Forecasting Technical Report

Student ID 31756387
Module code MATH6178
WORD COUNT 2000

1. Exponential Smoothing Method

1.1 Preliminary Analysis

1.1.1 Average Weekly Earnings: Service Sector (K55O)

This dataset contains 167 observations of average weekly earnings from year 2000 to year 2013.

From the time plot of the K55O which is shown as Figure 1, it is observed that there is a strong increasing trend pattern in this plot, and it also has a strong seasonal pattern. To validate that there is a strong seasonal pattern, a seasonal plot of K55O was plotted which is shown as Figure 2. We can observe that the average weekly earnings will be at peak in March every year and will be at bottom in April every year. This is probably caused by the Easter Holiday in UK. To further prove the Trend and seasonality, the ACF plot of K55O was drawn. From the Figure 3, it is observed that the peak will appear every 12^{th} time lag which it confirms the seasonality, and the spike show up at lag 1 and then towars zero which it also confirms the trend pattern in this time series data.

1.1.2 Producer Price Index: JVZ8

This dataset contains 215 observations of all manufacturing output from year 1996 to year 2013.

From the time plot of the JVZ8 which is shown as Figure 4, it is observed that there is a strong increasing trend pattern in this plot and there is no seasonal trend. To confirm the trend pattern, we plotted the ACF of JVZ8 which is shown as Figure 5, and it can be observed that the spike appears at lag 1 and then towards zero which can validate the existence of trend pattern.

1.1.3 Overseas Visits to the UK: GMAA

From the Figure 6 and Figure 7, we can observe and confirm the strong trend pattern and seasonal pattern in this dataset.

1.1.4 Overseas Visits to the UK: Earnings (GMAK)

From the Figure 6]8 and Figure 9, we can also observe and confirm the strong

trend pattern and seasonal pattern in this dataset.

1.2 Decomposition

In this part, we use MA method to estimate the underlying Trend-Cycle and additive decomposition or multiplicative decomposition to estimate the seasonality.

1.2.1 K55O

From the Figure 10, we applied 2×12-MA applied to the K55O. It is observed that the smooths line shows the seasonality and also the trend pattern. it is almost the same as the trend-cycle shown in Figure 1. And

1.2.2 JVZ8

From the Figure 11, we applied 2×12-MA applied to the JVZ8. It is observed that the smooths line shows no seasonality but the trend pattern. It is almost the same as the trend-cycle shown in Figure 4. There is also a large increase in this plot in year 2008. This may cause by the world economic crisis of 2008 year.

1.2.3 GMAA and GMAK

From the Figure 12 and Figure 13, we applied 2×12-MA applied to the GMAA and GMAK. It is observed that the smooth lines show no seasonality but the trend pattern. It is almost the same as the trend-cycle shown in Figure 6 and Figure 8.

Multiplicative Decomposition of four data are shown as Figure 20,21,22.

1.3 Data Preparation

1.3.1 Missing and Erroneous Data

In this study, we did not find any missing and erroneous data of the dataset.

1.3.2 Transformation

Calendar adjustment: This data is for calendar month and therefor need a calendar adjustment. The calendar adjustment is applied on K55O, GMAA and GMAK which are shown as Figure 14, 15 and 16. The line is much smoother than the original data.

Power Transform and Logarithm Transform:

The power and Logarithm transform are applied to K55O, GMAK, GMAA which are shown as Figure 17,18,19 to make sure the variance of random errors is time independent. For the K55O, applying the power transform is better; for the GMAA, it will apply to logarithm transform and GMAK will apply the power transform.

1.4 Model Selection

The four Exponential smoothing which is SES, LES, HWA and HWM methods were compared to find which model will be suitable for each dataset which indicates that the MSE will be the smallest.

		K55O	JVZ8	GMAA	GMAK
MSE	SES	205.82	0.095	108151.86	35290.99
	LES	205.05	0.065	110939.56	36236.02
	HWA	38.00	0.052	14055.86	6628.46
	HWM	37.21	0.052	16532.73	5965.04

Table 1: Comparison of Exponential Smoothing Methods

The Table 1 shows: for the K55O data the HWM method will be suitable; for the JVZ8 data the HMA and HWM method are all suitable; for the GMAA data the HWA method will be suitable; for the GMAK data the HWM method will be suitable.

1.5 Forecasting

After the model selection, we will use the model to forecast.

1.5.1 K55O

The result of forecasting K55O until December 2014 is shown as Figure 23. The forecast can fit the original line well except the year 2009. The parameters of the optimised model are shown as Table 2. From the Table, we can observe

that the MSE of the model is not small. The Residual ACF of HWM Model is shown as Figure 24. It can be observed that there is no seasonal pattern and trend pattern in this plot, and there is no spike at lag 1 and other lags. This Residual can be seen as white nose which means that the HWM model is suitable for the K55O data set

	alpha	beta	gamma	l0	b0	MSE
Value	0.24	0.00	0.60	387.63	1.32	37.21

Table 2: Parameter of HWM Model(K55O)

1.5.2 JVZ8

The result of forecasting JVZ8 until December 2014 is shown as Figure 25. The forecast can fit the original line perfect. The parameters of the optimised model are shown as Table 3. From the Table 3, we can observe that the MSE of the model is quite small. The Residual ACF of HWM Model is shown as Figure 26. It can be observed that there is no seasonal pattern and trend pattern in this plot, and there is no spike at lag 1 and other lags. This Residual can be seen as white nose which means that the HWM model is suitable for the JVZ8 data set.

	alpha	beta	gamma	l0	b0	MSE
Value	1.00	5.63 e-01	1.47 e-08	9.02 e+01	0.14	0.052

Table 3: Parameter of HWM Model (JVZ8)

1.5.3 GMAA

The result of forecasting GMAA until December 2014 is shown as Figure 27. The forecast can fit the original line perfect. The parameters of the optimised model are shown as Table 4. From the Table 4, we can observe that the MSE of the model is quite large. The Residual ACF of HWA Model is shown as Figure

28. It can be observed that there is no seasonal pattern and trend pattern in this plot, and there is no spike at lag 1 and other lags. This Residual can be seen as white nose which means that the HWM model is suitable for the GMAA data set.

	alpha	beta	gamma	l0	b0	MSE
Value	0.32	0.02	0.20	1483.7	2.15	14055.86

Table 4: Parameter of HWA Model (GMAA)

1.5.4 GMAK

The result of forecasting GMAK until December 2014 is shown as Figure 29. The forecast can fit the original line perfect. The parameters of the optimised model are shown as Table 5. From the Table 5, we can observe that the MSE of the model is quite large. The Residual ACF of HWM Model is shown as Figure 30. It can be observed that there is no seasonal pattern and trend pattern in this plot, and there is no spike at lag 1 and other lags. This Residual can be seen as white nose which means that the HWM model is suitable for the GMAK data set.

	alpha	beta	gamma	l0	b0	MSE
Value	2.52 e-01	1.10e-04	0.24	6.64 e+02	3.31	1.05 e+06

Table 4: Parameter of HWM Model (GMAK)

2. ARIMA Forecasting

2.1 Preliminary Analysis

From the section 1.5.2, we find that the result of the forecast and MSE value in JVZ8 dataset is better than others. Thus, in this part, we will focus on JVZ8 dataset.

2.2 Data Preparation

To use the ARIMA method, the data should be stationary which means that there is no trend and seasonal pattern. However, in the section 1, we found that the JVZ8 data though has not seasonal pattern it has trend pattern. To confirm that, we plotted the JVZ8 ACF plot and PACF plot which are shown as Figure 31 and Figure 32. It is observed that ACF after several lags is towards zero and PACF still has spike at lag 1. These are all indicate the data is not stationary.

To get the stationary data, we first differenced the data. The time plot, ACF and PACF plot are shown as Figure 33,34,35. However, the Plots still show the data not stationary.

Thus, we truncated the data and data is now from 1996 JAN to 2010 DEC and after 1st differencing. The time plot, ACF and PACF are shown as Figure 36, 37,38. The ACF drops so quickly to towards zero and PACF plot has not spike at lag 1. Therefore, this truncated data can be seen as stationary.

,

2.3 Model Selection

In this study, there is no seasonal appeared in this data, so the value of P, D, Q, S are zero. The initial models: ARIMA (0, d,0), ARIMA (2, d,2), ARIMA (1, d,0), ARIMA(0,d,1) are varied fitted. These models are built to find the best model which has the lowest Akaike's Information Criterion (AIC). The results are shown as Table 6:

We can observe that the Model ARIMA (0,1,2) has the lowest the AIC and therefore it is selected to forecast the further data.

Model	p	d	q	P	D	Q	S	AIC	MSE
1	0	1	1	0	0	0	0	-55.214	
2	0	0	0	0	0	0	0	1553.919	
3	0	1	0	0	0	0	0	-44.945	
4	1	0	0	0	0	0	0	-31.156	
5	0	0	1	0	0	0	0	1378.872	
6	1	0	1	0	0	0	0	-43.340	
7	1	1	0	0	0	0	0	-54.564	
8	1	1	1	0	0	0	0	-55.214	
9	0	1	2	0	0	0	0	-55.229	0.05
10	1	1	2	0	0	0	0	-54.138	
11	2	1	1	0	0	0	0	-53.322	
12	2	1	2	0	0	0	0	-53.260	

Table 6: ARIMA Model Comparison

2.3 Forecast and Evaluation

After the model ARIMA (0,1,2) forecasting, the forecast model is shown as Figure 39. And the residuals plot is shown as Figure 40. The ACF plot of the residuals from the ARIMA (0,1,2) model shows that all autocorrelations are within the threshold limits, indicating that the residuals are behaving like white noise. And the MSE of model is 0.05 which is smaller than the value of the HWM method in JZV8 data. This means the ARIMA method is more suitable than the exponential smoothing method in JVZ8 data.

3. Regression Prediction

3.1 Preliminary Analysis

In this part, we will use the GMAA, K55O and JVZ8 as input features to build regression model and predict the GMAK. We will use the same period data to forecast the GMAK. Thus, all the data of variables is from 2000 JAN to 2013 Nov.

To know the relationship among those variables, we made a correlation table among these four variables which is shown as Table 7:

It is observed that there is a strong correlation between GMAA and GMAK. This may bring the multicollinearity problem.

	K55O	JVZ8	GMAA	GMAK
K55O	1.0	-0.20	-0.25	-0.18
JVZ8	-0.20	1.0	0.14	0.13
GMAA	-0.25	0.14	1.0	0.83
GMAK	-0.18	0.13	0.83	1.0

Table 7: Correlation of Variables

3.2 Model Selection

In this part, there are four models built to find the best model (OLS method). And the model metrics are R^2 and regression p-value.

These four models are as follows:

$$GMAK = \beta_0 + \beta_1(GMAA) + \beta_2(JVZ8) + \beta_3(K55O) + \epsilon \quad (1)$$

$$GMAK = \beta_0 + \beta_1(GMAA) + \beta_2(JVZ8) + \beta_3(K55O) + \beta_4(D1) + \beta_5(D2) + \beta_6(D3) + \beta_7(D4) + \beta_8(D5) + \beta_9(D6) + \beta_{10}(D7 + \beta_{11}(D8) + \beta_{12}(D9) + \beta_{13}(D10) + \beta_{14}(D11) + \epsilon \quad (2)$$

$$GMAK = \beta_0 + \beta_1(GMAA) + \beta_2(JVZ8) + \beta_3(K55O) + \beta_4(D1) + \beta_5(D2) + \beta_6(D3) + \beta_7(D4) + \beta_8(D5) + \beta_9(D6) + \beta_{10}(D7 + \beta_{11}(D8) + \beta_{12}(D9) + \beta_{13}(D10) + \beta_{14}(D11) + \beta_{15}(t) + \epsilon \quad (3)$$

$$GMAK = \beta_0 + \beta_1(GMAA) + \beta_5(D2) + \beta_6(D3) + \beta_7(D4) + \beta_8(D5) + \beta_9(D6) + \beta_{10}(D7) + \beta_{14}(D11) + \beta_{15}(t) + \epsilon \quad (4)$$

After the regression, we obtained the four model R squared and P-value which is shown as Table 8:

	Model 1	Model 2	Model 3	Model4
R-squared	0.695	0.891	0.891	0.843
Prob(F-statistics)	1.08e-12	4.79e-14	2.18e-13	5.02e-16

Table 8: Regression Model Comparison

We can observe form the Table 8 that the largest R squared is Model 2 and Model 3, but the P value of the Model 2 is smaller than the Models 3. The smallest p-value is Model 4 but the value of R-squared is not much big. In sum up, we will choose the model 2 to forecast the 2014 Dec data.

3.3 Forecasting

In this part, we will use the LSE method to forecast the GMAA, JVZ8 and K55O variables. The reason why does not use HWM method is that the multiplicative method will not use the negative value to forecast. In this data, we differenced the input variables which include the negative value. The forecast plot of GMAA, JVZ8 and K55O are shown as Figure 41,42,43.

The final forecast graph is shown as Figure 44.

APPENDIX A

```
from sklearn.metrics import mean_squared_error
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import numpy as np
from statsmodels.formula.api import ols
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.tsa.api import ExponentialSmoothing
from statsmodels.tsa.api import Holt
from statsmodels.tsa.api import SimpleExpSmoothing
from numpy import sqrt
from pandas import DataFrame
from numpy import log
from statsmodels.tsa.seasonal import seasonal_decompose
from pandas.plotting import autocorrelation_plot
from matplotlib import pyplot
from pandas import read_excel
import matplotlib.pyplot as plt
import pandas as pd
series = read_excel('fore.xlsx', sheet_name='K55O', header=0,
                                index_col=0, parse_dates=True, squeeze=True)
series.plot(color='green')
plt.xlabel('Dates')
plt.ylabel('Average Weekly Earnings in £')
plt.title('Average Weekly Earnings from 2000 to 2013')
plt.show()

# Seasonal plot K55O
series = read_excel('fore.xlsx',
                                sheet_name='seasonal K55O', header=0,
                                index_col=0, parse_dates=True, squeeze=True)
```

```
series.plot()
pyplot.xlabel('Months')
pyplot.ylabel('Average Weekly Earnings in £')
pyplot.title('Seasonal Plot of K55O')
pyplot.show()


# ACF Plot K55O
series = read_excel('fore.xlsx', sheet_name='K55O', header=0, index_col=0,
                                parse_dates=True, squeeze=True)
autocorrelation_plot(series)
pyplot.show()

# Decompose K55O

# MAK55O
series = read_excel('fore.xlsx',
                                sheet_name='MAK55O', header=0,
                                index_col=0, parse_dates=True, squeeze=True)
series.plot()
pyplot.show()

# Multiplicative K550
series = read_excel('fore.xlsx', sheet_name='K55O', header=0,
                                index_col=0, parse_dates=True, squeeze=True)
#result = seasonal_decompose(series, model='additive')
result = seasonal_decompose(series, model='multiplicative')
result.plot()
plt.show()
```

```
# Data Preparation of K55O
# adcal
series = read_excel('fore.xlsx',
                    sheet_name='adcal K55O', header=0,
                    index_col=0, parse_dates=True, squeeze=True)
series.plot()
pyplot.show()

# log
series = read_excel('fore.xlsx',
                    sheet_name='K55O', header=0,
                    index_col=0, parse_dates=True, squeeze=True)
dataframe = DataFrame(series.values)
dataframe.columns = ['K55O']
dataframe['K55O'] = log(dataframe['K55O'])
pyplot.figure(1)
# line plot
pyplot.subplot(211)
pyplot.plot(dataframe['K55O'])
# histogram
pyplot.subplot(212)
pyplot.hist(dataframe['K55O'])
pyplot.show()

# Sqrt
series = read_excel('fore.xlsx',
                    sheet_name='K55O', header=0,
                    index_col=0, parse_dates=True, squeeze=True)
dataframe = DataFrame(series.values)
dataframe.columns = ['K55O']
```

```
dataframe['K55O'] = sqrt(dataframe['K55O'])
pyplot.figure(1)
# line plot
pyplot.subplot(211)
pyplot.plot(dataframe['K55O'])
# histogram
pyplot.subplot(212)
pyplot.hist(dataframe['K55O'])
pyplot.show()

# time plot JVZ8
series = read_excel('fore.xlsx', sheet_name='JVZ8', header=0,
                    index_col=0, parse_dates=True, squeeze=True)
series.plot(color='green')
plt.xlabel('Dates')
plt.ylabel('All Manufacturing Output ')
plt.title('Producer Price Index from 1996 to 2013')
plt.show()

# ACF JVZ8
series = read_excel('fore.xlsx', sheet_name='JVZ8', header=0, index_col=0,
                    parse_dates=True, squeeze=True)
autocorrelation_plot(series)
pyplot.show()

# Decompose JVZ8
# MA JVZ8
series = read_excel('fore.xlsx',
                    sheet_name='MAJVZ8', header=0,
                    index_col=0, parse_dates=True, squeeze=True)
```

```
series.plot()
pyplot.show()

# Multilicative JVZ8
series = read_excel('fore.xlsx', sheet_name='JVZ8', header=0,
                            index_col=0, parse_dates=True, squeeze=True)
#result = seasonal_decompose(series, model='additive')
result = seasonal_decompose(series, model='multiplicative')
result.plot()
plt.show()

# timeplot GMAA
series = read_excel('fore.xlsx', sheet_name='GMAA', header=0,
                            index_col=0, parse_dates=True, squeeze=True)
series.plot(color='green')
plt.xlabel('Dates')
plt.ylabel('Oversea Visits to UK')
plt.title('Oversea Visits to UK from 1980 to 2013')
plt.show()

# GMAK
series = read_excel('fore.xlsx', sheet_name='GMAK', header=0,
                            index_col=0, parse_dates=True, squeeze=True)
series.plot(color='green')
plt.xlabel('Dates')
plt.ylabel('Oversea Visits to UK Earnings')
plt.title('Oversea Visits to UK Earnings from 1980 to 2013')
plt.show()

# ACF GMAA
```

```
series = read_excel('fore.xlsx', sheet_name='GMAA', header=0, index_col=0,
                        parse_dates=True, squeeze=True)
autocorrelation_plot(series)
pyplot.show()

# GMAK
series = read_excel('fore.xlsx', sheet_name='GMAK', header=0, index_col=0,
                        parse_dates=True, squeeze=True)
autocorrelation_plot(series)
pyplot.show()

# Decompose
# MA GMAA
series = read_excel('fore.xlsx',
                        sheet_name='MAGMAA', header=0,
                        index_col=0, parse_dates=True, squeeze=True)
series.plot()
pyplot.show()

# MA GMAK
series = read_excel('fore.xlsx',
                        sheet_name='MAGMAK', header=0,
                        index_col=0, parse_dates=True, squeeze=True)
series.plot()
pyplot.show()

# Mulplicayive GMAA
series = read_excel('fore.xlsx', sheet_name='GMAA', header=0,
                        index_col=0, parse_dates=True, squeeze=True)
#result = seasonal_decompose(series, model='additive')
```

```
result = seasonal_decompose(series, model='multiplicative')
result.plot()
plt.show()

# Mulplicayive GMAK
series = read_excel('fore.xlsx', sheet_name='GMAK', header=0,
                            index_col=0, parse_dates=True, squeeze=True)
#result = seasonal_decompose(series, model='additive')
result = seasonal_decompose(series, model='multiplicative')
result.plot()
plt.show()

# Data Preparation of GMAA
# adcal
series = read_excel('fore.xlsx',
                            sheet_name='adGMAA', header=0,
                            index_col=0, parse_dates=True, squeeze=True)
series.plot()
pyplot.show()

# Data Preparation of GMAK
# adcal
series = read_excel('fore.xlsx',
                            sheet_name='adGMAK', header=0,
                            index_col=0, parse_dates=True, squeeze=True)
series.plot()
pyplot.show()



# log GMAA
```

```
series = read_excel('fore.xlsx',
                    sheet_name='tansGMAA', header=0,
                    index_col=0, parse_dates=True, squeeze=True)
dataframe = DataFrame(series.values)
dataframe.columns = ['GMAA']
dataframe['GMAA'] = log(dataframe['GMAA'])
pyplot.figure(1)
# line plot
pyplot.subplot(211)
pyplot.plot(dataframe['GMAA'])
# histogram
pyplot.subplot(212)
pyplot.hist(dataframe['GMAA'])
pyplot.show()

# sqrt GMAA
series = read_excel('fore.xlsx',
                    sheet_name='tansGMAA', header=0,
                    index_col=0, parse_dates=True, squeeze=True)
dataframe = DataFrame(series.values)
dataframe.columns = ['tansGMAA']
dataframe['tansGMAA'] = sqrt(dataframe['tansGMAA'])
pyplot.figure(1)
# line plot
pyplot.subplot(211)
pyplot.plot(dataframe['tansGMAA'])
# histogram
pyplot.subplot(212)
pyplot.hist(dataframe['tansGMAA'])
pyplot.show()
```

```
# log GMAK
series = read_excel('fore.xlsx',
                    sheet_name='transGMAK', header=0,
                    index_col=0, parse_dates=True, squeeze=True)
dataframe = DataFrame(series.values)
dataframe.columns = ['GMAK']
dataframe['GMAK'] = log(dataframe['GMAK'])
pyplot.figure(1)
# line plot
pyplot.subplot(211)
pyplot.plot(dataframe['GMAK'])
# histogram
pyplot.subplot(212)
pyplot.hist(dataframe['GMAK'])
pyplot.show()

# sqrt GMAK
series = read_excel('fore.xlsx',
                    sheet_name='transGMAK', header=0,
                    index_col=0, parse_dates=True, squeeze=True)
dataframe = DataFrame(series.values)
dataframe.columns = ['transGMAK']
dataframe['transGMAK'] = sqrt(dataframe['transGMAK'])
pyplot.figure(1)
# line plot
pyplot.subplot(211)
pyplot.plot(dataframe['transGMAK'])
# histogram
pyplot.subplot(212)
```

```
pyplot.hist(dataframe['transGMAK'])
pyplot.show()

# SES #alpha = 0.5
# K55O
series1 = read_excel('fore.xlsx', sheet_name='K55O', header=0,
                                index_col=0, parse_dates=True, squeeze=True)
fit1 = SimpleExpSmoothing(series1).fit()
# JVZ8
series2 = read_excel('fore.xlsx', sheet_name='JVZ8', header=0,
                                index_col=0, parse_dates=True, squeeze=True)
fit2 = SimpleExpSmoothing(series2).fit()
# GMAA
series3 = read_excel('fore.xlsx', sheet_name='GMAA', header=0,
                                index_col=0, parse_dates=True, squeeze=True)
fit3 = SimpleExpSmoothing(series3).fit()
# GMAK
series4 = read_excel('fore.xlsx', sheet_name='GMAK', header=0,
                                index_col=0, parse_dates=True, squeeze=True)
fit4 = SimpleExpSmoothing(series4).fit()

# Evaluating the errors
MSE1 = mean_squared_error(fit1.fittedvalues, series1)
print(MSE1)
MSE2 = mean_squared_error(fit2.fittedvalues, series2)
print(MSE2)
MSE3 = mean_squared_error(fit3.fittedvalues, series3)
print(MSE3)
MSE4 = mean_squared_error(fit4.fittedvalues, series4)
print(MSE4)
```

```
# LES
series1 = read_excel('fore.xlsx', sheet_name='K55O', header=0,
                     index_col=0, parse_dates=True, squeeze=True)
fit1 = Holt(series1).fit(optimized=True)
series2 = read_excel('fore.xlsx', sheet_name='JVZ8', header=0,
                     index_col=0, parse_dates=True, squeeze=True)
fit2 = Holt(series2).fit(optimized=True)
series3 = read_excel('fore.xlsx', sheet_name='GMAA', header=0,
                     index_col=0, parse_dates=True, squeeze=True)
fit3 = Holt(series3).fit(optimized=True)
series4 = read_excel('fore.xlsx', sheet_name='GMAK', header=0,
                     index_col=0, parse_dates=True, squeeze=True)
fit4 = Holt(series4).fit(optimized=True)

MSE1 = mean_squared_error(fit1.fittedvalues, series1)
print(MSE1)
MSE2 = mean_squared_error(fit2.fittedvalues, series2)
print(MSE2)
MSE3 = mean_squared_error(fit3.fittedvalues, series3)
print(MSE3)
MSE4 = mean_squared_error(fit4.fittedvalues, series4)
print(MSE4)

# HWA
series1 = read_excel('fore.xlsx', sheet_name='K55O', header=0,
                     index_col=0, parse_dates=True, squeeze=True)
series2 = read_excel('fore.xlsx', sheet_name='JVZ8', header=0,
                     index_col=0, parse_dates=True, squeeze=True)
```

```
series3 = read_excel('fore.xlsx', sheet_name='GMAA', header=0,
                             index_col=0, parse_dates=True, squeeze=True)
series4 = read_excel('fore.xlsx', sheet_name='GMAK', header=0,
                             index_col=0, parse_dates=True, squeeze=True)

fit1 = ExponentialSmoothing(
      series1, seasonal_periods=12, trend='add', seasonal='add').fit()
fit2 = ExponentialSmoothing(
      series2, seasonal_periods=12, trend='add', seasonal='add').fit()
fit3 = ExponentialSmoothing(
      series3, seasonal_periods=12, trend='add', seasonal='add').fit()
fit4 = ExponentialSmoothing(
      series4, seasonal_periods=12, trend='add', seasonal='add').fit()

MSE1 = mean_squared_error(fit1.fittedvalues, series1)
print(MSE1)
MSE2 = mean_squared_error(fit2.fittedvalues, series2)
print(MSE2)
MSE3 = mean_squared_error(fit3.fittedvalues, series3)
print(MSE3)
MSE4 = mean_squared_error(fit4.fittedvalues, series4)
print(MSE4)
# HWM
series1 = read_excel('fore.xlsx', sheet_name='K55O', header=0,
                             index_col=0, parse_dates=True, squeeze=True)
series2 = read_excel('fore.xlsx', sheet_name='JVZ8', header=0,
                             index_col=0, parse_dates=True, squeeze=True)
series3 = read_excel('fore.xlsx', sheet_name='GMAA', header=0,
                             index_col=0, parse_dates=True, squeeze=True)
series4 = read_excel('fore.xlsx', sheet_name='GMAK', header=0,
```

```
                        index_col=0, parse_dates=True, squeeze=True)

fit1 = ExponentialSmoothing(
    series1, seasonal_periods=12, trend='add', seasonal='mul').fit()
fit2 = ExponentialSmoothing(
    series2, seasonal_periods=12, trend='add', seasonal='mul').fit()
fit3 = ExponentialSmoothing(
    series3, seasonal_periods=12, trend='add', seasonal='mul').fit()
fit4 = ExponentialSmoothing(
    series4, seasonal_periods=12, trend='add', seasonal='mul').fit()

MSE1 = mean_squared_error(fit1.fittedvalues, series1)
print(MSE1)
MSE2 = mean_squared_error(fit2.fittedvalues, series2)
print(MSE2)
MSE3 = mean_squared_error(fit3.fittedvalues, series3)
print(MSE3)
MSE4 = mean_squared_error(fit4.fittedvalues, series4)
print(MSE4)

# FORCAST
# K55O
series1 = read_excel('fore.xlsx', sheet_name='K55O', header=0,
                        index_col=0, parse_dates=True, squeeze=True)
fit1 = ExponentialSmoothing(
    series1, seasonal_periods=12, trend='add', seasonal='mul').fit()
fit1.fittedvalues.plot(color='red')
fit1.forecast(12).rename(
    'HW-multiplicative seasonality').plot(color='red', legend=True)
series1.plot(color='black', legend=True)
```

```
pyplot.show()

MSE1 = mean_squared_error(fit1.fittedvalues, series1)
results = pd.DataFrame(index=[r"alpha", r"beta", r"gamma", r"l0", "b0", "MSE"])
params = ['smoothing_level', 'smoothing_trend',
          'smoothing_seasonal', 'initial_level', 'initial_trend']
results["HW model 1"] = [fit1.params[p] for p in params] + [MSE1]
print(results)

residuals1 = fit1.fittedvalues - series1
plot_acf(residuals1, title='Residual ACF of K55O', lags=50)
pyplot.show()

# JVZ8
series2 = read_excel('fore.xlsx', sheet_name='JVZ8', header=0,
                         index_col=0, parse_dates=True, squeeze=True)
fit2 = ExponentialSmoothing(
    series2, seasonal_periods=12, trend='add', seasonal='mul').fit()
fit2.fittedvalues.plot(color='red')
fit2.forecast(12).rename(
    'HW-multiplicative seasonality').plot(color='red', legend=True)
series2.plot(color='black', legend=True)
pyplot.show()

MSE2 = mean_squared_error(fit2.fittedvalues, series2)
results = pd.DataFrame(index=[r"alpha", r"beta", r"gamma", r"l0", "b0", "MSE"])
params = ['smoothing_level', 'smoothing_trend',
          'smoothing_seasonal', 'initial_level', 'initial_trend']
results["HW model 2"] = [fit2.params[p] for p in params] + [MSE2]
print(results)
```

```
residuals2 = fit2.fittedvalues - series2
plot_acf(residuals2, title='Residual ACF of JVZ8', lags=50)
pyplot.show()

# GMAA
series3 = read_excel('fore.xlsx', sheet_name='GMAA', header=0,
                                index_col=0, parse_dates=True, squeeze=True)
fit3 = ExponentialSmoothing(
      series3, seasonal_periods=12, trend='add', seasonal='add').fit()
fit3.fittedvalues.rename(
      'HW-additive seasonality').plot(color='red', legend=True)
series3.plot(color='black', legend=True)
pyplot.show()

MSE3 = mean_squared_error(fit3.fittedvalues, series3)
results = pd.DataFrame(index=[r"alpha", r"beta", r"gamma", r"l0", "b0", "MSE"])
params = ['smoothing_level', 'smoothing_trend',
                'smoothing_seasonal', 'initial_level', 'initial_trend']
results["HW model 2"] = [fit3.params[p] for p in params] + [MSE3]
print(results)

residuals3 = fit3.fittedvalues - series3
plot_acf(residuals3, title='Residual ACF of GMAA', lags=50)
pyplot.show()

# GMAK
series4 = read_excel('fore.xlsx', sheet_name='GMAK', header=0,
                                index_col=0, parse_dates=True, squeeze=True)
fit4 = ExponentialSmoothing(
```

```
    series4, seasonal_periods=12, trend='add', seasonal='mul').fit()
fit4.fittedvalues.plot(color='red')
fit4.forecast(12).rename(
    'HW-multiplicative seasonality').plot(color='red', legend=True)
series4.plot(color='black', legend=True)
pyplot.show()

MSE4 = mean_squared_error(fit4.fittedvalues, series3)
results = pd.DataFrame(index=[r"alpha", r"beta", r"gamma", r"l0", "b0", "MSE"])
params = ['smoothing_level', 'smoothing_trend',
          'smoothing_seasonal', 'initial_level', 'initial_trend']
results["HW model 2"] = [fit4.params[p] for p in params] + [MSE4]
print(results)

residuals4 = fit4.fittedvalues - series4
plot_acf(residuals4, title='Residual ACF of GMAK', lags=50)
pyplot.show()

# Rgression
series = read_excel('reg.xlsx', sheet_name='Sheet5', header=0,
                                squeeze=True, dtype=float)
# Correlation matrix for all the variables, 2 by 2
CorrelationMatrix = series.corr()
print(CorrelationMatrix)

# Model1

series = read_excel('reg.xlsx', sheet_name='Sheet6', header=0,
                                squeeze=True, dtype=float)
```

```
# reading the basic variables
DGMAK = series.DGMAK
DJVZ8 = series.DJVZ8
DGMAA = series.DGMAA
DK55O = series.DK55O

# Regression model(s)
formula = 'DGMAK ~ DK55O + DJVZ8 + DGMAA'

# Ordinary Least Squares (OLS)
results = ols(formula, data=series).fit()
print(results.summary())

# MODEL2

series = read_excel('reg.xlsx', sheet_name='Sheet6', header=0,
                                squeeze=True, dtype=float)

# reading the basic variables
DGMAK = series.DGMAK
DJVZ8 = series.DJVZ8
DGMAA = series.DGMAA
DK55O = series.DK55O

D1 = series.D1
D2 = series.D2
D3 = series.D3
D4 = series.D4
D5 = series.D5
D6 = series.D6
```

```
D7 = series.D7
D8 = series.D8
D9 = series.D9
D10 = series.D10
D11 = series.D11
formula = 'DEOM ~ DK55O + DJVZ8 + DGMAA +D1 + D2 +
D3+D4+D5+D6+D7+D8+D9+D10+D11'
results = ols(formula, data=series).fit()
print(results.summary())

# MODEL3
series = read_excel('reg.xlsx', sheet_name='Sheet6', header=0,
                                squeeze=True, dtype=float)

# reading the basic variables
DGMAK = series.DGMAK
DJVZ8 = series.DJVZ8
DGMAA = series.DGMAA
DK55O = series.DK55O

D1 = series.D1
D2 = series.D2
D3 = series.D3
D4 = series.D4
D5 = series.D5
D6 = series.D6
D7 = series.D7
D8 = series.D8
D9 = series.D9
D10 = series.D10
```

```
D11 = series.D11

time = series.time

formula = 'DGMAK ~ DK55O + DJVZ8 + DGMAA +D1 + D2 +
D3+D4+D5+D6+D7+D8+D9+D10+D11+time'
results = ols(formula, data=series).fit()
print(results.summary())

# MODEL 4
series = read_excel('reg.xlsx', sheet_name='Sheet6', header=0,
                                squeeze=True, dtype=float)

# reading the basic variables
DGMAK = series.DGMAK
DGMAA = series.DGMAA

D2 = series.D2
D4 = series.D4
D5 = series.D5
D6 = series.D6
D7 = series.D7
D11 = series.D11

formula = 'DGMAK ~ DGMAA + D2 +D4+D5+D6+D7+D11'
results = ols(formula, data=series).fit()
print(results.summary())

# FORCAST REGRESSION MODEL
series = read_excel('reg.xlsx', sheet_name='Sheet6', header=0,
```

```
                    squeeze=True, dtype=float)

DGMAK = series.DGMAK
DJVZ8 = series.DJVZ8
DGMAA = series.DGMAA
DK55O = series.DK55O



fit3 = Holt(DK55O).fit(smoothing_level=0.8,
                       smoothing_slope=0.2, optimized=False)
#fit3 = Holt(DK55O).fit(optimized=True)
fcast3 = fit3.forecast(6).rename("Additive 2 damped trend")
fit3.fittedvalues.plot(color='red')
fcast3.plot(color='red', legend=True)
DK55O.plot(color='black', legend=True)
plt.title('Forecast of DK55O with Holt linear method')
plt.show()

fit2 = Holt(DJVZ8).fit(smoothing_level=0.8,
                       smoothing_slope=0.2, optimized=False)
#fit3 = Holt(DK55O).fit(optimized=True)
fcast2 = fit2.forecast(6).rename("Additive 2 damped trend")
fit2.fittedvalues.plot(color='red')
fcast2.plot(color='red', legend=True)
DJVZ8.plot(color='black', legend=True)
plt.title('Forecast of DJVZ8 with Holt linear method')
plt.show()

fit1 = Holt(DGMAA).fit(smoothing_level=0.8,
                       smoothing_slope=0.2, optimized=False)
```

```
#fit3 = Holt(DK55O).fit(optimized=True)
fcast1 = fit1.forecast(6).rename("Additive 2 damped trend")
fit1.fittedvalues.plot(color='red')
fcast1.plot(color='red', legend=True)
DGMAA.plot(color='black', legend=True)
plt.title('Forecast of DGMAA with Holt linear method')
plt.show()


formula = 'DGMAK ~ DK55O + DJVZ8 + DGMAA'

results = ols(formula, data=series).fit()
results.summary()
b0 = results.params.Intercept
b1 = results.params.DGMAA
b2 = results.params.DJVZ8
b3 = results.params.DK55O

# putting the fitted values of the forecasts of AAA, Tto4, and D3to4 in arrays
a1 = np.array(fit1.fittedvalues)
a2 = np.array(fit2.fittedvalues)
a3 = np.array(fit3.fittedvalues)

# Building the fitted part of the forecast of DEOM
F = a1
for i in range(53):
      F[i] = b0 + a1[i]*b1 + a2[i]*b2 + a3[i]*b3

# putting the values of the forecasts of AAA, Tto4, and D3to4 in arrays
v1 = np.array(fcast1)
```

```
v2 = np.array(fcast2)
v3 = np.array(fcast3)

# Building the 6 values of the forecast ahead
E = v1
for i in range(6):
      E[i] = b0 + v1[i]*b1 + v2[i]*b2 + v3[i]*b3



# Joining the fitted values of the forecast and the points ahead
K = np.append(F, E)

DGMAKfull0 = read_excel('reg.xlsx', sheet_name='Sheet7', header=0,
                                squeeze=True, dtype=float)

DGMAKfull = DGMAKfull0.DGMAKfull
values = DGMAKfull[0:53]
print(values)
Error = values - F
MSE = sum(Error**2)*1.0/len(F)

LowerE = DGMAKfull0.LowerE
print(LowerE)
UpperE = DGMAKfull0.UpperE



# ARIMA
series = read_excel('fore.xlsx', sheet_name='JVZ8', header=0,
                            index_col=0, parse_dates=True, squeeze=True)
```

```
# ACF plot on 50 time lags
plot_acf(series, title='ACF of JVZ8', lags=50)

# PACF plot on 50 time lags
plot_pacf(series, title='PACF of JVZ8 time series', lags=50)
pyplot.show()

# FIRST DIFFERENCING
series = read_excel('fore.xlsx', sheet_name='JVZ8',
                                header=0,          index_col=0,          parse_dates=True,
squeeze=True)

X = series.values
diff = list()
for i in range(1, len(X)):
      value = X[i] - X[i - 1]
      diff.append(value)
pyplot.plot(diff)
pyplot.title('Time plot JVZ8 1st difference')

# ACF plot of time series
plot_acf(diff, title='ACF of JVZ8 1st difference', lags=50)

# PACF plot of time series
plot_pacf(diff, title='PACF of JVZ8 1st difference', lags=50)
pyplot.show()

#
series = read_excel('fore.xlsx', sheet_name='ARJVZ8',
                                header=0,          index_col=0,          parse_dates=True,
```

```
squeeze=True)

X = series.values
diff = list()
for i in range(1, len(X)):
	value = X[i] - X[i - 1]
	diff.append(value)
pyplot.plot(diff)
pyplot.title('Time plot truncated JVZ8 1st difference')

# ACF plot of time series
plot_acf(diff, title='ACF of truncated JVZ8 1st difference', lags=50)

# PACF plot of time series
plot_pacf(diff, title='PACF of truncated JVZ8 1st difference', lags=50)
pyplot.show()

# Forcast
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
plt.style.use('fivethirtyeight')

df = read_excel('fore.xlsx', sheet_name='ARJVZ8', header=0,
                    index_col=0, parse_dates=True, squeeze=True)
mod = sm.tsa.statespace.SARIMAX(df, order=(0,1,2),
                                    seasonal_order=(0,0,0,0))
results = mod.fit(disp=False)
print(results.summary())
```

```
results.plot_diagnostics(figsize=(15, 12))
plt.show()

pred = results.get_prediction(start=pd.to_datetime('2005 JAN'), dynamic=False)
pred_ci = pred.conf_int()
print(pred_ci)

ax = df['1996':].plot(label='Original data')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7)

ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.2)
plt.legend()
plt.show()

y_forecasted = pred.predicted_mean
y_truth = df['1992 JAN':]
# Compute the mean square error
mse = ((y_forecasted - y_truth) ** 2).mean()
print('MSE of the forecasts is {}'.format(round(mse, 2)))
```

Appendix B

1. Exponential Smoothing Method

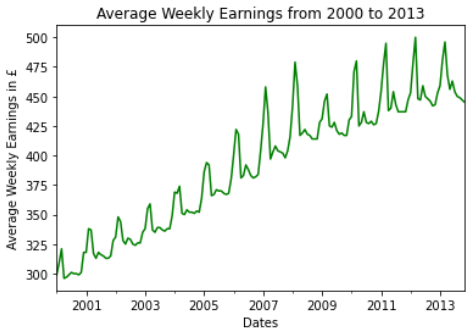

Figure 1: Time Plot of K55O

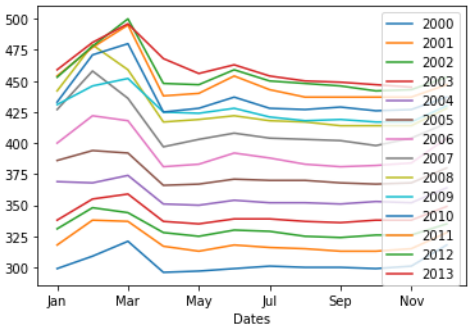

Figure 2: Seasonal Plot of K55O

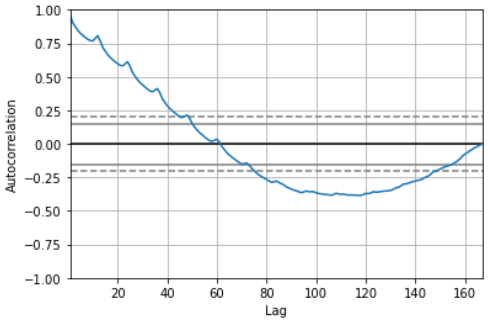

Figure 3: ACF Plot of K55O

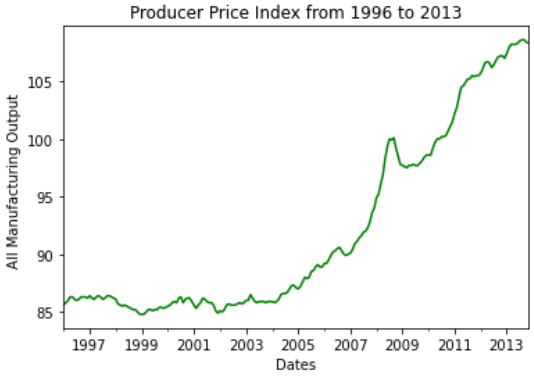

Figure 4: Time Plot of JVZ8

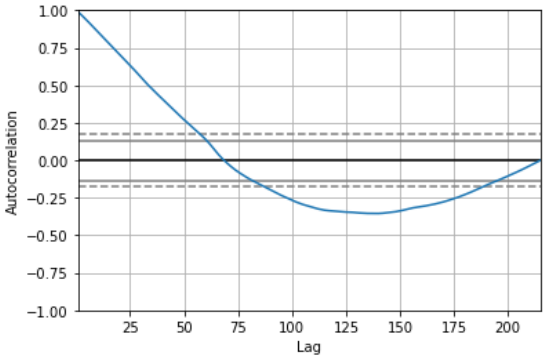

Figure 5: ACF of JVZ8

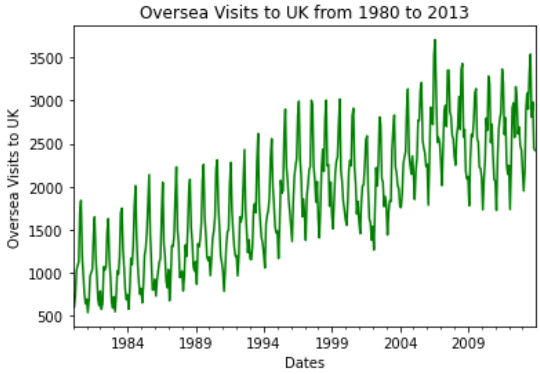

Figure 6: Time Plot of GMAA

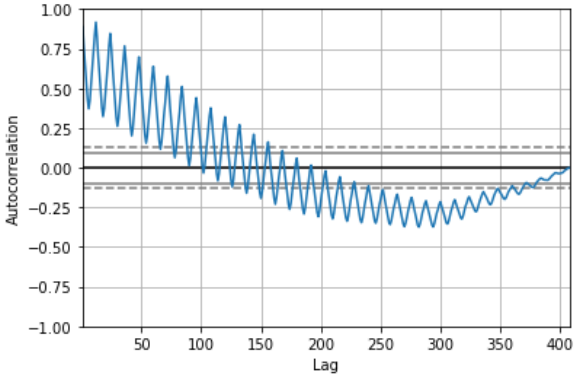

Figure 7: ACF of GMAA

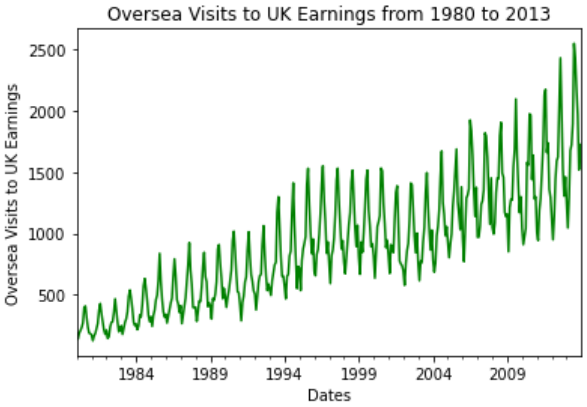

Figure 8 Time Plot of GMAK

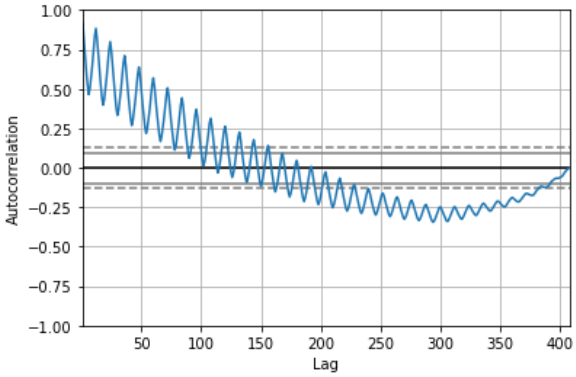

Figure 9 ACF Plot of GMAK

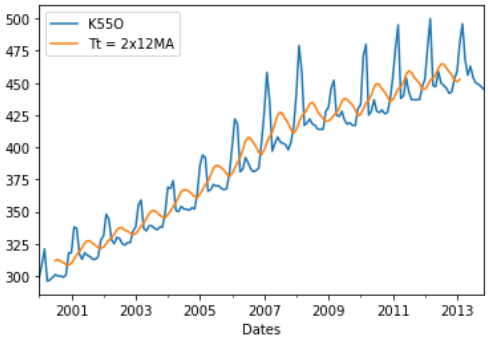

Figure 10: 2×12-MA Plot of K55O

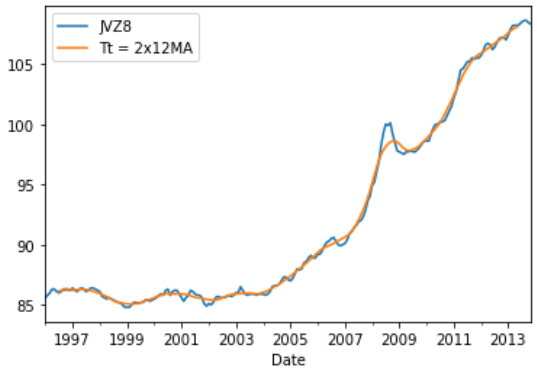

Figure 11: 2×12-MA Plot of JVZ8

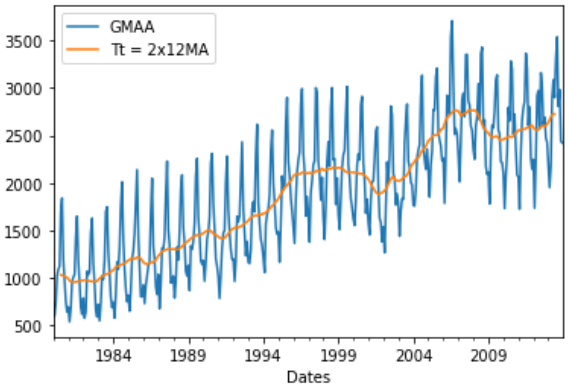

Figure 12: 2×12-MA Plot of GMAA

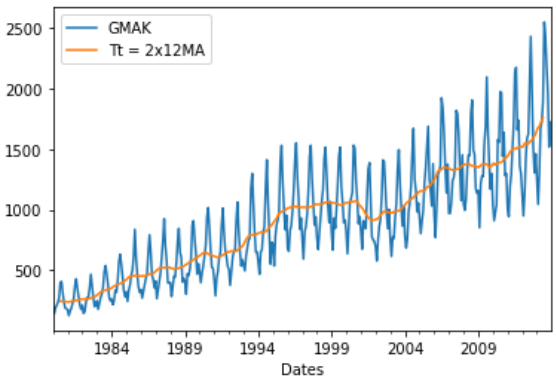

Figure 13: 2×12-MA Plot of GMAK

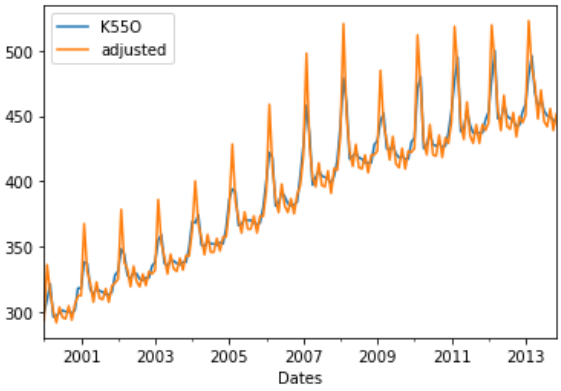

Figure 14: Calendar Adjustment of K55O

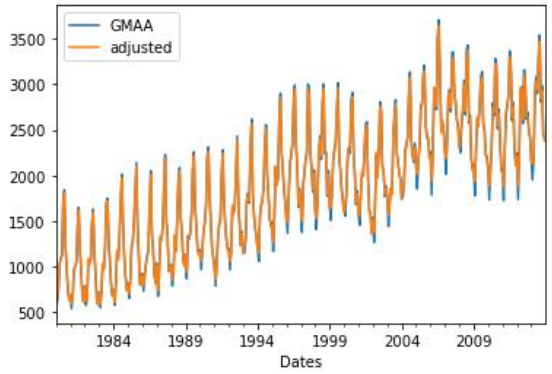

Figure 15: Calendar Adjustment of GMAA

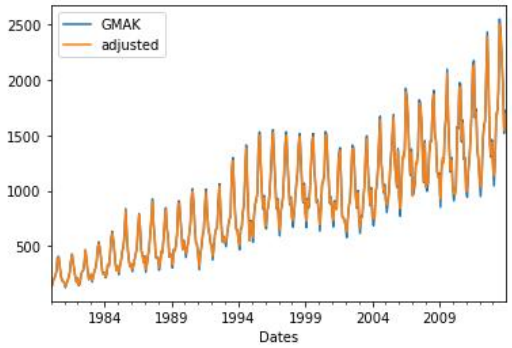

Figure 16: Calendar Adjustment of GMAK

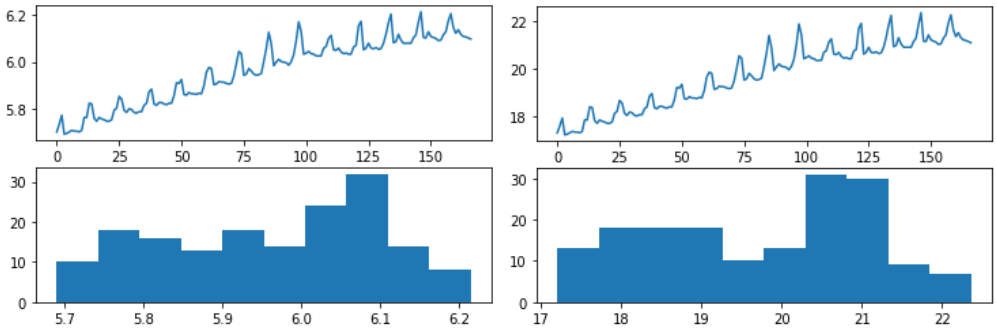

Figure 17 Logarithm Transform and Power Transform of K55O

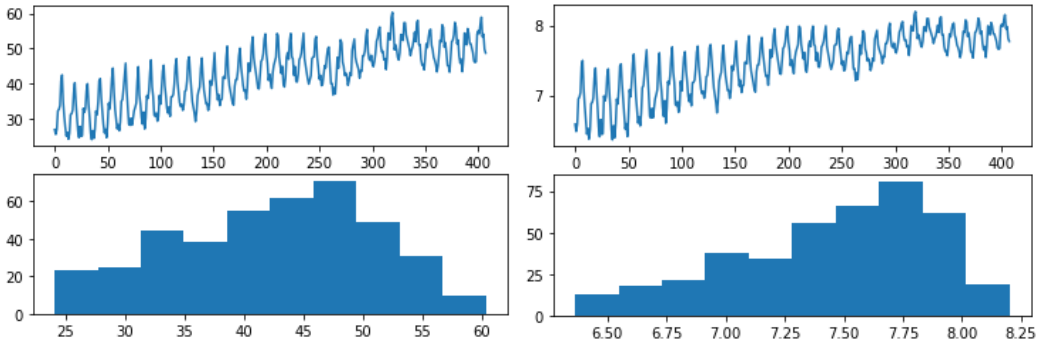

Figure 18 Logarithm Transform and Power Transform of GMAA

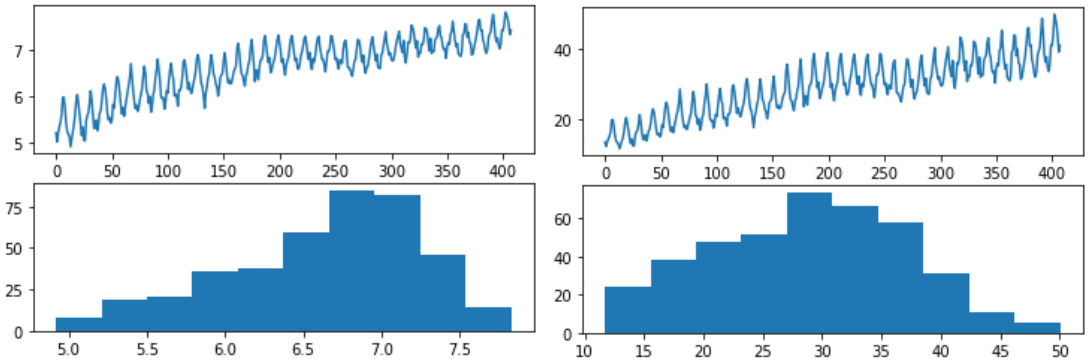

Figure 19 Logarithm Transform and Power Transform of GMAK

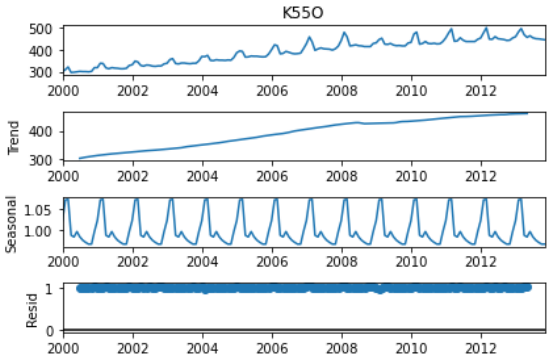

Figure 20 Multiplicative Decomposition of K55O

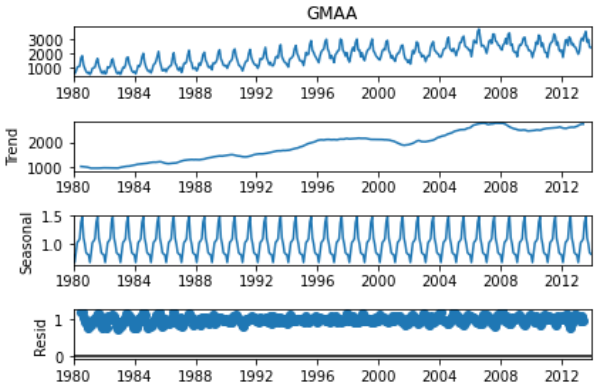

Figure 21 Multiplicative Decomposition of GMMA

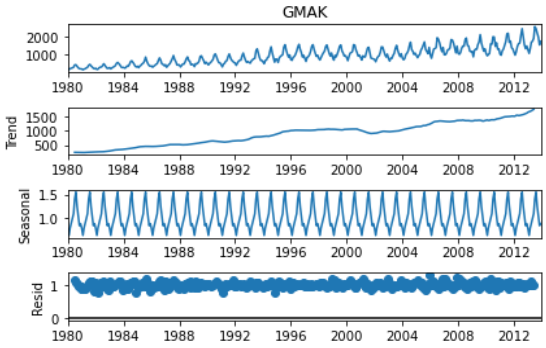

Figure 22 Multiplicative Decomposition of GMAK

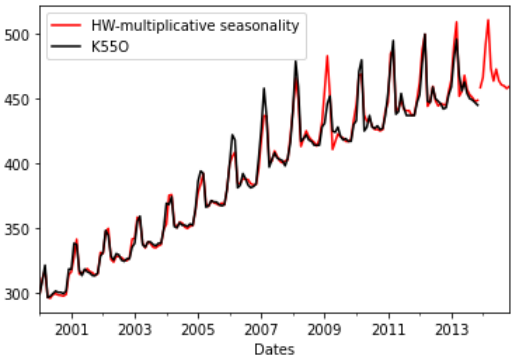

Figure 23: The Forecasting of K55O

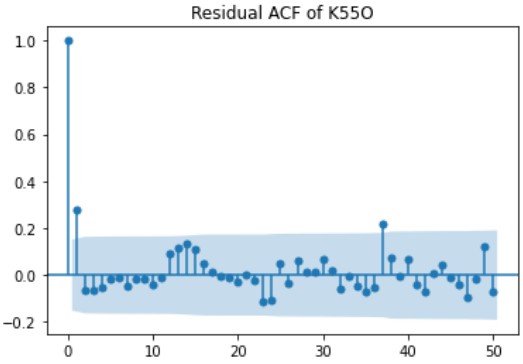

Figure 24: The Residual ACF of HWM Model (K55O)

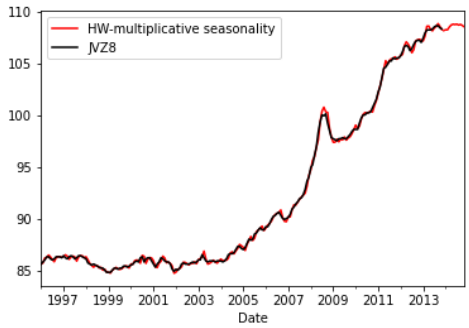

Figure 25: The Forecasting of JVZ8

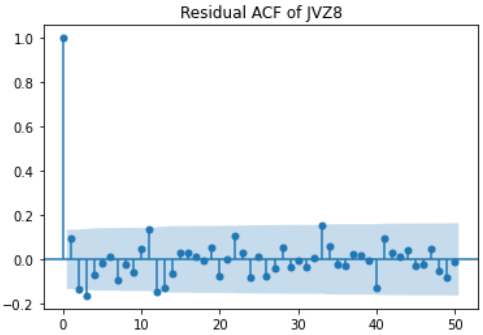

Figure 26: The Residual ACF of HWM Model (JVZ8)

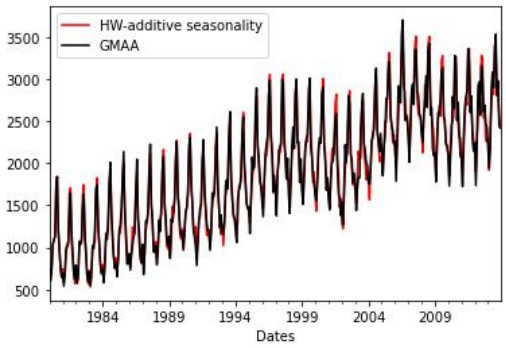

Figure 27: The Forecasting of GMAA

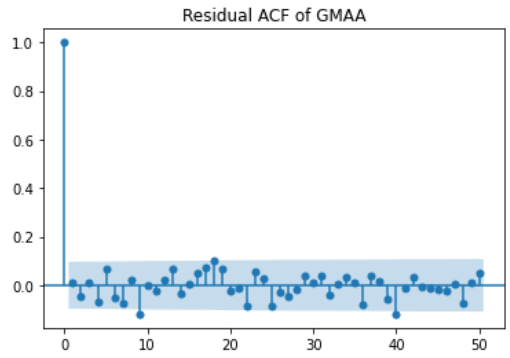

Figure 28 : The Residual ACF of HWA Model (GMAA)

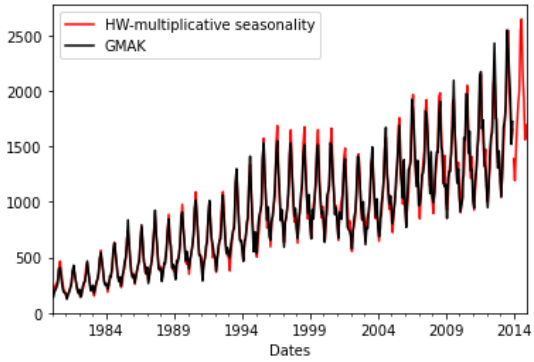

Figure 29: The Forecasting of GMAK

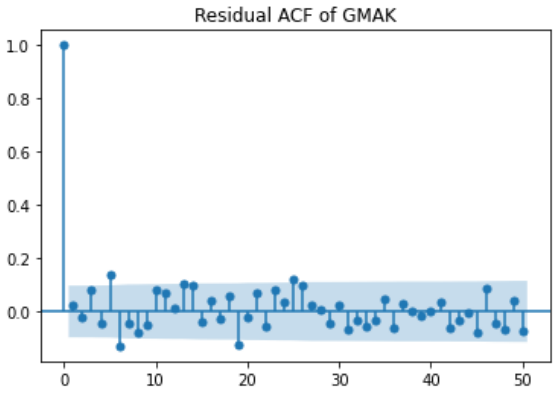

Figure 30: The Residual ACF of HWM Model (GMAK)

2. ARIMA Forecasting

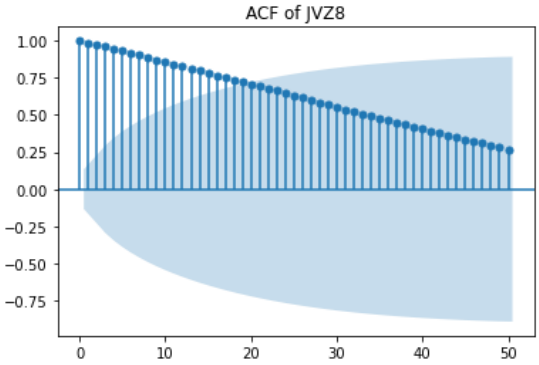

Figure 31: ACF plot of JVZ8

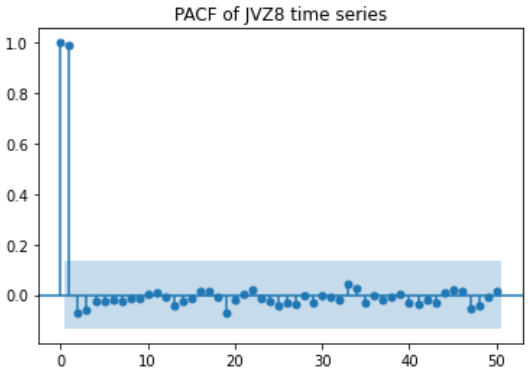

Figure 32: PACF plot of JVZ8

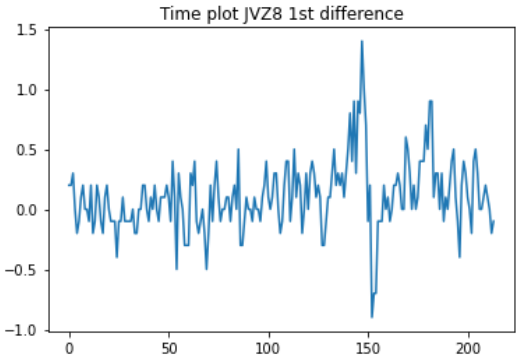

Figure 33: Time plot of 1st Difference

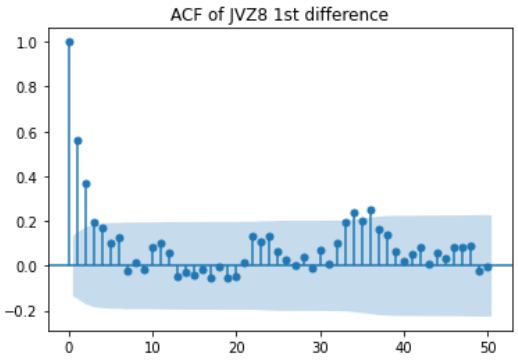

Figure 34 ACF of 1st Difference

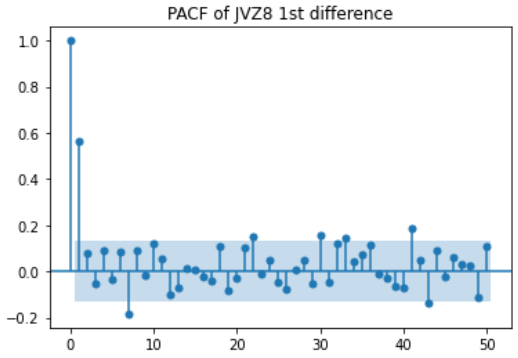

Figure 35 PACF of 1st Difference

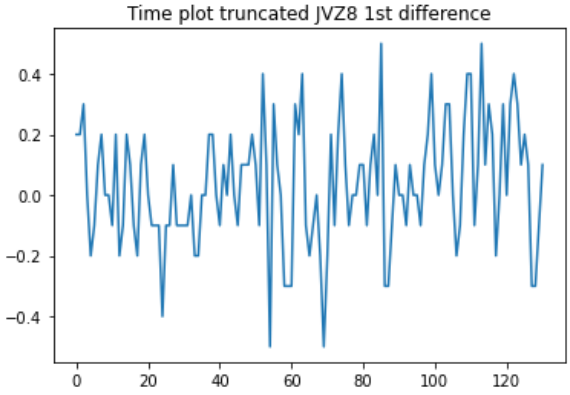

Figure 36 Truncated Time Plot of 1st Difference

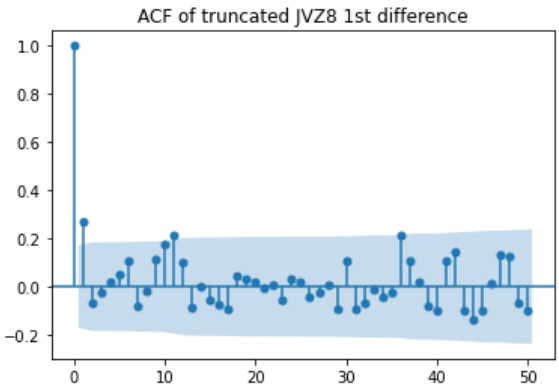

Figure 37 Truncated ACF of 1st Difference

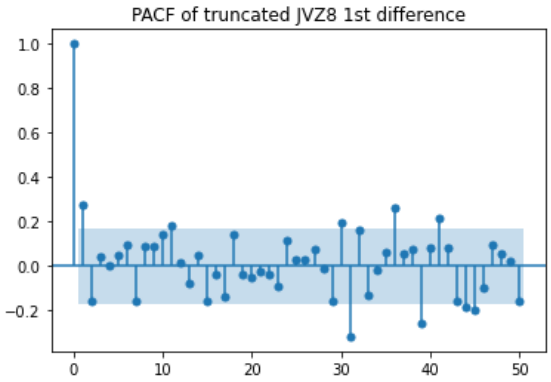

Figure 38 Truncated PACF of 1st Difference

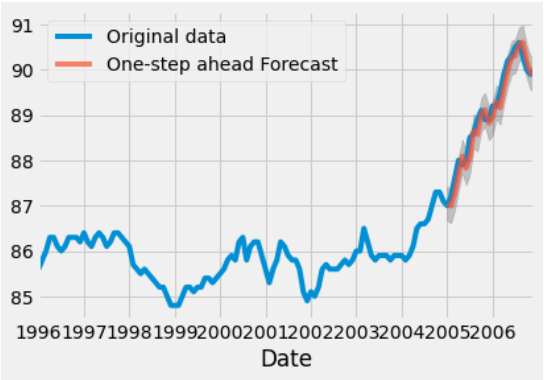

Figure 39: Forecast Graph of ARIMA Model

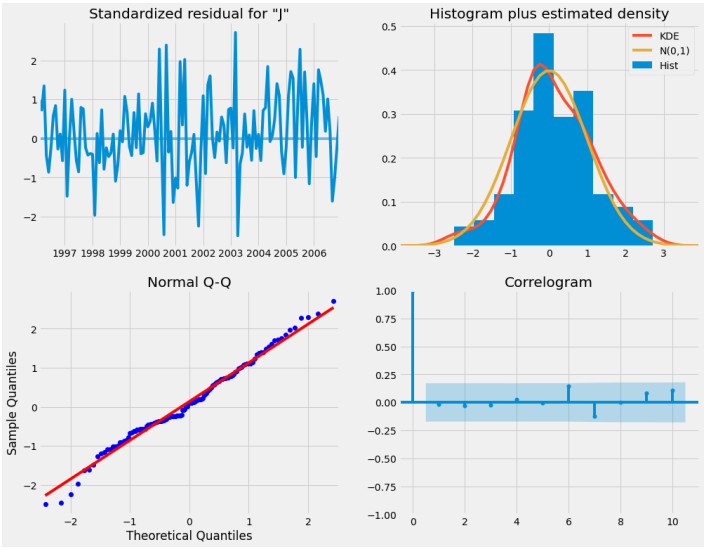

Figure 40: Residuals Plot of ARIMA Model

3. Regression Prediction

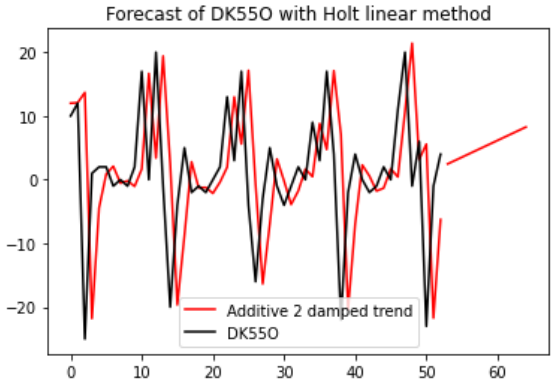

Figure 41: Forecast of K55O with LSE method

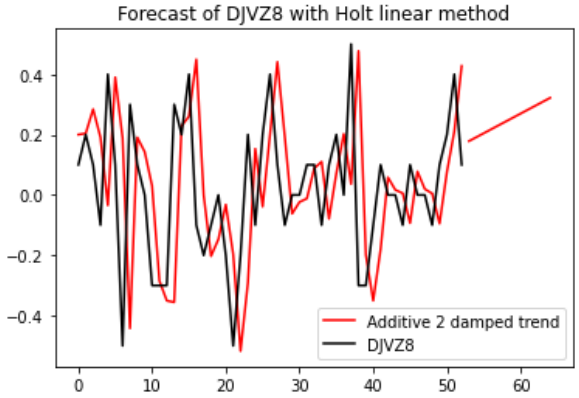

Figure 42: Forecast of JVZ8 with LSE method

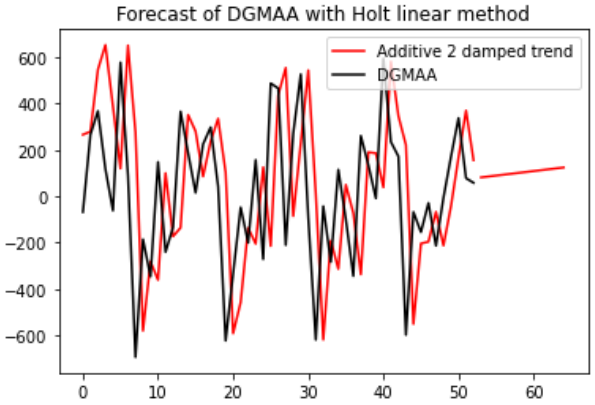

Figure 43: Forecast of GMAA with LSE mothod