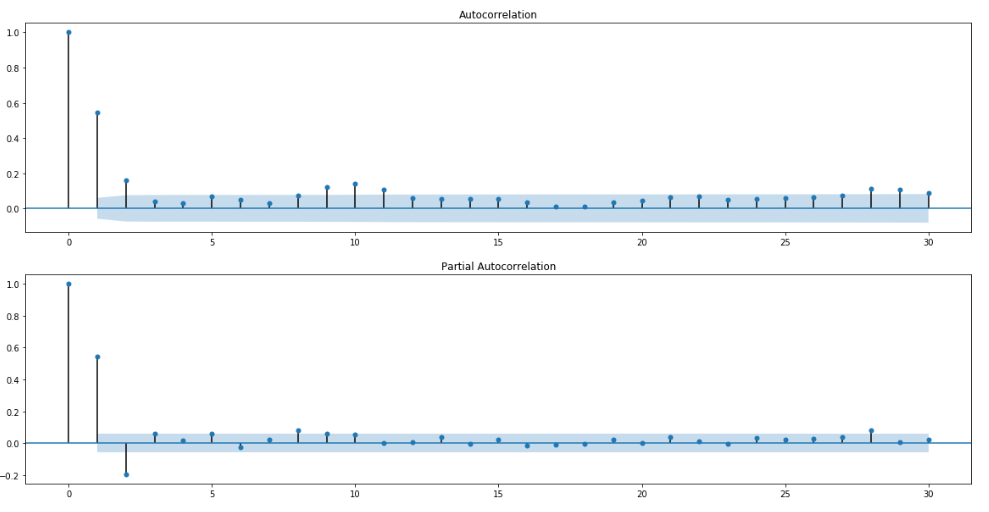


Fig 8: Shows the acf and pacf plots for monthly PM2.5.

### **3.2.1.2 Analysing ACF and PACF plots:**

We will use Rules for SARIMA model selection from ACF/PACF plots from this site: <https://www.datasciencecentral.com/profiles/blogs/tutorial-forecasting-with-seasonal-arima>

#### Rule for identifying d:

d=0 if the series has no visible trend or ACF at all lags is low.

d≥1 if the series has visible trend or positive ACF values out to a high number of lags.

In our case there is no visible trend and ACF is not low at all lags, So we will take d as 0 and 1.

#### Rule for Identifying the number of AR and MA terms (p and q)

p is equal to the first lag where the PACF value is above the significance level. So p will be 8.

q is equal to the first lag where the ACF value is above the significance level. So q will be 9.

#### Rule for Identifying the seasonal part of the model:

S is equal to the ACF lag with the highest value (typically at a high lag), So S may be 10.

D=1 if the series has a stable seasonal pattern over time.

D=0 if the series has an unstable seasonal pattern over time.

Since, In our case there is no stable seasonal pattern visible, So we'll take D = 0

**Rule of thumb: d+D≤2**

P≥1 if the ACF is positive at lag S, else P=0. Since ACF is positive at lag 10, So may take P as 1 or 2

Q≥1 if the ACF is negative at lag S, else Q=0. Since ACF is positive at lag 10, So we'll take Q as 0.

**Rule of thumb: P+Q≤2**

So, possible values for hyperparameters will be:  
  
p = 8  
d = 0 and 1  
q = 9  
P = 1 and 2  
D = 0  
Q = 0  
S = 10

**3.2.1.3 Defining Hyperparameters for SARIMA**

p, d, q, P, D, Q, S = 8, (0,1), 9, (1,2), 0, 0, 10

# lets fit the SARIMA model

for each\_d in d:

for each\_P in P:

mod = sm.tsa.statespace.SARIMAX(train\_PM25,

order=(p,each\_d,q),

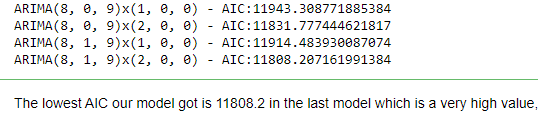
seasonal\_order=(each\_P,D,Q,S),

enforce\_stationarity=False,

enforce\_invertibility=False)

results = mod.fit()

print('ARIMA{}x{} - AIC:{}'.format((p,each\_d,q), (each\_P,D,Q), results.aic))



**3.2.1.4 SARIMA on Monthly data**

# define range for trend parameters Note: Since we have very less data so we'll keep possible range to 2

# define range for trend parameters Note: Since we have very less data so we'll keep possible range to 2

p = q = range(2)

d = range(0,2)

S = 12 # let's take S as 12 as our data frequency is monthly

# all combinations of (p,d,q)

pdq = list(itertools.product(p, d, q))

# all combinations of seasonal (P,D,Q)

seasonal\_PDQ = [(x[0], x[1], x[2], S) for x in list(itertools.product(p, d, q))]

best\_model = None

for trend\_param in pdq:

for seasonal\_param in seasonal\_PDQ:

try:

mod = sm.tsa.statespace.SARIMAX(train,

order=trend\_param,

seasonal\_order=seasonal\_param,

enforce\_stationarity=False,

enforce\_invertibility=False)

results = mod.fit()

if results.aic < min\_AIC:

min\_AIC = results.aic

best\_model = results

print('min till now- ',min\_AIC)

print('ARIMA{}x{} - AIC:{}'.format(trend\_param, seasonal\_param, results.aic))



**3.2.1.5 Challenge faced:**

SARIMA was not able to fit on daily data even after manual tuning of hyperparameters.

**3.2.2 FBProphet**

Model fit on Daily data

# load model

model = Prophet()

# fit model to our train\_PM25 data

model.fit(train\_PM25)

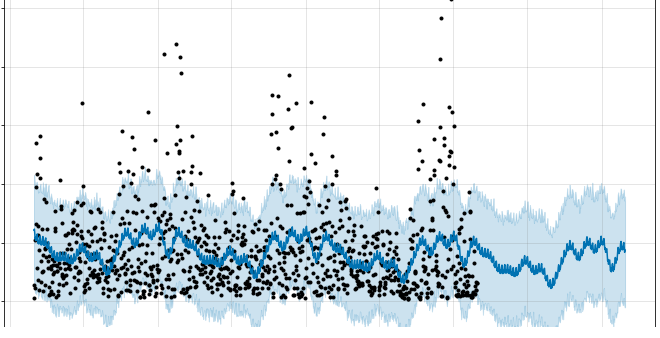
# specify period and frequency of predictions

# (We need to predict daily data for next one year so period will be 365 and frequency will be 'D')

future = model.make\_future\_dataframe(periods=365, freq = 'D')

# Predict using the Prophet model

forecast = model.predict(future)

Fig 8: FBProphet Fitting on Monthly data of PM2.5

#### Visualizing the forecast:

* Black dots are the data points(train data) used to train the model.
* The blue line in the graph represents the predicted values.
* Light blue line is the confidence interval

FBPhophet not able to fit into the data for daily PM2.5.

**3.3.3 FBPhrophet on monthly data of PM2.5**

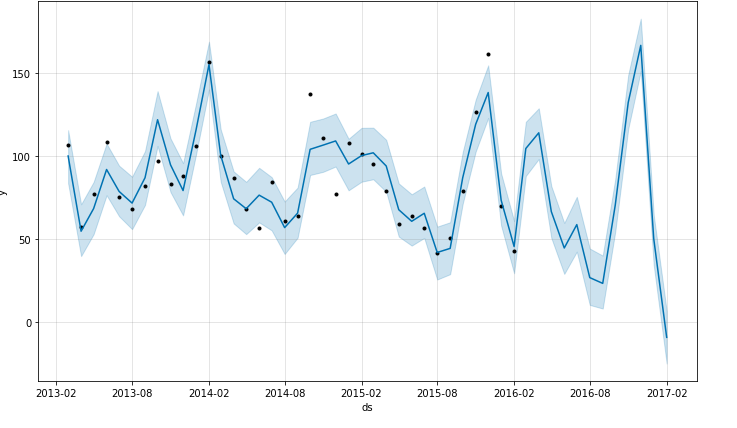


Fig: FBProphet fit better on monthly data than daily data of PM2.5.

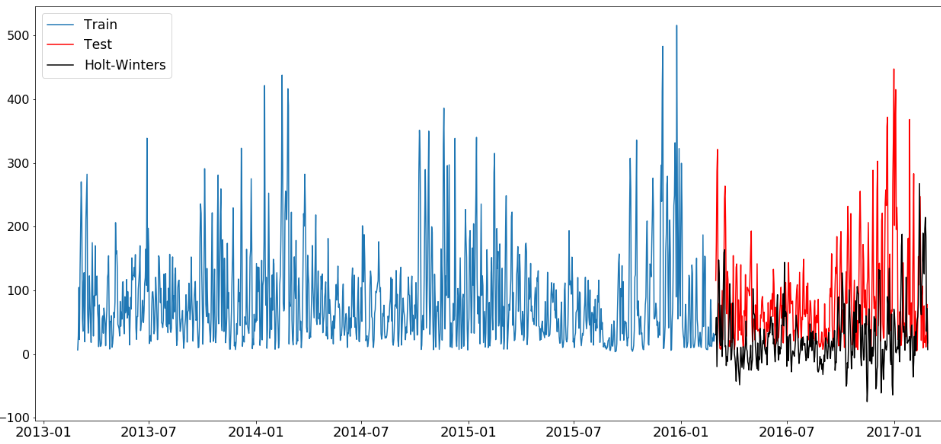
**3.2.3 Holt-Winter**

Daily data

model = ExponentialSmoothing(train\_PM25, seasonal='add', seasonal\_periods=365).fit()

# predict with the model

holt\_pred = model.predict(start=test\_PM25.index[0], end=test\_PM25.index[-1])

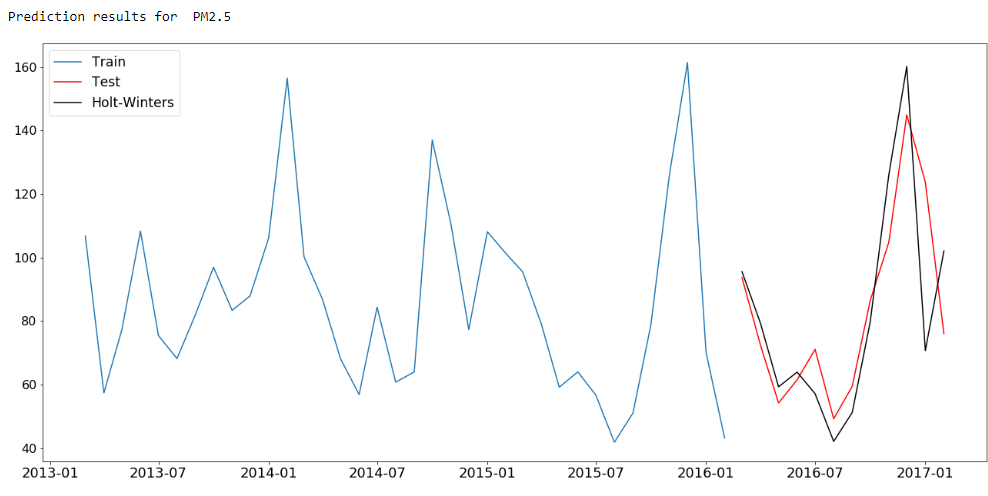


**3.2.3.1 Holt-Winter on monthly data**

model = ExponentialSmoothing(list\_train[i], seasonal='add', seasonal\_periods=365).fit()

# predict with the model

holt\_pred = model.predict(start=list\_test[i].index[0], end=list\_test[i].index[-1])



**3.2.4. LSTM**

Modelling Architecture for LSTM

# define generator object to produce batches for training/validation.

generator = TimeseriesGenerator(data = scaled\_train\_data, targets = scaled\_train\_data, length = time\_steps, batch\_size = batch\_size)

# Since we want to build LSTM network, In Keras we can simply stack multiple layers on top of each other,

# for this we need to initialize the model as Sequential().

model = Sequential()

# add input layer.

# units: We can take units as any no of dimensions(positive) for the outer space.

# activation: softmax, relu, softsign etc.

model.add(LSTM(units = 200, activation = activation, input\_shape = (time\_steps, n\_features)))

# Dropout: Every LSTM layer should be accompanied by a Dropout layer.

# 20% is often used as a good compromise between retaining model accuracy

# and preventing overfitting.

model.add(Dropout(0.25))

# add output layer the model. Since our model is making

# single predictions we'll take output layer as 1.

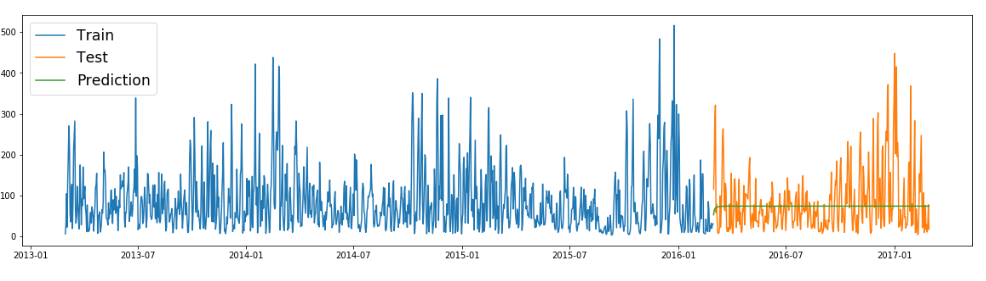
model.add(Dense(1))

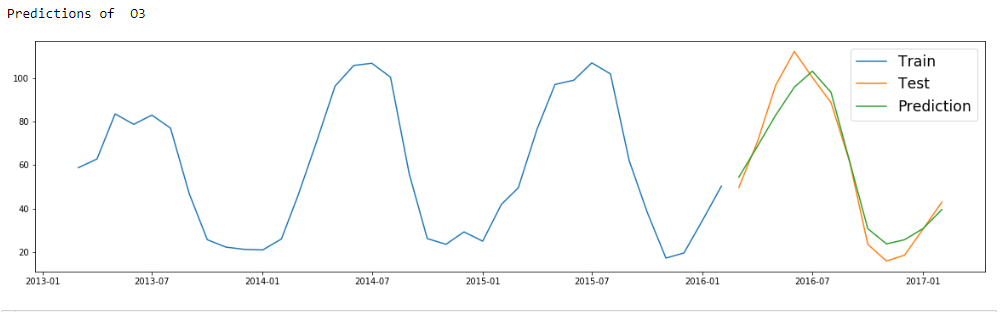
# compile the model. Generally we use 'adam' optimiser. Let's take loss as mean square error.

model.compile(optimizer='adam', loss='mse')

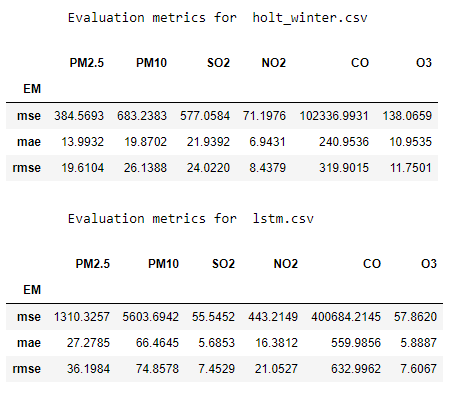
# fit the lstm neural network

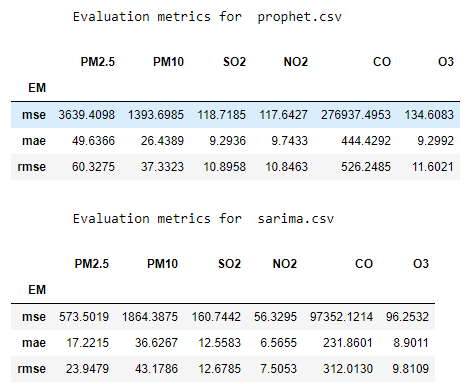
model.fit\_generator(generator, epochs = epochs)

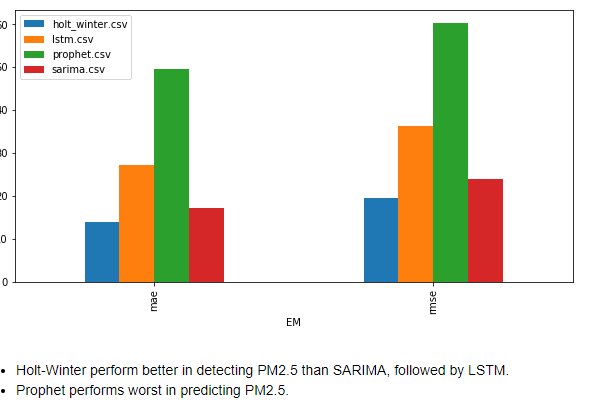


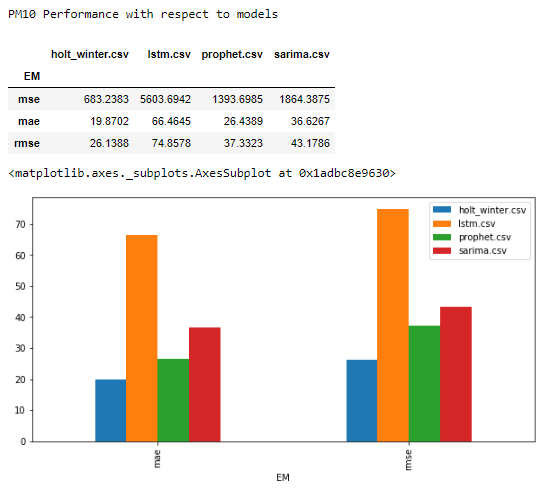


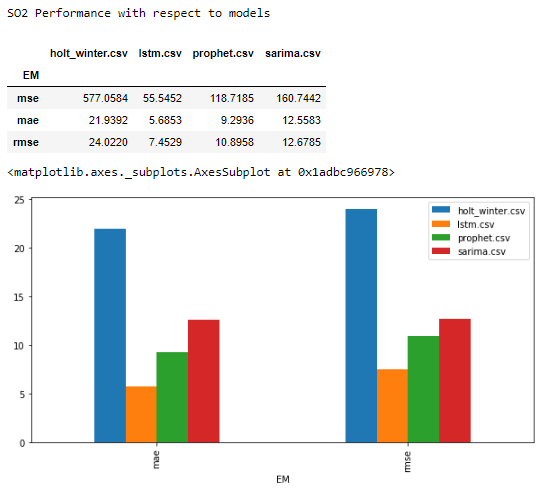
**4 Results:**

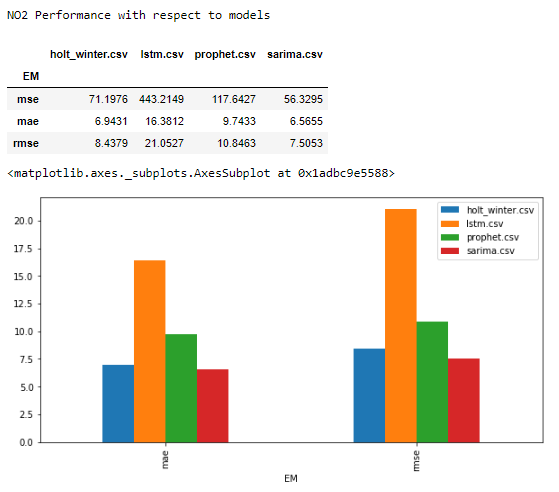


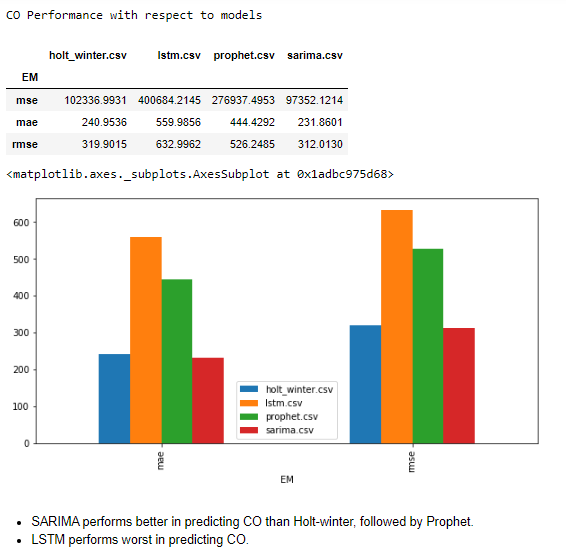


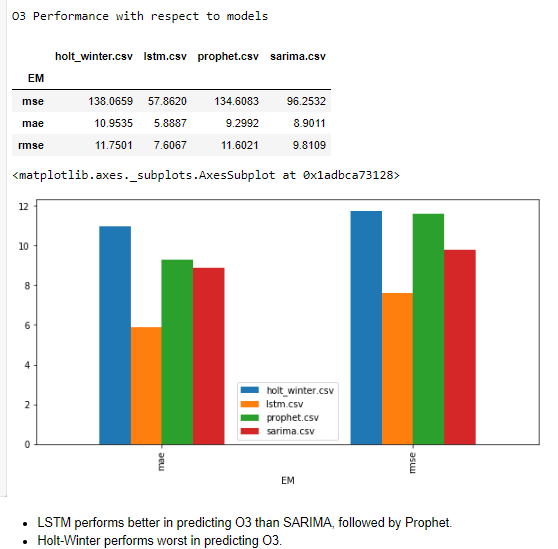












## **Summary**

* PM2.5 best fit : Holt-Winter
* PM10 best fit : Holt-Winter
* SO2 best fit : LSTM
* NO2 best fit : SARIMA
* CO best fit : SARIMA
* O3 best fit : LSTM

## **Conclusion**

Our selected benchmark model SARIMA was able to beat other models only in case of NO2 and CO. On the other hand, LSTM beat our Benchmark model in case of SO2 and O3.  
  
In this Time Series Analysis project, I just make an attempt to fit these models on real-time data as best as I can. There are still lots of aspects related to these models that are left uncovered due to limited time constraint. Like Hyperparameter Training, checking multiple methods for stationary data, series decomposition analysis, trend and seasonality analysis etc.  
  
These aspects affects greatly to the performance of the models and contrary takes much more time to tune and apply in real time. That's all for the conclusion part.  
Now we will look at some Future Scope that may improve the project

## **Future Scope**

* Hyperparameter Tuning
* Forecasting future predictions after achieving good performance score.
* Calculate AQI Index after accurately predicting all the pollutants.
* Deploy the Application to make predictions in real time.

### References:

1) [An end to end project on time series analysis and forecasting](https://towardsdatascience.com/an-end-to-end-project-on-time-series-analysis-and-forecasting-with-python-4835e6bf050b)  
2) [FBProphet](https://www.google.com/search?sxsrf=ALeKk01YKsxIDo-AuPRR01NvOGeTNqo8eQ%3A1582574836580&ei=9CxUXtKWI8uf4-EP85uVcA&q=prophet+facebook+wikipedia&oq=prophet+facebook+wi&gs_l=psy-ab.1.1.0j0i22i30l3.17066.19551..21671...0.2..0.435.927.2-2j0j1......0....1..gws-wiz.......0i71j0i67j0i20i263.1ym1Jo8udIk" \t "_blank)  
3) [Holt-Winters, SARIMA, FBProphet](https://towardsdatascience.com/a-quick-run-through-of-holt-winters-sarima-and-fb-prophet-c6370c6f32f1)  
4) [LSTM](https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/)  
5) [Time series with keras](https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/)  
6) [Understanding Input and Output shape in lstm](https://medium.com/@shivajbd/understanding-input-and-output-shape-in-lstm-keras-c501ee95c65e)