# Final Report

Ka Yin Ho A20436223 , Yin Yang A20450316

**Dataset: PM2.5 Data of Five Chinese Cities**
https://archive.ics.uci.edu/ml/datasets/PM2.5+Data+of+Five+Chinese+Cities

**Abstract**

Tackling climate change and alleviating air pollution emerge as the top agendum of the global community, so we are interested in studying pollution-related time series data to look for some insights. The data we are looking into is PM2.5 data of five Chinese cities provided by the UCI Machine Learning Repository. PM2.5, referring to atmospheric particulate matter with less than 2.5 micrometer, is one of the common pollutants from power plants and automobiles. It poses risk to the human's respiratory and heart system if the person is extensively exposed to high concentration of PM2.5. The data set includes the data of PM2.5 concentration over the period of January 1, 2010 and December 31, 2015 in the cities of Beijing, Shanghai, Guangzhuo, Shenyang, and Chengdu. Some climatological data like temperature, humidity, pressure, wind direction and speed, and precipitation are also included.

We aim to use the data to "predict" the data within the time range already included in the dataset, so we can match the predicted data to represent the accuracy of the actual data. In this report, we only focus on the data of Shanghai cities, but in fact, we have analyzed all five cities with multiple time ranges and you can find the code of the works in the GitHub link below.
https://github.com/AnnieYYang/TIMESERIES.PRJ.PM2.5.CHINA

## Contents

## A. Univariate Time Series Analysis

For univariate time series analysis, we intend to use different models and see which model can do the best job in forecasting the pollution level by the data of past PM2.5 concentration.

## 1 Non-seasonal ARIMA

### 1.1 Introduction

ARIMA (AutoRegressive Integrated Moving Average) models include autoregressive(AR) terms and/or moving average(MA) terms.

An autoregression(AR) model forecasts the variable of interest by a linear combination of past values of the variable. The term autoregression indicates that it is a regression of the variable against itself.

An autoregressive model of order p, AR(p), can be written as :

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t$$

where $\varepsilon_t$ is white noise.

A moving average(MA) model forecasts the variable of interest by a linear combination of past forecast errors in a regression-like model.

An moving average model of order q, MA(q), can be written as :

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

where $\varepsilon_t$ is white noise.

By combining the autoregression(AR) and a moving average(MA) model, we can obtain a non-seasonal ARIMA model.

The full model can be written as :

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

with the backshift notation(B), we can rewrite the full model of ARIMA(p,d,q) as :

$$(1 - \phi_1 B - \cdots - \phi_p B^p) \times (1 - B)^d y_t = c + (1 + \theta_1 B + \cdots + \theta_q B^q)\varepsilon_t$$

where p is the order of the autoregressive part.
d is the degree of differencing involved.
q is the order of the moving average part.

Note that the ARIMA model  can only be applied on the “stationary” time series, which means the data does not depend on the time at which the series is observed.

## 1.2 Application in Python

### 1.2.1 Data Preparation

We tried two types of data(df_1 and df_2) with different time lengths. In the beginning, we used hourly data(df_1) from Shanghai city with a duration of half a year (2015-01-01 00:00:00 ~ 2015-06-30 23:00:00).

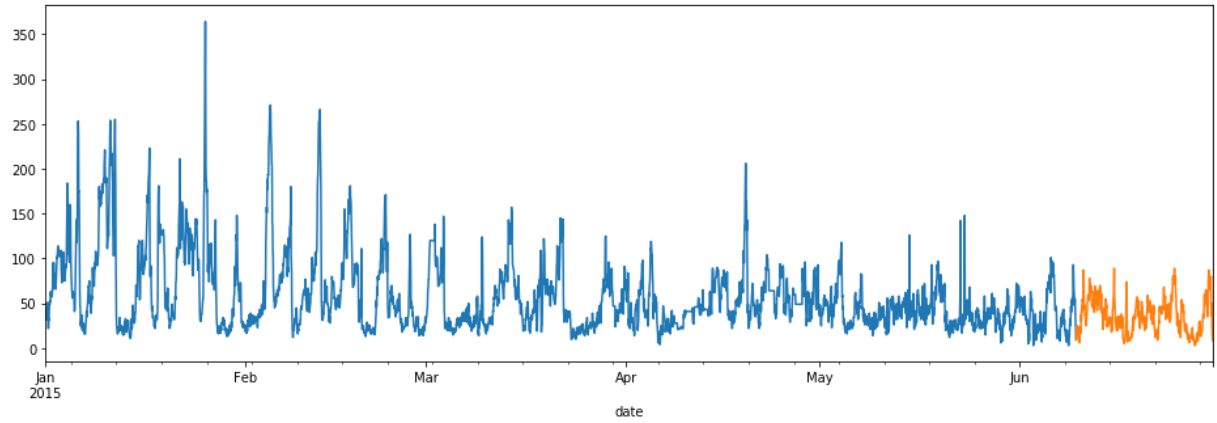



**Fig1.1** Show the time series df_1 and split it  into train set (80%) and test set (20%).

Then for the better results, we changed the time series from 6 months to 15 days (2014-03-28 00:00:00 ~ 2014-04-11 23:00:00), and in all subsequent univariate models, we use this time series(df_2) for analysis.

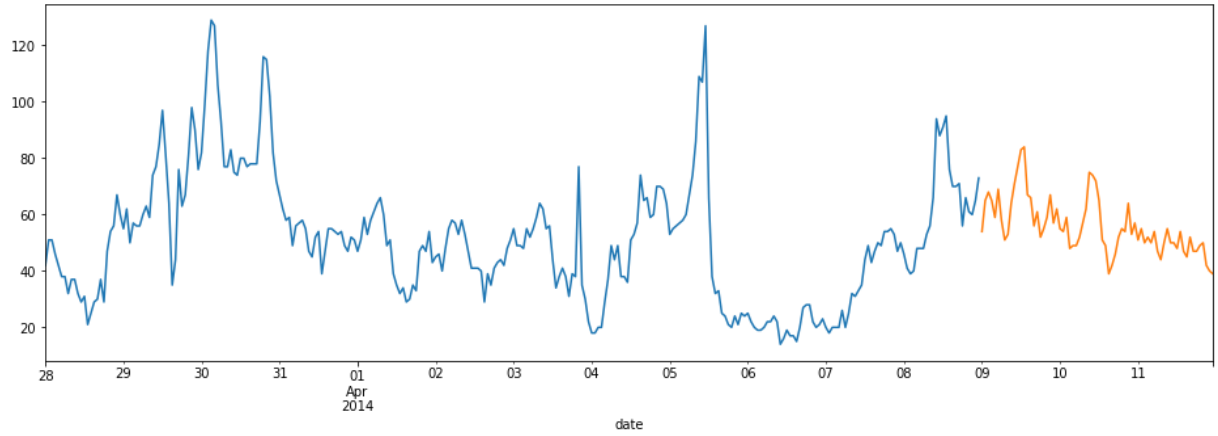



**Fig1.2** Show the time series df_2  and split it  into train-set (80%) and test-set (20%).

### 1.2.2 Stationarity and differencing

For an easier explanation, next we will use df_1 and df_2 to denote the time series of half a year and 15 days long, respectively.

We used the augmented Dickey–Fuller test to check the stationarity of the data.

Critical value(1%, 5%, 10%) can represent the comparison between the statistical value of the original hypothesis and the ADF Test result at different levels. If the ADF Test result is less than 1%, 5%, and 10% at the same time, it means that the hypothesis is rejected very well.
The null hypothesis of the ADF test is that there is a unit root. As long as the statistical value is less than 1%, the null hypothesis, the time series is non-stationary. , can be rejected very significantly, and the data is stable.

| | value |
|---|---|
| **Test Statistic Value** | -8.83326 |
| **p-value** | 1.74656e-14 |
| **Lags Used** | 16 |
| **Number of Observations Used** | 4327 |
| **Critical Value(1%)** | -3.43186 |
| **Critical Value(5%)** | -2.86221 |
| **Critical Value(10%)** | -2.56713 |

**Table1.1** Test stationarity for df_1 :
Since −8.3326 < −3.43186 the unit root hypothesis can be rejected at level 0.1.

| | value |
|---|---|
| **Test Statistic Value** | -4.28789 |
| **p-value** | 0.000466151 |
| **Lags Used** | 1 |
| **Number of Observations Used** | 358 |
| **Critical Value(1%)** | -3.44875 |
| **Critical Value(5%)** | -2.86965 |
| **Critical Value(10%)** | -2.57109 |

**Table1.2** Test stationarity for df_2 :
Since −4.28789 < -3.44875, the unit root hypothesis can be rejected at level 0.1.

Both result tables show that the ‘Test Statistic Value’ is less than the ‘Critical Value(1%)’, so we can reject the null hypothesis and say that both df_1 and df_2 time series are stationary data.

### 1.2.3 Order selection

#### 1.2.3.1 AutoCorrelation Function(ACF) / Partial AutoCorrelation Function(PACF)

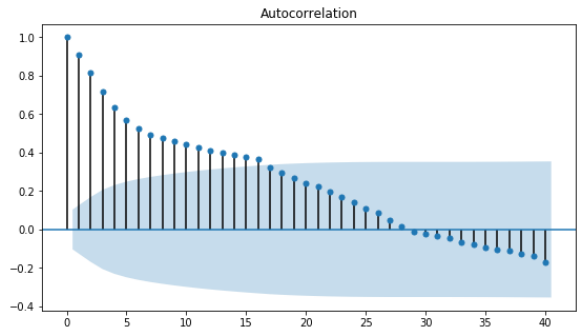
After checking the stationarity of the data, the next step in fitting an ARIMA model is to determine/estimate the number of AR or MA terms that are needed.

We used the ACF/PACF plot to estimate the p and q value, respectively. The ACF can be used to estimate the MA-part, i.e q-value, the PACF can be used to estimate the AR-part, i.e. p-value.

To estimate a model-order we look at

A. At which number of lag that ACF/PACF dies out sufficiently.

B. At which number of lag that ACF and PACF show significant interpretable peaks.



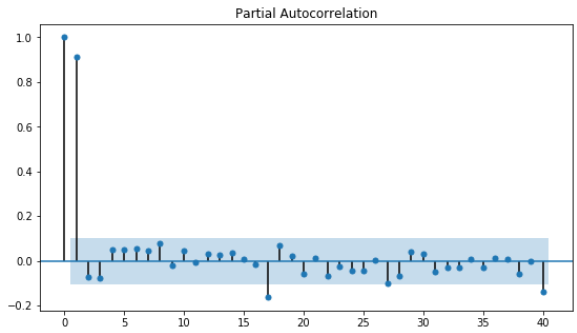
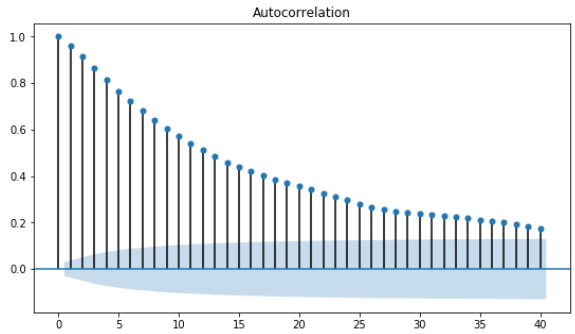




**Fig1.3** ACF/PACF plot for df_1 :
The PACF plot dies out after lag = 1 and shows spikes at lag = 1,17,40.



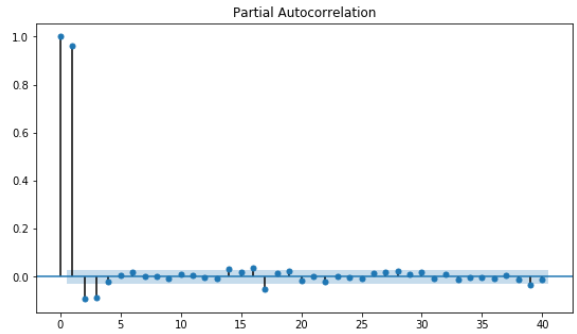




**Fig1.4** ACF/PACF plot for df_2 :
The PACF plot dies out after lag = 3 and shows spikes at lag = 1,2,3,17.

### 1.2.3.2 Akaike's Information Criterion (AIC)

$$AIC = -2\log(L) + 2(p + q + k + 1)$$

where $L$ is the likelihood of the data.

$k = 1$ if $c \neq 0$ and $k = 0$ if $c = 0$.

Akaike's Information Criterion (AIC), which was useful in selecting predictors for regression, is also useful for determining the order of an ARIMA model. The AIC value will allow us to compare how well a model fits the data and takes into account the complexity of a model, so models that have a better fit while using fewer features will receive a better (lower) AIC score than similar models that utilize more features.

By using the statsmodels module in Python,  we can determine the ARIMA(p,d,q) model for df_1 and df_2 to be ARIMA(1, 0, 2) and ARIMA(2, 0, 2), respectively.

### 1.2.4 Model Fitting and Residuals Testing

#### 1.2.4.1 Quantile-Quantile Plot

The quantile-quantile plot (Q-Q plot) is a graphical tool to help us assess if a set of data plausibly came from some theoretical distribution. It is a scatterplot created by plotting two sets of quantiles against one another. If both sets of quantiles came from the same distribution, we should see the points forming a line that's roughly straight.

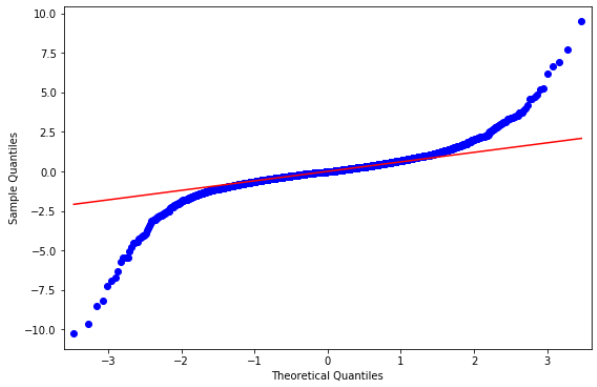



**Fig1.5** Q-Q plot after model fitted on df_1

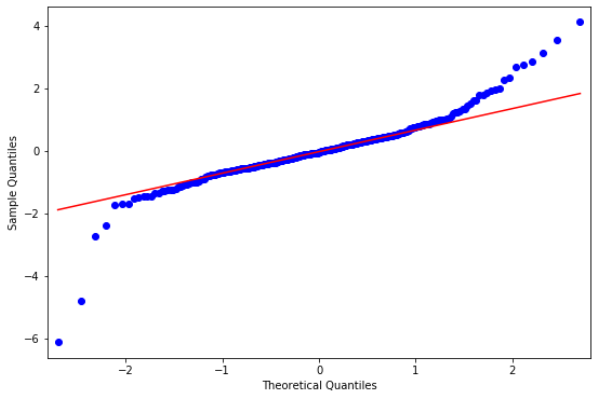



**Fig1.6** Q-Q plot after model fitted on df_2

In both Q-Q plots, we can see that points fall along a line in the middle of the graph, but curve off in the extremities.This behavior usually means that the data have more extreme values than would be expected if it truly came from the normal distribution.

### 1.2.5 Prediction & Validation

After we made the prediction from the fitted ARIMA model, we plotted a figure to compare the predictions and the true value and calculated Mean Squared Error(MSE) /  Root Mean Square Error (RMSE) values to validate the fitness of the ARIMA models.

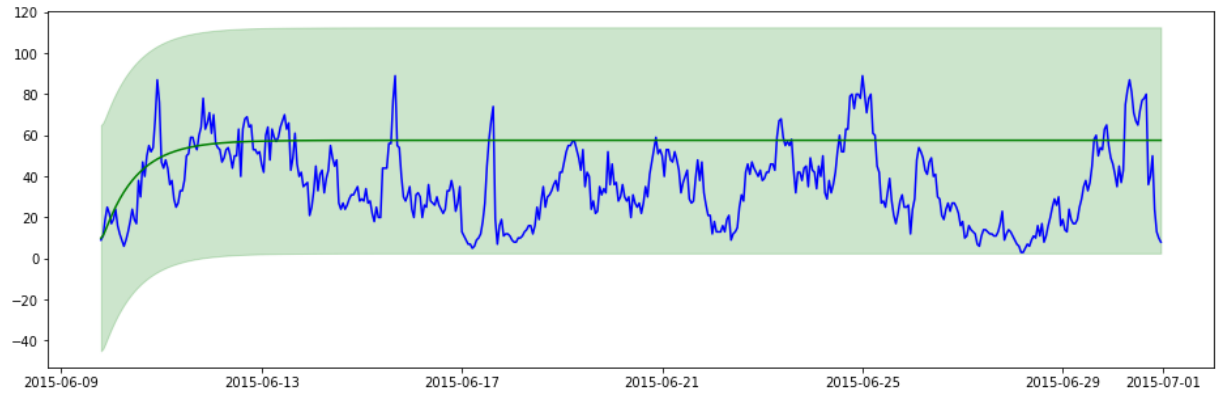



**Fig1.7** Predictions vs. Original data values in df_1

| | value |
|---|---|
| **MSE** | 883.243 |
| **RMSE** | 29.7194 |

**Table1.3** The MSE/RMSE values of predictions from the fitted model ARIMA(1,0,2)

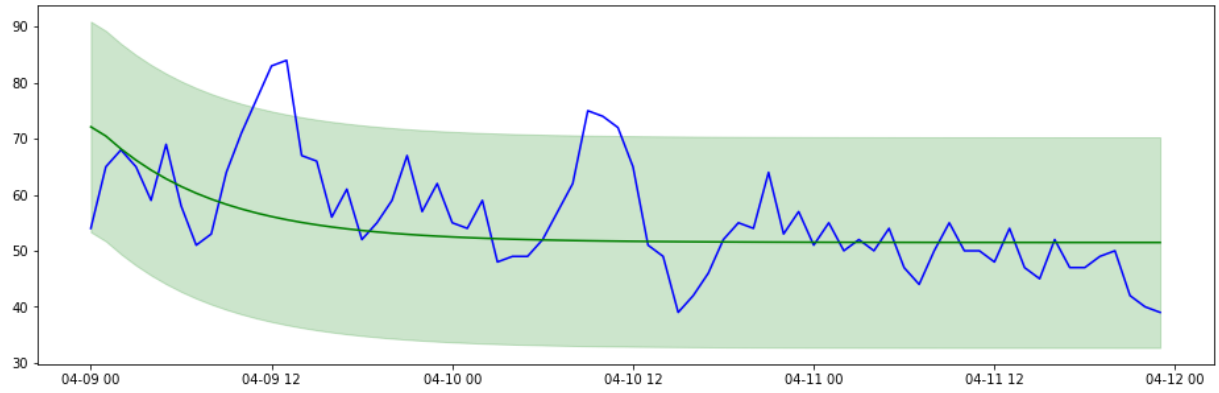



**Fig1.8** Predictions vs. Original data values in df_2

| | value |
|---|---|
| **MSE** | 88.2623 |
| **RMSE** | 9.3948 |

**Table1.4** The MSE/RMSE values of predictions from the fitted model ARIMA(2,0,2)

We were puzzled by the results of both predicting straight lines and soon realized that we should consider the seasonality in the data. So next, we applied seasonal ARIMA to the data, df_2.

## 2 Seasonal ARIMA

### 2.1 Introduction

A seasonal ARIMA(SARIMA) model is formed by including additional seasonal terms in the ARIMA models. The equation can be written as :

ARIMA( p , d , q )( P , D , Q )m

where m = number of observations per year

## 2.2 Application in Python

### 2.2.1 Decompose the Time Series

From the plot of our raw data, we can see that the model does not vary in frequency and amplitude over time. Therefore, we can conclude it as an additive time series model :

$$y(t) = \text{Level} + \text{Trend} + \text{Seasonality} + \text{Noise}$$

where Level: is the average value in the series.
Trend: is the increasing or decreasing value in the series.
Seasonality: is the repeating the short-term cycle in the series.
Noise: is the random variation in the series.

After applying the seasonal_decompose() function on df_2, we can get 4 resulting decompose plots that indicate a significant (daily) seasonality in the series.

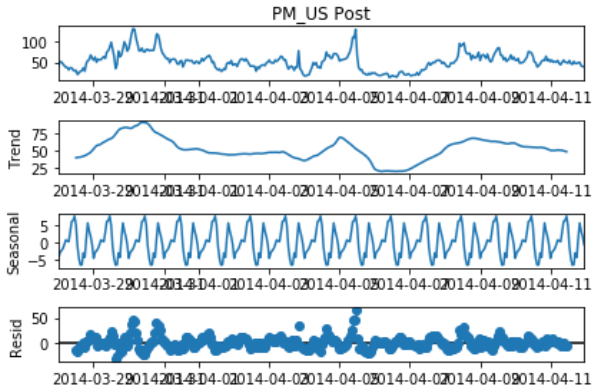



**Fig2.1** Decomposition of the time series df_2.

### 2.2.2 Order selection

The pyramid-arima library in Python contains an auto_arima() function that allows us to set a range of p,d,q,P,D,and Q values and then fit models for all the possible combinations. Then the model will keep the combination that reported back the best AIC value.

It took us a while to understand how to set the "m" value, which is related to the number of observations in each season cycle and must be known a priori. Finally, we realized that we can use the seasonal graph obtained by seasonal_decompose () to obtain the seasonality of the time series. As can be seen from the above figure, df_2 has "daily" seasonality. And because our data is an "hourly" time series, this means that a cycle contains 24 data points, so we set m to 24.

SARIMAX Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | PM_US Post | **No. Observations:** | 288 |
| **Model:** | SARIMAX(1, 0, 0)x(1, 1, [1], 24) | **Log Likelihood** | -986.904 |
| **Date:** | Sun, 26 Apr 2020 | **AIC** | 1981.807 |
| **Time:** | 20:28:35 | **BIC** | 1996.111 |
| **Sample:** | 03-28-2014 | **HQIC** | 1987.555 |
| | - 04-08-2014 | | |
| **Covariance Type:** | opg | | |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **ar.L1** | 0.9217 | 0.022 | 41.850 | 0.000 | 0.879 | 0.965 |
| **ar.S.L24** | -0.0285 | 0.093 | -0.308 | 0.758 | -0.210 | 0.153 |
| **ma.S.L24** | -0.9025 | 0.143 | -6.290 | 0.000 | -1.184 | -0.621 |
| **sigma2** | 88.4950 | 8.396 | 10.540 | 0.000 | 72.040 | 104.951 |

| | | | |
|---|---|---|---|
| **Ljung-Box (Q):** | 29.08 | **Jarque-Bera (JB):** | 374.33 |
| **Prob(Q):** | 0.90 | **Prob(JB):** | 0.00 |
| **Heteroskedasticity (H):** | 1.16 | **Skew:** | -0.50 |
| **Prob(H) (two-sided):** | 0.49 | **Kurtosis:** | 8.75 |

**Table2.1** Summary after applying the auto_arima() function to the df_2

From the summary table, we can see that the best arima model chosen by auto_arima() is SARIMAX(1,0,0)x(1, 1, [1], 24).

### 2.2.3 Model Fitting and Residuals Testing

#### 2.2.3.1 Quantile-Quantile Plot

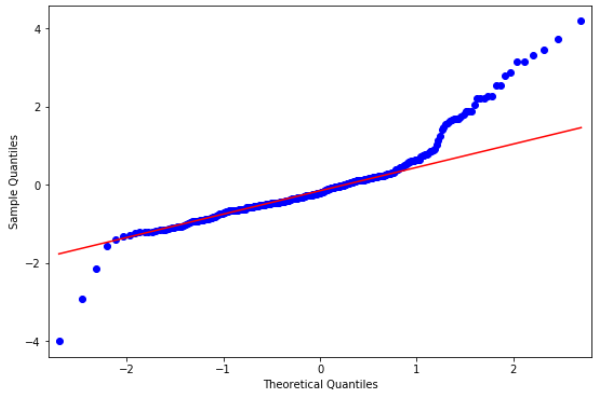



**Fig2.2** Q-Q plot after model fitted on df_2

### 2.2.4 Prediction & Validation

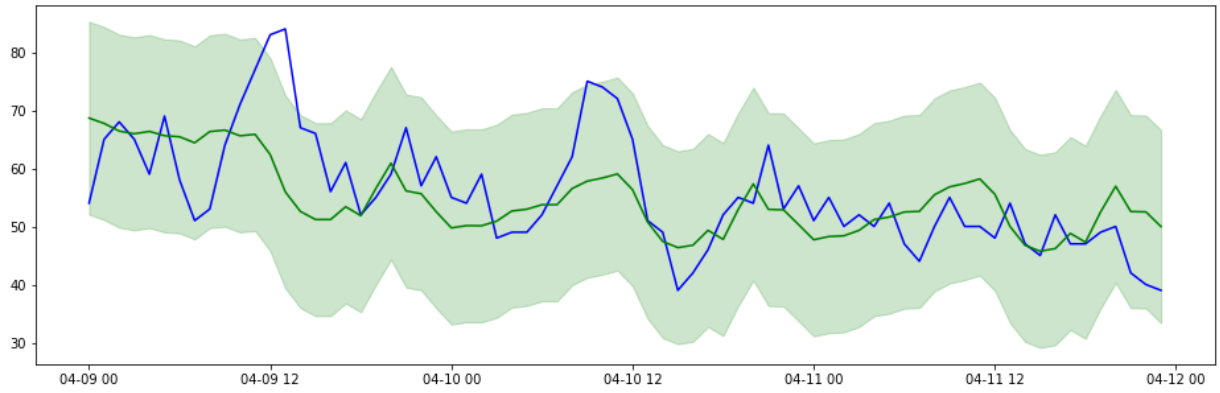



**Fig2.3** Predictions vs. Original data values in df_2

| | value |
|---|---|
| **MSE** | 69.0088 |
| **RMSE** | 8.30715 |

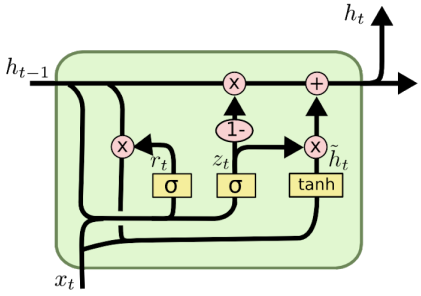
**Table2.2** The MSE/RMSE values of predictions from the fitted model SARIMAX(1,0,0)x(1, 1, [1], 24).

## 3 Long Short-Term Memory

### 3.1 Introduction

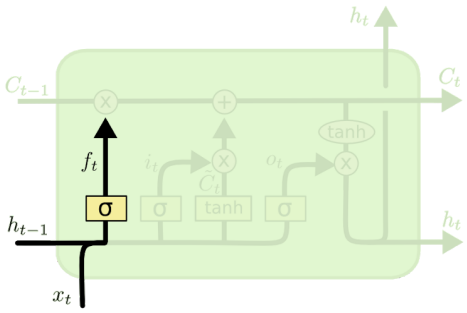
At first, when we tried to study why we always get straight-line prediction results in the ARIMA model, we also tried to use other methods to make predictions. While searching, we found that many people will use the Long Short-Term Memory(LSTM) to model univariate time series to solve prediction problems. Therefore, we spent some time studying and trying to see if this method can be used to obtain better prediction results.

Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. LSTMs are explicitly designed to avoid the long-term dependency problem by their chain-like structure with four neural network layers interacting in a very special way.



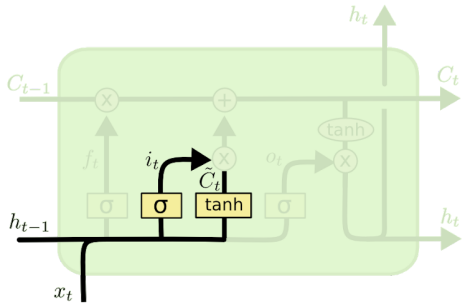

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

We'll briefly specify the core ideas behind this network and how each layer works as following:



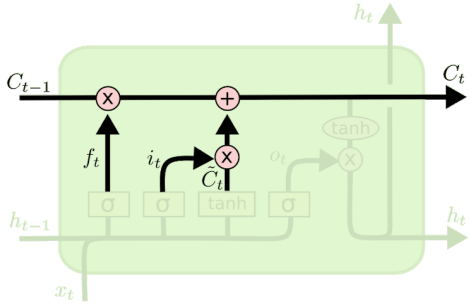

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

1. At the first layer, it will decide what information we're going to ignore from the cell state. It depends on the value of $h_{t-1}$ and $x_t$, and outputs a number between 0 and 1 for each number in the cell state $C_{t-1}$. A 1 represents "completely keep this" while a 0 represents "completely forget this."



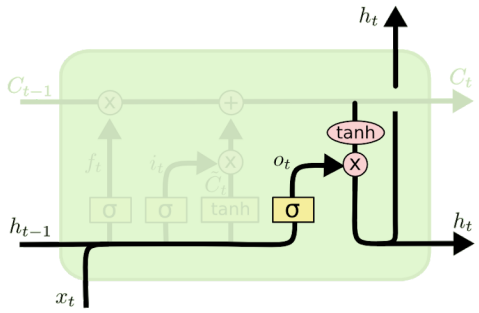

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

2. At the second layer, it will decide what information we're going to store in the cell state. It means that, in the process, which value needs to be updated and the new candidate value will be decided.




$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

3. Update the old cell state, $C_{t-1}$, into the new cell state $C_t$.




$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$



4. The Final step is that we need to decide what information we're going to output.

## 3.2 Application in Python

### 3.2.1 Scale Data

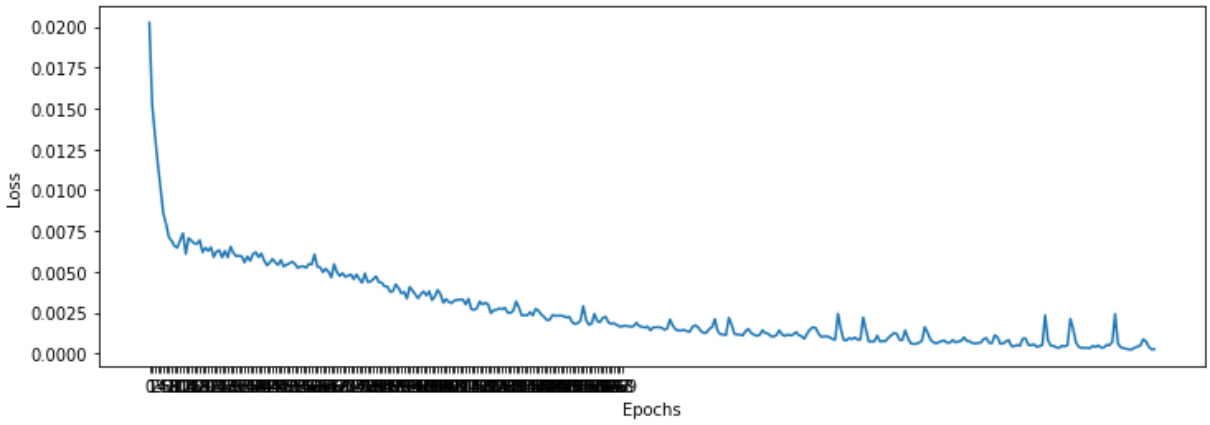
LSTMs are sensitive to the scale of the input data, therefore it can be a good practice to rescale/ normalize the data to the range of 0-to-1. We used the MinMaxScaler preprocessing class from the scikit-learn library in Python to scale the data.

### 3.2.2 Create the LSTM model

Before creating the LSTM model we should create a Time Series Generator object by setting the number of inputs, features and the value batch_size.When training neural networks, we generally feed data into them in mini-batches or just "batches". Keras has some handy functions which can extract training data automatically from a pre-supplied Python iterator/generator object and input it to the model. We use one of these Keras functions which is called fit_generato() to extract batches of data during the training process.

To specify the LSTM layer, first, we must determine the number of nodes in the hidden layer within the LSTM unit. For example, the number of cells in the forget gate layer and the input gate layer ... the number should be set between the size of the input layer and the size of the output layer.

Then, we can use the "Adam Optimizer" as the optimizer to compile the commands on the model to view the loss(MSE) graph to visualize the accuracy increase during training.




\

**Fig3.1** loss(MSE) graph

Finally, we can start training the Keras LSTM model by setting the number of training time in the fit_generator () function.

### 3.3 Prediction & Validation

Note that the predictions from the model will be scaled values between 0~1 as well, so we have to transform the values back to true values of predictions to do the validation.

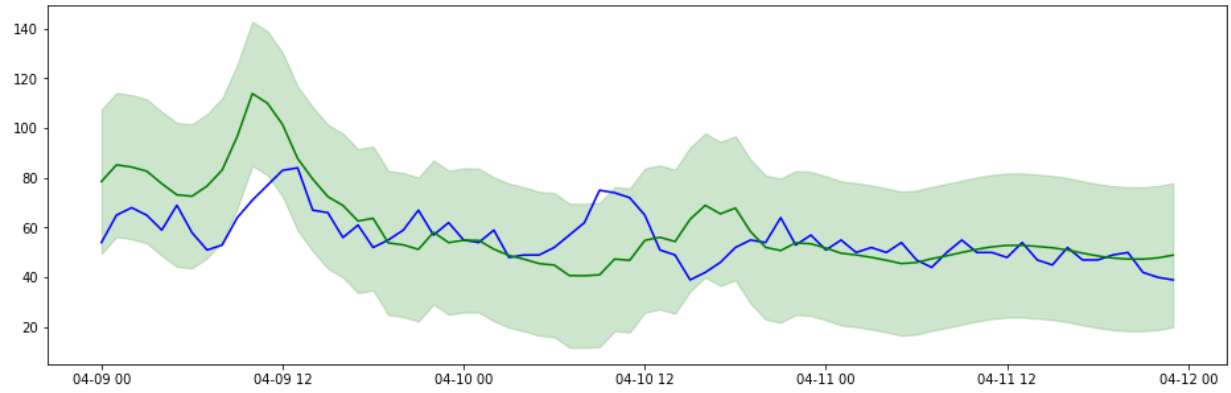



**Fig3.2** Predictions vs. Original data values in df_2

| | value |
|---|---|
| **MSE** | 209.317 |
| **RMSE** | 14.4678 |

**Table3.1** The MSE/RMSE values of predictions from the LSTM model.

## 4 Prophet

## 4.1 Introduction

Prophet is a procedure implemented in R and Python for forecasting time series data based on an additive regression model.

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t$$

where $g(t)$ is the non-periodic changes (trend)

$s(t)$ is the periodic change (eg weekly / annual seasonality)

$h(t)$ is the irregular holiday effects

$\varepsilon_t$ is the error term (normal distribution)

Prophet can automatically detect changes in trends by selecting changepoints from the data for the piecewise linear or logistic growth curve trend. And use Fourier series for seasonal components in the model and use dummy variables for detecting weekly seasonal components. Particularly, we can include the additional holidays effect in the model.

The Prophet procedure is robust to missing data and shifts in the trend, and typically handles outliers and is more applicable to business behavior data with obvious internal laws (or models).

## 4.2 Application in Python

Because the Prophet was developed for people who usually do not have a strong mathematical background but need to perform time series analysis. Using this process set, we do not need to set any parameters, just convert the input data to the required format and name. Our .csv data needs to contain two columns named "ds" and "y", representing the data time and the variables we want to predict.

## 4.3 Prediction & Validation

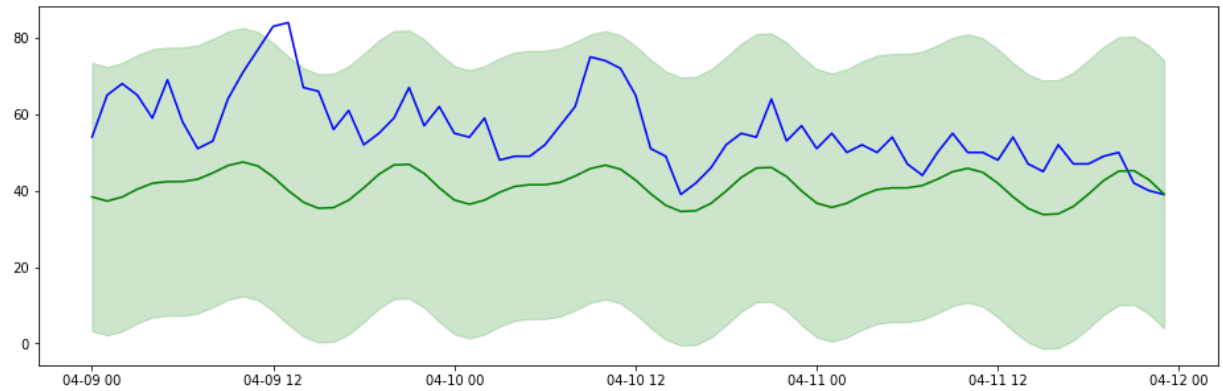



**Fig4.1** Predictions vs. Original data values in df_2

| | value |
|---|---|
| **MSE** | 308.192 |
| **RMSE** | 17.5554 |

**Table4.1** The MSE/RMSE values of predictions from the Prophet.

## 5 Predicting Results Comparison

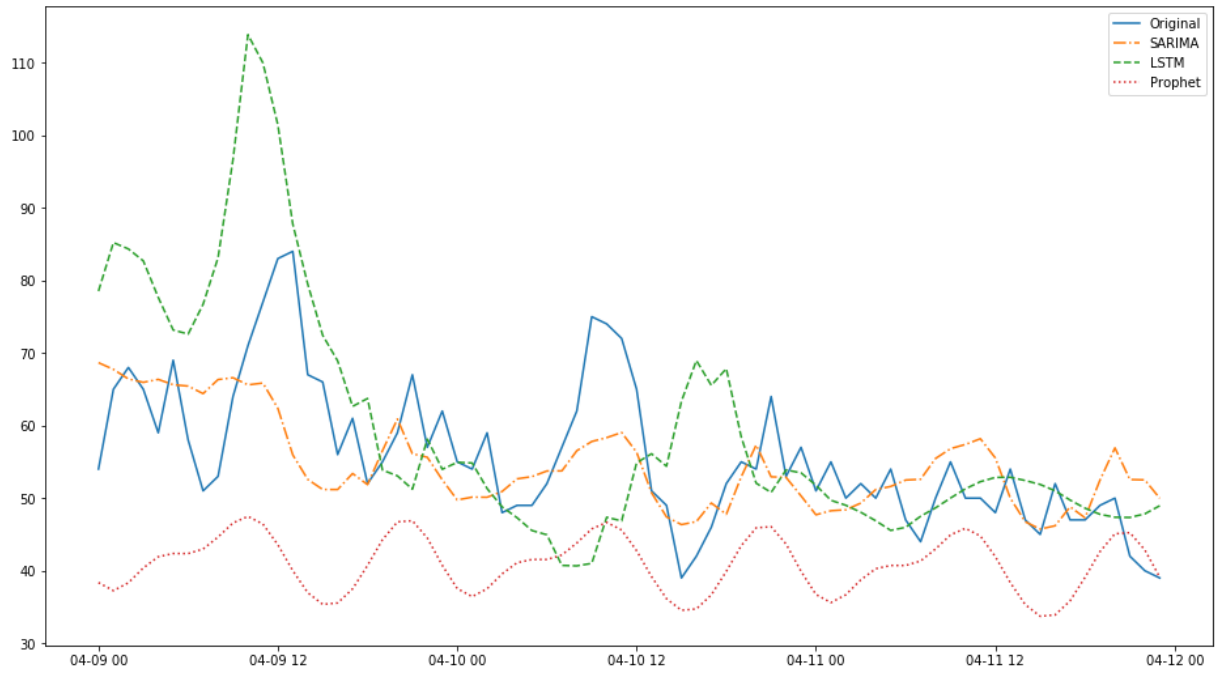



**Fig5.1** Predictions vs. Original data values in df_2

| | value | | |
|---|---|---|---|
| | SARIMA | LSTM | Prophet |
| **MSE** | 69.0088 | 209.317 | 308.192 |
| **RMSE** | 8.30715 | 14.4678 | 17.5554 |

**Table5.1** The MSE/RMSE values of predictions from Seasonal ARIMA/LSTM/Prophet.

From the validation results, we can find that for 15 days of hourly data df_2, the SARIMA model works better than the LSTM and Prophet methods. This may be reasonable because LSTM and Prophet are more suitable for time series with a longer time range. However, we may need a stronger computer (with a stronger CPU) to process more data to find out.

## B. Multivariate Time Series Analysis

After some extensive univariate time series analyses, we would also like to look into (a) the effect of climatological factors on the PM2.5 pollution and explore (b) the potential relationship of the PM2.5 pollution among the five cities. If these exogenous factors are highly correlated to the PM2.5 pollution of a specific city, the forecasting result is assumed to be more accurate, and ideally we could get a better forecasting model. In this section, we would focus on analyzing the city of Shanghai.

In this section, we decided to apply the Vector Autoregressive (VAR) model as an extension of our univariate time series analysis from a perspective of multivariate model. Meanwhile, our variable of interest, the PM2.5 concentration, is observed to be stationary. Even if other predictors are non-stationary, we could apply differencing to transform the data into stationary time series data for VAR analysis.

## 6. Vector Autoregression (VAR)

### 6.1 Introduction

A Vector Autoregression (VAR) model is formed by a linear combination of the product of the past value of the variable of interest and the past value of the exogenous variables in a coefficient matrix representation, the intercept vector to include non-zero intercept, and the white noise component. The process does not only regress the variable against itself, but also other variables taken into the model. The equation can be written as :

$$Y_t = \nu + A_1 Y_{t-1} + \ldots + A_p Y_{t-p} + u_t$$
$$u_t \sim \text{Normal}(0, \Sigma_u)$$

where Y_t is (K x 1) random vector,
v is the (K x 1) intercept vector,
Ai are fixed (K x K) coefficient matrices,
u_t is K-dimensional white noise.

Similar to the univariate AR model, the VAR model only accepts “stationary” time series. Supported by a Python package *statsmodel.tsa.api*, the modeling with VAR requires hyperparameter tuning on the number of lags (*h*) for order selection. And the prediction will be returned based on the fitted train data set, so if the fitted train data set is transformed with differencing, the final result needs to be rescaled for the actual value. For residual analysis, other than quantile-quantile plots, there are some VAR-specific test analysis techniques such as Durbin-Watson test for serial correlation of residual, impulse response analysis, and forecast error variance decomposition (FEVD), which will be discussed for each model.

## 6.2 Application in Python

### 6.2.1 Data Preparation

In all multivariate analyses, we used daily data from the city of Shanghai with a duration of two years (2013-05-01 03:00:00 ~ 2015-04-30 03:00:00), at which the missing value is limited.

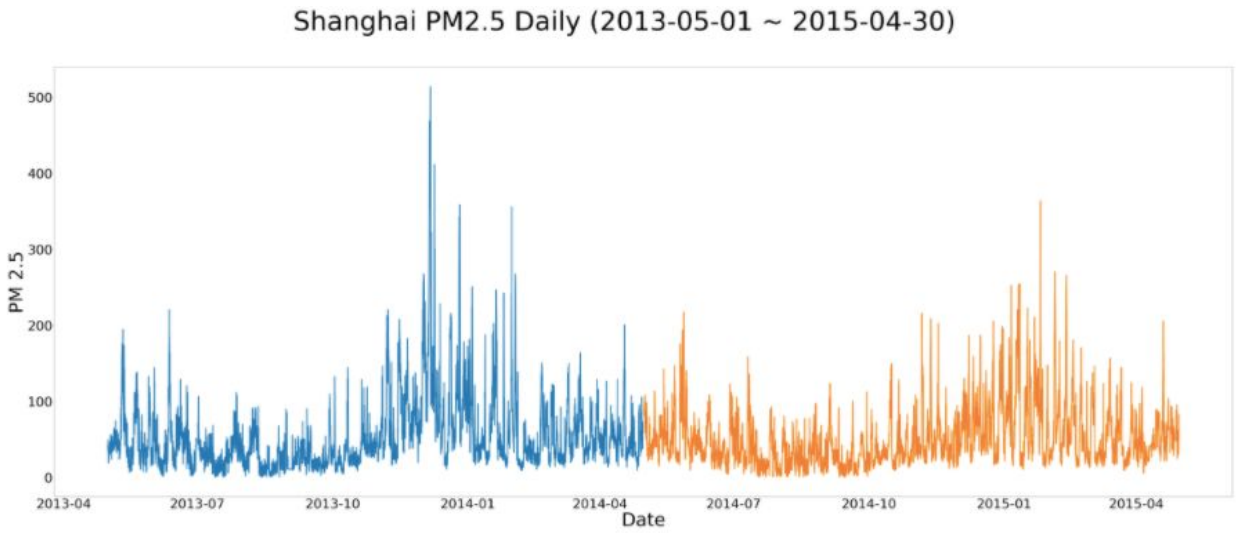



**Fig6.1** Show the daily data of Shanghai from 2013-05-01 03:00:00 to 2015-04-30 03:00:00 .

In the beginning, we tried to use hourly data so as to be coherent with our univariate analysis. However, we found the result fails to catch the spikes and drops as the figure shows below. Also. concerning the missing data and an observation that the prediction changes along every lag *h*, we think using hourly data might not be very reliable, so we turn to using daily data.

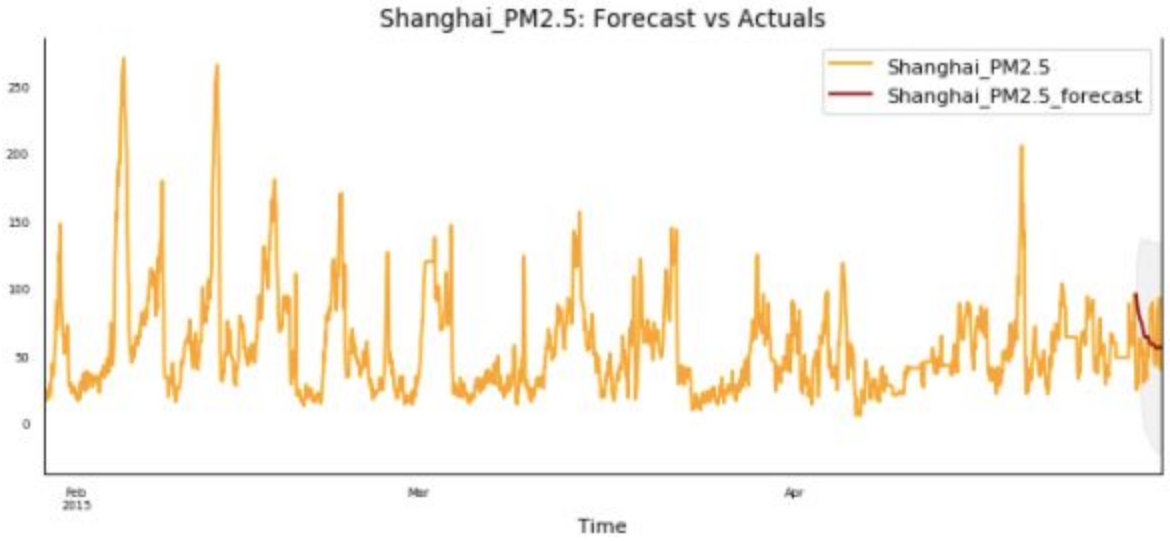



**Fig6.2** Result plot

Now we divide the discussion of modelling with respect to our problem statement.

**(a) The effect of climatological factors on the PM2.5 pollution of Shanghai**

### 6.2.2 Stationarity and differencing

We used the augmented Dickey–Fuller test to check for stationarity. The results show the ordinary data of PM2.5 concentration, humidity (HUMI), and cumulated wind speed (Iws) are "stationary", but pressure (PRES), dew point (DEWP), and temperature (TEMP) are "non-stationary".

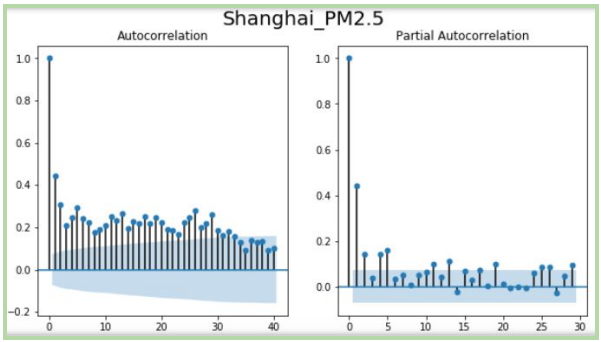



**Fig6.3** ACF/PACF

| | value |
|---|---|
| **Test Statistic Value** | -3.019492 |
| **p-value** | 0.033114 |
| **Lags Used** | 14 |
| **Number of Observations Used** | 569 |
| **Critical Value(1%)** | -3.441895 |
| **Critical Value(5%)** | -2.866633 |
| **Critical Value(10%)** | -2.569482 |

**Table 6.1** Test stationarity for Shanghai_PM2.5 : Since -3.019492 < -2.866633 the unit root hypothesis can be rejected at level 0.05.

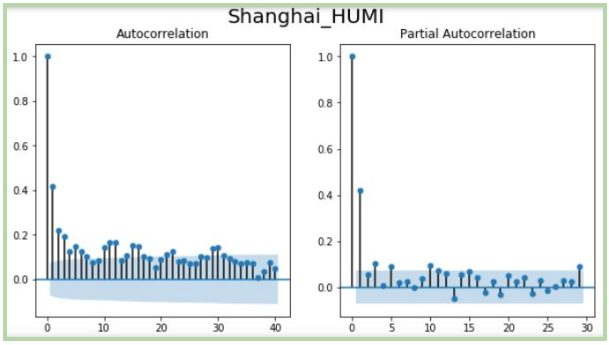



**Fig6.4** ACF/PACF

| | value |
|---|---|
| **Test Statistic Value** | -9.374875 |
| **p-value** | 7.212311e-16 |
| **Lags Used** | 2 |
| **Number of Observations Used** | 581 |
| **Critical Value(1%)** | -3.441655 |
| **Critical Value(5%)** | -2.866527 |
| **Critical Value(10%)** | -2.569426 |

**Table 6.2** Test stationarity for Shanghai_HUMI : Since -9.374875 < -2.866527 the unit root hypothesis can be rejected at level 0.05.

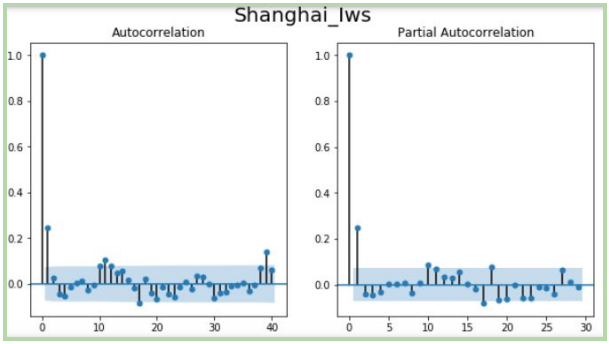



**Fig6.5** ACF/PACF for Shanghai_Iws

| | value |
|---|---|
| **Test Statistic Value** | -1.841663e+01 |
| **p-value** | 2.179747e-30 |
| **Lags Used** | 0 |
| **Number of Observations Used** | 583 |
| **Critical Value(1%)** | -3.441616 |
| **Critical Value(5%)** | -2.866510 |
| **Critical Value(10%)** | -2.569417 |

**Table 6.3** Test stationarity for Shanghai_Iws : Since -1.841663e+01 < -2.866510 the unit root hypothesis can be rejected at level 0.05.

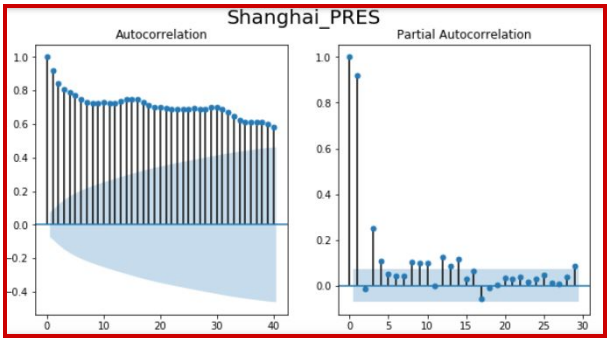



**Fig6.6** ACF/PACF for Shanghai_PRES

| | value |
|---|---|
| **Test Statistic Value** | -1.214146 |
| **p-value** | 0.667481 |
| **Lags Used** | 15 |
| **Number of Observations Used** | 568 |
| **Critical Value(1%)** | -3.441915 |
| **Critical Value(5%)** | -2.866642 |
| **Critical Value(10%)** | -2.569487 |

**Table 6.4** Test stationarity for Shanghai_PRES : Since -1.214146 > -2.866642 the unit root hypothesis fails to be rejected at level 0.05.

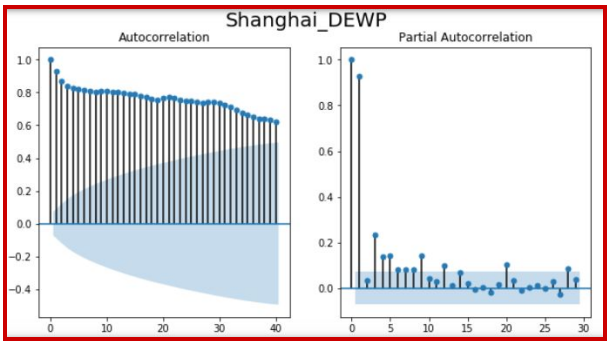



**Fig6.7** ACF/PACF for Shanghai_DEWP

| | value |
|---|---|
| **Test Statistic Value** | -1.118542 |
| **p-value** | 0.707621 |
| **Lags Used** | 14 |
| **Number of Observations Used** | 569 |
| **Critical Value(1%)** | -3.441895 |
| **Critical Value(5%)** | -2.866633 |
| **Critical Value(10%)** | -2.569482 |

**Table 6.5** Test stationarity for Shanghai_DEWP : Since -1.118542 > -2.866633 the unit root hypothesis fails to be rejected at level 0.0.5.

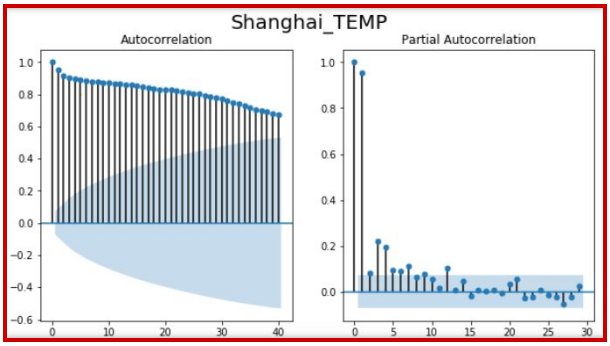



**Fig6.8** ACF/PACF for Shanghai_TEMP

| | value |
|---|---|
| **Test Statistic Value** | -0.933484 |
| **p-value** | 0.776746 |
| **Lags Used** | 11 |
| **Number of Observations Used** | 572 |
| **Critical Value(1%)** | -3.441834 |
| **Critical Value(5%)** | -2.866606 |
| **Critical Value(10%)** | -2.569468 |

**Table 6.6** Test stationarity for Shanghai_TEMP : Since -0.933484> -2.866606 the unit root hypothesis fails to be rejected at level 0.05.

After one-step differencing, we checked all variables with the augmented Dickey-Fuller test again. All time series show “stationary”, and are ready for modeling.

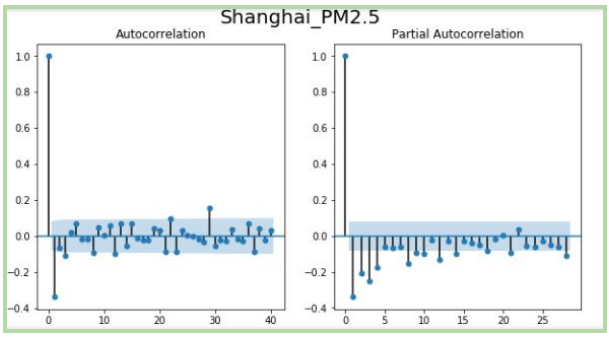



**Fig6.9** ACF/PACF for Shanghai_PM2.5

| | value |
|---|---|
| **Test Statistic Value** | -9.564868 |
| **p-value** | 2.371923e-16 |
| **Lags Used** | 17 |
| **Number of Observations Used** | 565 |
| **Critical Value(1%)** | -3.441977 |
| **Critical Value(5%)** | -2.866669 |
| **Critical Value(10%)** | -2.569502 |

**Table 6.7** Test stationarity for Shanghai_PM2.5 : Since -9.564868 < -2.866669 the unit root hypothesis can be rejected at level 0.05.

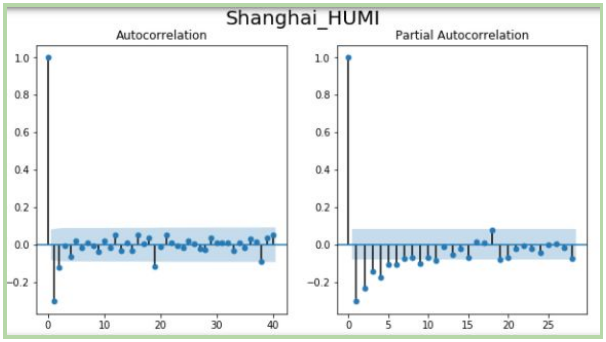



**Fig6.10** ACF/PACF for Shanghai_HUMI

| | value |
|---|---|
| **Test Statistic Value** | -8.542895 |
| **p-value** | 9.675236e-14 |
| **Lags Used** | 19 |
| **Number of Observations Used** | 563 |
| **Critical Value(1%)** | -3.441655 |
| **Critical Value(5%)** | -2.866687 |
| **Critical Value(10%)** | -2.569511 |

**Table 6.8** Test stationarity for Shanghai_HUMI : Since -8.542895 < -2.866687 the unit root hypothesis can be rejected at level 0.05.

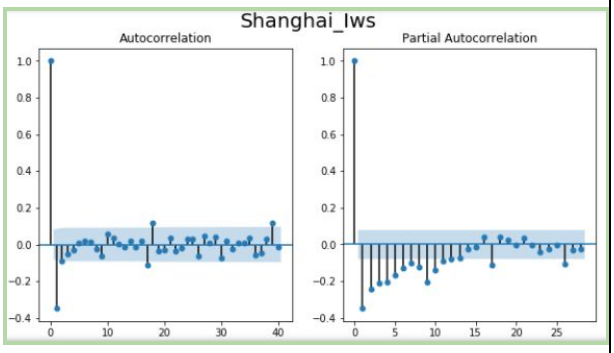



**Fig6.11** ACF/PACF for Shanghai_Iws

| | value |
|---|---|
| **Test Statistic Value** | -1.841663e+01 |
| **p-value** | 2.179747e-30 |
| **Lags Used** | 0 |
| **Number of Observations Used** | 583 |
| **Critical Value(1%)** | -3.441616 |
| **Critical Value(5%)** | -2.866510 |
| **Critical Value(10%)** | -2.569417 |

**Table 6.9** Test stationarity for Shanghai_Iws : Since -1.841663e+01 < -2.866510 the unit root hypothesis can be rejected at level 0.05.

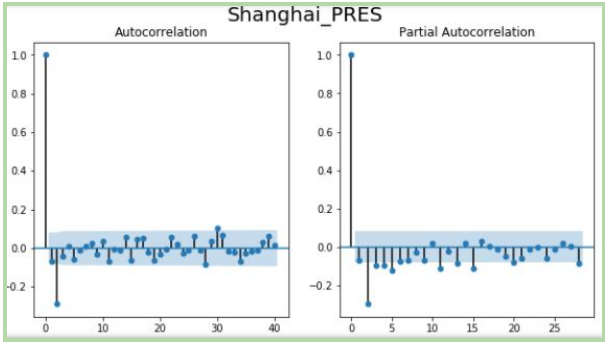



**Fig6.12** ACF/PACF for Shanghai_PRES

| | value |
|---|---|
| **Test Statistic Value** | -9.168309 |
| **p-value** | 2.426666e-15 |
| **Lags Used** | 14 |
| **Number of Observations Used** | 568 |
| **Critical Value(1%)** | -3.441915 |
| **Critical Value(5%)** | -2.866642 |
| **Critical Value(10%)** | -2.569487 |

**Table 6.10** Test stationarity for Shanghai_PRES : Since -9.168309 < -2.866642 the unit root hypothesis fails to be rejected at level 0.05.

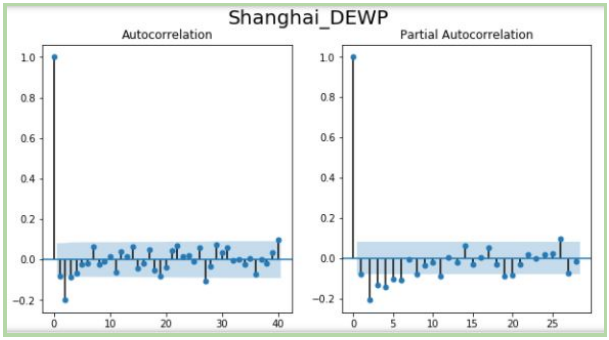



**Fig6.13** ACF/PACF for Shanghai_DEWP

| | value |
|---|---|
| **Test Statistic Value** | -1.062066e+01 |
| **p-value** | 5.526562e-19 |
| **Lags Used** | 10 |
| **Number of Observations Used** | 572 |
| **Critical Value(1%)** | -3.441834 |
| **Critical Value(5%)** | -2.866606 |
| **Critical Value(10%)** | -2.569468 |

**Table 6.11** Test stationarity for Shanghai_DEWP : Since -1.062066e+01 < -2.866606 the unit root hypothesis fails to be rejected at level 0.0.5.

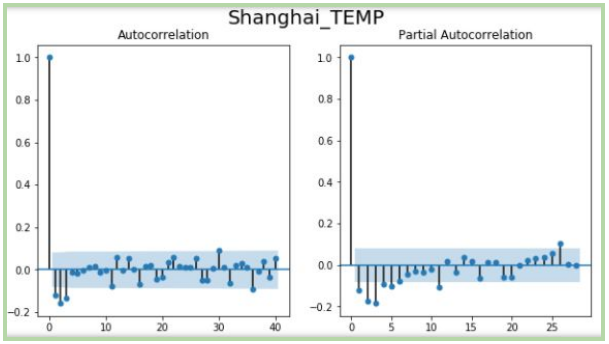



**Fig6.14** ACF/PACF for Shanghai_TEMP

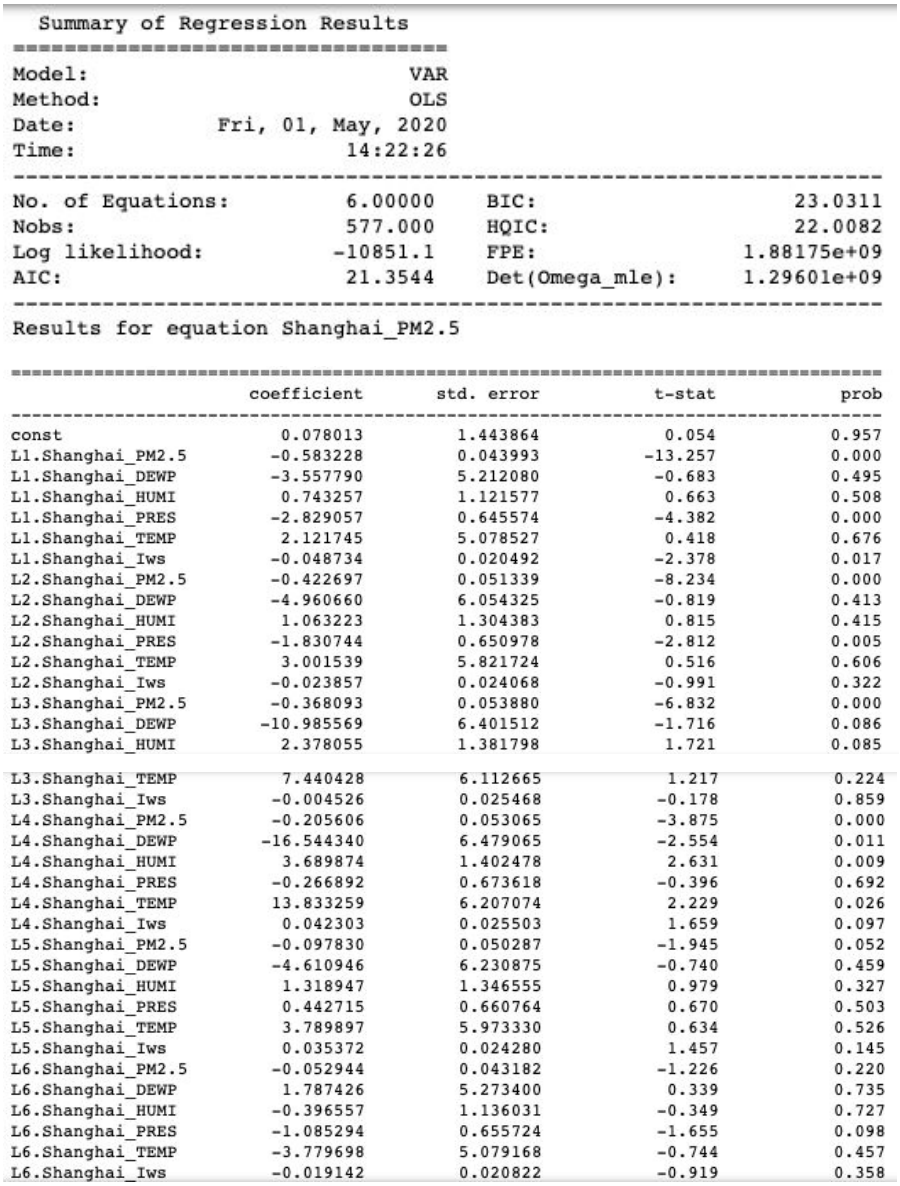
| | value |
|---|---|
| **Test Statistic Value** | -1.011137e+01 |
| **p-value** | 9.982817e-18 |
| **Lags Used** | 10 |
| **Number of Observations Used** | 572 |
| **Critical Value(1%)** | -3.441834 |
| **Critical Value(5%)** | -2.866606 |
| **Critical Value(10%)** | -2.569468 |

**Table 6.12** Test stationarity for Shanghai_TEMP : Since -1.011137e+01< -2.866606 the unit root hypothesis fails to be rejected at level 0.05.

### 6.2.3 Order selection

#### 6.2.3.1 Number of lag *h*

Unlike the univariate AR(*p*) model to fit in a maximum value of *p* to find the best *p* with the minimum AIC value, the VAR modeling requires the number of lag *h* for fitting a model. The modeling function of *.fit()* takes in a maximum lag (*maxlag*) to search the lag with minimum AIC, and our results show that the lag order of 6 should be selected.



Summary of Regression Results

| | | | |
|---|---|---|---|
| Model: | VAR | | |
| Method: | OLS | | |
| Date: | Fri, 01, May, 2020 | | |
| Time: | 14:22:26 | | |
| No. of Equations: | 6.00000 | BIC: | 23.0311 |
| Nobs: | 577.000 | HQIC: | 22.0082 |
| Log likelihood: | -10851.1 | FPE: | 1.88175e+09 |
| AIC: | 21.3544 | Det(Omega_mle): | 1.29601e+09 |

Results for equation Shanghai_PM2.5

| | coefficient | std. error | t-stat | prob |
|---|---|---|---|---|
| const | 0.078013 | 1.443864 | 0.054 | 0.957 |
| L1.Shanghai_PM2.5 | -0.583228 | 0.043993 | -13.257 | 0.000 |
| L1.Shanghai_DEWP | -3.557790 | 5.212080 | -0.683 | 0.495 |
| L1.Shanghai_HUMI | 0.743257 | 1.121577 | 0.663 | 0.508 |
| L1.Shanghai_PRES | -2.829057 | 0.645574 | -4.382 | 0.000 |
| L1.Shanghai_TEMP | 2.121745 | 5.078527 | 0.418 | 0.676 |
| L1.Shanghai_Iws | -0.048734 | 0.020492 | -2.378 | 0.017 |
| L2.Shanghai_PM2.5 | -0.422697 | 0.051339 | -8.234 | 0.000 |
| L2.Shanghai_DEWP | -4.960660 | 6.054325 | -0.819 | 0.413 |
| L2.Shanghai_HUMI | 1.063223 | 1.304383 | 0.815 | 0.415 |
| L2.Shanghai_PRES | -1.830744 | 0.650978 | -2.812 | 0.005 |
| L2.Shanghai_TEMP | 3.001539 | 5.821724 | 0.516 | 0.606 |
| L2.Shanghai_Iws | -0.023857 | 0.024068 | -0.991 | 0.322 |
| L3.Shanghai_PM2.5 | -0.368093 | 0.053880 | -6.832 | 0.000 |
| L3.Shanghai_DEWP | -10.985569 | 6.401512 | -1.716 | 0.086 |
| L3.Shanghai_HUMI | 2.378055 | 1.381798 | 1.721 | 0.085 |
| L3.Shanghai_TEMP | 7.440428 | 6.112665 | 1.217 | 0.224 |
| L3.Shanghai_Iws | -0.004526 | 0.025468 | -0.178 | 0.859 |
| L4.Shanghai_PM2.5 | -0.205606 | 0.053065 | -3.875 | 0.000 |
| L4.Shanghai_DEWP | -16.544340 | 6.479065 | -2.554 | 0.011 |
| L4.Shanghai_HUMI | 3.689874 | 1.402478 | 2.631 | 0.009 |
| L4.Shanghai_PRES | -0.266892 | 0.673618 | -0.396 | 0.692 |
| L4.Shanghai_TEMP | 13.833259 | 6.207074 | 2.229 | 0.026 |
| L4.Shanghai_Iws | 0.042303 | 0.025503 | 1.659 | 0.097 |
| L5.Shanghai_PM2.5 | -0.097830 | 0.050287 | -1.945 | 0.052 |
| L5.Shanghai_DEWP | -4.610946 | 6.230875 | -0.740 | 0.459 |
| L5.Shanghai_HUMI | 1.318947 | 1.346555 | 0.979 | 0.327 |
| L5.Shanghai_PRES | 0.442715 | 0.660764 | 0.670 | 0.503 |
| L5.Shanghai_TEMP | 3.789897 | 5.973330 | 0.634 | 0.526 |
| L5.Shanghai_Iws | 0.035372 | 0.024280 | 1.457 | 0.145 |
| L6.Shanghai_PM2.5 | -0.052944 | 0.043182 | -1.226 | 0.220 |
| L6.Shanghai_DEWP | 1.787426 | 5.273400 | 0.339 | 0.735 |
| L6.Shanghai_HUMI | -0.396557 | 1.136031 | -0.349 | 0.727 |
| L6.Shanghai_PRES | -1.085294 | 0.655724 | -1.655 | 0.098 |
| L6.Shanghai_TEMP | -3.779698 | 5.079168 | -0.744 | 0.457 |
| L6.Shanghai_Iws | -0.019142 | 0.020822 | -0.919 | 0.358 |

**Fig6.15** Regression result of VAR model shows the lag order of 6 should be selected

### 6.2.4 Model Fitting and Residuals Testing

#### 6.2.4.1 Quantile-Quantile Plot

In the Q-Q plot, we can see that points fall along a line in the middle of the graph, but curve off in the extremities.This behavior usually means that the data have more extreme values than would be expected if it truly came from the normal distribution, and the residual might be slightly heavy-tailed.

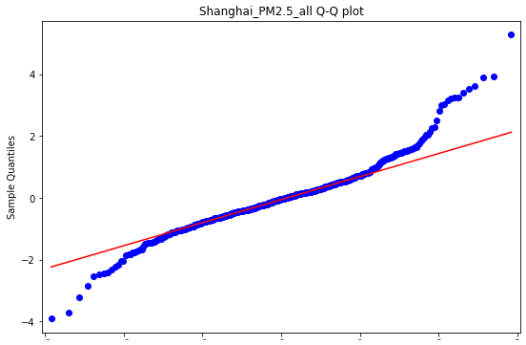



**Fig6.16** Q-Q plot

### 6.2.4.2 Durbin-Watson Statistics

The Durbin-Watson Statistics is a test statistic to check for Serial Correlation of Residuals (Errors). A statistic close to 2 means no significant serial correlation. The residuals of the model show no significant serial correlation among the variables.

| | Test value |
|---|---|
| **Shanghai_PM2.5** | 1.99 |
| **Shanghai_DEWP** | 1.99 |
| **Shanghai_HUMI** | 2.0 |
| **Shanghai_PRES** | 2.03 |
| **Shanghai_HUMI** | 2.0 |
| **Shanghai_TEMP** | 1.99 |

**Table6.13** Durbin-Watson Statistics of VAR model with all variables using $h = 6$

### 6.2.4.3 Impulse Response Analysis

Impulse response (IR) analysis is a VAR-specific test analysis technique to show effects of a shock at one innovation to a variable on the response of all variables. In VAR($p$) provess, assumed a MA($\infty$) representation, IR or forecast error impulse response (FEIR) can be computed with the following equation:

$$Y_t = \mu + \sum_{i=0}^{\infty} \Phi_i u_{t-i}$$

where $u_{t-i} \sim N(0, \boldsymbol{\Sigma}_u)$, $\boldsymbol{\Phi}$ as coefficient matrix of VAR($p$)

The plot is referring to the accumulated value of FEIR given a particular period. The presence of spikes explain the change of variable X “shocks” variable Y at a given period. The curve section with positive value (above the zero line) means the shock of variable X has positive impact over variable Y, and inversely for that with negative value then brings negative impact. A curve going up and down along the zero line means shocks to variable X will have asymmetric impacts on variable Y.

In our model, we can see shock to PM2.5 brings a positive impact on PM2.5, which is the reality; shocks to HUMI and TEMP are also positive to PM2.5. Impulse as PRES and DEWP might negatively affect PM2.5. And Iws might have an asymmetric impact. However, supplementary hypothesis tests such that the null hypothesis is the accumulated impulse response after 10 periods is zero could be done to support the statistical significance. But we are limited by the package to extract the impulse response data.

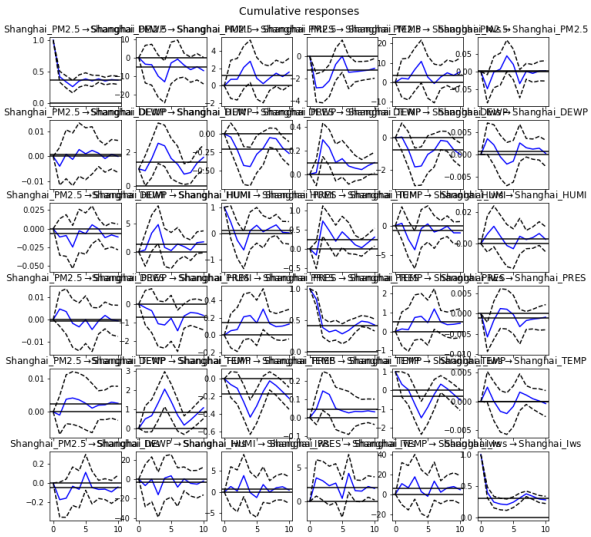



**Fig 6.17** Cumulative impulse response analysis for 10 periods after a shock among the variables

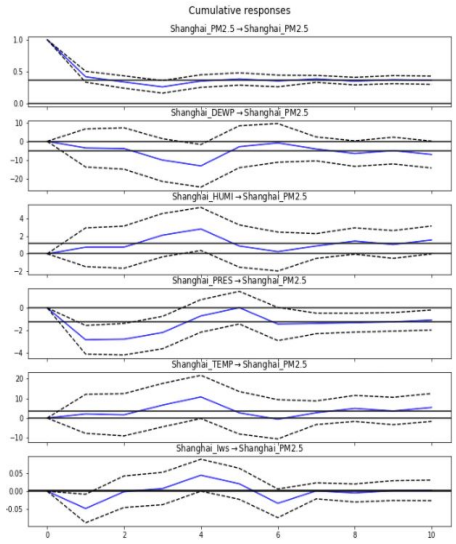



**Fig 6.18** Cumulative impulse response analysis for 10 periods after a shock of all variables towards PM2.5

### 6.2.4.4 Forecast Error Variance Decomposition

The forecast error variance decomposition (FEVD) is an extended study of the impulse response analysis. Using the orthogonalized impulse responses, which the variable order in the model matters in impulse response analysis, forecast errors of component *j* on *k* in a given step ahead forecast are decomposed, so as to see the relative importance of each shock of a variable in affecting the forecast error variance of all response variables. The FEVD can be written as follows.

$$\omega_{jk,i} = \sum_{i=0}^{h-1} (e_j' \Theta_i e_k)^2 / \mathrm{MSE}_j(h)$$

$$\mathrm{MSE}_j(h) = \sum_{i=0}^{h-1} e_j' \Phi_i \Sigma_u \Phi_i' e_j$$

where $\boldsymbol{\Theta}_i$ as orthogonalized impulse response,
$\boldsymbol{\Phi}$ as coefficient matrix of VAR(*p*),
$\boldsymbol{\Sigma}_u$ as error covariance matrix

In our model, by looking at the FEVD values in a 5-step ahead forecast in Table 6.14 and the first row of all five "black" boxes in the FEVD plot, the predictions of PM2.5 are mostly explained by the variable itself.

FEVD for Shanghai_PM2.5

| | Shanghai_PM2.5 | Shanghai_DEWP | Shanghai_HUMI | Shanghai_PRES | Shanghai_TEMP | Shanghai_Iws |
|---|---|---|---|---|---|---|
| 0 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 1 | 0.959304 | 0.007293 | 0.000778 | 0.024691 | 0.000542 | 0.007392 |
| 2 | 0.950984 | 0.009144 | 0.000814 | 0.024278 | 0.000653 | 0.014127 |
| 3 | 0.941562 | 0.013839 | 0.004014 | 0.024698 | 0.001747 | 0.014140 |
| 4 | 0.931015 | 0.013929 | 0.004789 | 0.029838 | 0.002268 | 0.018161 |

**Table 6.14** The FEVD table explain the shock to other variables on Shanghai_PM2.5

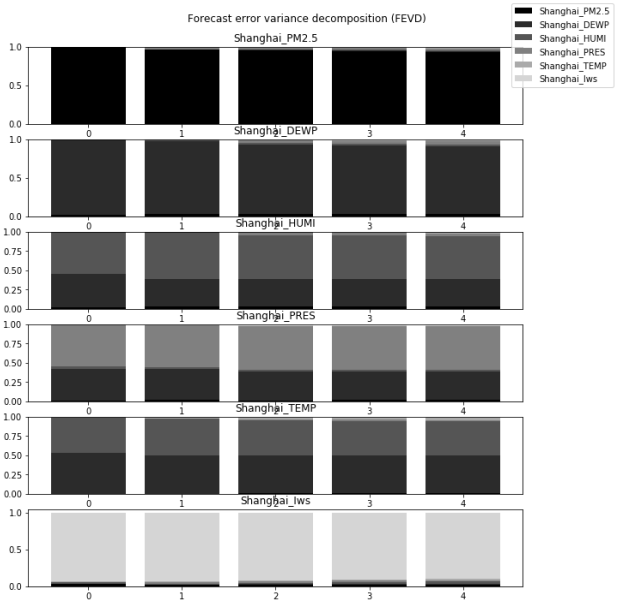



**Fig6.19** FEVD plots of all variables

## 6.2.5 Prediction & Validation

After we made the prediction from the fitted VAR model, we plotted the figures to compare the predictions and the true value and calculated Root Mean Square Error (RMSE) values to validate the fitness of the VAR model on all variables.

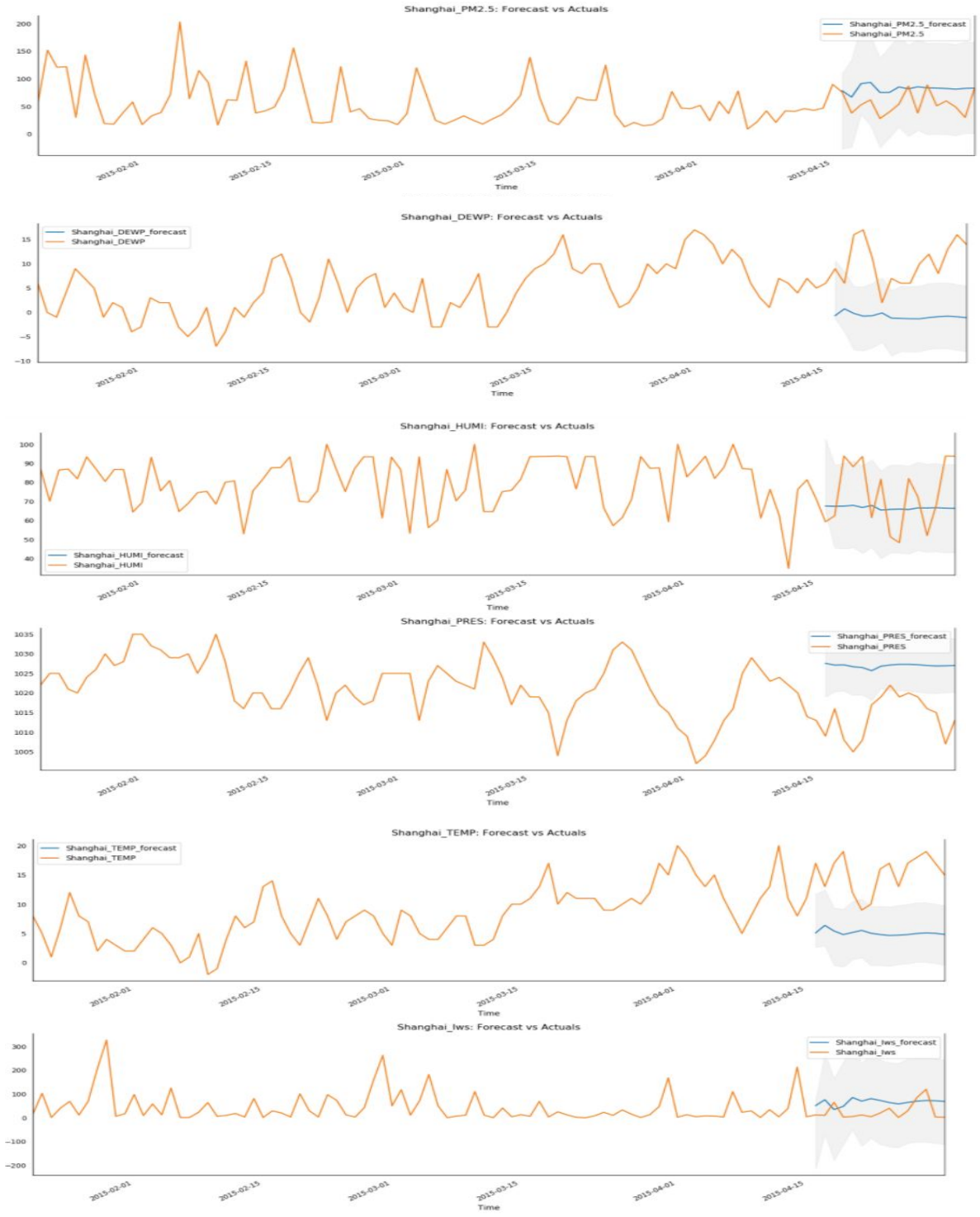



**Fig 6.20** Predictions of all variables with VAR(6) model

The following table shows the best RMSE value of PM2.5 forecast among different lags, and we deem this model is our best model with these variables.

| | RMSE value |
|---|---|
| **Shanghai_overall_forecast:** | 28.0540 |
| **Shanghai_PM2.5_forecast:** | 32.0945 |
| **Shanghai_DEWP_forecast:** | 11.8203 |
| **Shanghai_HUMI_forecast:** | 17.7858 |
| **Shanghai_PRES_forecast:** | 13.8042 |
| **Shanghai_HUMI_forecast:** | 10.6622 |
| **Shanghai_TEMP_forecast:** | 54.1461 |

**Table 6.15** Root Mean Square Error (RMSE) of the predictions of the overall model and each variable

### 6.2.6 Variable selection based on Granger causality for prediction

In VAR modeling, we could study the Granger causality among the variables. Granger causality means the tendency of one variable is a precedence of another variable, showing a causation relationship between two variables. By performing a hypothesis test that the null hypothesis is variable X does not explain variable Y, if the p-value returned from the Granger causality test is less than the significance level, we should reject the null hypothesis and say X statistically significantly causes Y.

From the following Granger causality matrix, we can see that HUMI and TEMP statistically does not explain the PM2.5 concentration under the significance level of 0.05, and other variables do.

| | Shanghai_PM2.5_x | Shanghai_DEWP_x | Shanghai_HUMI_x | Shanghai_PRES_x | Shanghai_TEMP_x | Shanghai_Iws_x |
|---|---|---|---|---|---|---|
| Shanghai_PM2.5_y | 1.0000 | 0.0365 | 0.1085 | 0.0000 | 0.2121 | 0.0082 |
| Shanghai_DEWP_y | 0.0025 | 1.0000 | 0.0047 | 0.0000 | 0.0024 | 0.0049 |
| Shanghai_HUMI_y | 0.0435 | 0.0000 | 1.0000 | 0.0000 | 0.0000 | 0.0957 |
| Shanghai_PRES_y | 0.0027 | 0.0080 | 0.1056 | 1.0000 | 0.0055 | 0.0000 |
| Shanghai_TEMP_y | 0.0807 | 0.0023 | 0.0026 | 0.0062 | 1.0000 | 0.0747 |
| Shanghai_Iws_y | 0.1132 | 0.2667 | 0.1962 | 0.0282 | 0.0084 | 1.0000 |

Therefore, we decided to try eliminating the non-Granger-causal variables and doing bivariate analysis between the PM2.5 concentration and each of the Granger-causal variables in our modeling. The results supported by the graphs below show that elimination of non-Granger-causal variables does not help the model prediction, but using the Granger-causal variables might help the prediction.

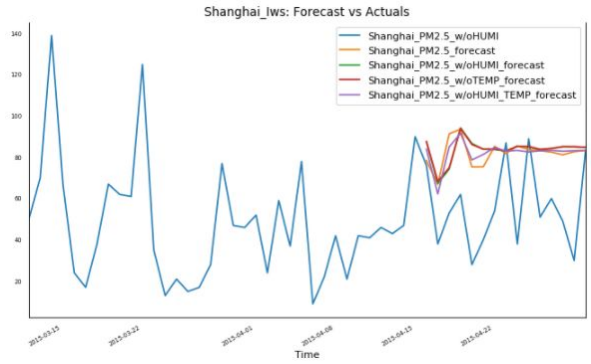



**Fig 6.21** Compare the forecasts of models with different non-Granger-causal variable eliminations

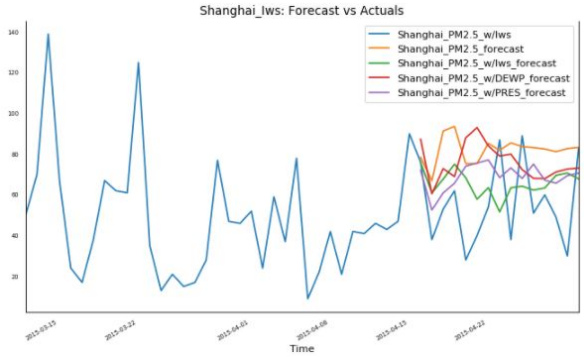



**Fig 6.22** Compare the forecasts of bivariate models each take one Granger-causal variable

**(b) The potential relationship of the PM2.5 pollution among the five cities**

Other than weather factors, we would like to study if there is any inter-city relationship like geographical relation and similar industrial activity on the PM2.5 pollution. Using the same data range from 2013-05-01 to 2015-04-30 for all five cities - Shanghai, Beijing, Chengdu, Shenyang, and Guangzhou, the principal variable is still Shanghai. Again, the modeling technique is VAR.

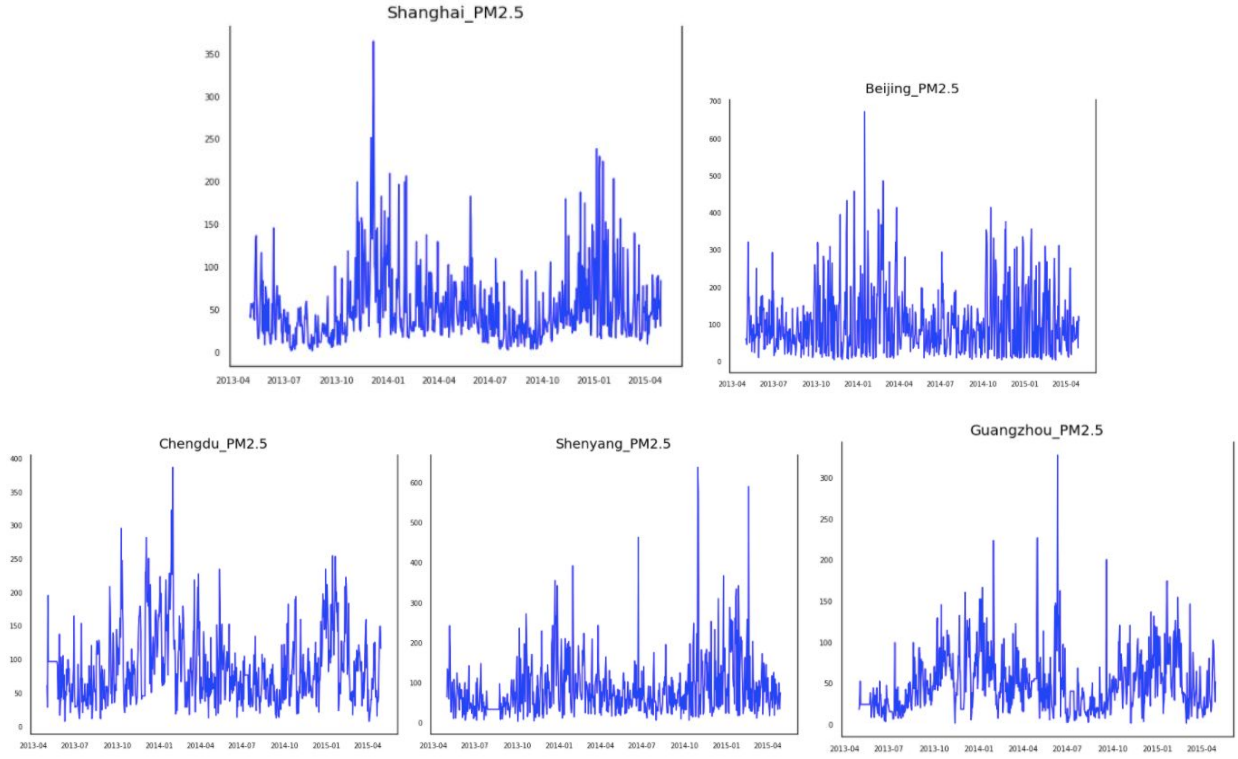



**Fig 6.23** Raw data plots for five cities with the time range from 2013-05-01 to 2015-04-30.

### 6.2.2 Stationarity and differencing

We used the augmented Dickey–Fuller test to check for stationarity. The results show the ordinary data of PM2.5 concentration in all five cities are “stationary”.

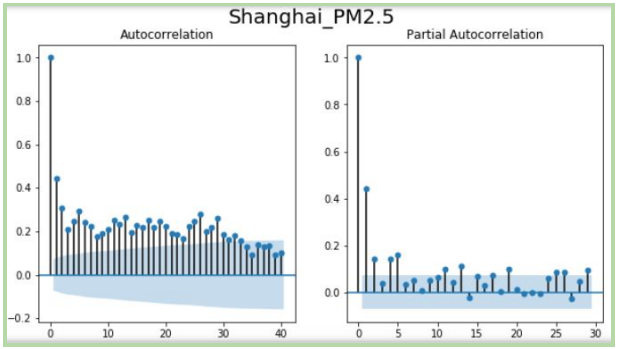



**Fig 6.24** ACF/PACF for Shanghai_PM2.5

| | value |
|---|---|
| **Test Statistic Value** | -3.019492 |
| **p-value** | 0.033114 |
| **Lags Used** | 14 |
| **Number of Observations Used** | 569 |
| **Critical Value(1%)** | -3.441895 |
| **Critical Value(5%)** | -2.866633 |
| **Critical Value(10%)** | -2.569482 |

**Table 6.16** Test stationarity for Shanghai_PM2.5 : Since -3.019492 < -2.866633 the unit root hypothesis can be rejected at level 0.05.

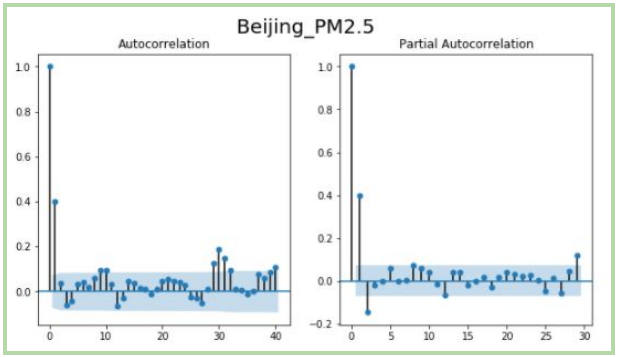
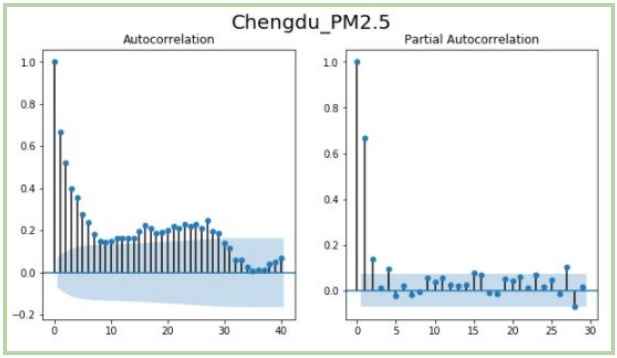



**Fig6.25** ACF/PACF for Beijing_PM2.5

| | value |
|---|---|
| **Test Statistic Value** | -1.449274e+01 |
| **p-value** | 6.110486e-27 |
| **Lags Used** | 1 |
| **Number of Observations Used** | 582 |
| **Critical Value(1%)** | -3.441636 |
| **Critical Value(5%)** | -2.866519 |
| **Critical Value(10%)** | -2.569422 |

**Table 6.17** Test stationarity for Beijing_PM2.5 : Since -1.449274e+01 < -2.866519 the unit root hypothesis can be rejected at level 0.05.




| | value |
|---|---|
| **Test Statistic Value** | -7.119539 |
| **p-value** | 3.753707e-10 |
| **Lags Used** | 3 |
| **Number of Observations Used** | 580 |
| **Critical Value(1%)** | -3.441675 |
| **Critical Value(5%)** | -2.866536 |
| **Critical Value(10%)** | -2.569431 |

**Fig6.26** ACF/PACF for Chengdu_PM2.5

**Table 6.18** Test stationarity for Chengdu_PM2.5 : Since -7.119539 < -2.866536 the unit root hypothesis can be rejected at level 0.05.

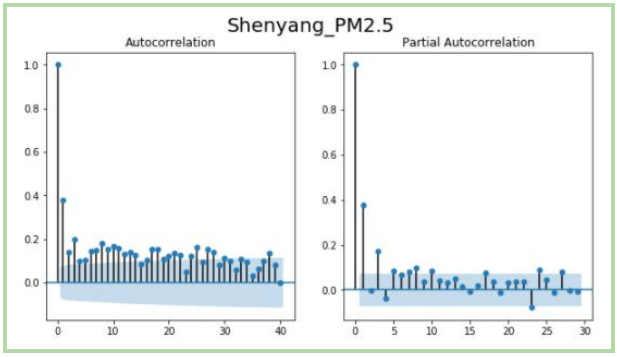



**Fig6.27** ACF/PACF for Shenyang_PM2.5

| | value |
|---|---|
| **Test Statistic Value** | -4.699282 |
| **p-value** | 0.000084 |
| **Lags Used** | 9 |
| **Number of Observations Used** | 574 |
| **Critical Value(1%)** | -3.441794 |
| **Critical Value(5%)** | -2.866588 |
| **Critical Value(10%)** | -2.569459 |

**Table 6.19** Test stationarity for Shenayng_PM2.5 : Since -4.699282 < -2.866588 the unit root hypothesis fails to be rejected at level 0.05.

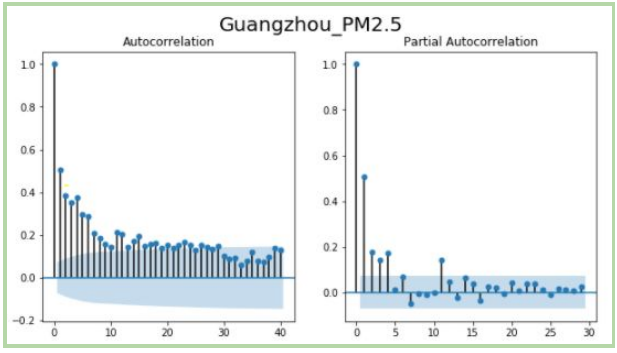



**Fig6.28** ACF/PACF for Guangzhou_PM2.5

| | value |
|---|---|
| **Test Statistic Value** | -4.435138 |
| **p-value** | 0.000257 |
| **Lags Used** | 10 |
| **Number of Observations Used** | 573 |
| **Critical Value(1%)** | -3.441814 |
| **Critical Value(5%)** | -2.866597 |
| **Critical Value(10%)** | -2.569463 |

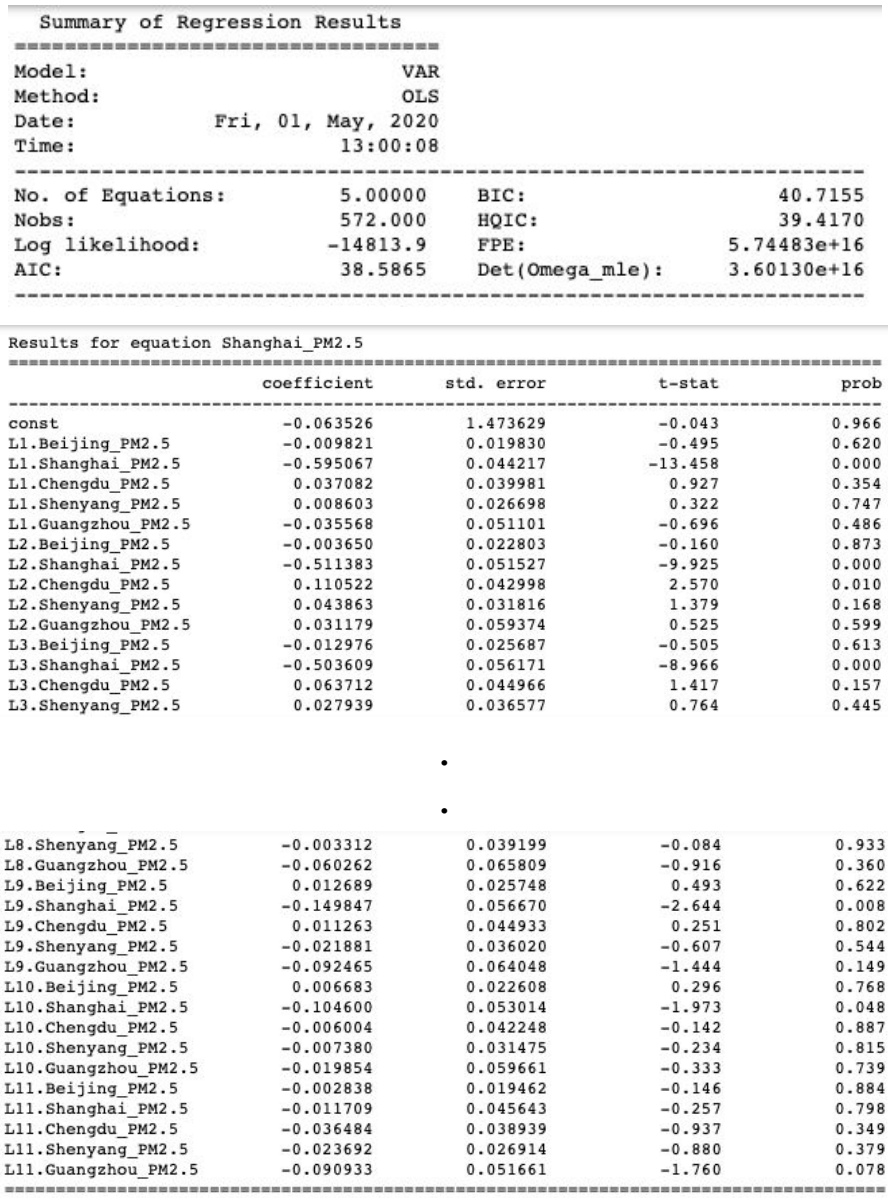
**Table 6.20** Test stationarity for Guangzhou_PM2.5 :
Since -4.435138 < -2.866597 the unit root hypothesis fails to be rejected at level 0.0.5.

We tried to use undifferenced data for modelling, but the prediction is unsatisfactory that a lag order of 1 is suggested and a straight line is returned. Therefore, we try to do one-step differencing, but the modeling result is more coherent with the actual data.

### 6.2.3 Order selection

#### 6.2.3.1 Number of lag *h*

With respect to the principle of minimum AIC in order selection, our results show that the lag order of 11 should be selected.



Summary of Regression Results

| | | | |
|---|---|---|---|
| Model: | VAR | | |
| Method: | OLS | | |
| Date: | Fri, 01, May, 2020 | | |
| Time: | 13:00:08 | | |
| No. of Equations: | 5.00000 | BIC: | 40.7155 |
| Nobs: | 572.000 | HQIC: | 39.4170 |
| Log likelihood: | -14813.9 | FPE: | 5.74483e+16 |
| AIC: | 38.5865 | Det(Omega_mle): | 3.60130e+16 |

Results for equation Shanghai_PM2.5

| | coefficient | std. error | t-stat | prob |
|---|---|---|---|---|
| const | -0.063526 | 1.473629 | -0.043 | 0.966 |
| L1.Beijing_PM2.5 | -0.009821 | 0.019830 | -0.495 | 0.620 |
| L1.Shanghai_PM2.5 | -0.595067 | 0.044217 | -13.458 | 0.000 |
| L1.Chengdu_PM2.5 | 0.037082 | 0.039981 | 0.927 | 0.354 |
| L1.Shenyang_PM2.5 | 0.008603 | 0.026698 | 0.322 | 0.747 |
| L1.Guangzhou_PM2.5 | -0.035568 | 0.051101 | -0.696 | 0.486 |
| L2.Beijing_PM2.5 | -0.003650 | 0.022803 | -0.160 | 0.873 |
| L2.Shanghai_PM2.5 | -0.511383 | 0.051527 | -9.925 | 0.000 |
| L2.Chengdu_PM2.5 | 0.110522 | 0.042998 | 2.570 | 0.010 |
| L2.Shenyang_PM2.5 | 0.043863 | 0.031816 | 1.379 | 0.168 |
| L2.Guangzhou_PM2.5 | 0.031179 | 0.059374 | 0.525 | 0.599 |
| L3.Beijing_PM2.5 | -0.012976 | 0.025687 | -0.505 | 0.613 |
| L3.Shanghai_PM2.5 | -0.503609 | 0.056171 | -8.966 | 0.000 |
| L3.Chengdu_PM2.5 | 0.063712 | 0.044966 | 1.417 | 0.157 |
| L3.Shenyang_PM2.5 | 0.027939 | 0.036577 | 0.764 | 0.445 |
| . | | | | |
| . | | | | |
| L8.Shenyang_PM2.5 | -0.003312 | 0.039199 | -0.084 | 0.933 |
| L8.Guangzhou_PM2.5 | -0.060262 | 0.065809 | -0.916 | 0.360 |
| L9.Beijing_PM2.5 | 0.012689 | 0.025748 | 0.493 | 0.622 |
| L9.Shanghai_PM2.5 | -0.149847 | 0.056670 | -2.644 | 0.008 |
| L9.Chengdu_PM2.5 | 0.011263 | 0.044933 | 0.251 | 0.802 |
| L9.Shenyang_PM2.5 | -0.021881 | 0.036020 | -0.607 | 0.544 |
| L9.Guangzhou_PM2.5 | -0.092465 | 0.064048 | -1.444 | 0.149 |
| L10.Beijing_PM2.5 | 0.006683 | 0.022608 | 0.296 | 0.768 |
| L10.Shanghai_PM2.5 | -0.104600 | 0.053014 | -1.973 | 0.048 |
| L10.Chengdu_PM2.5 | -0.006004 | 0.042248 | -0.142 | 0.887 |
| L10.Shenyang_PM2.5 | -0.007380 | 0.031475 | -0.234 | 0.815 |
| L10.Guangzhou_PM2.5 | -0.019854 | 0.059661 | -0.333 | 0.739 |
| L11.Beijing_PM2.5 | -0.002838 | 0.019462 | -0.146 | 0.884 |
| L11.Shanghai_PM2.5 | -0.011709 | 0.045643 | -0.257 | 0.798 |
| L11.Chengdu_PM2.5 | -0.036484 | 0.038939 | -0.937 | 0.349 |
| L11.Shenyang_PM2.5 | -0.023692 | 0.026914 | -0.880 | 0.379 |
| L11.Guangzhou_PM2.5 | -0.090933 | 0.051661 | -1.760 | 0.078 |

**Fig6.29** Regression result of VAR model shows the lag order of 11 should be selected

### 6.2.4 Model Fitting and Residuals Testing

#### 6.2.4.1 Quantile-Quantile Plot

In the Q-Q plot, despite some extreme points, we can see the residual normality is fairly maintained but might be slightly heavy-tailed.

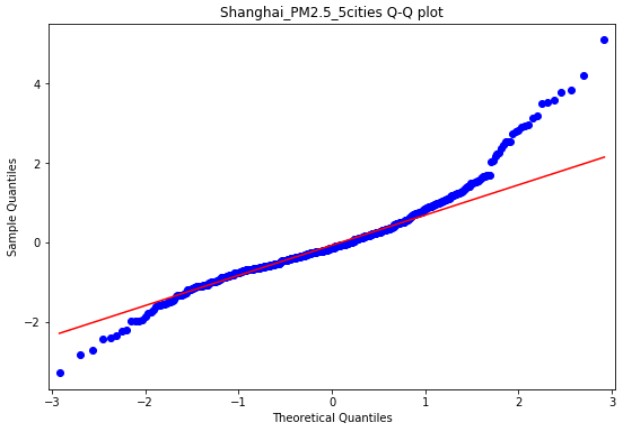



**Fig6.30** Q-Q plot

### 6.2.4.2 Durbin-Watson Statistics

The Durbin-Watson statistics show close to 2 for all variables, meaning no significant serial correlation of the residuals in the model.

| | Test value |
|---|---|
| **Shanghai_PM2.5** | 1.99 |
| **Beijing_PM2.5** | 2.0 |
| **Chengdu_PM2.5** | 2.0 |
| **Shenyang_PM2.5** | 1.99 |
| **Guangzhou_PM2.5** | 1.99 |

**Table6.21** Durbin-Watson Statistics of VAR model with all variables using $h = 11$

### 6.2.4.3 Impulse Response Analysis

In our model, we can see that in general the IR curves of PM2.5 concentration in other cities, especially Beijing and Shenyang, lay around the zero line, either telling that there is a mere asymmetric impact or it is not statistically significant to have that variable in the model. The shock to Chengdu_PM2.5 merely brings a positive impact on Shanghai_PM2.5. And the shocks to Guangzhou_PM2.5 might negatively affect Shanghai_PM2.5 at later periods. But again we are limited by the package to extract the impulse response data for further hypothesis testing.

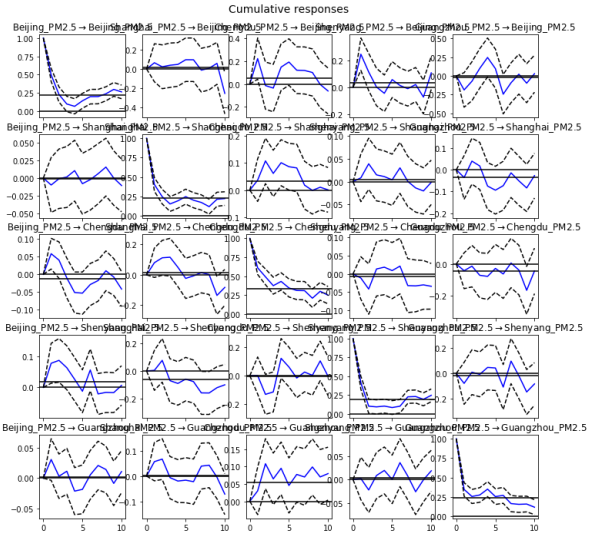



**Fig 6.31** Cumulative impulse response analysis for 10 periods after a shock among the variables

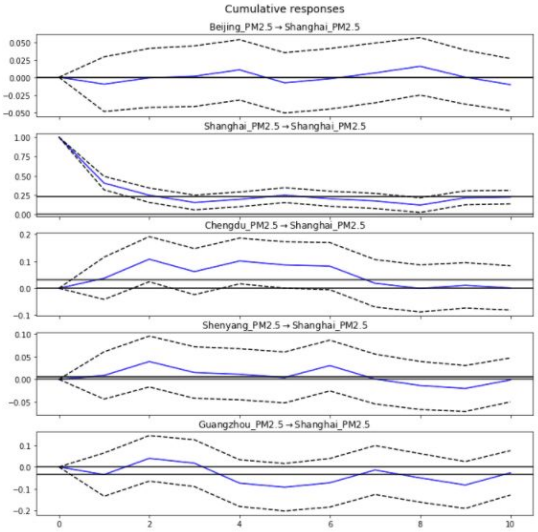



**Fig 6.32** Cumulative impulse response analysis for 10 periods after a shock of all variables towards PM2.5

### 6.2.4.4 Forecast Error Variance Decomposition

In our model, by looking at the FEVD values in a 5-step ahead forecast in Table 6.14 and the first row of all five "black" boxes in the FEVD plot, the predictions of PM2.5 are mostly explained by the variable itself.

FEVD for Shanghai_PM2.5

| | Beijing_PM2.5 | Shanghai_PM2.5 | Chengdu_PM2.5 | Shenyang_PM2.5 | Guangzhou_PM2.5 |
|---|---|---|---|---|---|
| 0 | 0.006663 | 0.993337 | 0.000000 | 0.000000 | 0.000000 |
| 1 | 0.005726 | 0.992540 | 0.000915 | 0.000130 | 0.000689 |
| 2 | 0.007915 | 0.980386 | 0.006061 | 0.001975 | 0.003663 |
| 3 | 0.007840 | 0.977351 | 0.007858 | 0.003066 | 0.003885 |
| 4 | 0.008012 | 0.972097 | 0.008508 | 0.003086 | 0.008296 |

**Table 6.22** The FEVD table explain the shock to other variables on Shanghai_PM2.5

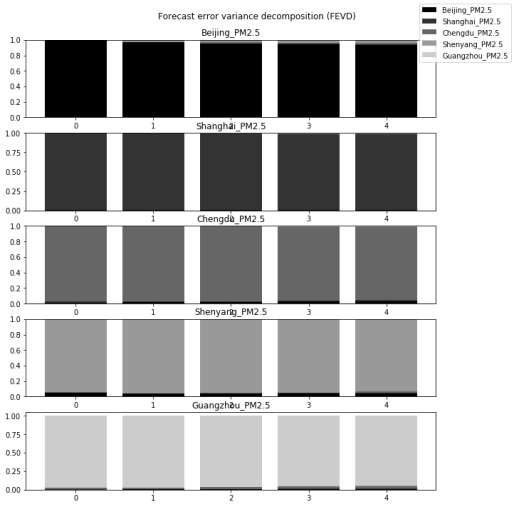



**Fig6.33** The FEVD plots of all variables

### 6.2.5 Prediction & Validation

After we made the prediction from the fitted VAR model, we plotted the figures to compare the predictions and the true value and calculated Root Mean Square Error (RMSE) values to validate the fitness of the VAR model on all variables.

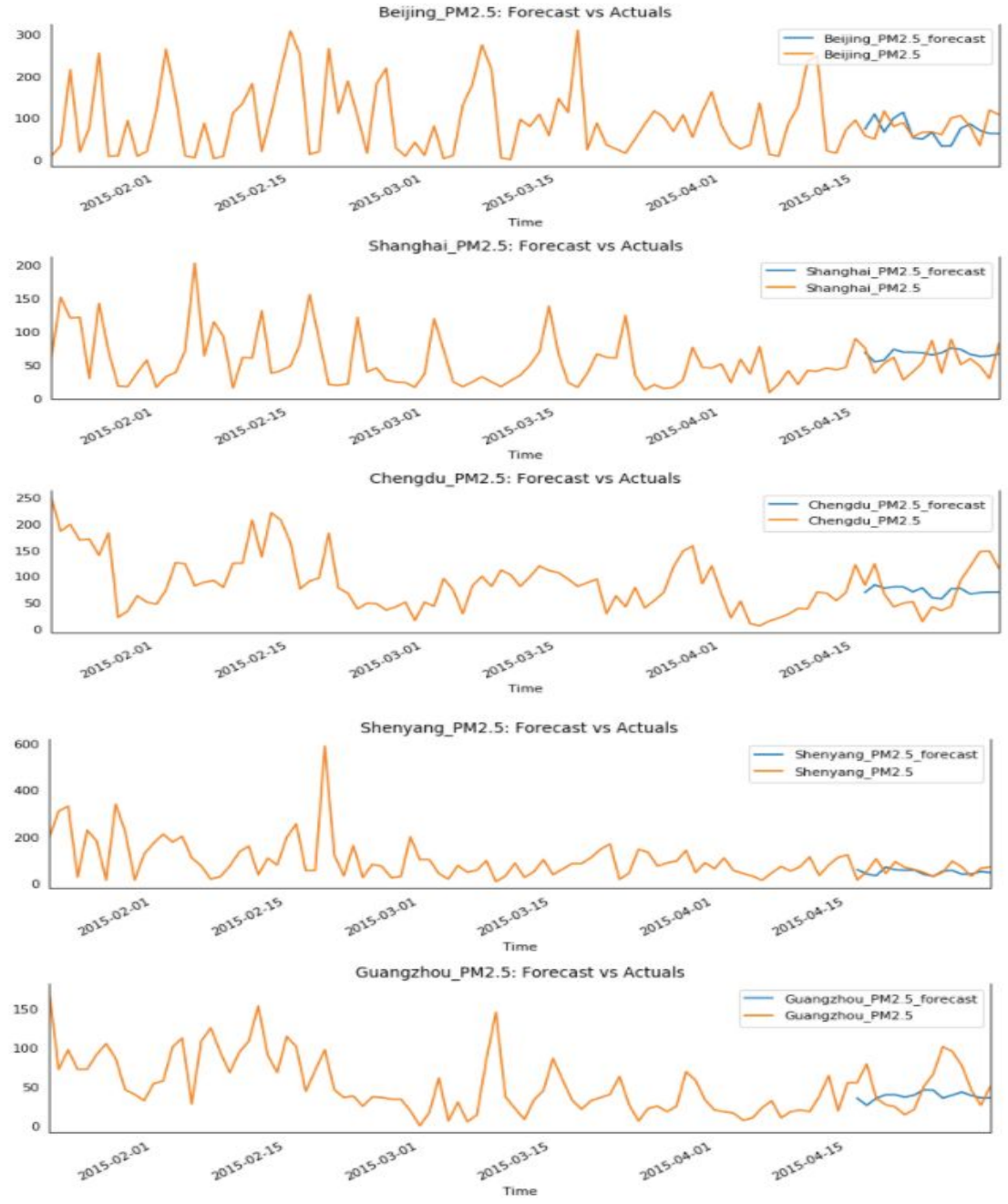


**Fig 6.34** Predictions of all variables with VAR(11) model

The following table shows the best RMSE value of PM2.5 forecast among different lags, and we deem this model is our best model with these variables.

| | RMSE value |
|---|---|
| **5cities_overall_forecas:** | 33.0745 |
| **Shanghai_PM2.5_forecast** | 21.8028 |
| **Beijinng_PM2.5_forecas:** | 36.7481 |
| **Chengdu_PM2.5_forecast** | 43.1145 |
| **Shenyang_PM2.5_forecas:** | 29.3794 |
| **Guangzhou_PM2.5_forecas:** | 30.3616 |

**Table 6.22** Root Mean Square Error (RMSE) of the predictions of the overall model and each variable

Earlier we mentioned the importance of Granger causality. We also analyzed this set of variables with Granger causality and all show a statistically significant causal relationship over Shanghai_PM2.5, our variable of interest. Therefore, we do not do further variable selection, and suggest the current model is the best model of all.

| | Beijing_PM2.5_x | Shanghai_PM2.5_x | Chengdu_PM2.5_x | Shenyang_PM2.5_x | Guangzhou_PM2.5_x |
|---|---|---|---|---|---|
| Beijing_PM2.5_y | 1.0000 | 0.1580 | 0.001 | 0.0003 | 0.0036 |
| Shanghai_PM2.5_y | 0.0023 | 1.0000 | 0.000 | 0.0000 | 0.0050 |
| Chengdu_PM2.5_y | 0.1144 | 0.0001 | 1.000 | 0.2600 | 0.0879 |
| Shenyang_PM2.5_y | 0.5559 | 0.0030 | 0.000 | 1.0000 | 0.0000 |
| Guangzhou_PM2.5_y | 0.0057 | 0.0000 | 0.000 | 0.0195 | 1.0000 |

**Fig 6.35** Granger causality matrix of all variables in the 5 cities model

## 6.3 Limitation in multivariate analysis - seasonality

In univariate analysis, we concluded that seasonal ARIMA gives the best prediction, so we think in multivariate analysis, the seasonality component might also be important in affecting the model performance. However, we failed to find a method with VAR in Python to eliminate the seasonality component. How to computationally and theoretically handle seasonal multivariate time series might be one of an important exploration in the due future.

## 7. Conclusion and Future Plans

After a series of univariate and multivariate analysis, with our best effort, we conclude that the seasonal ARIMA model performs the best in univariate analysis; and using Granger-causal variables in modeling might boost the multivariate forecasting performance.

With rounds of trials and mistakes, we understand that the selection of time frame and analysis unit is of paramount importance at the very first step of the analysis and modeling. This problem applies in both univariate and multivariate time series. Insights from the process of univariate analysis include:

- Seasonal ARIMA heavily requires a right seasonal period value for accurate prediction
- LSTM needs more guidance on the parameter tuning (e.g. n_input, batch_size)

Insights from the process of multivariate analysis include:

- The modeling is limited by the only parameter - lag order, and seasonality
- Importance of using Granger-causal variables in modeling
- Differencing might cause overfitting, but it might be unavoidable

In the future, we not only wish we could solve the problems now left unsolved, but also explore other models for better forecasting, such as using classical decomposition rather than differencing, and VARMAX or VECM in multivariate analysis. Also, we would like to enrich our scope of analysis by using the most updated data and also extra data sources which are believed to be highly correlated to the PM2.5 concentration level like industrial activity data.