Predicting Air Quality In India Based On Time Series Model

December 4, 2022

1 Introduction

Air pollution is one of the major issues threatening human's health. In order to implement corresponding police and measures in different period to ease such negative impaction, the quantification and prediction of air pollution is of great importance. Because AQI index has already been invented to quantify air pollution, the purpose of this article is to predict AQI base on Time Series Models. We firstly built a $SARIMA$ model to train and predict Indian AQI, then constructed an Intervention Model to modify this $SARIMA$ model to adjust to the significant impact of $COVID-19$. According to our experiment, Intervention Model provides a great improvement to prediction accuracy, so the combination of Intervention Model and Origin $SARIMA$ Model is desirable.The MAPE of Intervention Model equals to 26%.

2 Background and Data Description

2.1 Background

The air quality of India represents significant seasonal pattern before 2020. After the outbreak of $COVID-19$, Indian Government implemented a nationwide lockdown restriction from 24 March 2020 to 30 May 2020, which inhibited industrial and traffic movement, and thence decrease the level of air pollution in that period.

2.2 AQI

AQI index indicates the level of air pollution in certain area and time. It is computed hourly based on 7 types of air condition data. The detailed computation rules is presented as follows:

(1) The AQI calculation uses 7 measures: $PM2.5$, $PM10$, SO_2, NO_x, NH_3, CO and O_3.

(2) For $PM2.5$, $PM10$, SO_2, NO_x and NH_3 the average value in last 24-hrs is used with the condition of having at least 16 values.

(3) For CO and O_3 the maximum value in last 8-hrs is used. Each measure is converted into a Sub-Index based on pre-defined groups.

(4) Sometimes measures are not available due to lack of measuring or lack of required data points.

(5) Final AQI is the maximum Sub-Index with the condition that at least one of $PM2.5$ and $PM10$ should be available and at least three out of the seven should be available.

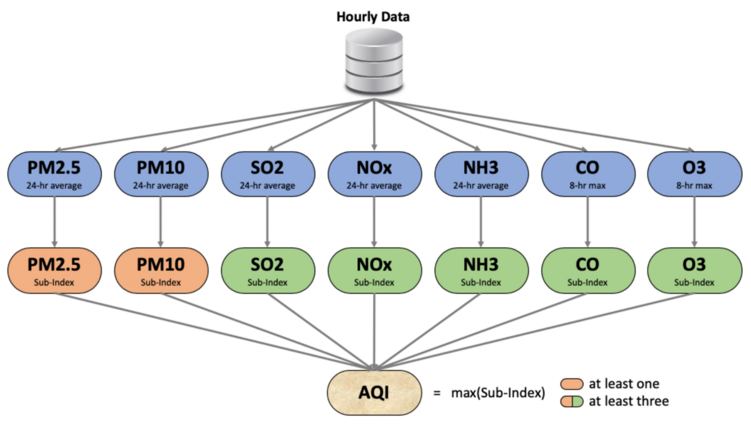


Figure 1: AQI Calculation Stream

2.3 Data Content

The dataset contains air quality data and AQI (Air Quality Index) at daily level of various stations across multiple cities in India, from April 5, 2015 to July 5, 2020. It has been made publicly available by the Central Pollution Control Board, which is the official portal of Government of India. Nevertheless, the biggest drawback of this dataset is value missing. As we further examine this dataset, it turns out that there are five cities containing acceptable amount of missing data (Fig. 1): Bengaluru, Chennai, Delhi, Hyderabad and Lucknow. Therefore, we decided to transform daily value to weekly value using the mean of AQI within a week, fill the missing data base on Interpolate technic, then calculated the mean of AQI of these five cities to represent the overall AQI of India in each week. A time series graph of Indian Weekly AQI is shown in (Fig. 1), indicating a significant seasonal pattern.

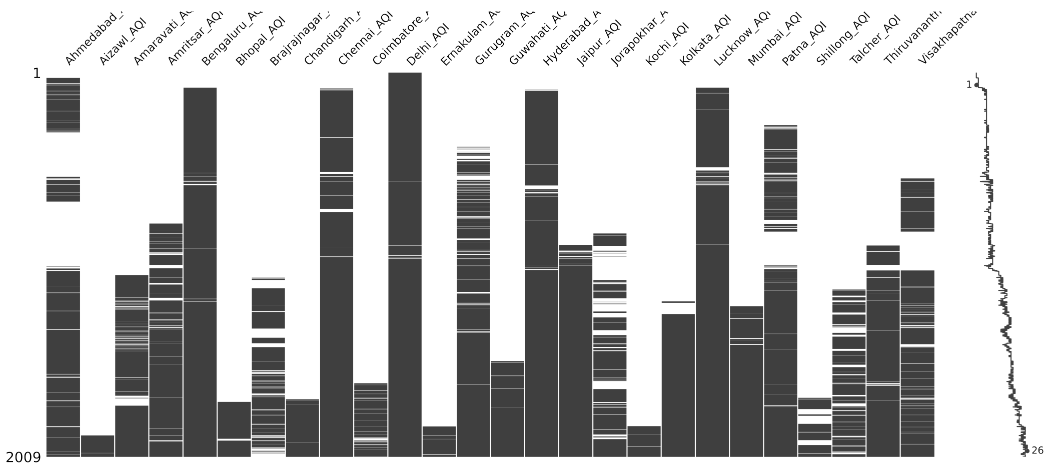


Figure 2: Data Integrity of Different Cites

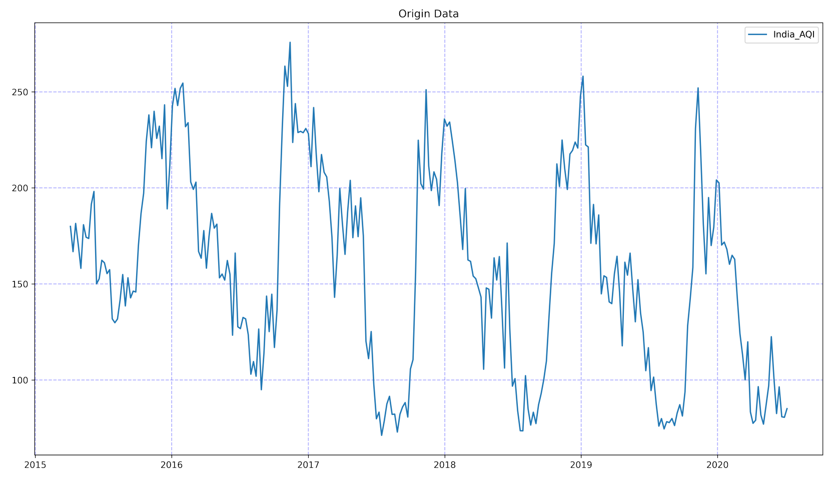


Figure 3: Time Plot of Indian AQI index

We drew this plot by msno.matrix in Python, a useful tool to check the data completeness. In Fig. 2, each vertical column represents one corresponding city, and data completeness of a city is shown by its vertical column. Within a vertical column, the existence of a data is shown by color, in time series from the top to the bottom. The top of a vertical column represents the start date, April 5, 2015, and similarly the bottom of a vertical column represents the end date, July 5, 2020. If the data exist in a certain day, it is shown black, while it is shown white (blank) if it doesn't exist. More specifically, if there is a blank part of a vertical column, that means there are no existing values in the corresponding period of that part. For the five cities mentioned above, few blank/white parts are shown in their corresponding column, indicating that these five cities don't have a large amount of consecutive missing data, which is desirable for Interpolate Data Filling.

3 Model Construction I

An auto-regressive integrated moving average, or ARIMA, is a statistical analysis model that uses time series data to either better understand the data set or to predict future trends. A problem with ARIMA is that it does not support seasonal data which is a time series with a repeating cycle. ARIMA expects data that is either not seasonal or has the seasonal component removed, e.g. seasonally adjusted via methods such as seasonal differencing.In our first model we focus on SARIMA model.

3.1 Log Transformation

After the data pre-processing, we computed the mean of five cities' daily AQI (Bangalore,Chennai,Delhi,Hyderabad and Lucknow)as the sample. We directly performed a Dickey-Fuller test on this time series and found that the null hypothesis can be rejected under 95% significance level (null hypothesis is the original data-set is non-stationary), and under 99% significance level we accept null hypothesis.

Test Statistic:	-3.333672
p-value:	0.013439
Lags Used:	0.000000
Number of Observations Used:	274.000000
Critical Value (1%):	-3.454444
Critical Value (5%):	-2.872147
Critical Value (10%):	-2.572422

Table 1: Dickey-Fuller Test for Log Transformation

From the table 1, It is a bit reluctant for original data to pass the stationarity test. At the same time, we plotted the time series image of the original data and the log transformed data respectively.

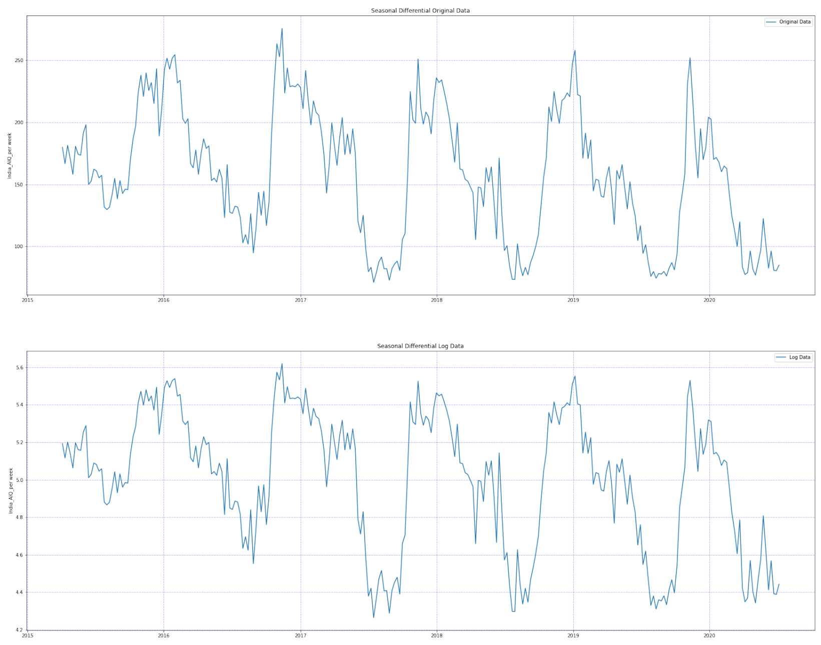


Figure 4: Comparison between Original data and log transformed

It is apparent that the variance of the data after log transformation is smaller than original, which can effectively reduce the heteroscedasticity. Therefore, we used log-transformed data in subsequent model processing.

3.2 Seasonal Difference

We began with drawing the ACF and PACF diagrams of the undifferentiated data.From below figures, the data has an extremely strong seasonality, so we need to make seasonal difference.

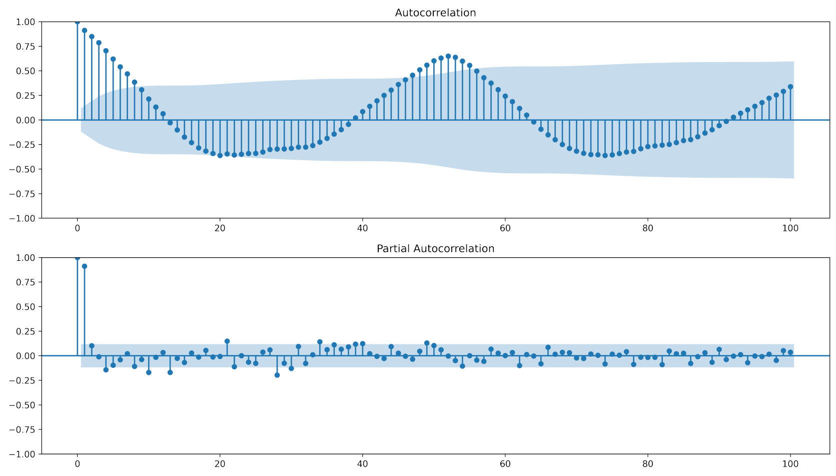


Figure 5: ACF and PACF for Origin Data

When both the seasonal difference and the first order difference are utilized,the result after changing the order is still the same. However, if the seasonal characteristics of the data are relatively strong, we recommended performing seasonal difference first, because sometimes the data after seasonal difference is stable enough leading to no need for subsequent differentiation. If we do the first order difference primarily, we will still need to do a seasonal difference. after seasonal difference,we do Augmented Dickey-Fuller test on the data. Then, we noticed that under 99.999% significance level we should reject null hypothesis.

Test Statistic	-4.168751
p-value	0.000744
Lags Used	2.000000
Number of Observations Used	220.000000
Critical Value (1%)	-3.460428
Critical Value (5%)	-2.874769
Critical Value (10%)	-2.573821

Table 2: Dickey-Fuller Test for log transformation after seasonal difference

3.3 Model Construction

We can summarize three models from ACF and PACF (figure 6):
1. $SARIMA(0, 1, 1) \times (0, 1, 1)_{52}$ (ACF cut off,PACF decay quickly)
2. $SARIMA(2, 1, 0) \times (1, 1, 0)_{52}$(PACF cut off ,ACF decay quickly)
3. $SARIMA(2, 1, 1) \times (1, 1, 1)_{52}$(PACF and ACF noth decay quickly)

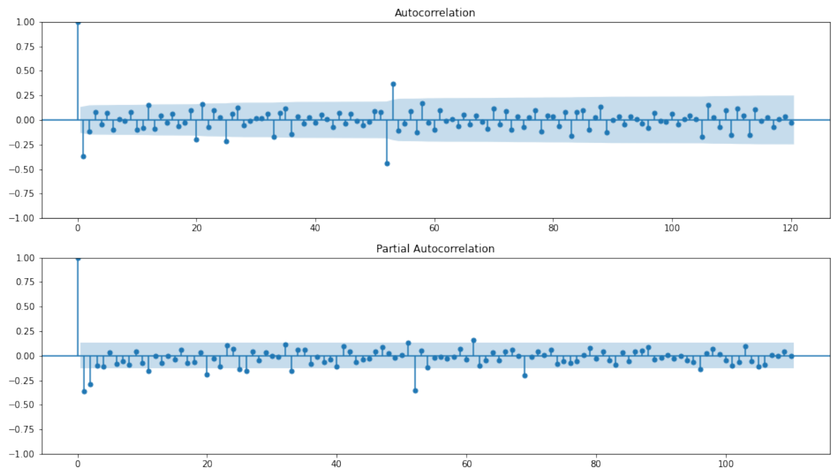


Figure 6: ACF and PACF after seasonal and 1st difference

In addition, We have determined all possible combinations of the parameters of $SARIMA(p,d,q) \times (P,D,Q)_s$ (p is the number of autoregressive terms, d is the number of nonseasonal differences needed for stationarity, and q is the number of lagged forecast errors in the prediction equation; the term $(P,D,Q)_s$ gives the order of the seasonal part,$s$is the lag term).
We used itertools package to traverse all possible combinations, using AIC, BIC and HQIC as their indicators to test the goodness of the model. We demonstrate the top five in table3.

Parameters	AIC	BIC	HQIC	Mean
$SARIMA(0,1,1) \times (0,1,1)_{52}$	-206.946806	-197.205733	-203.000839	-202.384459
$SARIMA(1,1,2) \times (0,1,1)_{52}$	-207.073804	-194.085708	-201.812516	-200.990672
$SARIMA(0,1,2) \times (0,1,1)_{52}$	-206.932219	-193.944123	-201.670930	-200.849091
$SARIMA(1,1,1) \times (0,0,1)_{52}$	-206.489096	-193.501000	-201.227807	-200.405968
$SARIMA(2,1,0) \times (0,0,1)_{52}$	-205.564055	-192.575959	-200.302766	-199.480927

Table 3: Top 5 results

From the table above, it is obvious $SARIMA(0,1,1) \times (0,1,1)_{52}$ is the best one. After we determine the order of the model, we based on statsmodels.api package to estimate the value of each parameter. The tabulation illustrates each parameter is significant since its P-value close to zero.This also proves that our model is reasonable.

Dep. Variable:	log India
Model:	$SARIMAX(0,1,1) \times (0,1,1)_{52}$
No. Observations:	243
Log Likelihood	106.473
AIC	-206.946806
BIC	-197.205733
HQIC	-203.000839

Table 4: SARIMAX Results

	coef	std err	z	$P > \|z\|$	[0.025	0.975]
ma.L1	-0.5250	0.063	-8.331	0.000	-0.648	-0.401
ma.S.L52	-0.8464	0.293	-2.884	0.004	-1.422	-0.271
sigma2	0.0144	0.004	4.101	0.000	-0.008	0.021

Ljung-Box (L1) (Q):	0.4	Jarque-Bera (JB):	3.18
Prob(Q):	0.51	Prob(JB):	0.20
Heteroskedasticity (H):	1.29	Skew:	-0.08
Prob(H) (two-sided):	0.31	Kurtosis:	3.62

Table 5: Model Summary

3.4 Model Prediction

According to 3.3, we trained the model on 243 samples, and the remaining 29 samples were used as the test set. The orange line in the figure below is the value we fitted, and the blue line is the true value. From the figure, the predicted values from December 2019 to March 2020 are good as they close to true values. However, after March 2020, the fitted values are always greater than the true values, so we guess it is influenced by the epidemic. Then, we utilised the intervention model to make predictions.

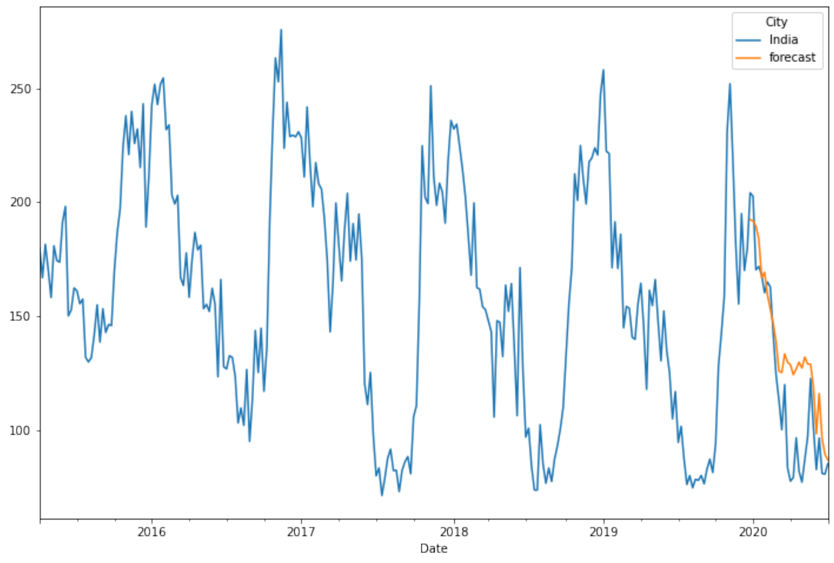


Figure 7: Forecast on test data

3.5 Residual Test

Finally, we test residual. The "residual" in a time series model are what is left over after fitting a model. For many (but not all) time series models, the residuals are equal to the difference between the observations and the corresponding fitted values: $e_t = y_t - \hat{y}_t$ From the following figures,it is obvious that the residual corresponds to white noise.

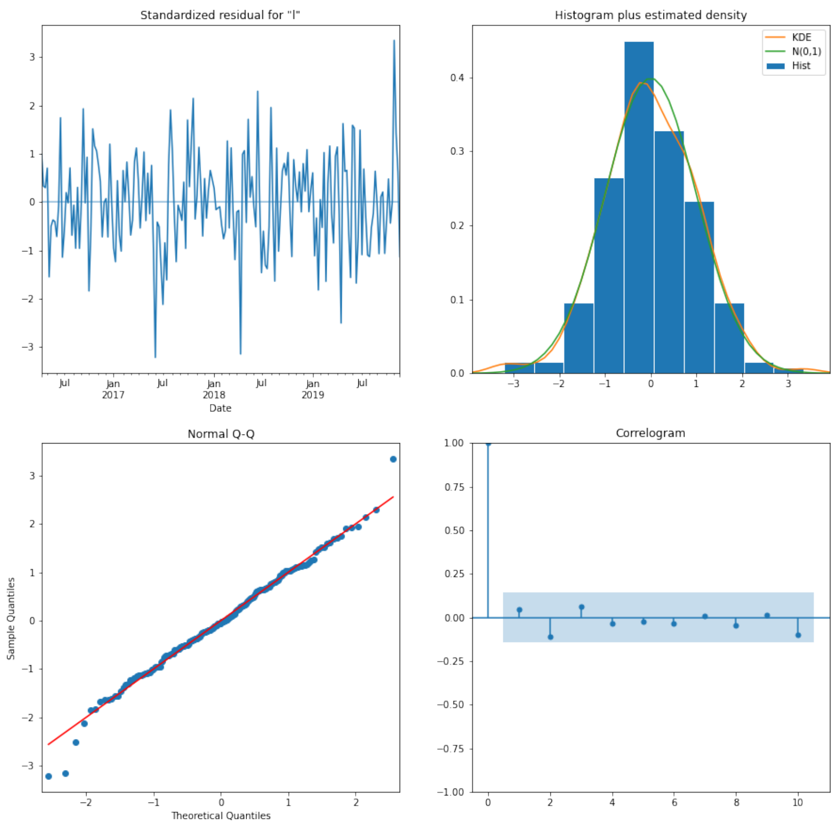


Figure 8: Residual Test Results

4 Model Construction II

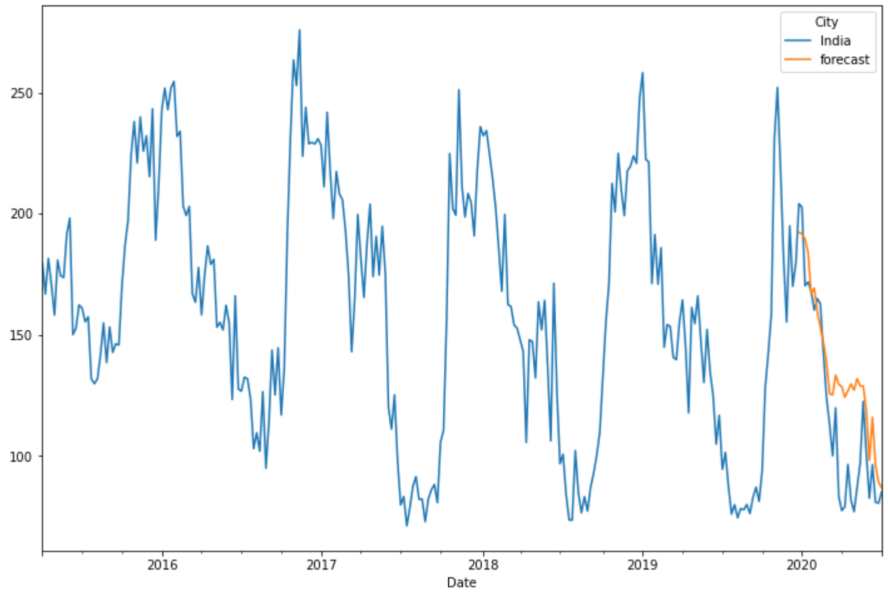


Figure 9: Forecast value of Model I

Referring to the result from the Model I, the model's performance seemed great in the beginning. But things changed from around March 2020 when the prediction error became much larger and unacceptable. So, what happened in India that time?
In fact, we found that due to the COVID-19, India began a 21-day national lockdown on March 25, 2020. 21 days later, the lockdown still not be cancelled. On April 11, the Indian government decided to extend the national foot ban until May 3. And then the lockdown of the controlled area was extended to October 31, 2021. In general, since March 2020, India has been almost completely closed. Many states in India implemented a complete lockdown, and states that did not implement blockade also restricted the movement of people in the state.
We think the lockdown is an intervention event for the air quality, but our model did not capture this information, which is the main reason that result become such poor. So, we tried to fit a intervention model to get better result.

4.1 Introduction to intervention model

The intervention model is based on the basic time series model, taking the intervention event into consideration, adjusting and combining the time series model, and finally establishing a complete and accurate forecasting model. The most important issue in the establishment of an intervention model is to quantify the effect of intervention.

4.1.1 Intervention model

Consider a simple single intervention. After proper transformation, the time series can have a form like:

$$Y_t = m_t + N_t \tag{1}$$

where m_t represents the change of the mean function, and the model of N_t is the ARIMA process (or SAIMA process). We assume that the time series got intervention at time T, which means m_t just equal to 0 if $t < T$.

4.1.2 Representations of intervention variables

There are two representations of intervention variables: One is the continuous intervention variable, which means that the time series is subject to intervention at time T, and the sequence is continuously affected after the intervention occurs. Then the intervention variable is represented by a step function:

$$S_t^T = \begin{cases} 1, & t < T \\ 0, & t > T \end{cases} \tag{2}$$

The second is a transient intervention variable, which means that the time series is interfered at a certain moment, and the intervention only has an impact at that moment. Then the intervention variable is expressed by a unit impulse function:

$$P_t^T = \begin{cases} 1, & t = T \\ 0, & t \neq T \end{cases} \tag{3}$$

The second is a transient intervention variable, which means that the time series is interfered at a certain moment, and the intervention only has an impact at that moment. Then the intervention variable is expressed by a unit impulse function:

4.1.3 Model for intervention variables

Generally, we can build an ARIMA model to fit $\{m_t\}$:

$$m_t = \frac{\omega(B)}{\delta(B)} \cdot P_t^T or \frac{\omega(B)}{\delta(B)} \cdot S_{t-1}^T \tag{4}$$

4.1.4 Modeling process

The specific steps of building intervention model modeling are as follows:

(1) Use the time series data before the intervention to establish a time sequence model(Y_t^{before}), and then obtain the predicted value of the time sequence when no intervention event occurs.

(2) Build Model for intervention variables. Using

$$m_t = Y_t^{observed} - \hat{Y}_t^{before} | t > T \tag{5}$$

to get the value of the intervention sequence. And then build ARIMA model for it.

(3) Calculate the data after excluding the influence of the intervention and obtain the sequence of excluding the influence of the intervention:

$$N_t = Y_t^{observed} - \hat{m}_t \tag{6}$$

And then build a univariate time series model for it.

(4) Obtain the final intervention analysis model

$$Y_t = m_t + N_t \tag{7}$$

4.2 Intervention model for Indian AQI

4.2.1 Building model based on the data before the intervention

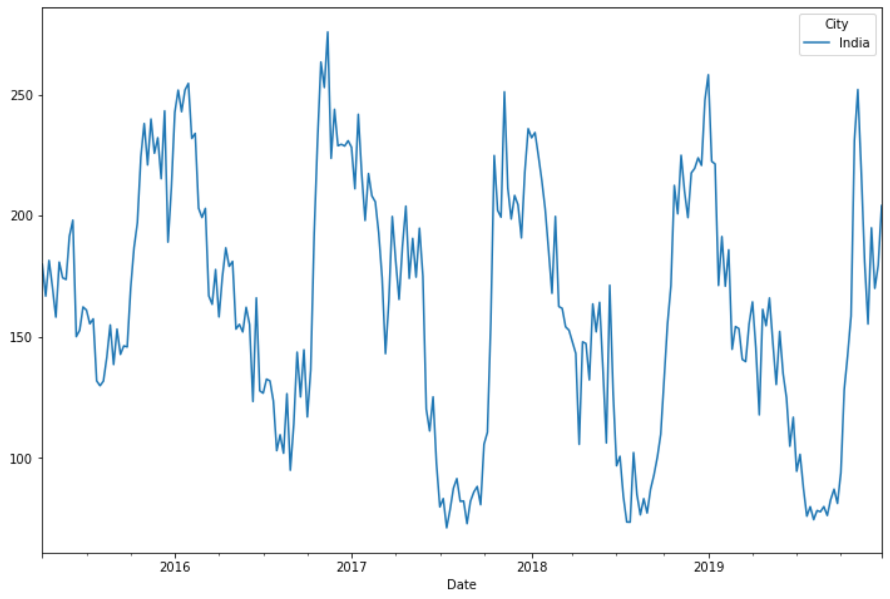


Figure 10: Log transformed Data

Since the Model I used the data from 2015-04-05 to 2019-12-01 which have not met intervention. So we can just use the model I as the model based on the data before the intervention.
We got the model:

$$\bigtriangledown^{d=1}\bigtriangledown^{D=52}Y_t^{before} = e_t - 0.5250 * e_{t-1} - 0.8464 * e_{t-52} + 0.5250 * 0.8464 * e_{t-53} \tag{8}$$

4.2.2 Build Model for intervention variables

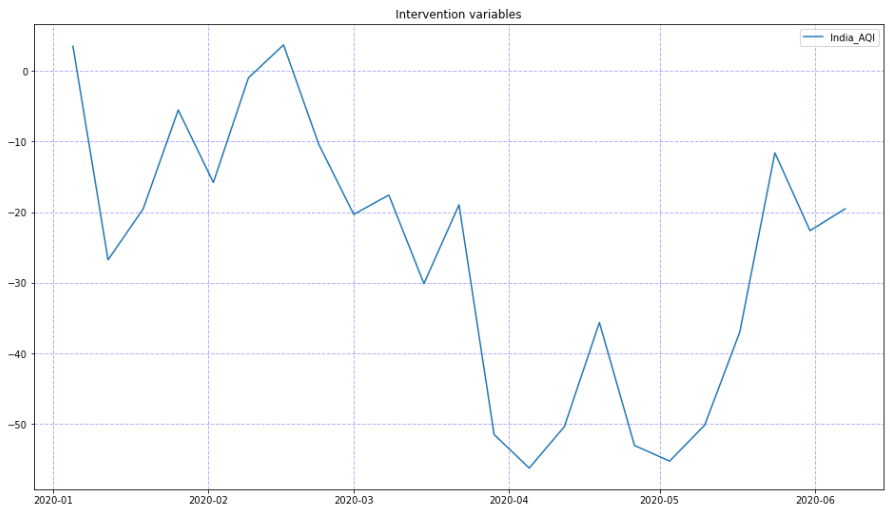


Figure 11: Intervention variable graph

Stationarity transformation Since ACF and PACF decay quickly, we think this time series data is stationary. Thus, we do not do any transformation of it.

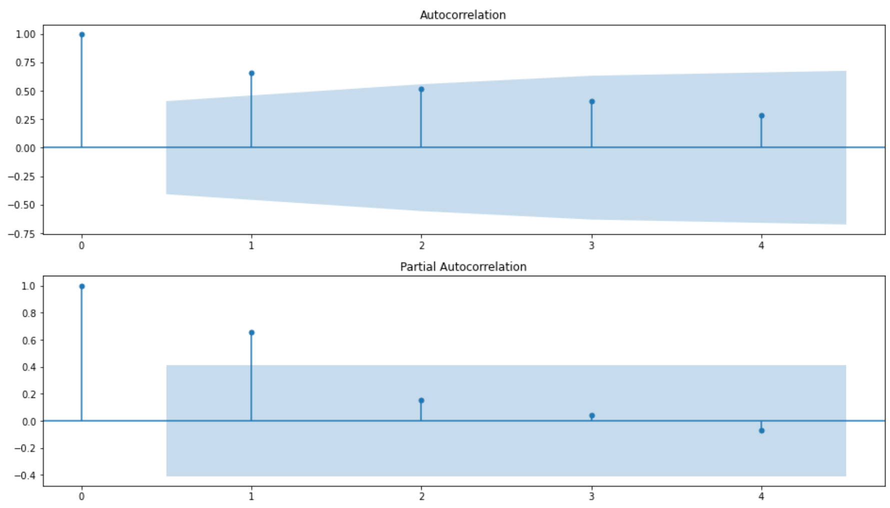
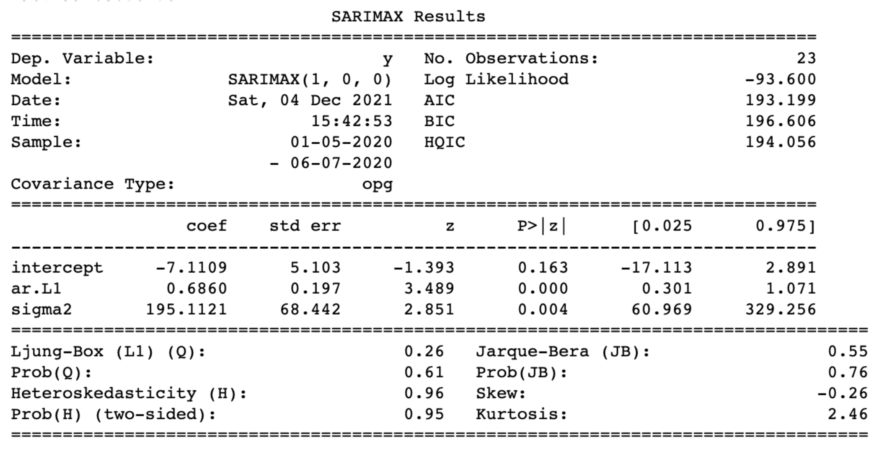


Figure 12: ACF and PACF

Model specification, Fitting and diagnostics For the ACF and PACF, there are two ways to think about it:
Firstly we can think the ACF cut off at lag 1 and PACF decay quickly, which lead to MA(1) model. Secondly we can also think the PACF cut off at lag 1 and ACF decay quickly, which lead to AR(1) model. We tried both model, the result was as blow:



```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   23
Model:               SARIMAX(1, 0, 0)   Log Likelihood                 -93.600
Date:                Sat, 04 Dec 2021   AIC                            193.199
Time:                        15:42:53   BIC                            196.606
Sample:                    01-05-2020   HQIC                           194.056
                         - 06-07-2020
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -7.1109      5.103     -1.393      0.163     -17.113       2.891
ar.L1          0.6860      0.197      3.489      0.000       0.301       1.071
sigma2       195.1121     68.442      2.851      0.004      60.969     329.256
===================================================================================
Ljung-Box (L1) (Q):                   0.26   Jarque-Bera (JB):                 0.55
Prob(Q):                              0.61   Prob(JB):                         0.76
Heteroskedasticity (H):               0.96   Skew:                            -0.26
Prob(H) (two-sided):                  0.95   Kurtosis:                         2.46
===================================================================================
```

Figure 13: AR(1) model

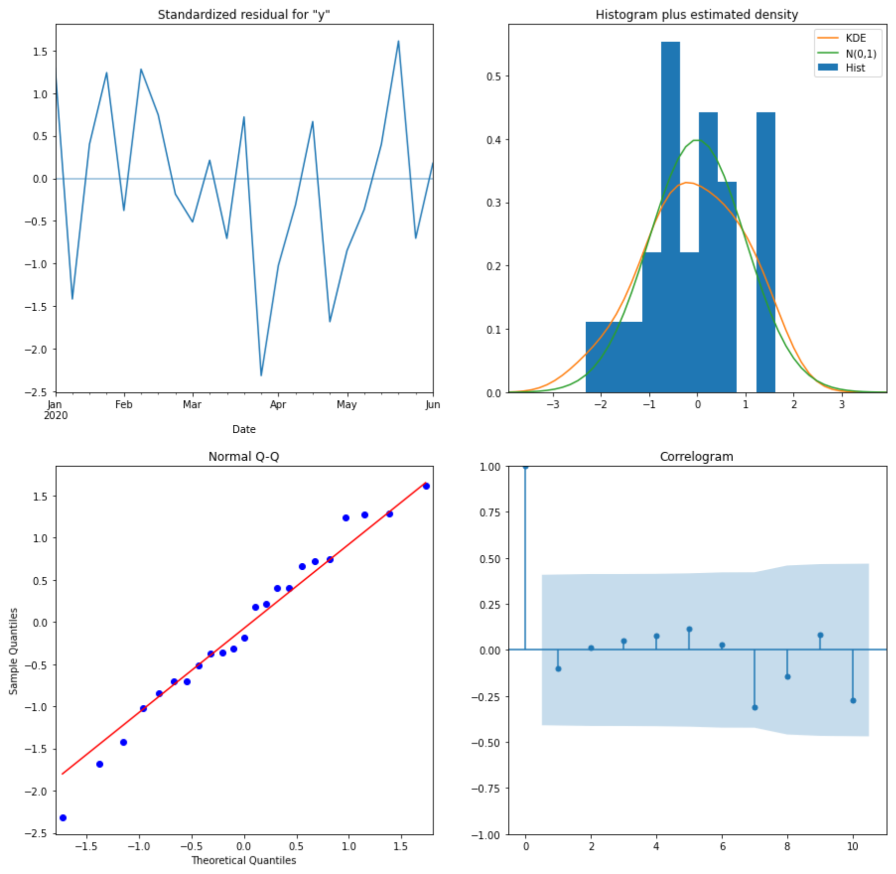


Figure 14: Residual Test

Based on the LB test result, the P value is large. At the 0.05 level, we have no reason to reject the null hypothesis, that is, it can be considered that there is no correlation for the residual of the AR(1) model. Besides, the Scatter plot and ACF of residual all indicate that the it is a white noise. And the distribution of residual is very close to normal based on the QQ plot.

```
                               SARIMAX Results                                
==============================================================================
Dep. Variable:                      y   No. Observations:                   23
Model:               SARIMAX(0, 0, 1)   Log Likelihood                 -94.878
Date:                Sat, 04 Dec 2021   AIC                            195.755
Time:                        15:43:10   BIC                            199.162
Sample:                    01-05-2020   HQIC                           196.612
                         - 06-07-2020                                         
Covariance Type:                  opg                                         
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept    -36.1028      5.108     -7.068      0.000     -46.114     -26.091
ma.L1          0.5486      0.223      2.455      0.014       0.111       0.986
sigma2       220.6996     94.658      2.332      0.020      35.173     406.226
===================================================================================
Ljung-Box (L1) (Q):                   0.52   Jarque-Bera (JB):                 1.11
Prob(Q):                              0.47   Prob(JB):                         0.57
Heteroskedasticity (H):               0.94   Skew:                            -0.38
Prob(H) (two-sided):                  0.94   Kurtosis:                         2.23
===================================================================================
```

Figure 15: MA(1) model

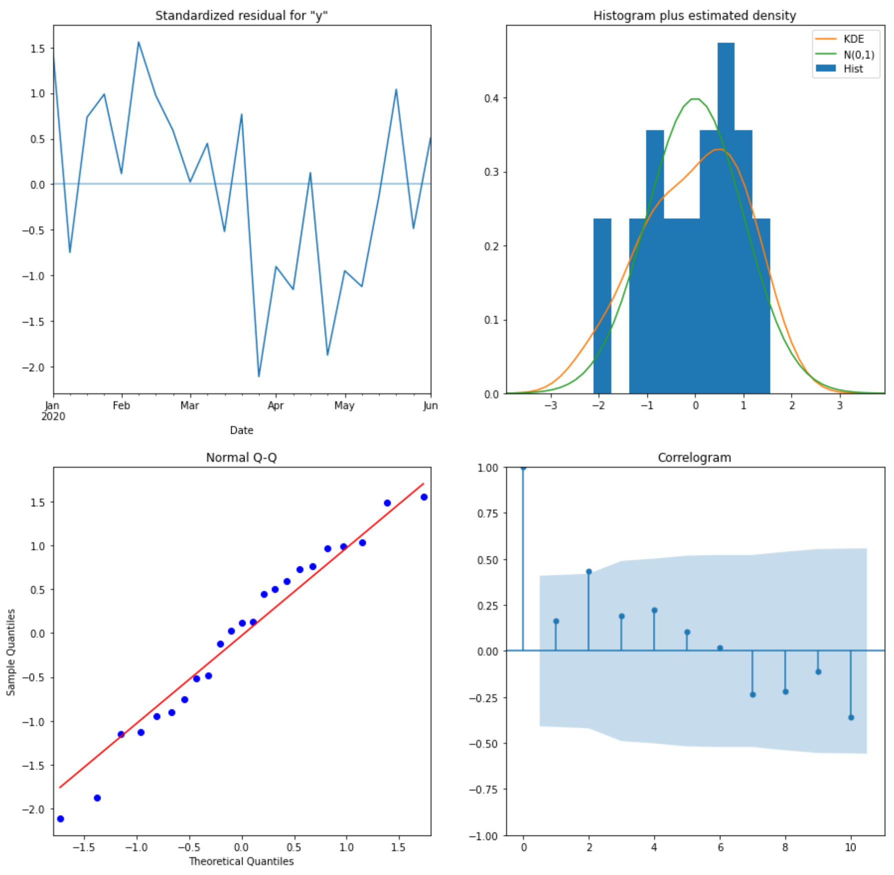


Figure 16: Residual Test

Based on the LB test result, the P value is large. At the 0.05 level, we have no reason to reject the null hypothesis, that is, it can be considered that there is no correlation for the residual of the MA(1) model. Besides, the Scatter plot and ACF of residual all indicate that the it is a white noise. And the distribution of residual is very close to normal based on the QQ plot.
Thus both two model are suitable model. Since the AIC for AR(1) model is a little smaller than MA(1) model, we finally chose the AR(1) model. For our case, we think COVID-19 is a continuous intervention variable, thus we use a step function to represent it.

$$S_t^T = \begin{cases} 1, & t > T \\ 0, & t < T \end{cases} \tag{9}$$

Then we got the model for intervention variables:

$$m_t = 0.686 m_{t-1} - 7.1109_t^T \tag{10}$$

The AR form of intervention model means that the intervention may only gradually affect the mean function, and its full impact can only be fully manifested after a period of time, which is also reasonable for our case.

4.2.3 Build Model for the sequence excluding the influence of the intervention

Stationarity transformation Following the same procedure from Model Construction I, we found after log-transformation, no-seasonal first order difference and seasonal first order difference, we achieved stationarity.

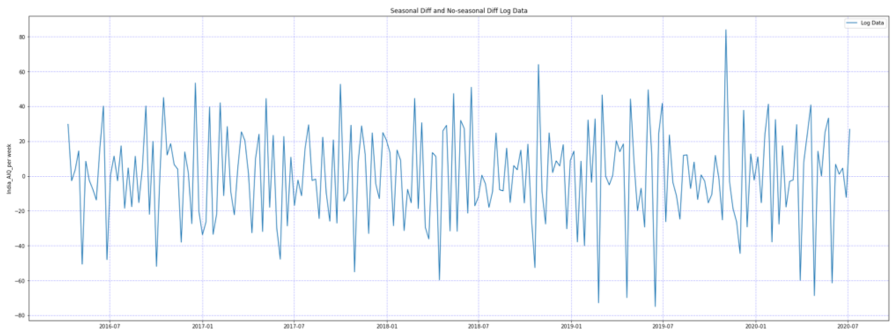


Figure 17: Seasonal Difference Graph

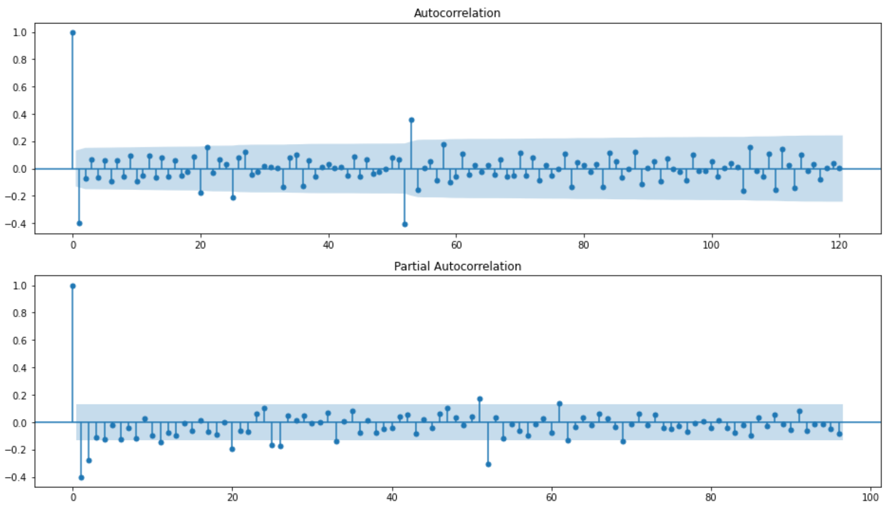
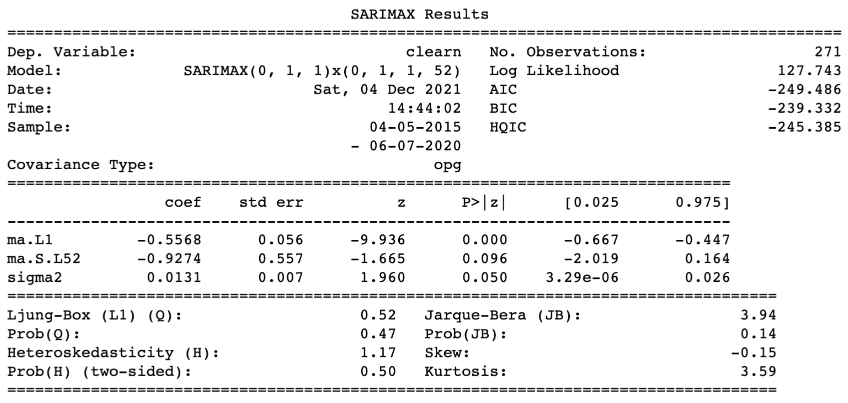


Figure 18: ACF and PACF of Seasonal Difference Data

Through the scatter chart, we can see that the average value of the data remains stable.data fluctuates around the average value, and the amplitude of the fluctuation is relatively stable. ACF and PACF also decay quickly.

Model specification, Fitting and diagnostics The patterns of ACF and PACF are very similar to Model Construction I, after carefully model specification, we still choose $SARIMA(0,1,1)(0,1,1)_{52}$ to fit the data.



```
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                               clearn   No. Observations:                271
Model:             SARIMAX(0, 1, 1)x(0, 1, 1, 52)   Log Likelihood                 127.743
Date:                              Sat, 04 Dec 2021   AIC                           -249.486
Time:                                      14:44:02   BIC                           -239.332
Sample:                                  04-05-2015   HQIC                          -245.385
                                       - 06-07-2020
Covariance Type:                                opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.5568      0.056     -9.936      0.000      -0.667      -0.447
ma.S.L52      -0.9274      0.557     -1.665      0.096      -2.019       0.164
sigma2         0.0131      0.007      1.960      0.050    3.29e-06       0.026
===================================================================================
Ljung-Box (L1) (Q):                   0.52   Jarque-Bera (JB):                 3.94
Prob(Q):                              0.47   Prob(JB):                         0.14
Heteroskedasticity (H):               1.17   Skew:                            -0.15
Prob(H) (two-sided):                  0.50   Kurtosis:                         3.59
===================================================================================
```

Figure 19: SARIMA Results of Model II

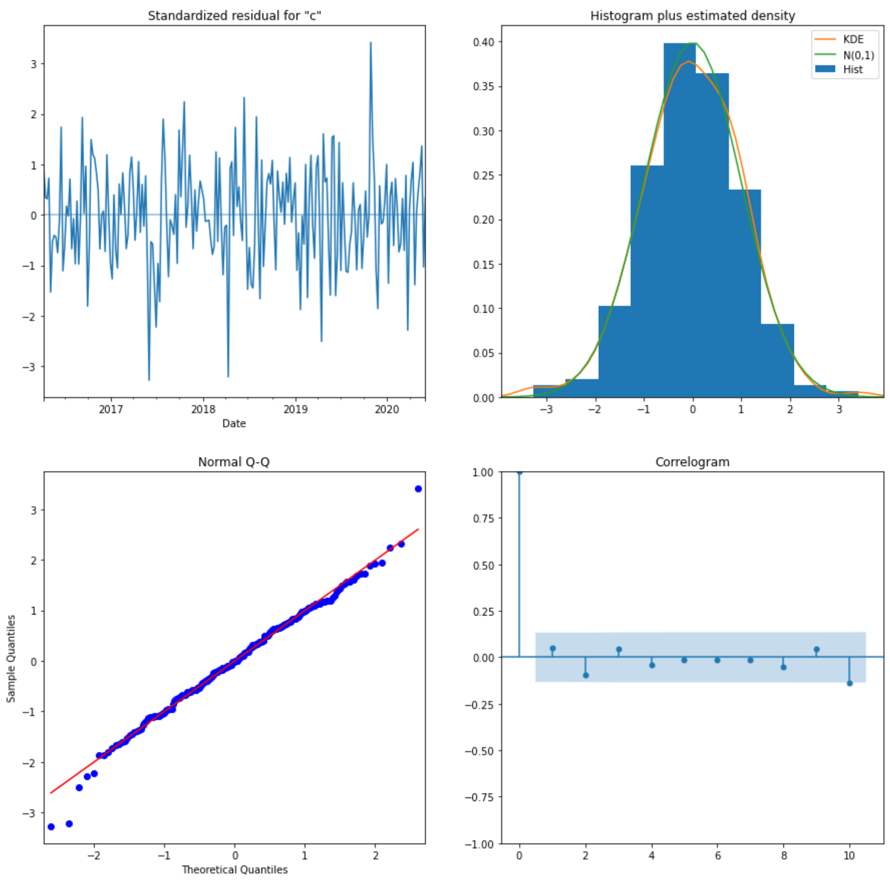


Figure 20: Residual Test

Based on the LB test result, the P value is large. At the 0.05 level, we have no reason to reject the null hypothesis, that is, it can be considered that there is no correlation for the residual. Besides, the Scatter plot and ACF of residual all indicate that it is a white noise. And the distribution of residual is very close to normal based on the QQ plot.
We got the model:

$$\bigtriangledown^{d=1} \bigtriangledown N_t = e_t - 0.5568e_{t-1} - 0.9274e_{t-52} + 0.5568 \times 0.9274 \times e_{t-53} \tag{11}$$

4.3 Final Results

We finally got the model:

$$Y_t = m_t + N_t \tag{12}$$

$$m_t = 0.686m_{t-1} - 7.1109S_t^T \tag{13}$$

$$\bigtriangledown^{d=1} \bigtriangledown N_t = e_t - 0.5568e_{t-1} - 0.9274e_{t-52} + 0.5568 \cdot 0.9274 \cdot e_{t-53} \tag{14}$$

From the picture, the intervention model has achieved very good results

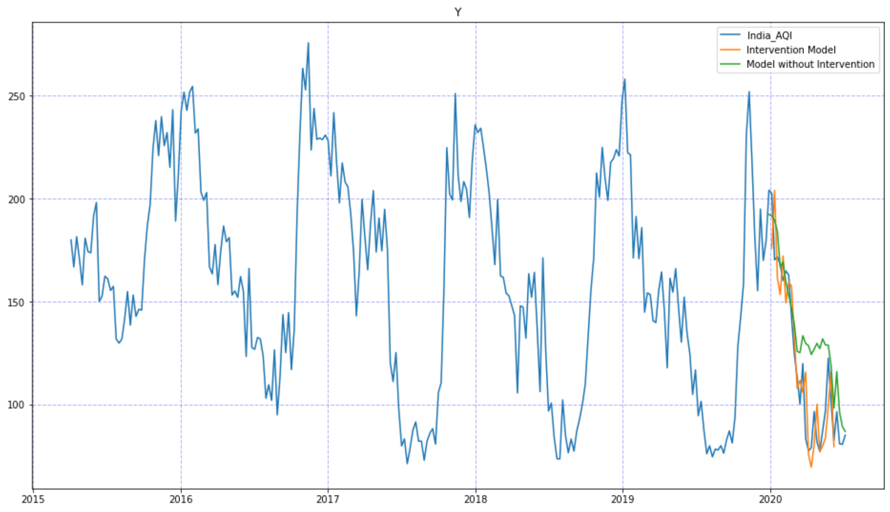


Figure 21: there should be a title here

5 Conclusion

The outcome of prediction shows that Intervention Model provides an evident improvement to the original SARIMA model, with 61.5% decrease of MSE and 25% decrease of MAPE. Intervention Model achieves a good performance with MAPE equals to 26%, indicating that it is a successful model for predicting air pollution in India before and after the outbreak of COVID-19 since 2020.

	MSE	MAPE
SARIMA	1475.4	0.3465
Intervention Model	568.3	0.25931
Improvement	61.5%	25%

Table 6: Evaluation of SARIMA and Intervention Model

Appendix

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


#####SARIMA Method


# In[1]:


import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
import numpy as np


# In[2]:


#显示所有行和列
pd.options.display.max_columns = None
pd.options.display.max_rows = None
np.set_printoptions(threshold=np.inf)


# In[3]:


df = pd.read_csv('./city_day.csv')
df['Date'] = pd.to_datetime(df['Date'])
df.head(10)


# In[4]:


# 查看每个column 有多少na 值
df.isna().sum()


# In[5]:
```

```
cities_all = df.pivot_table(values='AQI', index=['Date'],
columns='City')
cities_all=cities_all.add_suffix('_AQI')
cities_all.isna().sum()
# Bengaluru_AQI,Chennai_AQI,Delhi_AQI,Hyderabad_AQI,Lucknow_AQI 相对来说
缺失值较少
# 且分配均匀，所以用这五个城市作为 india 的均值


# In[6]:


msno.matrix(cities_all)
plt.show()
# 根据缺失值的数量和缺失值的分布选
Bengaluru_AQI,Chennai_AQI,Delhi_AQI,Hyderabad_AQI,Lucknow_AQI 作为研究对
象


# In[7]:


# 线性填补
df_India = pd.DataFrame()
df_India =
cities_all[['Bengaluru_AQI','Chennai_AQI','Delhi_AQI','Hyderabad_AQI','
Lucknow_AQI']]
df_India=df_India.resample(rule='W').mean()
print(df_India.isna().sum())
print('----------------')

df_India = df_India.interpolate('linear')
df_India['India'] = df_India.mean(axis=1)
print(df_India.isna().sum())
print('----------------')

# 线性填补不了的数据是最前面连着缺失的数据，这些数据我们将其删掉
df_India  = df_India.dropna()
print(df_India.isna().sum())


# In[8]:
```

```python
df_India.head()


# In[9]:


# 后面使用线性填补的数据作为分析

plt.figure(figsize=(16,9))
plt.title('Origin Data')
plt.plot(df_India['India'], label='India_AQI')
plt.grid(color='b', linestyle='--', linewidth=1, alpha=0.3)
plt.legend()
plt.show()


# In[10]:


from statsmodels.tsa.stattools import adfuller
# 此时先用 ADF 对原数据做一下平稳性检验

dftest = adfuller(df_India['India'],autolag='BIC')
dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-
value','#Lags Used','Number of Observations Used'])
for key,value in dftest[4].items():
    dfoutput['Critical Value (%s)'%key] = value

print(dfoutput)
# 可以看到通过检验，但有点勉强


# In[16]:


'''
Note!!!!!
Rejecting the null hypothesis of a unit root after applying the ADF
test does not imply that the series is stationary in all respects.
The ADF test is devised to detect non-stationarity in the long-term
cycle, i.e. the trend of the series. Non-stationarity in other cycles
are not inspected by this test. As long as the series exhibits a
stationary
trend pattern, sources of non-stationarity such as a changing seasonal
pattern or a switch in the variance of the series will most
likely result in rejecting the null hypothesis of the ADF test.
```

```
'''


# In[17]:


# 画一下 ACF 和 PACF 的图
# 可以看出有强季节性
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
fig, ax = plt.subplots(2,figsize=(16, 9))
ax[0] = plot_acf(df_India['India'], lags=100, ax=ax[0])
ax[1] = plot_pacf(df_India['India'],lags=100, ax=ax[1], method='ywm')
plt.show()


# In[18]:


# 原数据稍微有点异方差，对原数据取 Log
df_India['log_India'] = df_India['India'].apply(np.log)


# In[19]:


# 下面先做差分,先做季节性差分
'''
当季节性差值和第一差值都被使用时，两者的先后顺序并不会影响结果—变换顺序后的结
果仍是一样的。
然而，如果数据的季节性特征比较强，我们建议先进行季节性差分，因为有时经过季节性
差分的数据已经足够平稳，
没有必要进行后续的差分。如果先进行第一差分,我们仍将需要做一次季节性差分。
Reference:
https://otexts.com/fppcn/stationarity.html
'''
diff_season = [(df_India['India'][i+52]-df_India['India'][i]) for i in
range(0,len(df_India['India'])-52)]
diff_log_season = [(df_India['log_India'][i+52]-
df_India['log_India'][i])
                  for i in range(0,len(df_India['log_India'])-52)]
# 差分后少了 52 个数据，所以要在这里插入 52 个 None
for i in range(52):
    diff_season.insert(0,None)
    diff_log_season.insert(0,None)
# 更新一下 dataframe
```

```
df_India['diff_season'] = diff_season
df_India['diff_log_season'] = diff_log_season
# check 一下新表
print(df_India.isna().sum())
df_India.head()


# In[20]:


# 差分之后的图像
plt.figure(figsize=(30,24))

plt.subplot(211)
plt.plot(df_India['India'],label='Original Data')
#plt.plot(df_India['diff_season'], label='diff_season')
plt.grid(color='b', linestyle='--', linewidth=1, alpha=0.3)
plt.title('Seasonal Differential Original Data')
plt.ylabel('India_AIQ_per week')
plt.legend()

plt.subplot(212)
plt.plot(df_India['log_India'],label='Log Data')
#plt.plot(df_India['diff_log_season'], label='log_diff_season')
plt.grid(color='b', linestyle='--', linewidth=1, alpha=0.3)
plt.title('Seasonal Differential Log Data')
plt.ylabel('India_AIQ_per week')
plt.legend()

# plt.subplot(313)
# plt.plot(df_India['diff_season'], label='diff_season')
# plt.plot(df_India['diff_log_season'], label='log_diff_season')
# plt.grid(color='b', linestyle='--', linewidth=1, alpha=0.3)
# plt.title('Seasonal Differential')
# plt.ylabel('India_AIQ_per week')
# plt.legend()
plt.show()

# 感觉取了 log 之后再差分会平稳很多


# In[21]:


# 对季节性差分后的数据做 ADFtest
```

```python
from statsmodels.tsa.stattools import adfuller

print('---ADF test for seasonal differential method---')
dftest = adfuller(df_India['diff_season'].dropna(),autolag='BIC')
dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-
value','#Lags Used','Number of Observations Used'])
for key,value in dftest[4].items():
    dfoutput['Critical Value (%s)'%key] = value

print(dfoutput)

print(r'---ADF test for seasonal differential method (log)---')
dftest = adfuller(df_India['diff_log_season'].dropna(),autolag='BIC')
dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-
value','#Lags Used','Number of Observations Used'])
for key,value in dftest[4].items():
    dfoutput['Critical Value (%s)'%key] = value

print(dfoutput)
# 可以看到p 值非常接近0，平稳
#


# In[22]:


# 画一下ACF 和PACF 的图
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
fig, ax = plt.subplots(2,figsize=(16, 9))
ax[0] = plot_acf(df_India['diff_season'].dropna(), lags=50, ax=ax[0])
ax[1] = plot_pacf(df_India['diff_season'].dropna(),lags=50, ax=ax[1],
method='ywm')
plt.show()


# In[23]:


# Log 版
# 画一下ACF 和PACF 的图
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
fig, ax = plt.subplots(2,figsize=(16, 9))
```

```
ax[0] = plot_acf(df_India['diff_log_season'].dropna(), lags=50,
ax=ax[0])
ax[1] = plot_pacf(df_India['diff_log_season'].dropna(),lags=50,
ax=ax[1], method='ywm')
plt.savefig('log_acf_pacf.png')
plt.show()


# In[24]:


# 感觉下降的不是很快，所以再做一次差分
diff1 = [(df_India['diff_season'][i+1]-df_India['diff_season'][i]) for
i in range(52,len(df_India['diff_season'])-1)]
diff1_log = [(df_India['diff_log_season'][i+1]-
df_India['diff_log_season'][i])
             for i in range(52,len(df_India['diff_log_season'])-1)]

for i in range(53):
    diff1.insert(0,None)
    diff1_log.insert(0,None)

df_India['diff1'] = diff1
df_India['diff1_log'] = diff1_log
df_India


# In[25]:


# 画一下ACF 和PACF 的图
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
fig, ax = plt.subplots(2,figsize=(16, 9))
ax[0] = plot_acf(df_India['diff1_log'].dropna(), lags=120, ax=ax[0])
ax[1] = plot_pacf(df_India['diff1_log'].dropna(),lags=110, ax=ax[1],
method='ywm')
plt.savefig('log_acf_pacf_diff.png',dip=600)
plt.show()


# In[298]:


from pmdarima import auto_arima
# 使用AutoArima 拟合一下看看
# P 指的是在 PACF 中有多少个滞后点
# Q 指的是在 ACF 中有多少个滞后点
auto_arima(y=df_India['log_India'],start_p=1,start_P=1,start_q=1,start_
Q=1,
           seasonal=True,W=52, stepwise=True).summary()


# In[299]:


# 设置区间搜索参数
from itertools import product
ps = range(0,3)
d = 1
qs = range(0,3)
Ps = range(0,2)
D = 1
Qs = range(0,2)
s = 52

parameter = product(ps,qs,Ps,Qs)
parameter_list = list(parameter)
print(len(parameter_list))
# 总共迭代 36 次


# In[300]:


get_ipython().system('pip install tqdm')


# In[301]:


from tqdm import tqdm_notebook
def optimizeSARIMA(parameter_list,d,D,s):
    results = []
    bestaic = float("inf")

    for param in tqdm_notebook(parameter_list):
        try:
            model =
sm.tsa.statespace.SARIMAX(df_India['log_India'][0:247],
                                            order=(param[0], d,
param[1]),
                                            seasonal_order=(param[2],D,p
aram[3],s)).fit(disp=5)
        except:
            continue
        aic = model.aic
        bic = model.bic
        hqic = model.hqic
        results.append([param,aic,bic,hqic])

    results_table = pd.DataFrame(results, columns=['parameter', 'aic',
'bic', 'hqic'])
    results_table =
results_table.sort_values(by='aic',ascending=True).reset_index(drop=Tru
e)

    return results_table


# In[302]:


results_table = optimizeSARIMA(parameter_list,d,D,s)


# In[303]:


results_table['mean_eva'] =
results_table[['aic','bic','hqic']].mean(axis=1)
results_table.sort_values(by='mean_eva',ascending=True).reset_index(dro
p=True)


# In[358]:


# 参数最后选择 1201
import statsmodels.api as sm

# 一般来说做 seasonal 会要求有 5 个完整的 season，这里不足，所以会报一个 error
model=sm.tsa.statespace.SARIMAX(df_India['log_India'][0:243],order=(0,
1, 1),seasonal_order=(0,1,1,52))
results=model.fit(disp=5)
aic = results.aic
print(aic)
print(results.summary())
```

```
#### 这里的pdq PDQ 是看上面的ACF 和PACF 选择的，和autoarima 稍微有点不同


# In[306]:


df_India['forecast']=results.predict(start=247,end=275,dynamic=True).ap
ply(np.exp)
df_India[['India','forecast']].plot(figsize=(12,8))
plt.show()


# In[307]:


# 残差性检验
residual = results.predict(start=0,end=246,dynamic=True) -
df_India['log_India'][0:247]
x_label = sorted(residual)
y_label =
sorted(np.random.normal(np.mean(x_label),np.std(x_label),len(x_label)))
plt.plot(x_label,y_label)
line_x = np.arange(np.min(x_label),np.max(x_label),0.01)
plt.plot(line_x,line_x,c='r')
plt.show()


# In[308]:


test = df_India['forecast'][247:275]
true = df_India['India'][247:275]
n = len(test)
mape = 1/n*sum(abs((test-true)/true))
mse = 1/n*sum((test-true)**2)
print(mape)
print(mse)


# In[309]:


df_India['forecast'][271:275]


# In[310]:
```

```
#要不要report 这个？
test = df_India['forecast'][271:275]
true = df_India['India'][271:275]
n = len(test)
mape = 1/n*sum(abs((test-true)/true))
mse = 1/n*sum((test-true)**2)
print(mape)
print(mse)


# ## 干预模型

# In[361]:


df_India['forecast']=results.predict(start='2019-12-29',end='2020-06-
28',dynamic=True)
df_India['forecast_exp'] = df_India['forecast'].apply(np.exp)
df_India[['India','forecast_exp']].plot(figsize=(12,8))
plt.show()


# 2020 年，3 月 25 日，印度开始进行为期 21 天的全国封锁，4 月 11 日，印度政府决定
延长全国禁足令至 5 月 3 日，根据印度内政部发布的“解封 5.0”措施，严控区的封锁延长
至 10 月 31 日
# 2021 年，面对第二波疫情，印度多邦实施全面封锁，没有进行封锁的邦也限制邦内人员
流动。
#
https://zh.wikipedia.org/wiki/2019%E5%86%A0%E7%8B%80%E7%97%85%E6%AF%92%
E7%97%85%E5%8D%B0%E5%BA%A6%E7%96%AB%E6%83%85#2020%E5%B9%B4
#
# 但是因为从 3 月开始的话，数据太少，所以还是以 2020-01-01 分开数据

# In[362]:


#拆分
train = df_India['2015-04-05':'2020-01-01'] #没有干预的时候的模型
test = df_India['2020-01-02':'2020-06-07']
forcast = df_India['2020-06-14':'2020-07-05'] #用于预测的部分


# ## 拟合没有干预的时候的模型
# In[364]:


import statsmodels.api as sm
# 一般来说做 seasonal 会要求有 5 个完整的 season，这里不足，所以会报一个 error
model=sm.tsa.statespace.SARIMAX(df_India['log_India'][0:243],order=(0,
1, 1),seasonal_order=(0,1,1,52))
results_raw=model.fit(disp=5)
aic = results_raw.aic
print(aic)
print(results.summary())
#### 这里的 pdq PDQ 是看上面的 ACF 和 PACF 选择的，和 autoarima 稍微有点不同


# In[365]:


#用 train 集合上面的数据得到的 model 来 fit test 集合
df_India['forecast_test']=results_raw.predict(start ='2020-01-01',end =
'2020-06-07',dynamic=True)
df_India['forecast_test_exp'] = df_India['forecast_test'].apply(np.exp)
df_India[['India','forecast_test_exp']].plot(figsize=(12,8))
plt.show()


# ## 对于预序列拟合模型

# In[366]:


# 干预项序列：就是用真实值-train model 的 fit 值
Z = df_India['India'] - df_India['forecast_test_exp']
Z = Z['2020-01-01':'2020-06-07']
Z


# In[367]:


# Z 序列的图
plt.figure(figsize=(16,9))
plt.title('Intervention variables')
plt.plot(Z, label='India_AQI')
plt.grid(color='b', linestyle='--', linewidth=1, alpha=0.3)
plt.legend()
plt.show()
```

```
# In[368]:


#平稳性检验
from statsmodels.tsa.stattools import adfuller
dftest = adfuller(Z,autolag='BIC')
dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-
value','#Lags Used','Number of Observations Used'])
for key,value in dftest[4].items():
    dfoutput['Critical Value (%s)'%key] = value
print(dfoutput)


# In[369]:


# 画一下 ACF 和 PACF 的图
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
fig, ax = plt.subplots(2,figsize=(16, 9))
ax[0] = plot_acf(Z, lags=4, ax=ax[0])
ax[1] = plot_pacf(Z,lags=4, ax=ax[1], method='ywm')
plt.show()


# In[371]:


#可以尝试模型 AR 1 或者  MA 1
import statsmodels.api as sm
# 一般来说做 seasonal 会要求有 5 个完整的 season，这里不足，所以会报一个 error
model_train=sm.tsa.statespace.SARIMAX(Z,order=(1, 0,
0),seasonal_order=(0,0,0,0),
                                    enforce_stationarity = True,
enforce_invertibility = True, trend = "c")
results_Z=model_train1.fit(disp=5) #这个参数是什么用处啊
aic = results_Z.aic
print(aic)
print(results_Z.summary())


# In[373]:
```

```python
import statsmodels.api as sm
# 一般来说做 seasonal 会要求有 5 个完整的 season，这里不足，所以会报一个 error
model_train=sm.tsa.statespace.SARIMAX(Z,order=(0, 0,
1),seasonal_order=(0,0,0,0),
                                    enforce_stationarity = True,
enforce_invertibility = True,trend = "c")
results_ZZ=model_train.fit(disp=5) #这个参数是什么用处啊
aic = results_ZZ.aic
print(aic)
print(results_ZZ.summary())


# In[374]:


results_Z.plot_diagnostics(figsize=(15,15))
plt.savefig('residual white noise test.png',dip=600)
plt.show()


# In[375]:


results_ZZ.plot_diagnostics(figsize=(15,15))
plt.savefig('residual white noise test.png',dip=600)
plt.show()


# ## 得到去除干预模型后的数据 净化序列

# In[376]:


df_India['clearn'] = df_India['India']
df_India['clearn']['2020-01-05':'2020-06-07'] =
df_India['India']['2020-01-05':'2020-06-07'] - results_Z.fittedvalues
#df_India['clearn']['2020-01-05':'2020-06-07'] =
df_India['India']['2020-01-05':'2020-06-07'] - Z


# In[377]:


Y = df_India['clearn']['2015-04-05':'2020-06-07']
```

```
# In[378]:


plt.figure(figsize=(16,9))
plt.title('Sequence Excluding the Influence Of the Intervention:')
plt.plot(Y, label='India_AQI')
plt.grid(color='b', linestyle='--', linewidth=1, alpha=0.3)
plt.legend()
plt.show()


# In[379]:


#平稳性检验
from statsmodels.tsa.stattools import adfuller
dftest = adfuller(Y,autolag='BIC')
dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-
value','#Lags Used','Number of Observations Used'])
for key,value in dftest[4].items():
    dfoutput['Critical Value (%s)'%key] = value
print(dfoutput)


# In[380]:


# 画一下ACF 和PACF 的图
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
fig, ax = plt.subplots(2,figsize=(16, 9))
ax[0] = plot_acf(Y, lags=50, ax=ax[0])
ax[1] = plot_pacf(Y,lags=50, ax=ax[1], method='ywm')
plt.show()


# In[381]:


Y_diff = Y.diff()
Y_diff = Y_diff.diff(52)


# In[382]:
```

```python
# 差分之后的图像
plt.figure(figsize=(30,24))
plt.subplot(212)
plt.plot(Y_diff,label='Log Data')
#plt.plot(df_India['diff_log_season'], label='log_diff_season')
plt.grid(color='b', linestyle='--', linewidth=1, alpha=0.3)
plt.title('Seasonal Diff and No-seasonal Diff Log Data')
plt.ylabel('India_AIQ_per week')
plt.legend()
plt.show()


# In[383]:


# 画一下 ACF 和 PACF 的图
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
fig, ax = plt.subplots(2,figsize=(16, 9))
ax[0] = plot_acf(Y_diff.dropna(), lags=120, ax=ax[0])
ax[1] = plot_pacf(Y_diff.dropna(),lags=96, ax=ax[1], method='ywm')
plt.show()


# In[384]:


log_Y = np.log(Y)


# In[385]:


import statsmodels.api as sm
# 一般来说做 seasonal 会要求有 5 个完整的 season，这里不足，所以会报一个 error
model_Y=sm.tsa.statespace.SARIMAX(log_Y[0:271],order=(0, 1,
1),seasonal_order=(0,1,1,52),
                                enforce_stationarity = True,
enforce_invertibility = True)
results_Y=model_Y.fit(disp=5)
aic = results.aic
print(aic)
print(results_Y.summary())


# In[386]:
```

```
results_Y.plot_diagnostics(figsize=(15,15))
plt.savefig('residual white noise test.png',dip=600)
plt.show()


# In[387]:


#最终结果X :
#Y-- 用真实值 - 干预序列估计的arima model + 干预项序列：就是用真实值-train
model 的fit 值
#是不是不应该用fittedvalues (fittedvalues  =  predict dynamic= false
df_India['final_fitted'] = np.exp(results_Y.fittedvalues)
df_India['final_fitted']["2020-01-05":"2020-06-07"] =
np.exp(results_Y.fittedvalues)["2020-01-05":"2020-06-07"] +
results_Z.fittedvalues


# ## 结果

# In[388]:


plt.figure(figsize=(16,9))
plt.title('Y')
plt.plot(df_India['India'], label='India_AQI')
plt.plot(df_India['final_fitted']["2020-01-05":"2020-06-07"],
label='Intervention Model')
plt.plot(df_India['forecast'], label='Model without Intervention')
#plt.plot(df_India['forecast_exp'], label='forecast')
plt.grid(color='b', linestyle='--', linewidth=1, alpha=0.3)
plt.legend()
plt.show()


# ## 干预模型的预测

# In[390]:


XX = results_Z.predict(start = "2020-06-14",end = "2020-07-
05",dynamic=True) + np.exp(results_Y.predict(start = "2020-06-14",end =
"2020-07-05",dynamic=True))
```

```python
# In[392]:


plt.figure(figsize=(16,9))
plt.title('X')
plt.plot(XX, label='forecast')
plt.plot(df_India['India']['2020-01-02':'2020-07-05'],
label='India_AQI')
#plt.plot(df_India['forecast']['2020-06-14':'2020-07-05'],
label='forecast')
plt.grid(color='b', linestyle='--', linewidth=1, alpha=0.3)
plt.legend()
plt.show()


# In[393]:


test = XX
true = df_India["India"]['2020-06-14':'2020-07-05']
n = len(test)
mape = 1/n*sum(abs((test-true)/true))
mse = 1/n*sum((test-true)**2)
print(mape)
print(mse)
```