NANYANG TECHNOLOGICAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

AI6123: Time Series Analysis
Assignment 2

Name: Syed Anas Majid

Matriculation Number: G2204717J

Email: syed0062@e.ntu.edu.sg

Table of Contents

1. Problem Statement....3
2. Approach....3
3. Exploratory Analysis....3
    3.1. Basic Analysis with Pandas....3
    3.2. Time Plot....4
    3.3. Seasonal Decomposition....4
    3.4. ACF and PACF....5
4. Data Transformation....6
    4.1. Box-Cox transformation....6
    4.2. Differencing for Trend and Seasonal Components....7
    4.3. Train-Test Split....9
5. Models and Forecasting....9
    5.1. SARIMA Models....9
        5.1.1. ACF and PACF....9
        5.1.2. SARIMA Fitting....11
        5.1.3. Diagnostics....13
        5.1.4. Forecasting Result....14
    5.2. Holt Winters' Model....14
        5.2.1. Motivation....14
        5.2.2. Implementation....14
        5.2.3. Holt Winters' Model Fitting....16
        5.2.4. Forecasting Result....16
6. Conclusion....17
7. References....18

1. Problem Statement

The first data is for monthly anti-diabetic drug sales in Australia from 1992 to 2008. Total monthly scripts for pharmaceutical products falling under ATC code A10, as recorded by the Australian Health Insurance Commission.

Please build a good model to predict the drug sales.

Aim: Demonstrate that you understand how to fit an appropriate ARIMA model in the real situations.
Expectation:
State the background of the time series.
How do you come up with an initial model (by looking at ACF, PACF, differencing, transformation)?
How do you improve your model by considering ACFs in the residuals and using ARIMA models?
How do you arrive at the final model ?
Forecast the time series for three years (or three months) ahead.
Include the R code/python.

2. Approach

This project uses Python to conduct exploratory analysis and forecast monthly anti-diabetic drug sales in Australia. First, exploratory analysis is conducted on the data provided using the *pandas* library. Subsequently, a SARIMA model and a Holt Winters' model is used to fit the data and predict drug sales for a period of 3 years, from the last entry in the data. Finally, the accuracies of each model are compared. The libraries used in this project are: *pandas, numpy, matplotlib, scipy, sklearn, pmdarima, statsmodels.*

3. Exploratory Analysis

3.1. Basic Analysis with Pandas

The *drug.txt* file contains 204 entries, with no null values in the data. The data was first loaded into a *pandas dataframe*, and basic information was gathered using *pandas* library functions. This is shown below:

```
27    # read data and format
28    df = pd.read_csv('drug.txt')
29    df['date'] = pd.to_datetime(df['date'], format='%Y-%m-%d')
30    df = df.set_index(df['date'])
31    df = df.drop(columns=['date'])
32    print(df.head())        # visualize first 5 rows
33    print(df.tail())        # visualize last 5 rows
34    print(len(df.index))    # Get number of entries
35    print(df.dtypes)        # See datatype of entries
36    print(df[df.isnull().any(axis=1)])
```

Figure 1: Code used to load data and perform simple exploration.

	value
date	
1991-07-01	3.526591
1991-08-01	3.180891
1991-09-01	3.252221
1991-10-01	3.611003
1991-11-01	3.565869

Figure 2: First 5 entries of data in the *dataframe*.

3.2. Time Plot

The time plot of the original data is shown below:

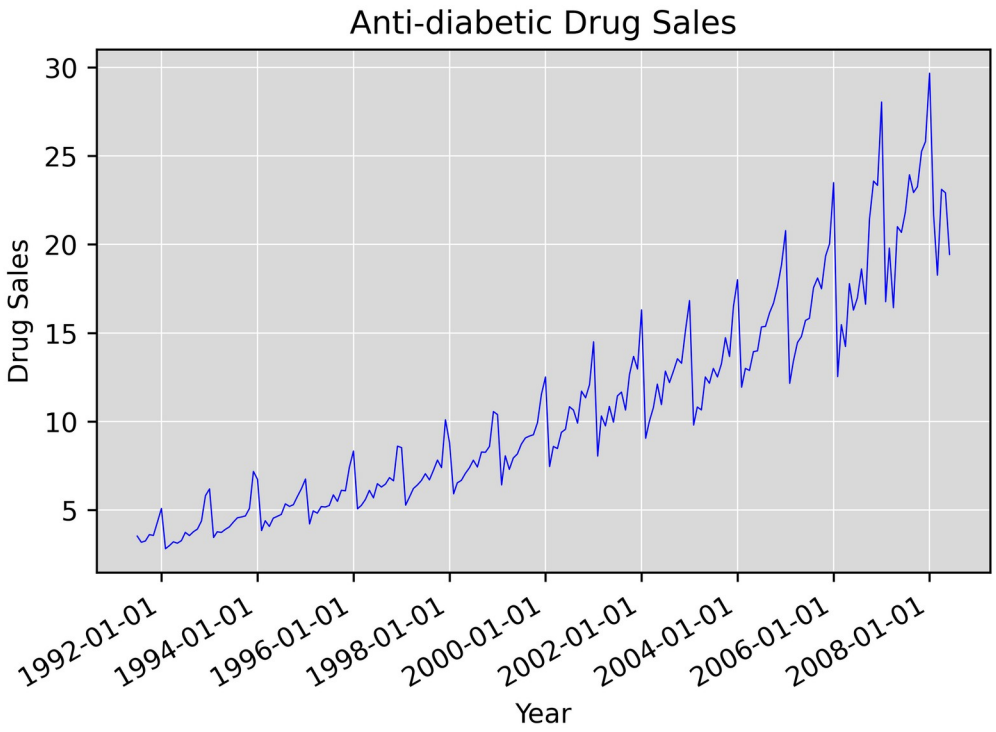


Figure 3: Time plot of the original data.

From the time plot, we observe that the data has an upward trend component, and a seasonal component.

3.3. Seasonal Decomposition

Seasonal decomposition was performed using the *statsmodels* function *seasonal_decompose()*. The figure below shows the plot obtained:

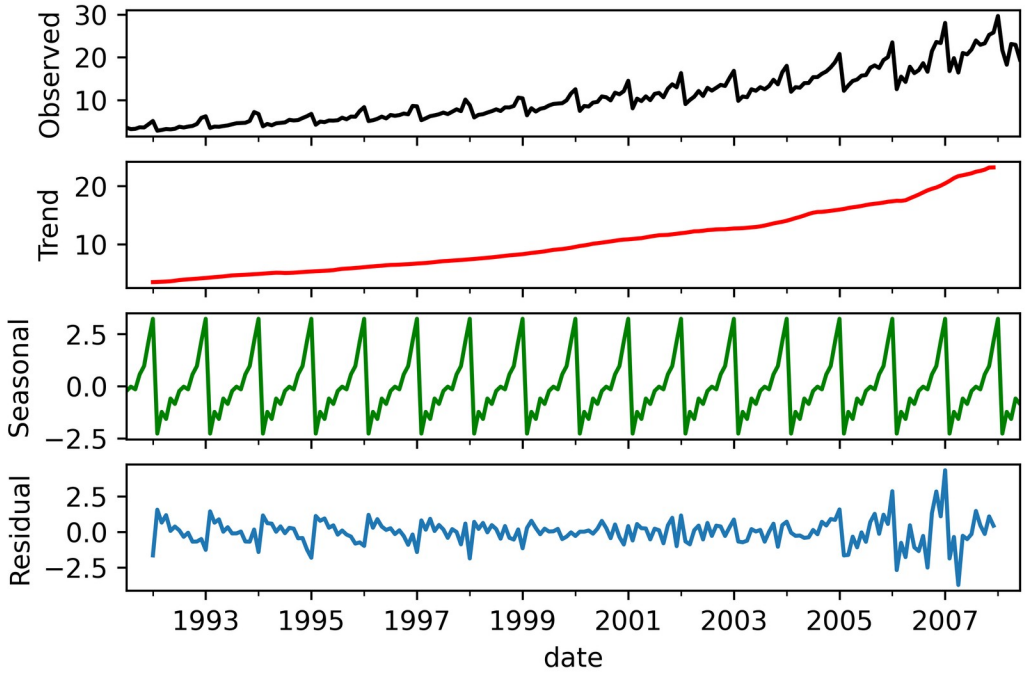


Figure 4: Seasonal decomposition result.

From the above plot, we clearly see that there is a trend, and a yearly seasonal pattern.

3.4. ACF and PACF

The plots below show the ACF and PACF of the original data, which are obtained using *statsmodels* *plot_acf()* and *plot_pacf():*

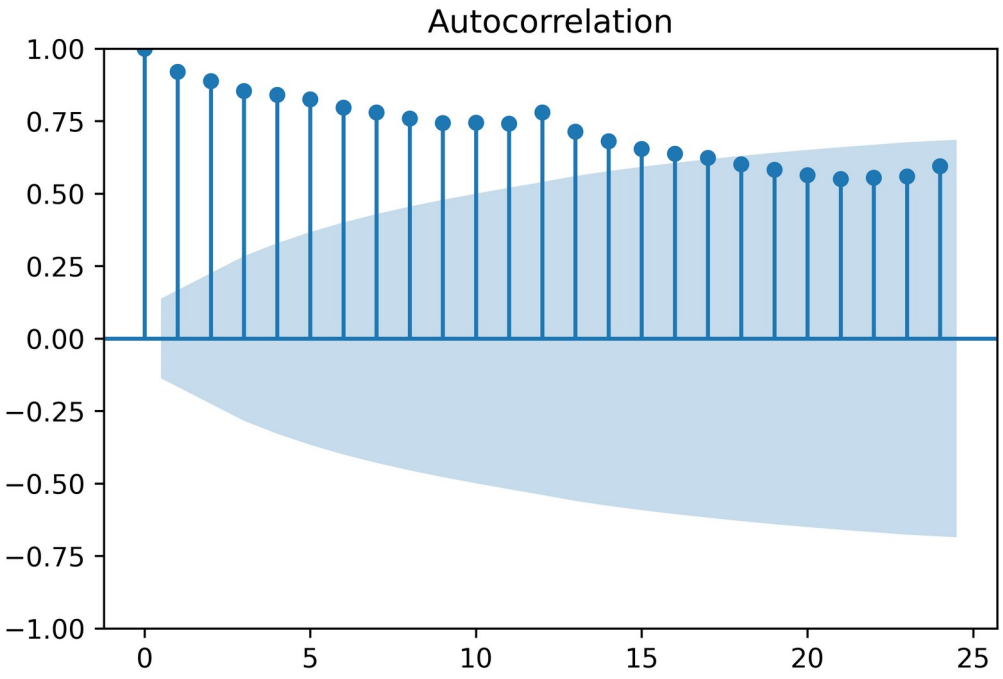


Figure 5: ACF of original data.

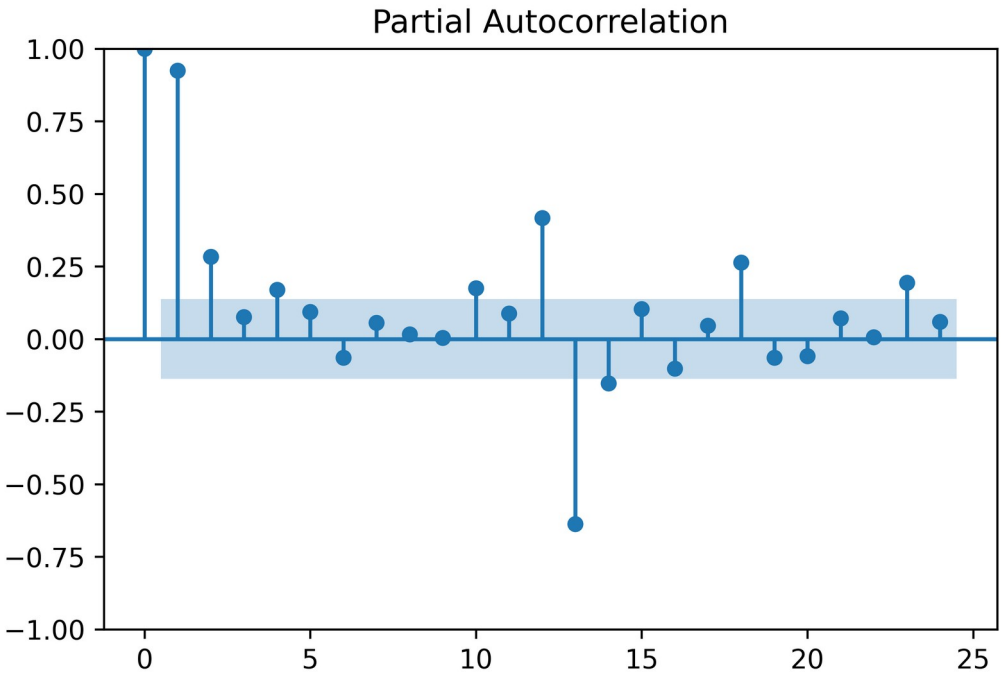


Figure 6: PACF of original data.

From the ACF plot, we observe that the ACF dies down slowly, indicating that the data is non-stationary.

4. Data Transformation

4.1. Box-Cox transformation

From the original time plot (Figure 3), we observe that the data has increasing variance for each season. The Box-Cox transformation was therefore chosen to remedy this and stabilize the variance. For implementation, the *boxcox()* function from the *scipy* library was used. Using the function, the lambda value is chosen by maximizing the log-likelihood function. This value is found to be **0.061505584870954325**. The figure below shows the time plot after the Box-Cox transformation is applied:

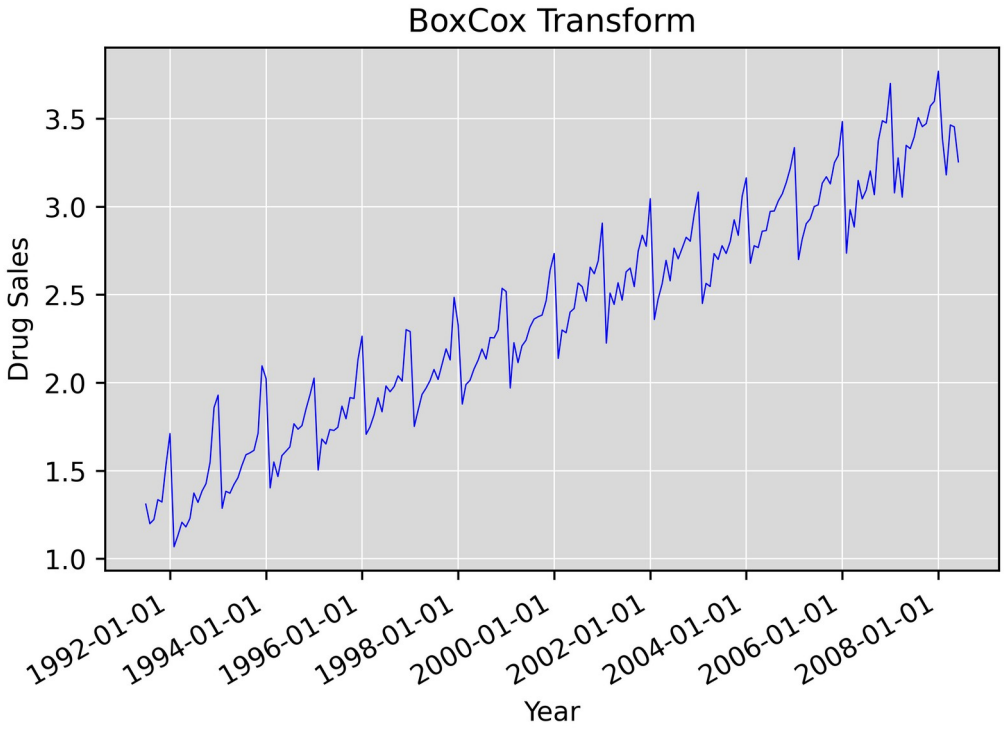


Figure 7: Time plot after Box-Cox Transformation.

From the plot above, we clearly see that the originally increasing variance for each season is largely minimized.

4.2. Differencing for Trend and Seasonal Components

From the original time plot (Figure 3), we observe that the data contains both trend and seasonal components.

Firstly, to remove the trend component, one-time differencing is applied to the data. The general equation for one-time differencing is as follows:

$$Z_t = X_t - X_{t-1}$$

For the implementation, the *pandas* function *diff()* is used. The time plot after one-time differencing for trend is shown below:

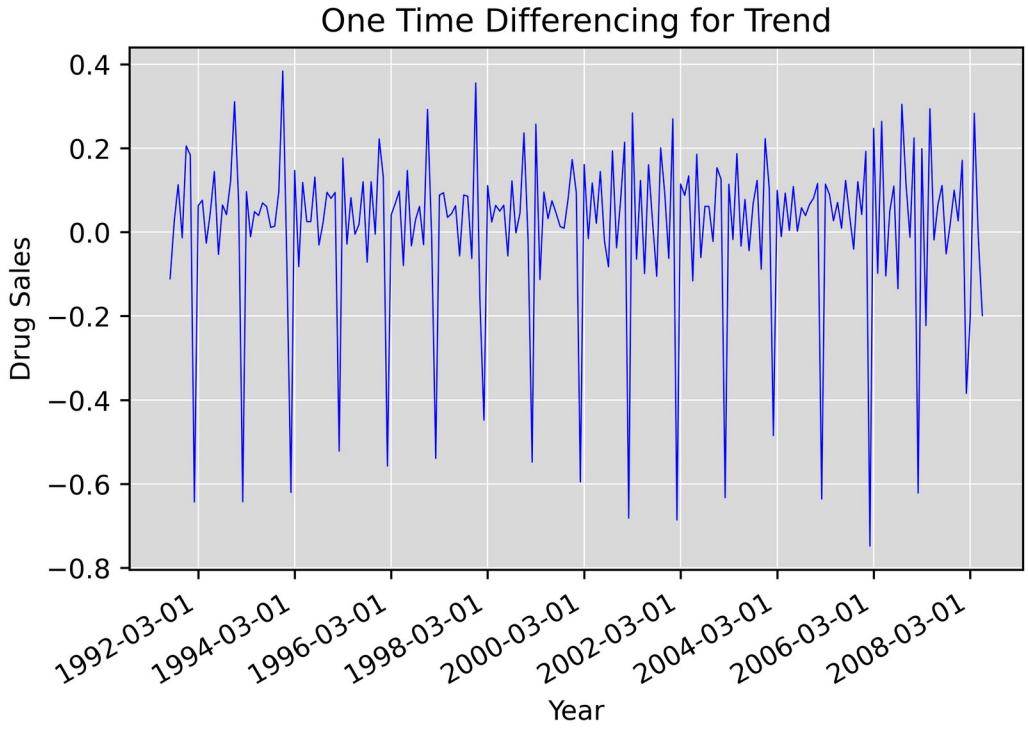


Figure 8: Time plot after one-time differencing for trend.

Secondly, to handle the seasonal component, seasonal differencing is also applied. The general formula for differencing is modified as follows to achieve this:

$$Z_t = X_t - X_{t-12}$$

For the implementation, the same *diff()* function was used, but with *periods=12*. The following figure shows the resulting time plot:

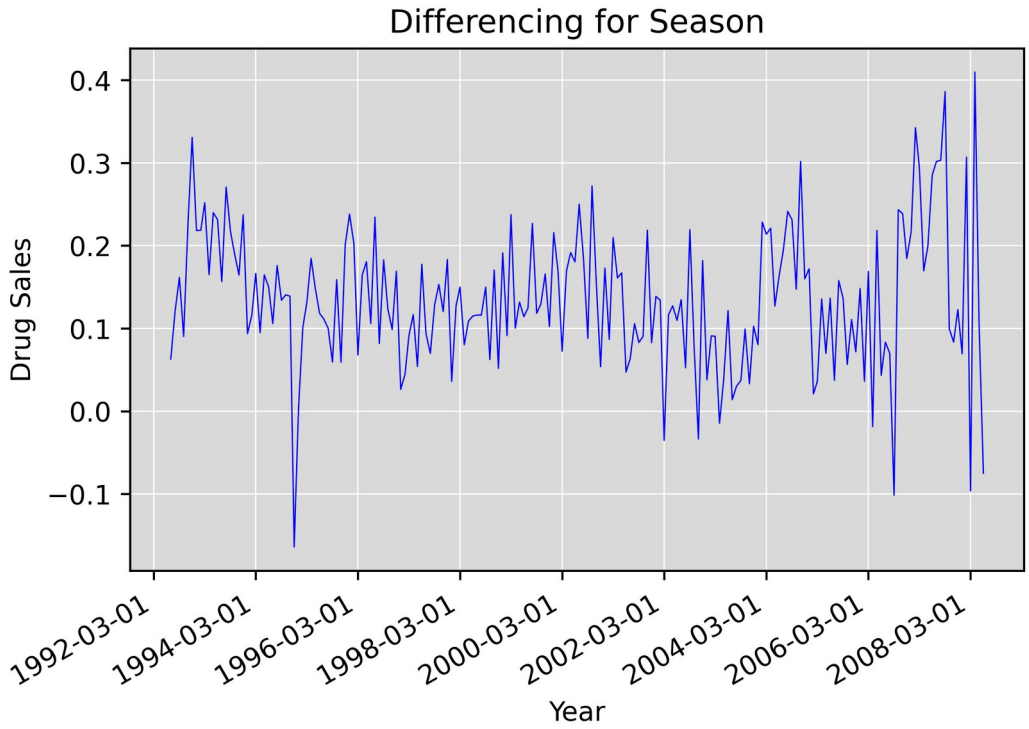


Figure 9: Time plot after differencing for seasonality.

4.3. Train-Test Split

The dataset was split into train and test sets, with a ratio of 80% of samples for train, and 20% for test. This gives us 163 training datapoints and 41 test datapoints. In the implementation, the *train_test_split()* function was used from the *scikit-learn* library. Two other *dataframes* containing the train and test samples were created for ease of manipulation during model training.

5. Models and Forecasting

5.1. SARIMA Models

5.1.1. ACF and PACF

The figures below show the ACF and PACF of the data, after all the transformations stated in Section 4 have been applied:

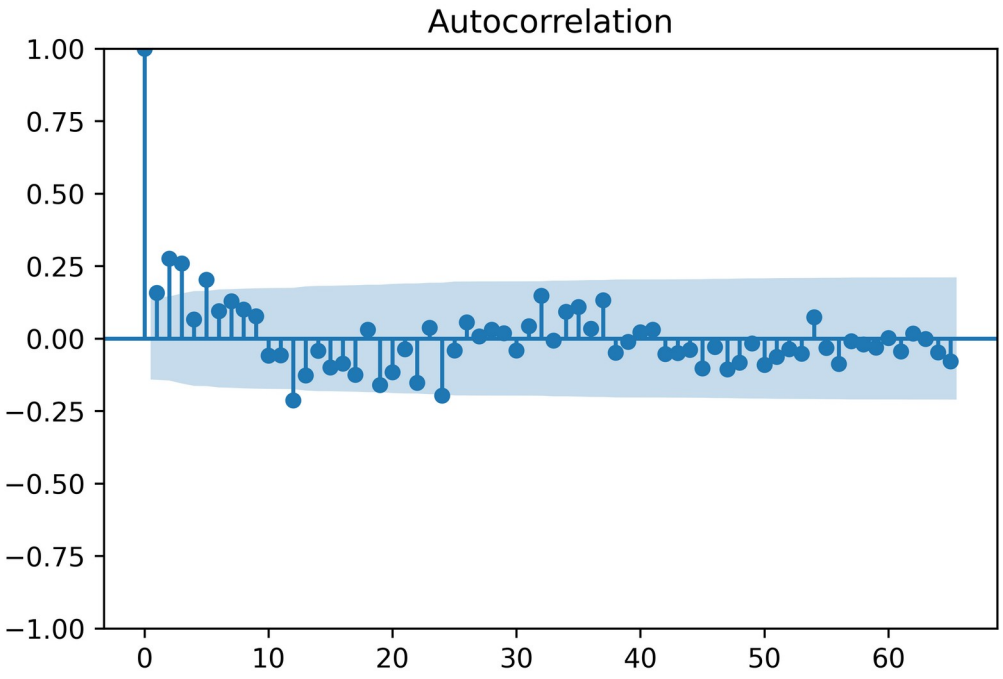


Figure 10: ACF of transformed data.

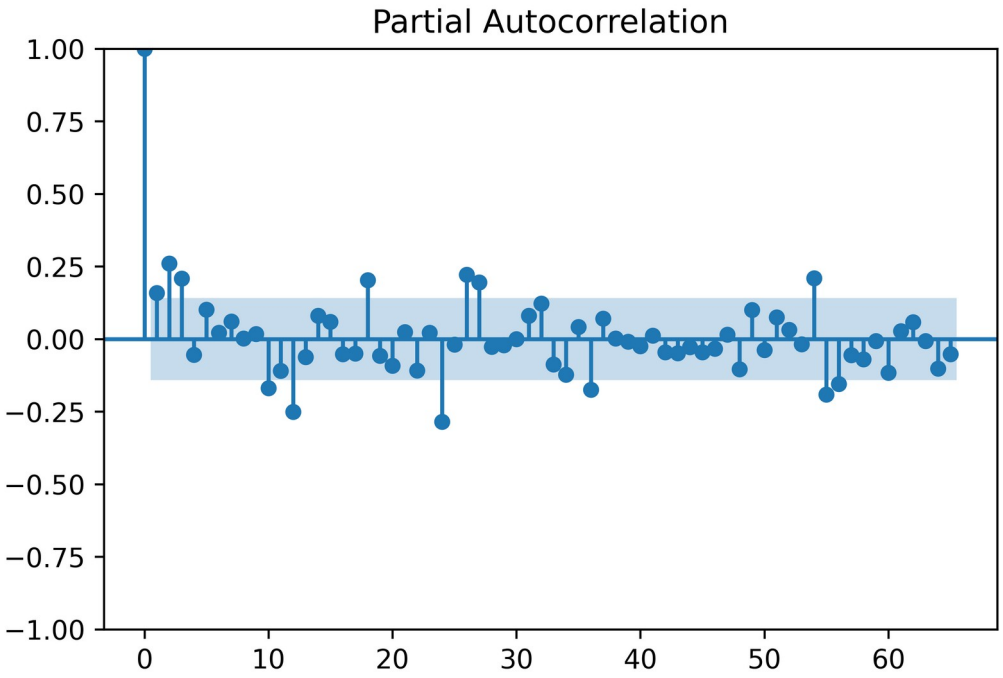


Figure 11: PACF of transformed data

From the above figures, we can choose the values of p, q, P, Q, d and D for the SARIMA models. These values are determined to be 5, 4, 0, 2, 1, 1 respectively.

For the implementation, of the SARIMA model however, the *auto_arima()* function from the *pmdarima* library was chosen. A description of the function from the documentation is as follows:

Auto-ARIMA works by conducting differencing tests (i.e., Kwiatkowski–Phillips–Schmidt–Shin, Augmented Dickey-Fuller or Phillips–Perron) to determine the order of differencing, d, and then fitting models within ranges of defined start_p, max_p, start_q, max_q ranges. If the seasonal optional is enabled, auto-ARIMA also seeks to identify the optimal P and Q hyper- parameters after conducting the Canova-Hansen to determine the optimal order of seasonal differencing, D. [1]

This means that the optimal values for the hyperparameters are chosen by the function after conducting differencing tests, and fitting multiple models within the *max* and *start* ranges defined, to identify the model with the best performance. The model chosen will be discussed in the next sub-section.

5.1.2. SARIMA Fitting

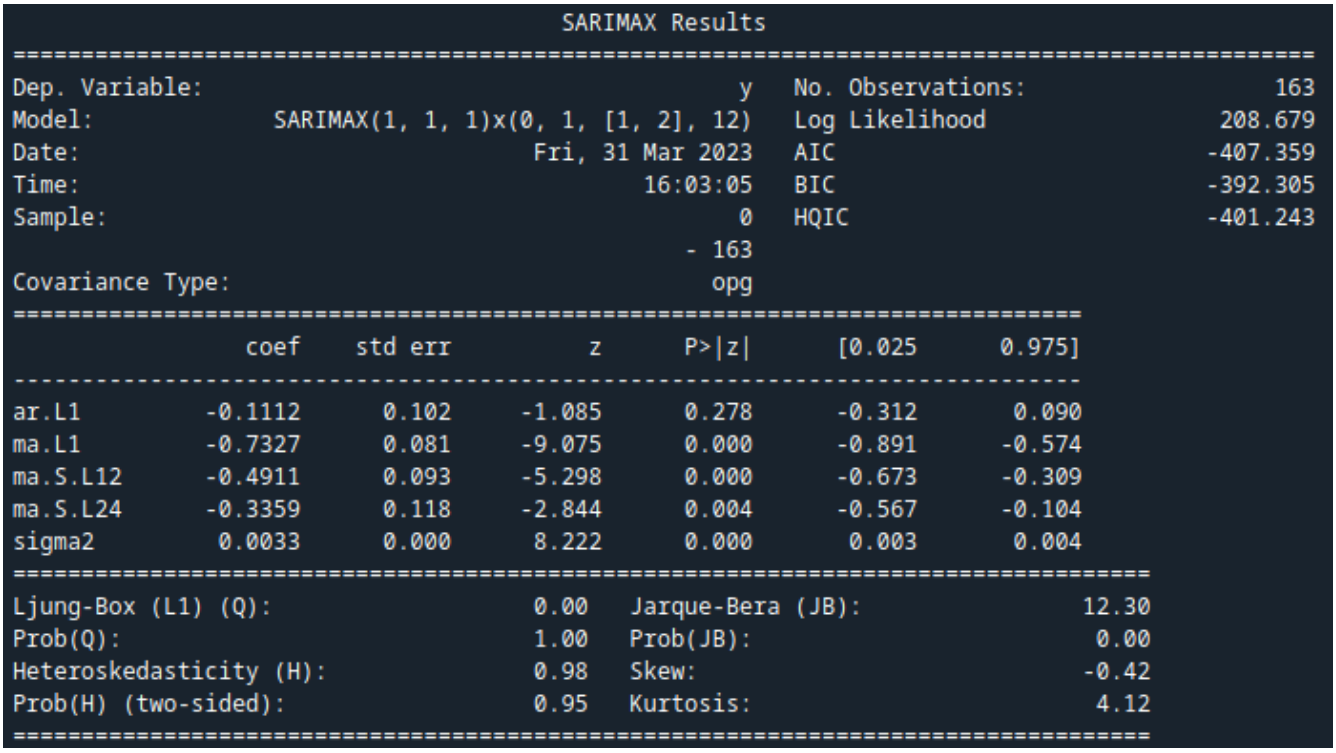
The figure below shows the code used to fit the SARIMA models, with the *max_p, max_q* and *start_P* parameters set as described in the previous sub-section:

```
sarima = pm.auto_arima(train["box_cox_value"], start_p=1, start_q=1,
                       test='adf',
                       max_p=5, max_q=4,
                       m=12, # 12 is the frequncy of the cycle
                       start_P=0,
                       seasonal=True, # set to seasonal arima
                       d=1, # apply one time differencing for seasonality
                       D=1,
                       trace=False,
                       error_action='warn',
                       suppress_warnings=True,
                       stepwise=True)
```

Figure 12: Code to fit SARIMA models.

The parameter *seasonal* is set to *true,* to allow for SARIMA models (and not ARIMA models) to be used. The frequency of the seasonal cycle is 12 (yearly, as described in Section 3), and therefore *m=12*.

After the model is fit, a summary of the chosen model's parameters is shown below:



SARIMAX Results

Dep. Variable:	y	No. Observations:	163
Model:	SARIMAX(1, 1, 1)x(0, 1, [1, 2], 12)	Log Likelihood	208.679
Date:	Fri, 31 Mar 2023	AIC	-407.359
Time:	16:03:05	BIC	-392.305
Sample:	0	HQIC	-401.243
	- 163		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.1112	0.102	-1.085	0.278	-0.312	0.090
ma.L1	-0.7327	0.081	-9.075	0.000	-0.891	-0.574
ma.S.L12	-0.4911	0.093	-5.298	0.000	-0.673	-0.309
ma.S.L24	-0.3359	0.118	-2.844	0.004	-0.567	-0.104
sigma2	0.0033	0.000	8.222	0.000	0.003	0.004

Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	12.30
Prob(Q):	1.00	Prob(JB):	0.00
Heteroskedasticity (H):	0.98	Skew:	-0.42
Prob(H) (two-sided):	0.95	Kurtosis:	4.12

Figure 13: Summary of chosen SARIMA model.

The table below summarizes the model parameters that were chosen:

Parameter	Value
p	1
d	1
q	1
P	0
D	1
Q	[1,2]
m	12

Table 1: SARIMA model parameters.

5.1.3. Diagnostics

For diagnostic checks, the *plot_diagnostics()* function from the *pmdarima* library was used. The following figure shows the diagnostic output:

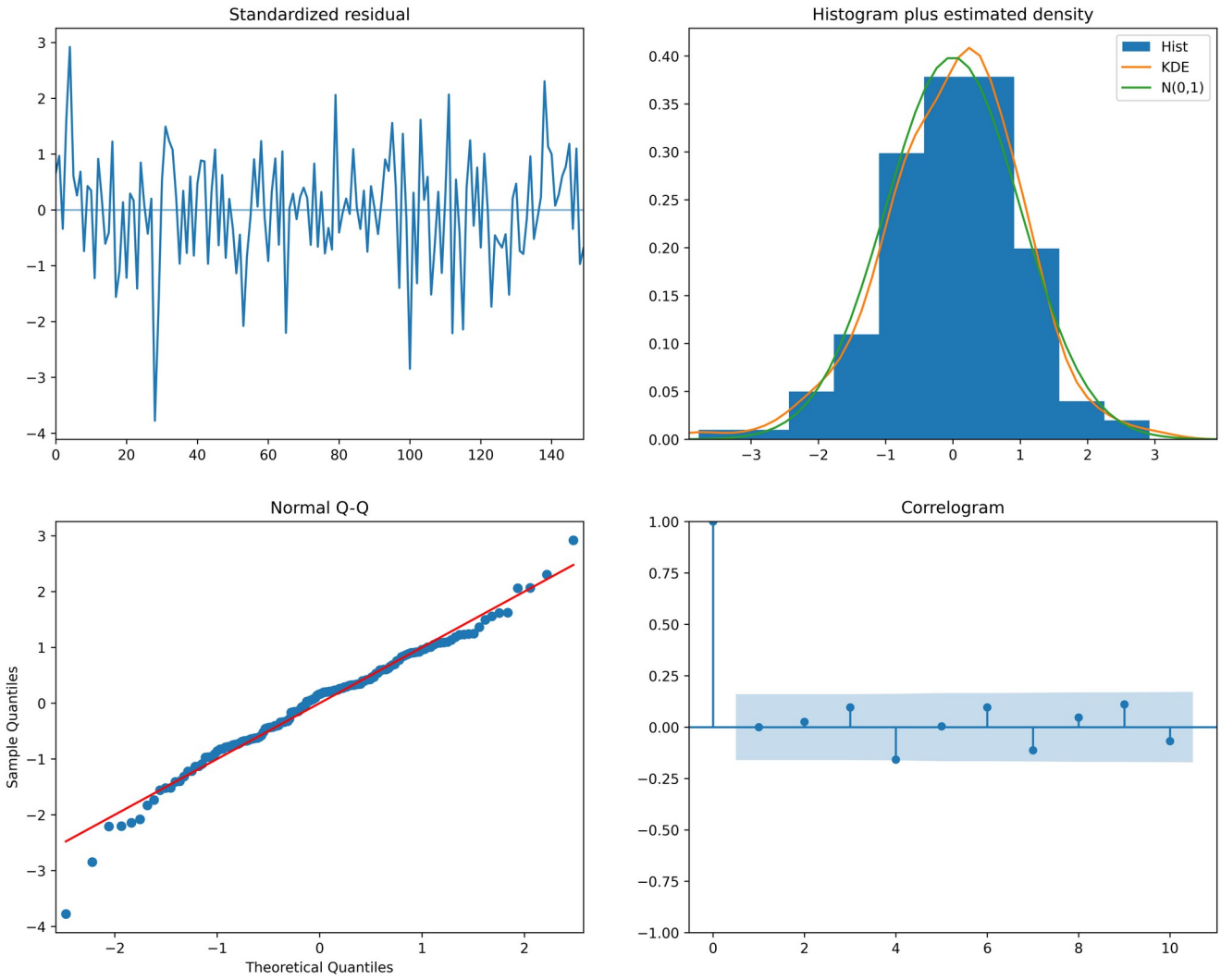


Figure 14: SARIMA model diagnostic output.

From the plots above, we make the following observations:

1. **Standardized residual:** There are no obvious patterns in the residuals, with values having a mean of zero and having a uniform variance.

2. **Histogram plus KDE estimate:** The KDE curve is very similar to the normal distribution (N(0,1)).

3. **Normal Q-Q:** Most of the data points lie on the straight line.

4. **Correlogram:** 95% of correlations for lag greater than zero are not significant. The grey area is the confidence band, the values within them are not statistically significant. In our case, there are a few values outside of this area, and therefore we may need to add more predictors to make the model more accurate**.**

Hence, we conclude that the model is adequate for our forecasting task.

5.1.4. Forecasting Result

The trained model was used for predictions for the period of the test set and for a period of three years after the last entry in the dataset. The figure below shows the forecasting result:

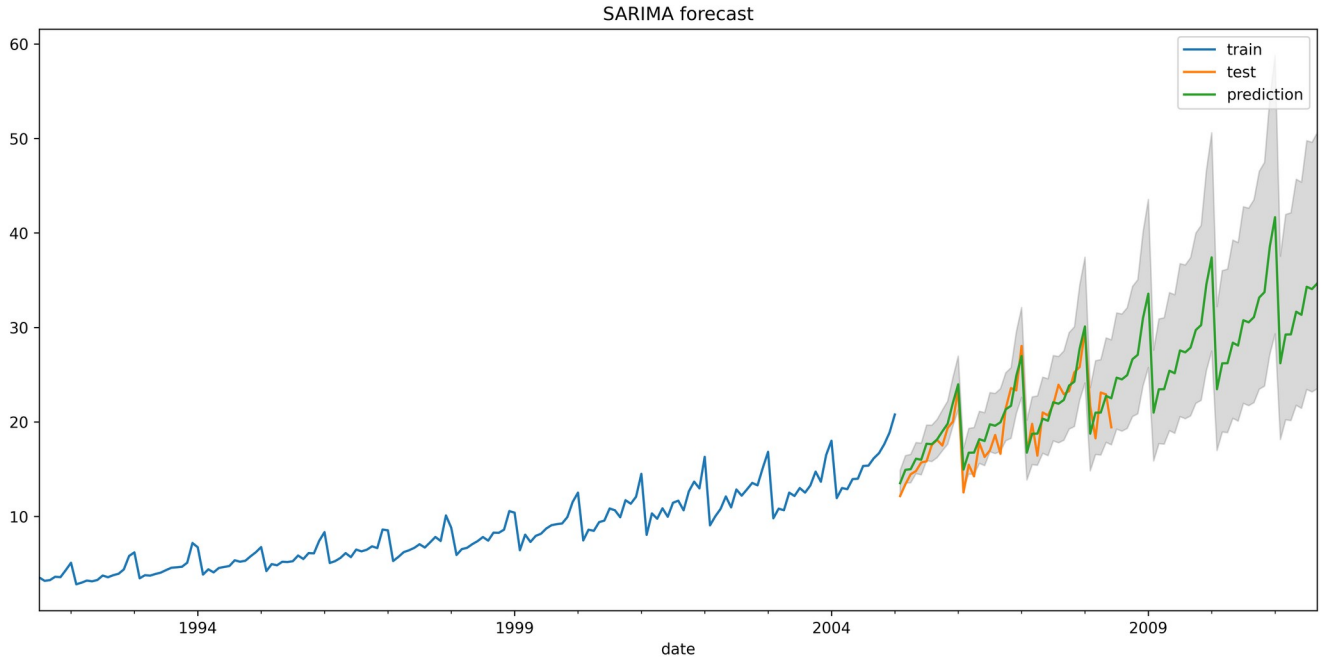


Figure 15: Forecasting result using the trained SARIMA model.

The RMSE was calculated against the test set, and the result was found to be **1.614**.

5.2. Holt Winters' Model

5.2.1. Motivation

Holt Winters' method models three aspects of time series behaviour: a typical value (average), a slope (trend) over time, and a cyclical repeating pattern (seasonality). Since the exploratory analysis in Section 1 showed that this data contained both trend and seasonality components, the Holt Winters' model was also tested for this forecasting task.

5.2.2. Implementation

Holt Winters' model was was implemented in this project using the *holtwinters.SimpleExpSmoothing()* and *holtwinters.ExponentialSmoothing()* functions from the *statsmodels* library.

First, single exponential smoothing was applied on the data, and the result of this is shown in the figure below:

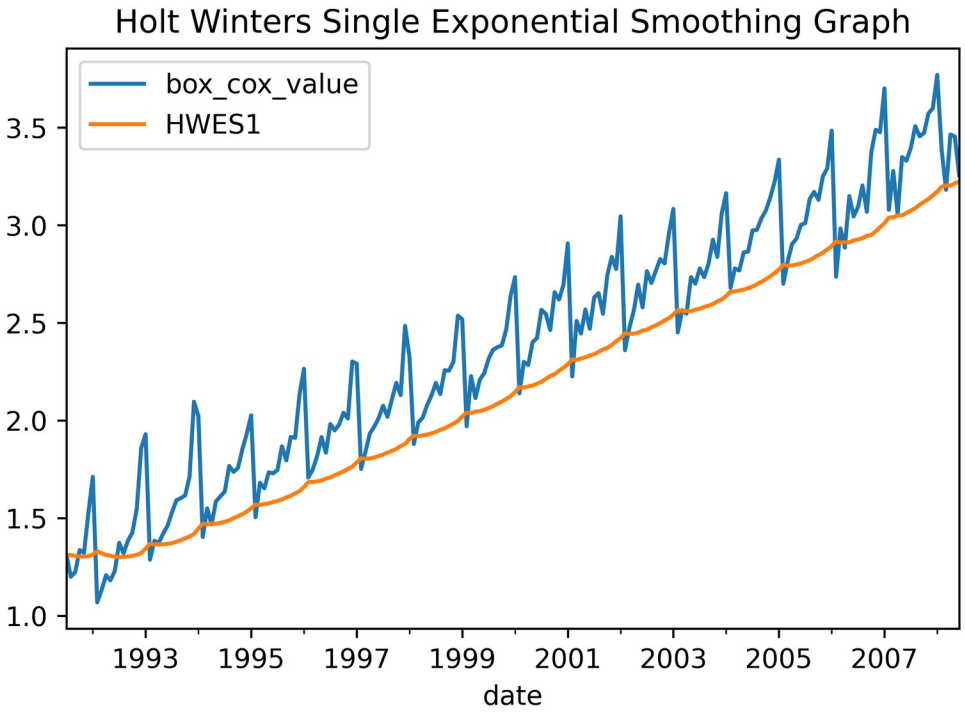


Figure 15: Single exponential smoothing using Holt Winters'.

Then, double exponential smoothing was applied, with both additive and multiplicative trends. The figure below shows the result of this:

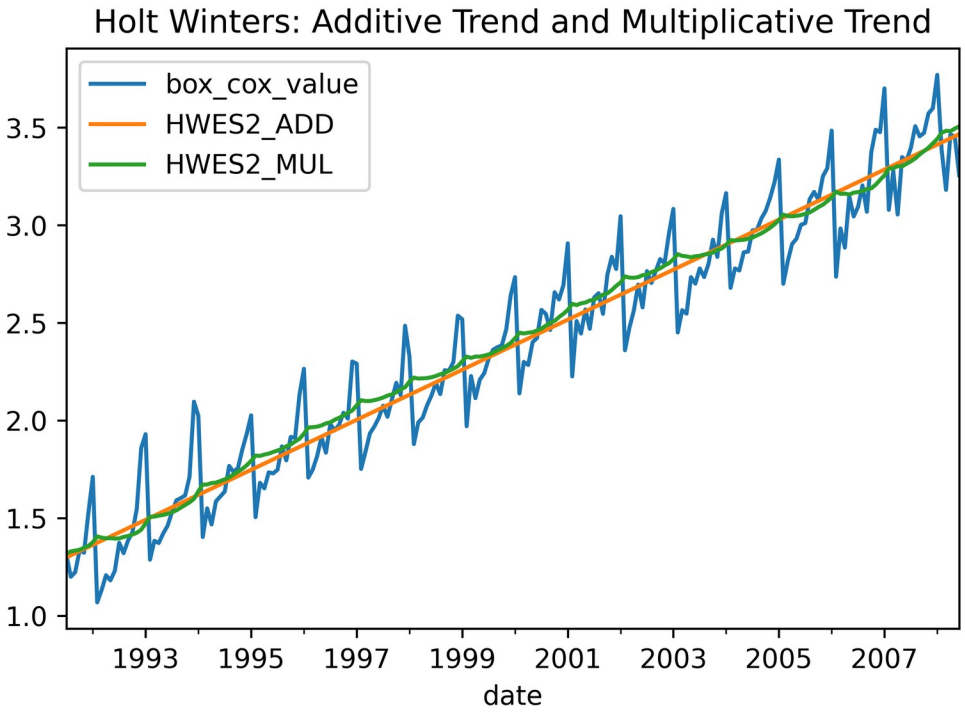


Figure 16: Double exponential smoothing, with additive and multiplicative trends.

5.2.3. Holt Winters' Model Fitting

To fit the model, the *holtwinters.ExponentialSmoothing()* function from the *statsmodels* library was used. To explore the effect of damping on model performance, two models were trained, one with damping and the other without, as shown in the figure below:

```
# Fit the model
hw_model = ExponentialSmoothing(train["box_cox_value"],trend='mul',seasonal='mul',seasonal_periods=x).fit()
hw_model_damped = ExponentialSmoothing(train["box_cox_value"],trend='mul',seasonal='mul',seasonal_periods=x, damped=True).fit()
```

Figure 17: Code used for fitting the Holt Winters' models.

The summary of the final trained model (without damping) is shown below:

ExponentialSmoothing Model Results

Dep. Variable:	box_cox_value	No. Observations:	163
Model:	ExponentialSmoothing	SSE	0.730
Optimized:	True	AIC	-849.654
Trend:	Multiplicative	BIC	-800.154
Seasonal:	Multiplicative	AICC	-844.904
Seasonal Periods:	12	Date:	Fri, 31 Mar 2023
Box-Cox:	False	Time:	17:08:53
Box-Cox Coeff.:	None		

	coeff	code	optimized
smoothing_level	0.1876579	alpha	True
smoothing_trend	0.000000	beta	True
smoothing_seasonal	0.7391421	gamma	True
initial_level	1.2666382	l.0	True
initial_trend	1.0043073	b.0	True
initial_seasons.0	1.0254346	s.0	True
initial_seasons.1	0.9512123	s.1	True
initial_seasons.2	0.9673301	s.2	True
initial_seasons.3	1.0456082	s.3	True
initial_seasons.4	1.0567344	s.4	True
initial_seasons.5	1.2186287	s.5	True
initial_seasons.6	1.3298900	s.6	True
initial_seasons.7	0.8452621	s.7	True
initial_seasons.8	0.8800438	s.8	True
initial_seasons.9	0.9053805	s.9	True
initial_seasons.10	0.8842230	s.10	True
initial_seasons.11	0.8950059	s.11	True

Figure 18: Holt Winters' model (without damping) summary.

5.2.4. Forecasting Result

The trained models were used for predictions for the period of the test set and for a period of three years after the last entry in the dataset. The figure below shows the forecasting result:

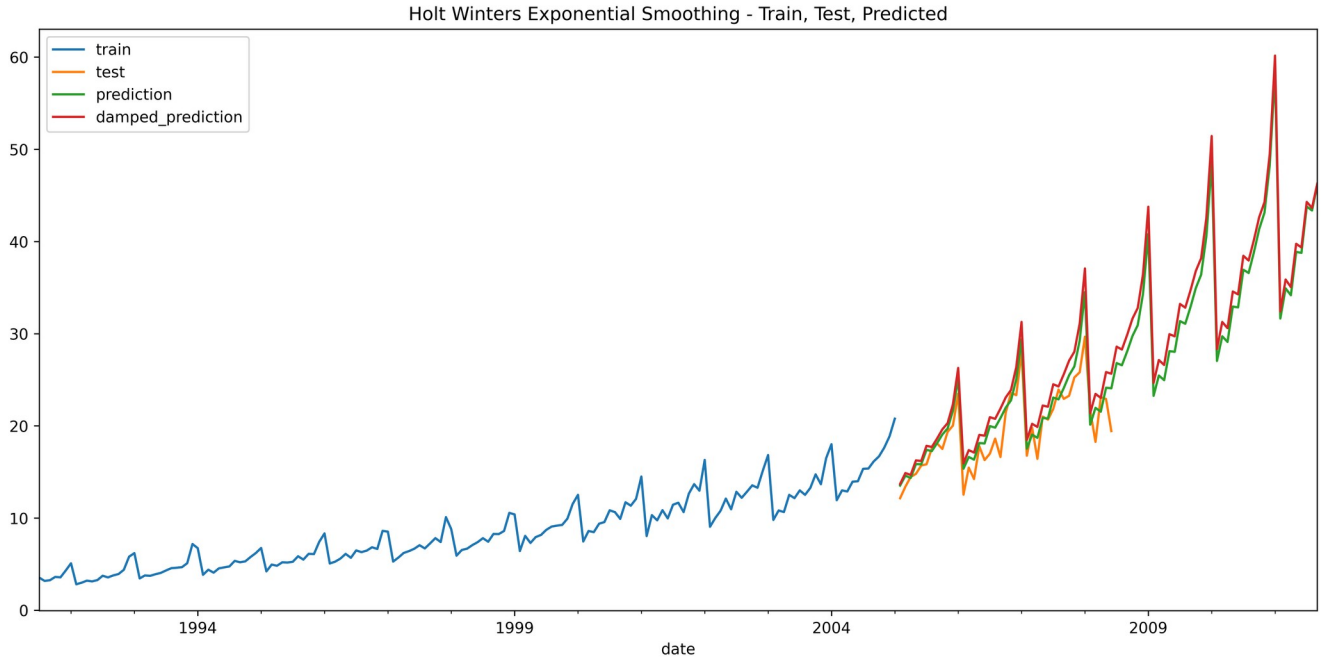


Figure 19: Holt Winters' models' predictions.

The RMSE was calculated against the test set, and the result was found to be **1.971** for the model without damping, and **2.897** for the model with damping. The model without damping therefore has significantly better performance in this case.

6. Conclusion

In summary of this project, exploratory analysis was first conducted on the time series data, which indicated that the data was non-stationary, and had both trend and seasonality components.

For data augmentation, the Box-Cox transformation was applied to the original data, with the lambda value of ~0.061506. Differencing was performed on the dataset for trend, followed by differencing with a period (lag) of 12 for seasonality. Subsequently, the data was split into train and test sets with the ratio of 80% of the samples for training and 20% of the samples for testing.

SARIMA models were then trained on the data, and the optimal model parameters were found to be 1, 1, 1, 0, 1, [1,2] for p, d, q, P, D, and Q respectively. These values can also be found in the table in Section 5.1.2. The RMSE of the trained model against the test set was found to be 1.614.

The Holt Winters' model was also used to model this data and perform forecasting. The effects of applying damping was tested on the Holt Winters' models, and it was found that the model without

damping performed better. The RMSE values for the Holt Winters' models were 1.971 for the model without damping, and 2.897 for the model with damping.

From the RMSE values, it can be seen that the model with the lowest error (of the tested models) is the SARIMA model, and is therefore the most appropriate to model this data.

7. References

[1] "PMDARIMA.ARIMA.AUTO_ARIMA¶," *pmdarima.arima.auto_arima - pmdarima 2.0.3 documentation*. [Online]. Available: https://alkaline-ml.com/pmdarima/modules/generated/pmdarima.arima.auto_arima.html. [Accessed: 31-Mar-2023].