Radboud University Nijmegen

Faculty of Science

Optimal Demand Forecasting in-practice

Time series analysis by SARIMA, state-space-models, and random forest

Thesis MSc Information Sciences

Supervisor:
Yuliya Shapovalova

Author:
Sander Ter Stege

Supervisor:
Daniel Walda

Second reader:
Tom Heskes

January 24th, 2020

Abstract

The growing production figures and pace of the open market make it more and more important to enact accurately to new developments. Demand forecasting has the potential to analyse patterns and support demand planning for increased overall performance. Forecasting models differ in predictive performance and practical use. The paper compares two different *estimation models* through SARIMA and ARMA state-space models and compares two different *models* through SARIMA and random forest on univariate time series. Concluding, the SARIMA model performs better on predictive performance than the random forest on the given time series. Still, in practice, the random forest for time series does have potential due to its efficient scalability to other time series. The non-convergent state-space model results in an inability for comparison in predictive performance. Convergent ARMA state-space estimations do reveal the potential to outperform the SARIMA models, though it will be at the expenses of practical inefficiencies in terms of computational costs.

Keywords — demand forecasting, time series analysis, SARIMA, state-space model, random forest

Acknowledgements

I want to thank the Radboud University for allowing me to write the thesis at Deloitte. The provision of resources and the flexibility provided in formulating the thesis have had a great impact on the end product. Both the Radboud University and Deloitte were resourceful and assisted both by providing supportive tools as through knowledge exchange.

In particular, I would like to thank the Radboud University supervisor Yuliya Shapovalova for the active supervision throughout the project. The discussions on subjects handled throughout the paper and extracted insights from the discussions were valuable in the formation of the thesis.

Also, I would like to show appreciation to my supervisor from Deloitte - Daniel Walda - for his progressive mindset and continuous feedback on the process. Throughout the project, Daniel has been important by providing the right resources and by providing the right framework for communication.

Contents

List of Figures

List of Tables

Listings

1 Introduction

1.1 Relevance

In the current globalised world, productions of large organisations have risen to enormous quantities. The digitalisation has caused an increased necessity to act quickly on the continuously changing environment as it increased the pace of the open market. The growing production figures make it more and more important to plan accurately to new developments as the growth will enlarge the impact on the organisation. Consequently, it is crucial in *demand planning* to strive for increased accuracy and efficiency of orders planning. Demand forecasting models have the potential to analyse and forecast demand planning based on the history of data. These systems could complement current human expertise in demand planning decision-making. Previously, less computationally intensive models as Radial Basis Functions, ARIMA, and GARCH forecasted time series for demand planning [11][12]. The development of digital systems resulted in a wider range of models being applicable in-practice due to the advancements in terms of computational power, such as the utilisation of Extreme Learning Machines and Deep Learning [47] [17] [18].

The focus of this paper will be on a demand planning case with real data from an organisation in the food industry. In the food industry, shortly perishable products make stock and production planning to minimise products loss very challenging [4]. Due to this challenge, a demand forecasting system which is both accurate and efficient is important in this field.

A popular way to execute demand forecasting is ARIMA: because of the large knowledge-base and widely supported tools, it is a relatively easy model to implement. Besides its ease of implementation, it also is an efficient model making it applicable in quickly changing environments. However, more computationally extensive models, such as the state-space models, potentially have higher accuracy. These particular models can be implemented through the use of the ARMA principle, resulting in a comparison of two models with similar methods and different model estimations. Though state-space models demand higher computational power [18], it is interesting to see whether the trade-off between efficiency and accuracy is worth using state-space models in practice. Note that the ARIMA and state-space models are both theoretically-driven models, requiring the designer of such algorithms to thoughtfully select components and parameters for efficient and effective execution of the forecast.

These theoretically-driven models are interesting to compare to machine learning models, which utilise their capability to continuously learn from the data to form a model applicable to the data. Instead of selecting components and estimating all parameters on forehand as is the case with ARIMA and ARMA state-space models, machine learning models estimate and correct the parameters itself. Through the learning process, machine learning tools can be utilised for the identification of time series patterns in the past and forecasting the future. For time series analysis, machine learning models have been used in the past, though literature lacks demand planning forecasting random forests [21][45]. Random forest has been proven to be effective in predictions and differs in its underlying model principles. Therefore, this paper compares random forest to the ARIMA and ARMA state-space models in terms of predictive performance and its applicability in practice [6].

The methods should get a use in-practice to add value to society and thereby should not only perform well in terms of predictions but also be easy to understand and implement to get a function in the industry. Therefore, the performance of all three models is captured not only through predictive performance measures but also through the practical applicability of the models. Practical insights concerning ease of implemen-

tation, ease of understanding, and computational efficiency are investigated as well in order to conclude which model could fulfil the prediction task best, both in predictive performance as practically.

1.2 Research Questions

Consequently, the formed research questions measure the predictive performances of ARIMA, ARMA state-space models, and random forest in an experiment and compare the models from the practical perspective. In more detail, the models are optimised for a demand forecasting time series case in terms of accuracy with an overarching goal to improve the forecasting precision. The best parameters for each model cause optimization to compare the models' potential fairly. Besides, the complexity of implementation, ease of understanding, and computational efficiency investigate its applicability in practice. The following research questions should lead to that goal:

i. What are the optimised models and according (hyper)parameters for SARIMA, state-space models and random forests?

ii. To which extent do the models perform in terms of predictive performance?

iii. To which extent are the tested methods feasible for use in-practice?

The following sections investigate the research questions in order to give a conclusion on the models' comparison systematically. First, the methods section - section 2 - presents the theory used for implementing the models. Afterwards, the specific experimental setup is given at section 3. The results section, section 4, will reveal the results of the research, which is further interpreted and compared in the discussion section - section 5. The discussion also includes opportunities for future works. The conclusion in section 6 wraps up the paper with the final findings.

2 Methods

The methods section is supportive and illustrates the principles used in the Experimental Setup. The introduction of each principle will start by an explanation of time series analysis and stationarity in sections 2.1 and 2.2. Afterwards, the seasonality principles utilised are explained and grounded at section 2.3. Subsequently, section 2.4 splits out the foundations of ARIMA and further extends to the Seasonal ARIMA model which offers extra, ARIMA related functionalities. As the ARIMA model needs sufficient parameters for successful execution, the methods to do so are defined in the ARIMA section as well. State-space model principles and its application on ARMA time series are given in section 2.5, followed by the random forest principles and its application on time series analysis in section 2.6. The methods section is finalised by arguing choices for prediction accuracy metrics in section 2.7, which enables comparison of the predictive performance of all three methods.

2.1 Time series Analysis

Time series analysis uses a series of data points sequenced by constant time intervals, which usually is per hour, day, week, month or year. Through analysis of such a sequence - or time series - one can identify patterns that occurred in the past to predict future patterns. More practically for demand forecasting, one can identify the total orders per week in the past and can use the information to forecast future orders. This information can enable anticipation on, e.g. making sure one has the right supplies to produce the incoming orders of the future. Usual patterns to identify in a time series are of long-term trends, short-term seasonality, and "unexplained" data - residuals. There are several methods to identify trends, seasonality and the residuals. This paper focuses on three models to identify the underlying patterns of the data: SARIMA, state-space model, and random forest. SARIMA, state-space models and random forests are further described at sections 2.4, 2.5 and 2.6. Before using those models, pre-processing the time series can be necessary, and the stationarity section identifies the necessity further.

2.2 Stationarity

A stationary time series is a time series data without long-term trends or seasonality. Though rapid, short-term changes or impulses may exist in time series, only as long as their nature seems to be for the short-term [25]. As ARMA models assume and assume stationary data for the time series analysis to execute, it is important to transform the data to a stationary time series. If one does not deal with stationarity, the linear nature of the underlying prediction model can cause problems for forecasting on time series with trends and seasonality. Usual methods to deal with seasonality and trends is by seasonality and trends correction, which will be further explained in section 2.3. This section focuses on testing stationarity and testing whether a time series is stationary can be done in several ways. In this paper, two common and widely used methods to test for stationarity, the Augmented Dickey-Fuller test and the Kwiatkowski–Phillips–Schmidt–Shin test, are utilised.

The ADF and KPSS tests analyse the time series and check whether the mean and variance of the time series stay the same throughout the time series. Since the time series should not have any trends or seasonality, the mean and variance of the residuals should stay the same over time. In order to do so, current time steps are compared to lagged time steps - a time step in the past. Comparing the current point with a previous point, say 15 weeks ago, should not make a difference in terms of mean and variance. The tests analyse the differences between points for each time step and form its judgements by converting the calculations of mean and variances into a "stationarity

score". Subsequently, the stationarity score leads to the probability that a time series is stationary. If the data turns out to be non-stationary, trends and seasonal corrections are necessary to transform the time series data into a stationary time series. As the time series in this paper only has a seasonal pattern, only the seasonal pattern will be focused on and can be found in section 2.3,

The next subsections further cover the stationarity tests. First, the differences of the KPSS and ADF tests is given through a brief, separate explanation at sections 2.2.1 and 2.2.2. The strengths of combining of both tests are argued at section 2.2.3. Also, lags selection is an important parameter for the successful execution of stationarity tests. Thereby, lags selection section finalises the stationarity tests through an explanation of and the approach for lags selection.

2.2.1 KPSS test

The Kwiatkowski–Phillips–Schmidt–Shin (KPSS) test calculates the probability that the time series has a unit-root - or is non-stationary - and is based on the following hypotheses:

i. Null-hypothesis: Time series has no unit-root and is stationary

ii. Alternative hypothesis: Time series has unit-root and is non-stationary

Through this null-hypothesis, the KPSS function assumes a stationary time series and tries to gather enough proof to reject the null hypothesis for a non-stationarity judgement. The rejection-border of the hypothesis bases on the choice for the Critical Value (α), which represents the percentage at which rejection of the null hypothesis is relevant. For a stationary time series through the KPSS test, there should be a lack of evidence to reject the null-hypothesis to make a case for stationary data.

2.2.2 ADF test

The Augmented Dickey-Fuller (ADF) test calculates the probability that the data set has no unit-root - or is stationary - and has the following hypotheses:

i. Null-hypothesis: Time series has unit-root and is non-stationary

ii. Alternative hypothesis: Time series has no unit-root and is stationary

The mechanics of the ADF test is the same as the KPSS test, though the hypotheses differ. The ADF test tries to gather enough proof to claim a stationary time series. As well as the KPSS test, the ADF test bases its rejection-border on the choice for the Critical Value (α), which indicates the proof necessary for the null-hypothesis (in %) to be rejected. For a stationary time series, there should be enough evidence to reject the null hypothesis.

2.2.3 Combining KPSS and ADF

Stationarity tests investigate whether the time series has a unit-root. Since the absence of a unit-root implies stationarity, unit-root tests are suitable as testing tools. In this paper, a combination of ADF and KPSS is used to test for stationarity. Both tests have the same mechanics to calculate the probability for a unit-root, but differ in statistical hypotheses: whereas the null-hypothesis for the KPSS test is a *stationary* time series, the ADF test's null-hypothesis is a *non-stationary* time series. Combining these methods covers the weaknesses and enlarges the strengths of both methods. Using only the KPSS test could raise the risk for a type II error: accepting the null-hypothesis while

the alternative hypothesis is true in practice. The other way around is the case for the ADF test, which has a risk for a type I error. Besides, the combination has proven to be strong in the past, having high testing power and being consistent in its results [8, 22]. To utilise both the power of the KPSS test and ADF test, and to reduce the risk for either type I or type II errors, this paper proposes the combination of both tests to test for stationarity.

2.2.4 Lags Selection

Both principles need lags as input to calculate their values. Stationarity tests compare the current time to the lagged time step, which is the lag value of time steps in the past. Say, if the lag is to be 10, the test compares values belonging to the current time steps with the values for ten time steps earlier. The comparison repeats for all possible time steps in the time series. Due to a possible remainder of seasonality or trends, lags selection can have a great influence on the success and power of a stationarity test. Inaccurate lag selection can cause a decrease of the testing power and increases the chance for the error types described in section 2.2.3 to occur. To deal properly with lag selection, one can utilise the process proposed by Ng and Perron through the usage of Schwert's formula, which covers the lags selection accurately and simple [48][39][29]. Equation 1 denotes Schwert's formula, in which p_{max} is the maximum lag size and T is the seasonal frequency. p_{max} is the starting point for the lags selection through the Ng and Perron procedure. If the absolute test statistic of the stationarity tests for the lag p_{max} is greater the value of 1.6, the given lags selection should be utilised. If not, the lag size should decrement until a test statistic has the correct value. Other procedures can be applied as well for lag selection. Though the procedures base its outcome on the same rule of thumbs, those procedures seem to be more time-extensive than the proposed method. Therefore, combination Schwert's formula and NG and Perron's procedure is chosen as the manner in solving the lag selection [48].

$$p_{max} = [12 * (\frac{T}{100})^{\frac{1}{4}}] \tag{1}$$

2.3 Seasonality

A time series is seasonal if the time series has short-term trends with an interval of a year at maximum. Detection and removal of seasonality can be important to make the data more stationary and thus is used in this paper as applied the time series seems to have seasonal patterns. Different manners exist to apply seasonal corrections. As all time series are different, the different methods are applied to see which method is best applicable to the used time series. This paper identifies three methods, which are seasonal differencing, seasonal decomposition and Fourier seasonality. Seasonal differencing is applied in the SARIMA model and is explained in section 2.3.1. Afterwards, the concepts decomposition and Fourier seasonality are identified in sections 2.3.2 and 2.3.3.

2.3.1 Seasonal Differencing

Seasonal differencing uses the same principle explained at the integrated component of the ARIMA models in section 2.4.2 and differs the current seasonal period with the previous seasonal period. To obtain the seasonally differenced time series, one starts at the end of the time series and substracts that data point with the data point, which has is one seasonal period in the past. Hence, the time series is weekly, and the seasonality is throughout the whole year, the seasonal period is 52 time steps for each week within

that year. For seasonal differencing, one should subtract the current time step with 52 time steps in the past and do so for all time series data points to obtain a seasonally differenced time series.

Though seasonal differencing can effectively increase the stationarity of a time series, applying seasonal differencing also has a downside: The function returns a total of time series points of the order of differencing times the seasonal period less than the original time series. In other words, if the seasonal period is 52 time steps, and the original time series has 260 time steps, the result of seasonal differencing is a time series of 208 time steps. Due to the sketched situation, the trade-off between stationarity and a possible increase of forecasting errors as a result of fewer data should be handled with care.

2.3.2 Seasonal Decomposition

Decomposition of a time series is splitting up a time series in a trend, seasonal and residuals components. Several manners exist to split up the components. This paper investigates the averaging method, which follows three steps. First, the method identifies the trend through a moving average. Second, the seasonal component is found through averaging over the seasonal periods. At last, the extraction of the trend and seasonal components results in the remainder component of the original values.

2.3.3 Fourier Seasonality

Fourier seasonality has been widely used and applied in many fields and is based on the Fourier transform. The Fourier transform decomposes the time series into constituent frequencies. In other words, the Fourier Transform expresses movements of data in sinusoids of different frequencies or time intervals in cosine functions [14]. The principle generates sinusoids for the desired number of frequencies and the summation of these functions should result in one or more "power" vectors, which are the vectors at time t for which the seasonal relationship between data points is highest [42]. The first power vector which is found can be used as seasonal arguments and will result in the best seasonal correction Fourier seasonality can achieve.

For improved computational efficiency and a minimal loss in output performance, one can utilise the fast Fourier transform in Equation 2 [14]. In the formula, N is the maximum seasonal period is utilised in the value formation for k. So, in case of a weekly time series, the maximum seasonal period is 52, and the maximum value for k is 26. Note that k is only expressable in Natural numbers.

$$y(t) = \sum_{i=1}^{k} A_i sin(2\pi f_i t + \varphi_i) \tag{2}$$

where $k = N/2$.

2.4 ARIMA(p, d, q) Model

ARIMA is a model for time series analysis which is based on three components: Autoregressive (AR), Integrated (I), and Moving Average (MA) [5]. The three components are in place to make sure a wide variety of time series fit into the model. Through ARIMA, one can implement the components which are necessary for the time series to be analysed, which can vary from all three components to only one component. What components to implement depends on the type of time series, which can be caused by differences in variances of the time series. setting values for the parameters p, d, and q sets the choice for components. After the specification of the parameters, the model can be used for forecasting - in this case, demand forecasting. The further subsections

outline each part of the ARIMA model and provide theories for parameter choices and model selection.

2.4.1 Auto-Regression

The AR function, of which the variable used in ARIMA(p, d, q) is p, assumes an auto-correlated and stationary data set. By the hand of the assumption of auto-correlation, the AR component calculates the correlation with a previous point to make statements for future points. In other words, a first-order AR function - AR(1) - calculates the correlation between the current time step and the previous time step. An AR(2) function assumes a correlation of the current time step with the previous two time steps, and so on. Thereby, the AR function tries to make sense of the data by identifying the correlation of the current time step with previous time steps.

In order to do so, the AR is calculated through the mean (μ) plus the sum of the multiplicity (ϕ_1) of the previous value (Y_{t-1}), and an error term ϵ. ϕ thereby represents the strength of autocorrelation with the previous point. A common method to obtain the right order p is by analyzing Partial-Autocorrelation Function outputs, which is explained further at section 2.4.6. Equation 3 illustrates an AR(1) function and Equation 4 is the general function for AR(p). Note that ARIMA utilises μ only once, as the MA component will have the same variable in its formula.

$$\hat{Y}_t = \mu + \phi_1 Y_{t-1} \tag{3}$$

$$\hat{Y}_t = \mu + \sum_{i=1}^{p} \phi_i \hat{Y}_{t-i} + \epsilon_t \tag{4}$$

2.4.2 Integrated

The Integrated process (d) is built-in for improving stationarity in case the data has a trend. The method ARIMA uses for trends correction is differencing, at which the function transforms the data by subtracting - or differencing - time series points at time t by the previous time series point $(t-1)$. Stationary data is required for the other components (p and q) to perform as intended and reduces the risk of an invalid model.

For the differencing method, the order of differencing indicates the type of differencing to be applied. The equations illustrate the distinctions between orders of differencing. At Equations 5, 6 and 7, the orders of differencing p are 0, 1 and 2. In the function, Y_t is the value at time t, whereas variations through Y_{t-1} and Y_{t-2} indicate values at the previous time steps. As one can observe, the number of differencing indicates the number of time steps at which the chain of differencing should continue. Once the order of differencing grows above 2, the formula extends by the same pattern as given in the equations.

$$y_t = Y_t \tag{5}$$

$$y_t = Y_t - Y_{t-1} \tag{6}$$

$$y_t = (Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2}) = Y_t - 2Y_{t-1} + Y_{t-2} \tag{7}$$

2.4.3 Moving Average

The MA averages several values and moves or iterates through the time series to take the average for each value [24]. In case of an ARIMA model, the order of q decides the number of times to repeat the moving average. According to Box and Jenkins, analysis of the Autocorrelation Function (ACF) output can be used as in deciding for the value

for q [5], which is further explained at section 2.4.6. In Equation 8, the specific form for MA(1) is given, whereas the generalized form is illustrated in 9. For both functions, q is the order of the MA, θ is the parameter of the model, ϵ represents the error, and μ is the mean of the time series. Note that the μ is also in the formula of AR and for completeness, ARIMA uses μ only once.

$$Y_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} \tag{8}$$

$$Y_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + ... + \theta_q \epsilon_{t-q} \tag{9}$$

2.4.4 SARIMA (p, d, q)(P, D, Q)

Seasonal ARIMA(p,d,q)(P,D,Q) uses the same techniques as ARIMA and extends the model through three seasonality components. Whereas the under case letters describe the orders of the ARIMA model, the capital letters describe the orders for seasonality and use the same techniques. The capital letters implicate P to be the seasonal autoregressive order, D to be the seasonal order of differencing, and Q to be the order of moving averages. The underlying concepts are all explained at sections 2.4.1, 2.4.2 and 2.4.3. However, whereas the (p,d,q) values use a standard lag of 1, the seasonal components are applied on the seasonal lag. The seasonal lag equals the number of time series data points to fulfil a seasonal period. In the case of weekly data for a year, the seasonal period would be 52 time steps, and the seasonal lag is 52.

2.4.5 AIC and BIC

The Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) are model comparison measures, which are useful at choosing the optimal SARIMA model. The metrics are approximately unbiased estimates of the Kullback–Leibler index, which estimates the fitted model relative to the true model [41][9]. AIC and BIC have the same calculations, though BIC corrects stronger on the order of p, d, and q values in the SARIMA algorithm. A higher order also implies a higher error which results in a wider prediction range, and this could outweigh the model's possible improvements [7]. In general, lower AIC and BIC values indicate a better model. As a rule of thumb, a model is hardly different if the difference between AIC or BIC values of two models differs with less than two [5].

2.4.6 ACF and PACF

The Autocorrelation Function (ACF) and Partial-Autocorrelation Function (PACF) are useful tools in estimating the parameter values in a SARIMA model as both plots show patterns which are the basis for correct values for the parameters. The mechanics of both methods are quite similar.

ACF is used for estimating the q value and calculates the correlation of a time series points with a lagged version of itself. In other words, it calculates the correlation of the current time step with a previous time step for different lag sizes. In estimating the q value, ACF plots are useful to analyze. If the correlation has a strong peak at the first lags of the plot, the number of significant correlations equals the order of q in the MA model. For seasonal corrections, an occurrence implies the Q variable to be applicable on the time series. Finally, the differencing orders can also be extracted from the ACF plots, as a trend will show a slowly increasing or decreasing ACF value over the year, whereas for seasonality the ACF value will slowly increase and decrease throughout the year into a fluent line around the x-axis.

PACF also calculates the correlation between two time spots, though it removes the correlation of variables between the two time spots. PACF thereby only uses the true

correlation between two time spots as it removed the influences of other time spots on the correlation. For estimating the right order, the same trick as in the ACF plot works for estimating the p value, based on the PACF plots. If the correlation of the PACF plot has a strong peak at the first lags of the plot, the number of significant correlations equals the order of p in the AR model. The same counts for P when analysing the seasonal lag.

The ultimate goal of a SARIMA model is to miss enough evidence for significant correlations for both the ACF and PACF plots. In other words, the underlying residuals of the models' estimation should be uncorrelated to make SARIMA model perform best.

2.5 State-space Model

State-space models calculate the next state solely based on the previous state and the history of data of comparable state situations. State-space models can be used for a wide variety of applications, such as analysis of neuroscience data, speech recognition, and image processing [30][27][37]. Besides, the identification of current patterns in time series and utilisation of the model to predict future values is within the range of the state-space estimations as well. This paper combines the state-space estimations with the ARMA model in order to model and forecast on the time series. In this section, first, the general state-space model principles are illustrated which count for all types of state-space models carried out in this paper. In order to predict future values, the Kalman Filter is utilised, which is depicted in section 2.5.1. Finally, section 2.5.2 outlines the state-space representation of ARMA.

In general, the state-space model has two types of representations: The continuous-time representation and the discrete-time representation. As the analysis concerns a time series analysis, the discrete-time representation - which follows the structure of time series - is used. The discrete-time representation results in the two equations given below, which are the observation and state equation.

$$y_t = F_t\theta_t + v_t \tag{10}$$

$$\theta_t = G_t\theta_{t-1} + w_t \tag{11}$$

$$v_t \sim N(0, V_t) \qquad\qquad w_t \sim N(0, W_t) \tag{12}$$

Observation equation y_t describes the underlying data at time-step t. The observation equation follows the Markov model, which bases future states solely on the previous state, forming a Markov chain [15] [43]. The state-space representation goes one step further by introducing the state equation θ_t. The observation equation estimates variables in the form of the state equation, as seen in the equation. The state equation θ_t is a "hidden variable" which is not directly visible, though the output, dependent on the state, is visible. Thus, through the state equation, the Markov chain extends to the hidden Markov model [35]. The variables F, G, V, and W represent the structure of the state-space representation. For the models implemented in this paper, the state-space models' parameters are Gaussian, which normally distributes the variables v_t and w_t with variances V_t and W_t. The variances V_t and W_t are the variables utilised for the implementation of the ARMA model as well, which is further explained in the ARMA section.

2.5.1 Kalman Filter

The Kalman filter predicts the next observed value based on the estimations of the current state variables and its uncertainties. Once the next point is observed, the new

situation corrects the predicted estimates using a weighted average. It bases its prediction on an estimation of a joint probability distribution over the variables for each time step, which results in a matrix of chances for a variable value x to occur at time step t.

In practice, after initialization, the Kalman filter follows a two-step process. First, the Kalman filter is initialised through building the model by the hand of states and errors calculations. The first step in two-step process is predicting. At the prediction step, the Kalman filter calculates its prediction based on the previous state, and the error covariance describes the uncertainty of the prediction. Equation 13 forms the prediction $\hat{\prime x}_t$ with an estimated uncertainty in Equation 14 of $\prime P_t$ in terms of the covariance for a normally distributed - Gaussian - error. In the second step, the actual value updates the predicted value and its according variables. The current estimations, P_t and $\hat{x}_t$ in Equations 15 and 16, are updated by using the previous estimations - $\prime\hat{P}_t$ and $\hat{\prime x}_t$ - and the actual value y_t. The optimal Kalman gain K in Equation 17 is utilised in the calculation for covariances of P_t and state $\hat{x}_t$ estimations to take possible errors into account for future predictions. Once the new situations led to updated values, the Kalman filter is ready to repeat the process for future time steps predictions.

Kalman Prediction

$$\prime\hat{P}_t = AP_{k-1}A^T + Q \tag{13}$$

$$\hat{\prime x}_t = A\hat{x}_{t-1} + B\mu_t \tag{14}$$

Kalman Update Estimations

$$P_t = (I - K_k C\prime\hat{P}_t) \tag{15}$$

$$\hat{x}_k = \hat{\prime x}_t + K_t(y_t - C\hat{\prime x}_t) \tag{16}$$

$$K_t = (\prime\hat{P}_t C^t)/(C\prime\hat{P}_t C^T + R) \tag{17}$$

2.5.2 ARMA

The ARMA model relies, as well as ARIMA, on the AR and MA models illustrated at sections 2.4.1 and 2.4.3. In the state-space model, ARMA can be applied through incorporating the principles in the variable construction of the state-space estimation model. Particularly, two variables are affected: v_t and w_t, both introduced in the formula of the general state-space model form in Equations 10 and 11. In matrix v_t, the variables entered for the AR specification are included, whereas in w_t the variables are included for the MA specification. Several methods to write the variables down are proposed in literature and Harvey's annotation is given in Equation 18 [36]. In this equation, d represents the $max(p, q+1)$, ϕ represents the AR coefficients, whereas θ represents the MA coefficients.

$$v_t \sim N(0, V_t) \qquad w_t \sim N(0, W_t)$$

$$V_t = \begin{bmatrix} \phi_1 & 1 & 0 & ... & 0 \\ \phi_2 & 0 & 1 & ... & 0 \\ \phi_3 & 0 & 0 & \ddots & 0 \\ \vdots & 0 & 0 & & 1 \\ \phi_d & 0 & 0 & ... & 0 \end{bmatrix} \qquad W_t = \begin{bmatrix} 1 \\ \theta_1 \\ \vdots \\ \vdots \\ \theta_{d-1} \end{bmatrix} \tag{18}$$

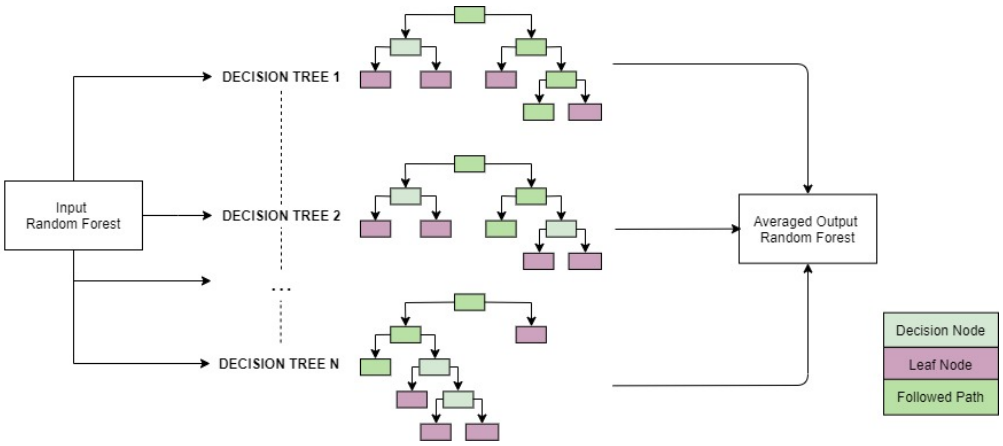

Figure 1: Random forest Example

2.6 Random forest for time series

Random forest is a supervised machine learning technique which builds decision trees in order to make its judgement about the input variable(s). Random forests can be utilised for classification and regression, and as this paper focuses on continuous, univariate time series, regression is outlined in this explanation of the random forest. The decision trees random forest utilises consist of if-else choices which lead the input variable(s) to an end-point of the tree, consisting of a statement and the probability the statement is true. The random forest typically uses multiple decision trees and combines all judgements of the trees to make its final judgement. Figure 1 represents an example of a random forest, which uses a collection of decision trees. As shown, each decision tree is built differently and is built in the learning phase through bootstrap aggregation - or bagging - and requires a training data set to do so. After building the trees, the input variable goes through the if-else statements, and in the end, the random forest combines the outcomes of all trees for a final judgement. For a deeper understanding of the principles used in the random forest, the decision trees and its bagging methodology will be clarified in more detail in section 2.6.1. As random forests cannot process plain time series, the theory to pre-process the time series for analysis through random forests is given as well in section 2.6.2.

2.6.1 Decision Trees

A decision tree can be used for regression and classification and uses if-then-else statements to estimate the to-be predicted, dependent variable. An example of the decision tree is given in Figure 2 at which the input variable is $time = 44$. Following the if-else blocks, the dark-green pattern is the if-else statements the input variable fulfils, ending up at a judgement block at the bottom of the tree. In this case, the $time = 44$ causes the quantity to have a value between 19.000 and 21.000, with a certainty of 95%. As shown in the example, decision trees consist of decision nodes and leaf nodes which connect through branches. Decision nodes involve a statement and at least connect to two or more other nodes, either a decision or a leaf node. Leaf nodes are end-nodes and contain a judgement, and the probability the judgement is true. Once the input reaches the end-note, the tree is finished and the result of the tree is clear. The depth of a tree is decided by the number of layers of nodes. The deeper the tree is, the more detailed the decision nodes are, and the higher the chance the judgement in the leaf node will

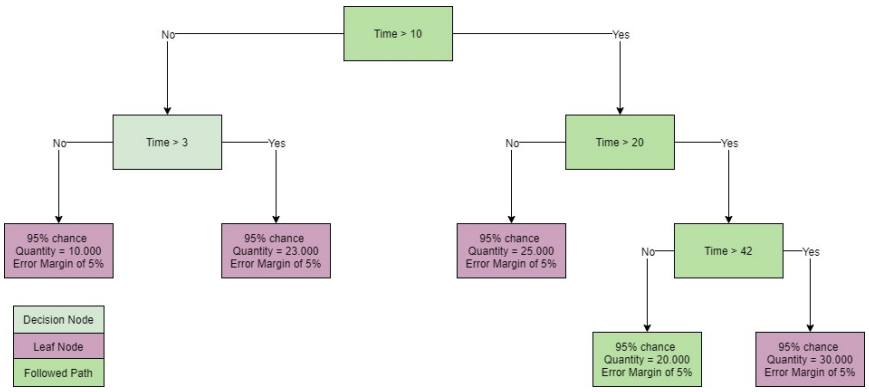

Figure 2: Decision Tree Example

converge to the original data. Doing that too much will tend the decision tree to copy previous data - or overfit the data - and should be prevented. Due to possible issues in building the trees, the formation of a tree is an important practice and is key for accurate execution of the trees.

Decision trees build on the underlying data at which one can use different techniques: randomised bagging and adaboost are two examples of techniques one can choose to apply [38][10]. Adaboost or adaptive boosting is a variant relying on the boosting method, which has as the main principle to learn from previous mistakes. In particular, adaboost does so by increasing the weight of misclassified or inaccurately regressed data points. Following this practice, adaboost can discover and adapt the inaccurately regressed and misclassified data points to a more accurate value. Randomised bagging is based on the bagging method and uses multiple machine learning algorithms together and averages its outcomes. Randomised bagging extends the principle of bagging by adding noises to different machine learning algorithms. It has proven to be more accurate and efficient in computability than the other methods and therefore is used in this paper[6][38].

2.6.2 Time Series application

The random forest regression algorithm typically utilises data cases at which time does not matter. However, for a time series analysis of a univariate time series, time should be one of the important aspects in basing the models' judgement. Dealing with two main challenges is necessary before using time series in random forests.

First, random forests have challenges with seasonality and trends caused by the replacement of values by values with an error, implemented through the bagging method. Replacing the value could cause the seasonality and trend in the data to disappear, while it is likely for the seasonal pattern to appear in the future as well. To deal with the issue, the random forest implements the same techniques of trends corrections used in the ARIMA model in section 2.4 and seasonal corrections in section 2.3 for pre-processing the time series into a series that can be processed by random forest models. The pre-processing results in the time series to be relatively constant and reduce the impact of wrong replacements adapting the time series' trends and seasonality.

Second, random forests are unable to predict indefinitely in the future, which implies the model needs at least as many historical time series points as the time series points it is trying to predict. It should be taken into account when choosing to apply random forests. Since the training set is larger than the test set, it will not be an issue for the used time series, and therefore, methods for solving this issue are not discussed.

2.7 Prediction Accuracy metrics

As forecasting is the goal of the models utilised in this paper, the comparison of forecasts is one of the core indicators for the performance of a forecasting model. Therefore, choosing the metrics to capture that performance is an important process [13]. Several measures have been investigated. Due to the case of having one univariate time series, the issues in the choice for error measures are less than issues which might arise at having several time series [20]. However, the decision is to combine several measures to get as most as possible out of each measures' strength and to prevent the loss of potentially valuable information. The information possibly could be useful at comparisons cross time series which is outside the scope of this research [3]. Therefore, the following five measures are investigated: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Percentage Error (MPE), Mean Absolute Percentage Error (MAPE) and Theil's-U. One should acknowledge the distinction between relative performance measures, which are accuracy measures in percentages, and absolute performance measures, which are accuracy measures returning data-related values. The following paragraphs describe each method briefly.

The *Root Mean Squared Error* is the root of the squared *ME* and is interpretative as relative values to the real values within the time series. The method on which it is based, Mean Square Error, was broadly adopted and supported at early stages of forecasting papers [3], though has been judged of its incapability to compare across different time series [1]. Taking the root of MSE does not solve the inability of comparison across different time series. Nevertheless, it will not form a problem for this paper, which is why it is used.

Mean Absolute Error is the mean of absolute errors. In other words, the mean absolute error averages all errors in real terms. Consequently, all negative and positive differences are made positive and then averaged to get a view on the real average error. MAE should give an insight into the deviation of the forecasts.

Mean Percentage Error is interesting for investigating the balance of positive or negative forecasts. The function returns the average percentual error of the average error summing both positive and negative errors. A negative percentage means an overall prediction which is too low, whereas a positive percentage means an overall prediction which is too high. This measure could indicate a possible combination of models if one model has a practice of predicting too high, whereas other models have a practice of predicting too low.

Mean Absolute Percentage Error indicates the prediction error by averaging the percentages derived from the absolute error in each period, divided by the observed values that belong to that period [23]. Among other methods, literature widely supports the measure [46], and the measure is useful in getting a grasp on the percentual deviation from real values.

Theil's-U investigates the predictive power of a function and thereby differs from the methods above. It does so by introducing the random walk into the error measures' functions [2]. In general, values for $Theil's\text{-}U < 1$ indicate the forecast reveals predictive behaviour, which is better than a random walk. A strive should be to obtain values of Theil's U close to zero since this value represents a strong predictive power by the forecasting function.

3 Experimental Setup

The experimental setup enlights all principles mentioned in the methods section from the implementation perspective. This section encourages reproducibility of the concepts to use for other time series and is described step-by-step for thorough understanding. First, section 3.1 discusses stationarity in which defines the methods and parameters. Subsequently, implementation choices and prediction methods for SARIMA, state-space model, and random forest are defined in-detail in sections 3.3, 3.4, and 3.5. This chapter finalises by a description of the used time series in section 3.6. Note that Appendix I contains the packages and functions used for implementation and Appendix V contains code for methods mentioned in this section.

3.1 Stationarity

The ADF and KPSS tests execute the stationarity tests. A visual analysis of time series graphs checks and validates the outcome of the stationarity tests. The stationarity tests adopted are the KPSS test and ADF test.

Two variables to choose before executing the stationarity tests are the critical value and lag. The critical values for both tests were formed based on the critical symmetric power value theory, proposed by Charemza and Syczewska [8]. The paper proposes equal critical values (α) for the ADF and KPSS test. To prevent type errors and to maintain the power of both methods, $\alpha = 0.05$ is adopted to test for stationarity. Lags selection utilises Ng and Perron's procedure with Schwert's starting point proposed in section 2.2.4, which results in a starting lag size of 10.

The second method for stationarity testing, the *visuals analysis*, is executed by checking whether the transformed data tends to a stationary time series. This method is more of a controlling action and relies on the interpretation of the researcher. The information could enrich the overall understanding, which helps to formulate an explanation for certain stationary time series and its forecasting behaviour.

3.2 Seasonal Transformations

The seasonal transformations applied are given in section 2.3 and involve differencing, decomposition and Fourier. Before analysing the models' output, the time series should be stationarity and applying these forms of seasonal transformations can do so.

The components of the provided package define the state-space model, which supports both Fourier and regular seasonal corrections. In this way, the state-space models deviate from the implementation done for the other two models as it forms the techniques through a state-space representation. If one would choose to implement the seasonal corrections in the same manner as the other methods, the impact of the state-space estimation reduces. Therefore, utilising all state-space models components is necessary for a full evaluation of the state-space estimation method.

3.3 SARIMA Models

In order to compare SARIMA models with the other investigated models, the optimal model is identified using the core components of SARIMA. The search for optimal models implies discovering the appropriate values for p, d, and q at implementing the ARIMA models, but also includes the optimal orders of seasonal corrections (P,D,Q). A successful search for optimal orders is crucial in finding the optimal SARIMA method and will be grounded in the model specification in section 3.3.1. Afterwards, methods for comparing the SARIMA models is given in model comparison (Section 4.5). At last, the prediction methods describe the way of adopting the rolling forecasts.

3.3.1 Model Specification

The choice of parameters specifies the SARIMA models and can be obtained through the application of two methods: the theoretical approach and model generation. Besides model specification for SARIMA, the model generation approach for F-SARIMA model specifies the optimal model using Fourier.

The theoretical approach bases its methods on the Box and Jenkins approach described earlier, which checks ACF and PACF to come to the proper parameters. As an input, the SARIMA model requires a stationary time series which is transformed through seasonal transformations and checked through stationarity tests. Afterwards, visual checks on the ACF and PACF plots of the residuals uses the theory in section 2.4.6 to theoretically choose the right parameters. If certain patterns come up at analysis of the plots, the values can be adapted in such a manner to get the desired result: residuals which are as uncorrelated as possible. In the end, the procedure forms the SARIMA parameters step-by-step and results in a theoretically-based SARIMA model.

The second method - model generation - generates a high variety of models and compares these through AIC and BIC metrics. The variations of models differ in the model specifications through different SARIMA parameters. Since the information criteria differ strongly per level of differencing - which can bias the overall results - the models are categorised by level of differencing and the models with the best AIC and BIC values of each category are investigated further in the model comparison.

In order to find the best Fourier SARIMA model, the order of k has to be determined first. As the underlying time series will always reveal the same seasonal pattern regardless of the SARIMA parameters, investigating all values of k through AIC and BIC metrics is investigated first. The information criteria for k determines the value for k used in the next step. The next step copies the model generation for SARIMA, implicating the model generation investigates a given set of SARIMA parameters, categorises those on levels of differencing, and chooses the model specification with the best information criteria in its category.

3.3.2 Model comparison

All models extracted from the steps at the model specification uses two steps for comparison: the models' AIC and BIC values, and the prediction errors analysis. The information criteria indicate the strength of the underlying model, whereas the prediction errors capture the predictive performance. In theory, the models with the lowest information criteria should predict best.

At last, the predictions' error measures compares each model on predictive performance. The prediction error measures, proposed in section 2.7, are investigated for all models to see what model performs best in terms of predictions. Also, graphs of the best models are the source for the visual representation to see how the forecast behaves and to see whether a combination of models could cause an improved forecasting performance. However, the papers' content does not include graphs.

3.3.3 Prediction Methods

The data case used is a case in the food industry, and therefore the production cycles are assumed to be short due to the use of fresh and perishable products. Therefore, a relevant prediction interval is assumed to be weekly. The weekly interval for a period of 52 time steps results in a situation that the forecasting model could learn from the new situation at each time step. Rolling forecasts, forecasting for one time step at each time step, therefore is a useful method to apply. However, packages providing rolling forecasts were not found. Therefore, prediction algorithms regenerate the models at

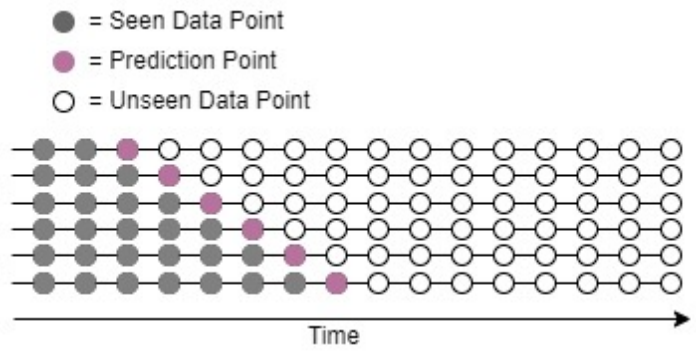

Figure 3: Rolling Forecast Principle

each time step at which it includes the newly available data, as shown in Figure 3. For each time step, the "newly available data" will be used to build a model and forecast for one time step. For rolling forecast implementation, the rolling forecast uses the forecast function of the Forecast package [19].

3.4 State-space models

The components used in the composition of the optimal state-space model are the polynomial, seasonal, and ARMA component. Each component needs hyperparameters dV and dW, which are important for correctly weighting each component. Also, the ARMA components need values for coefficients of θ and ϕ. The maximum likelihood estimation (MLE) and the Gibbs sampler search for the right values of the hyperparameters. The mechanics of the estimation principles are further explained in section 3.4.1. For exploring the best ARMA model and thus the best values for θ and ϕ, the SARIMA model's results are basis since the underlying time series is the same as in the state-space representation. The experimental setup for prediction is further explained in section 3.4.2.

3.4.1 Hyperparameter Estimation

The Gibbs sampler and MLE estimate the correct hyperparameters in this paper. In order to estimate the hyperparameters correctly, the estimators should converge to a constant mean and variance. Convergence implies a correct estimation of hyperparameters and thereby is essential in forming an optimal state-space model. In the case of the MLE, the function returns its statement on the convergence of the parameter estimations and thereby should not need any further investigations.

For a convergent hyperparameter estimation through Gibbs sampling, several parameters should be given on forehand. The parameter value choice can have a strong influence on the efficiency and effectivity of the estimations process. The input parameters for the Gibbs sampler are the starting point of estimations of the observation and state equations' variables. They can have a high impact on the iterations necessary for an accurate estimation. For simplifying the process of picking the estimations for the observation and state equations' variables, the original time series is normalised to a scale between -5 and +5. As the time series values are normalised, it makes it easier for the Gibbs sampler to adapt the initial estimations to actual estimations. The number of iterations is standard set at 1000, of which the first 100 estimations are not used - or 'burned'. If the sampler does not manage to converge within 1000 iterations, the number of iterations increases to a higher number for investigating convergence at later points of iterations.

3.4.2 Prediction

In the other used methods, SARIMA and random forest, the rolling forecast is applied. However, because the hyperparameter estimation is a time-extensive process, using the rolling forecast would take several days or even weeks from scratch and thereby is not feasible in this case. Besides, as the estimation of parameters is unlikely to change a lot at including the new data points. Therefore, it should not be an issue to drop recalculation of hyperparameters at each time step. However, one should consider the room for improvement at the interpretation of the results.

In the end, it resulted in a 52-step prediction without rolling forecast. To prepare the information for forecasting, first, the model is built, and the Kalman filter uses the model for filtering. Subsequently, the forecast function utilises the Kalman filtered time series and forecasts for 52 time steps.

3.5 Random Forest

To specify the random forest, one can give as input the number of trees to grow, the depth of the trees, and an indication for the number of splits for each node. To set the right depth of trees and the correct number of splits for each node, models generation finds the best hyperparameters by saving the value for MAPE. As the random forest uses a random error, outcomes of the forecasts differ each time the random forecast executes. Therefore, for choosing the right value of each parameter, the parameters are analysed separately by searching for a constant, low value of MAPE at each parameter. In other words, for deciding the right value of tree depth, the tree depth values are analysed for each value of node splits. If one can observe a low, constant performance for a particular value of tree depth, the optimised random forest model uses that particular value. The same is the case for node splits which checks all MAPE values for the different tree depths. The combination of both optimal parameters should lead to the optimal random forest model. The number of trees is set such that an increase in trees hardly affects the forecasting accuracy. The idea of the number of trees is that the random forest calculates each input row at least a few times. To prevent data points which are hardly estimated and to keep the random forest efficient, the number of trees is set to 1500. Thereby, only outliers of data points might never be estimated or once.

3.5.1 Time Series Pre-processing

Before running the random forest, the time series needs pre-processing in order to make it work properly. To do so, the model utilises decomposition to remove the trend in the data. After the transformation, the time series has no trend while still having the seasonal component. Random forest deals with the seasonal component through Fourier. As explained in 2.3.3, the value for k has to be set. The underlying seasonal pattern does not change between usage of different models, and therefore, Fourier SARIMA is the source for the value for k. The data without trend and the extracted Fourier component are the input time series for predictions.

3.5.2 Prediction

As is the case in predictions of SARIMA, rolling forecast is applied, which is explained in more detail at section 3.3.3. As well as in the SARIMA case, the model is rebuilt on the new data for each time step and subsequently predicts for one time step. As the model predicts a time series without trend, the trend needs to be reunited afterwards. To do so, ARIMA is used to recognise and forecast the trend for 52 time steps. Afterwards, the predicted trend and predictions combined form the finalised predictions.

3.6 Data

The time series contains five years of order and production details of a food manufacturer. A total of 2.2 million rows describe the order details in terms of quantity, week, year, customer, product-type, order-number, production plant, and shipping country. The production quantity is an important column and reveals the ordered quantity for the organisation necessary to produce. Derived from that statistic, the organisation can calculate the required resources to produce the product, which makes it a relevant variable to predict.

For the time series analysis, it is relevant to split the data up into equal periods. As the date in the data is weekly, the time series follows that pattern resulting in a time series with 52 data-points each year, or 260 data points in total. The variable used beside week and year, quantity, is summed per week to match the time series format. A training and validation time series divide the time series for training and validating the models. Based on results in the past in terms of training and validating the model accordingly, this paper utilises the 80/20 division of the time series [28] [16]. Thereby, the training set contains the first four years of data resulting in a total of 208 data-points to train upon, and the validation set contains the last year of data which will compare the forecasts to the actual values.

The plots in Figure 4 show the ordered quantity per week. As extracted from the plots, the ordered quantity per week tends to a seasonal pattern and tends to have no trend over the course of five years. Table 1 summarises the statistics of the time series, giving insights in averages and distribution of the time series values.

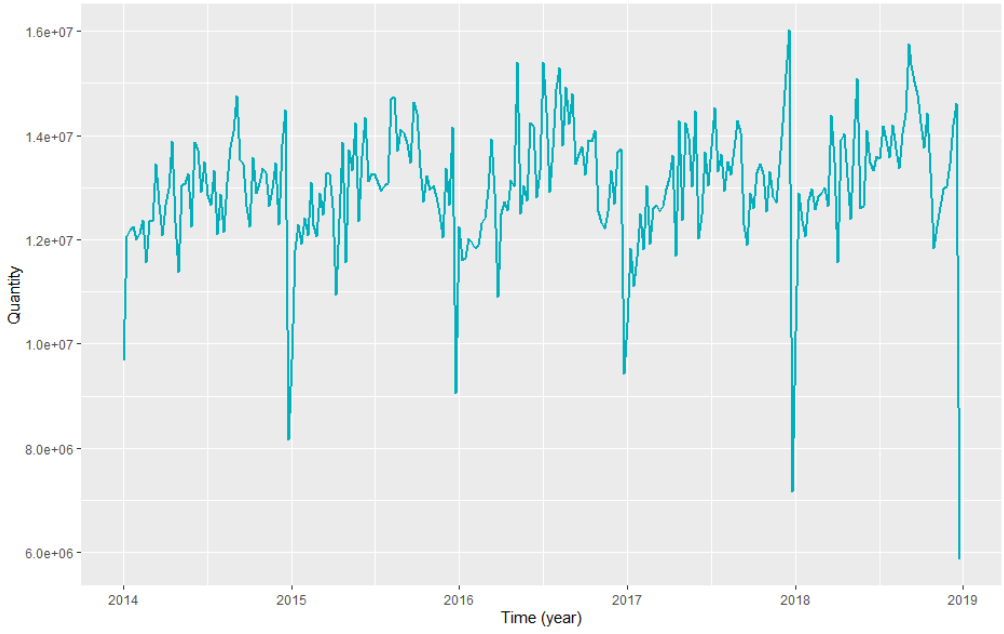

Figure 4: Time Series - Quantity per week over 5 years

Minimum	1st Quantile	Median	Mean	3rd Quantile	Maximum
5.869.133	12.419.782	13.040.087	13.024.744	13.762.191	16.019.201

	St. Deviation	Skewness	kurtosis	Standard Error	
	1.233.309	-1,53	6,53	76.487	

Table 1: Summary Statistics Time Series

4 Results

4.1 Stationarity

The first step in building SARIMA or ARMA models is passing the tests for stationary time series. For executing the stationarity tests, the critical value α and lag size require values first. The applied methods described in section 3.1 result in $\alpha = 0.05$ and maximum lag = 10. After running the tests, all tests pass the absolute value requirement of 1.6, which sets all lag sizes to 10. Though the original time series fulfils the requirements for stationarity according to the tests, a form seasonal or trend correction is applied to investigate whether data transformations could improve the forecasts. Also, through the visual analysis, one can still observe a seasonal pattern in the original data in 4. Applying seasonal corrections resulted in three stationary time series which are input for the models used in this paper. All seasonal transformations resulted in stationary time series. As illustrated in Table 2, the untransformed and seasonally differenced time series just passed the ADF test for stationarity, for which $p < 0.05$ for stationarity. The decomposed time series easily pass the ADF test. As for all time series $p > .10$ and $p > .05$ is the requirement, all time series manage to pass the KPSS test.

The transformed time series are given visually in Figures 5, 6, and 7. Comparing the results with the original time series one can see a great improvement in stationarity due to the more constant means and variances one can observe. To observe whether the differences in seasonal differencing result in differences in predictive performance, the SARIMA uses all four time series in model specification. Note that for SARIMA the untransformed and seasonal differenced time series are applied in one SARIMA model since SARIMA can apply differencing through parameter choices.

Abbreviation	Data Transformation	ADF	KPSS
UTS	Untransformed TS	$p = 0.046$	$p > 0.10$
SDif	Seasonally Differenced TS	$p = 0.040$	$p > 0.10$
SDec1	Seasonally Decomposed TS	$p = 0.029$	$p > 0.10$
SDec2	Seasonally Decomposed by STL TS	$p = 0.020$	$p > 0.10$

Table 2: Stationarity Tests for lag = 10

4.2 Optimal SARIMA Model

4.2.1 Model Specification

In this section, the theoretical approach and model generation mentioned in section 3.3.1 form the models. First, the theoretical approach specifies the models, followed by the models established by model generation. Through model generation, the Fourier SARIMA model finalises the model specification.

Theoretical approach

The theoretical approach utilises the theory provided by Box and Jenkins, further grounded at section 2.4.5.

Differencing and Seasonally differencing order d and D

The stationarity section applies the seasonal corrections and thereby, application of seasonal differencing is unnecessary for the seasonally transformed time series. As this is the case for the two seasonally decomposed time series, these will be used to find the

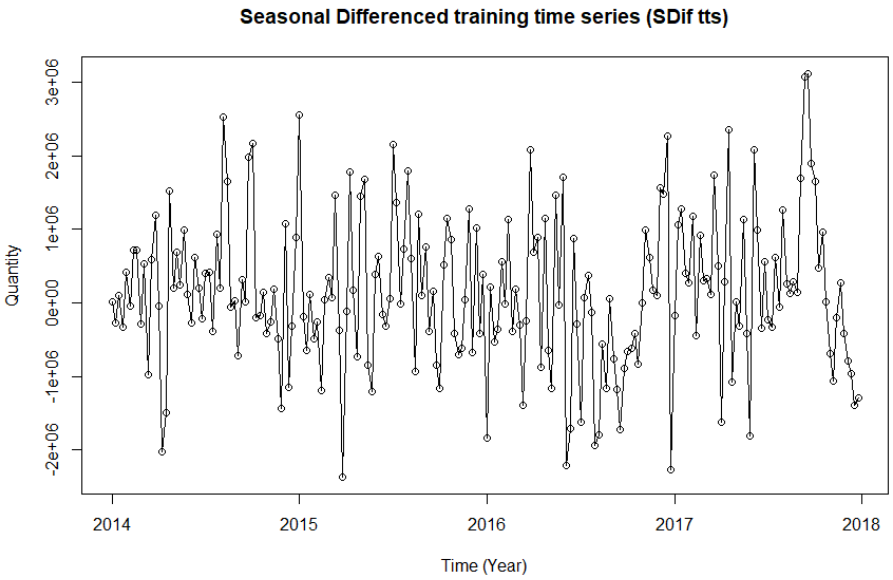

Figure 5: Time Series - Seasonally Differenced training set

best model. However, for a performance indicator of solely SARIMA, the original time series is put in as well. Analyzing the ACF plot for the original time series in figure 8, one can clearly see a "wave pattern" indicating seasonality. Besides, figure 4 confirms a time series tending to a seasonal pattern. Due to these facts, seasonal differencing - $D = 1$ - is applied to make the time series stationary. Differencing is unnecessary due to the absence of a trend, which indicates $d = 0$.

Seasonal Order of P and Q

The order of the seasonal autoregressive component P and seasonal moving average component Q are decided by the hand of the ACF and PACF plots in Figures 9 and 10. If the plots reveal significant correlation peaks at the seasonal lag size, which is 52 in this case, seasonal orders are suitable for the time series, either seasonal autoregressive or moving average. As the PACF plot illustrates a strong, significant, and negative peak at lag 52, $Q = 1$ is the correct value for the seasonal moving average. The other two time series reveal the same pattern as one can observe in Appendix II. Therefore, Q value for the seasonally decomposed time series follow the decided value of $Q = 1$. As one can observe in the next ACF and PACF plots, the correction has caused the peak at lag = 52 to disappear in all three time series.

Order of p and q

The order of p and q are derived from the correlation plots ACF and PACF applied on the residuals of the applied model, which has $Q = 1$ (Figures 11 and 12). Whereas the orders of P and Q follow patterns on the seasonal lag (52), patterns of the plots at the start of the year decide the orders of p and q. The plots indicate a strong peak at the ACF plot at lag = 0 and a significant peak at and until lag = 1, which implies a moving average model and thereby $q = 1$. As the plots for the other two time series in Appendix II reveal the same pattern, the same q applies to the other time series.

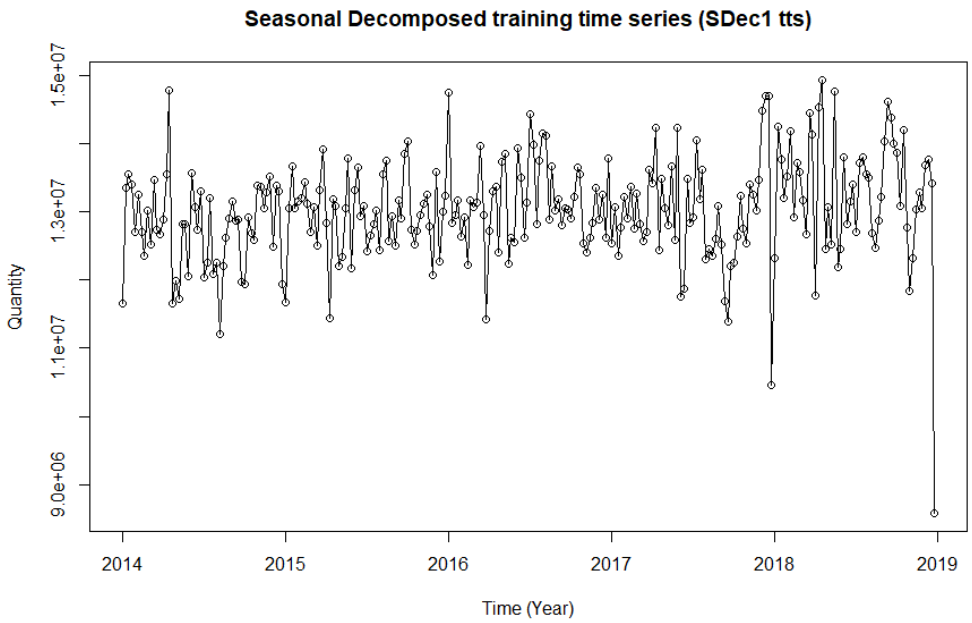

Figure 6: Time Series - Seasonally Decomposed training set

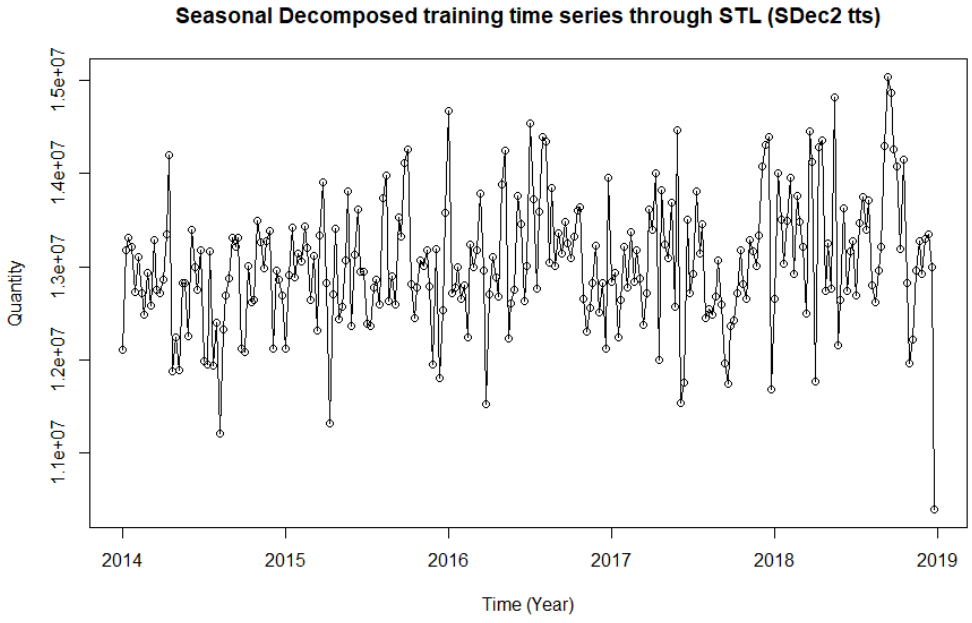

Figure 7: Time Series - Seasonally Differenced training set through STL

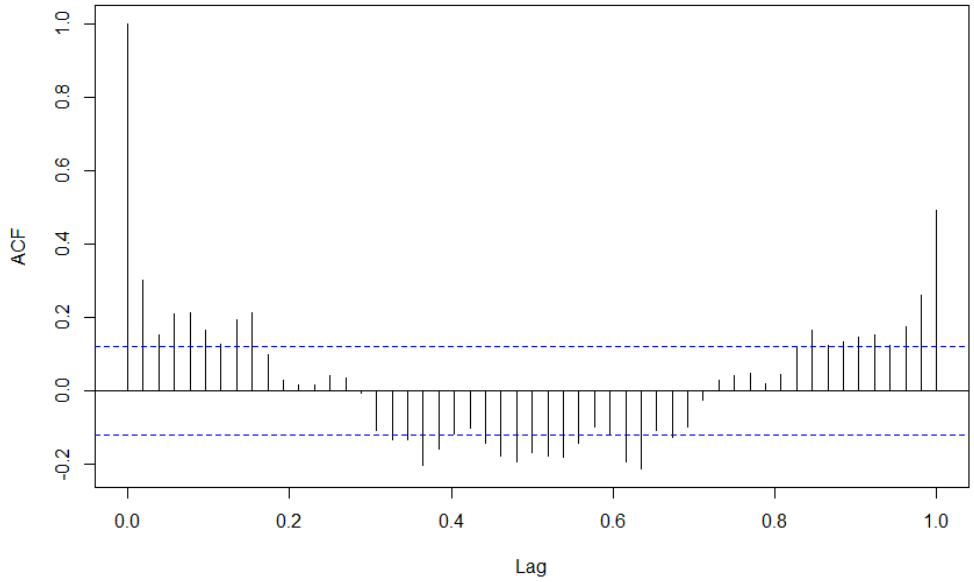

Figure 8: ACF - Original Time Series

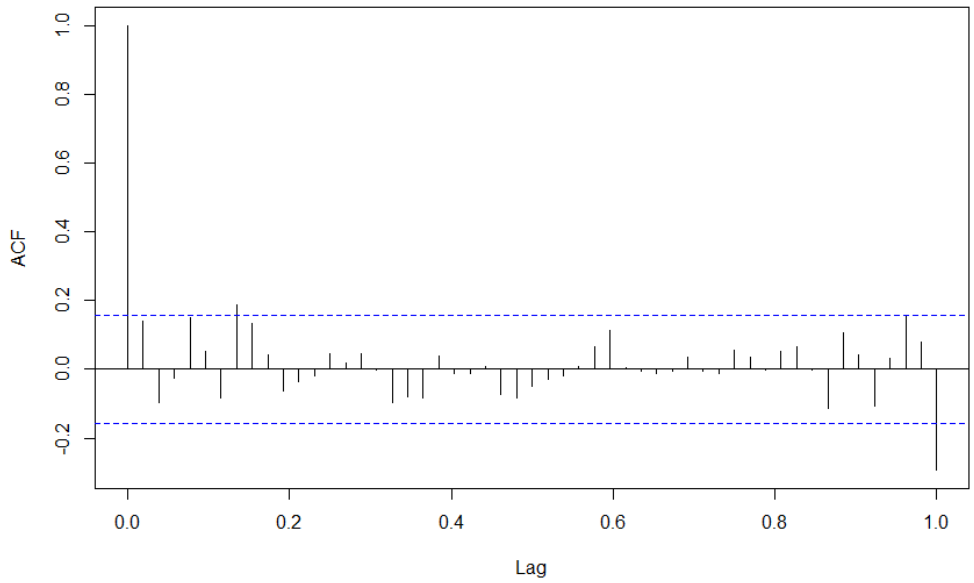

Figure 9: ACF - Seasonally Differenced Time Series

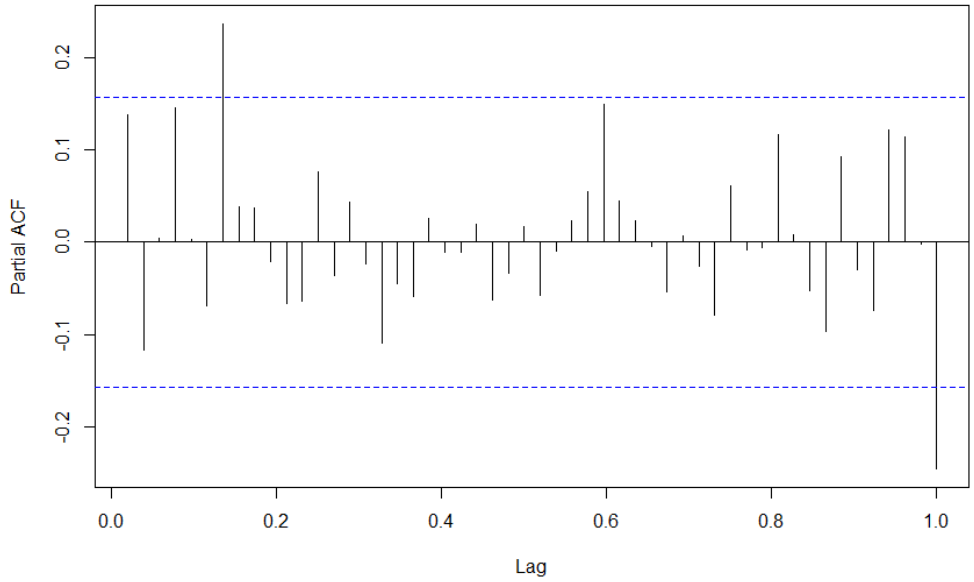

Figure 10: PACF - Seasonally Differenced Time Series

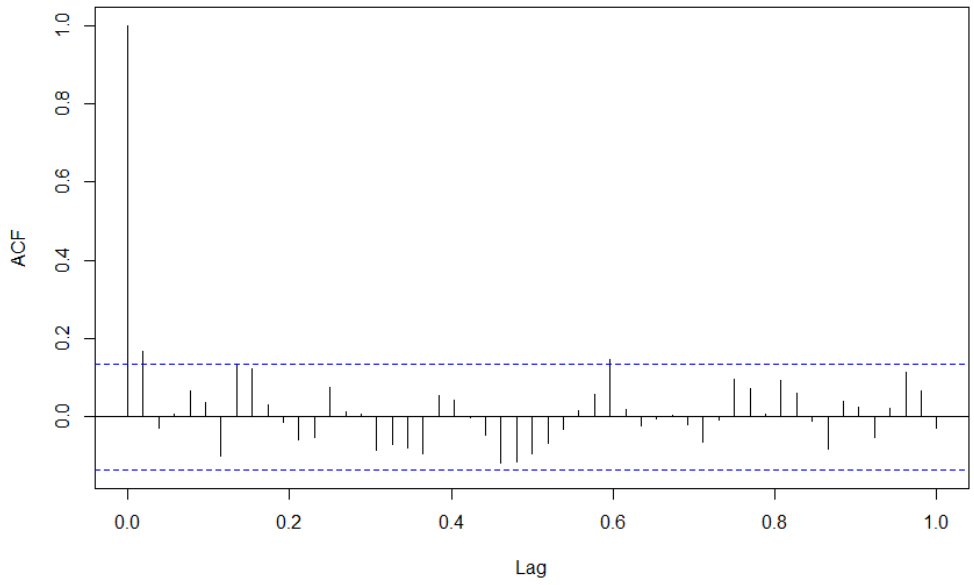

Figure 11: ACF - Residuals Seasonally Differenced Time Series, Q = 1

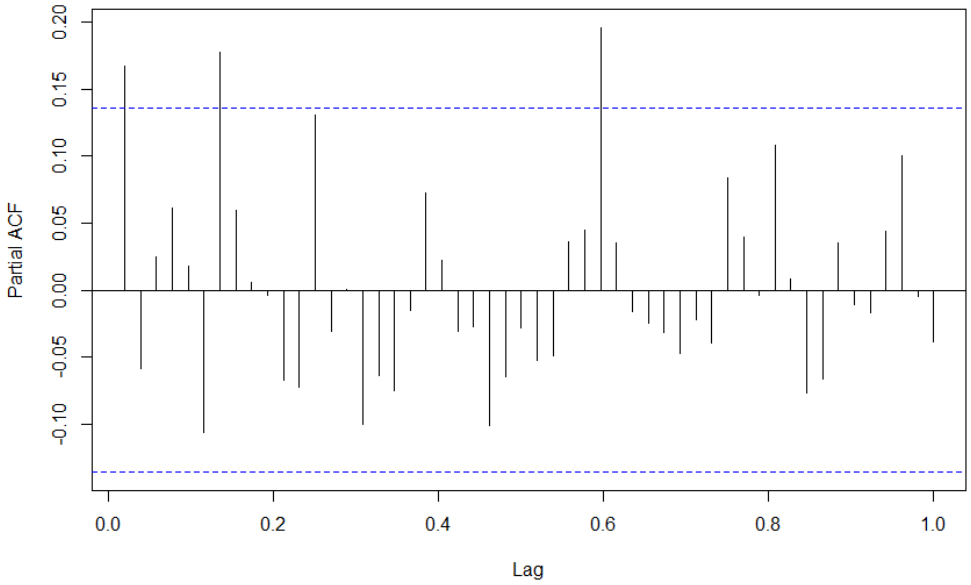

Figure 12: PACF - Residuals Seasonally Differenced Time Series, Q = 1

Final models

The final models for all three time series are given in Table 3, including the values for AIC and BIC. The values for AIC and BIC will be further used in the model comparison at section 4.5.

Time Series	Model	AIC	BIC
UTS	SARIMA(0,0,1)(0,1,1)	4725	4737
SDec1	SARIMA(0,0,1)(0,0,1)	6148	6165
SDec2	SARIMA(0,0,1)(0,0,1)	6082	6099

Table 3: Theoretically generated SARIMA models

Model Generation - SARIMA

In the model generation process, models with different values for $(p,d,q)(P,D,Q)$ were built and compared on their AIC and BIC values. The generation of models starts from SARIMA(0,0,0)(0,0,0) and will iterate through all possible combinations until the model reaches the ranges of the parameters. The choice for parameter ranges bases its values on the stationarity assumption required for the successful execution of the model. For some values for the seasonality component of SARIMA, the optimisation algorithm converges to numbers too small to handle, which is one of the borders of the SARIMA function for the testing range of parameters. Besides, the values for p, q, P and Q are capped by 2, as a higher order also increases the error margins which outweighs the possible improvements by the parameters. Table 4 represents the ranges of the parameters used for model generation.

The orders of differencing, either through trend or seasonal transformations, categorizes the AIC and BIC values as differencing causes the AIC and BIC values to differ strongly. In each category, the comparison of models results in the best model if it has the lowest AIC and BIC values. The total comparison through model generation results in a comparison of 108 model specifications per time series. Besides the seasonally transformed time series, the model generation uses the original time series as well to get an indication of the performance of solely SARIMA.

Table 5 presents the best models in each category. For the full range of results for each time series per category, one can observe Appendix III. As one can extract from the best models in each category, seasonal differencing has a very positive effect on the AIC and BIC values as these values far lower than the seasonally untransformed time series. Differencing also has a positive effect in terms of a decrease in the information criteria values. Whether the predictive performance improves is investigated in section 4.5, which identifies the optimal SARIMA model.

Parameter	p	d	q	P	D	Q
Parameter Values	2	1	2	0	1	2

Table 4: SARIMA Model Generation - Parameters

Time Series	Model	AIC	BIC
UTS	SARIMA(0,0,1)(0,0,2)	6336	6356
UTS	SARIMA(0,1,1)(0,0,2)	6310	6330
UTS	SARIMA(0,0,1)(0,1,1)	4725	4737
UTS	SARIMA(0,1,2)(0,1,1)	4701	4713
SDec1	SARIMA(0,0,1)(0,0,2)	6146	6166
SDec1	SARIMA(0,1,2)(0,0,2)	6124	6144
SDec1	SARIMA(0,0,1)(0,1,1)	4731	4743
SDec1	SARIMA(0,1,2)(0,1,1)	4707	4719
SDec2	SARIMA(1,0,2)(0,0,1)	6080	6107
SDec2	SARIMA(2,1,2)(0,0,1)	6058	6085
SDec2	SARIMA(0,0,1)(0,1,1)	4725	4737
SDec2	SARIMA(0,1,2)(0,1,1)	4701	4713

Table 5: SARIMA Model Generation - AIC and BIC

Model Generation - Fourier SARIMA

As an alternative for decomposition and differencing, Fourier corrects for seasonality as well. The Fourier transform utilises the SARIMA model afterwards, resulting in an F-SARIMA model. To combine Fourier and SARIMA, first, model generation over different values for k results in an analysis of the values through the analysis of the information criteria. Second, model generation for SARIMA parameters earlier described at section 3.3.1 executes over the best order of k. To find the best parameters of SARIMA at the given time series and to increase comparability, this model generation for F-SARIMA uses the same parameters given in Table 4. Since Fourier is a form of seasonal correction, only the Fourier transformed time series is the input of the SARIMA function and seasonal differencing is not applied.

In Appendix IV one can find the results for searching the best order of k for the Fourier function. Following the AIC and BIC values, the best order for k is $k = 26$. The two categories result in two generated models given in Table 6, which the model comparison further considers.

Model	AIC	BIC
F-SARIMA(1,0,1)(0,0,2)	7726	7932
F-SARIMA(1,1,1)(0,0,2)	7704	7911

Table 6: SARIMA Model Generation - AIC and BIC

4.2.2 Model Comparison

Model comparison compares through two manners in order to evaluate both the underlying models and the models' predictive performance. The AIC and BIC values evaluate the underlying model, whereas error measures of rolling forecasts for one year capture the predictive performance. Though both the underlying models and predictive performance are measured, the predictive performance is as a key indicator of a well-performing model in model comparison.

The predictive performance of the SARIMA models is captured in Table 8. Applying the theoretical approach and model generation in section 4.2.1 resulted in one duplicate model at the UTS time series. Therefore, the duplicate UTS model is removed from the model generation accuracy part, resulting in three generated models for the UTS time series, instead of four.

Following the results, one can argue that all types of seasonal correction result in similar results by model generation: The best model of all time series have accuracy values which are similar to each other. It is interesting to observe that only through the application of the original time series (UTS) in combination with solely SARIMA results in the best model by the theoretical approach. For the other time series, the theoretical approach did not find the best model. Other variations of SARIMA in the model generation for those time series outperform the theoretically formed models.

Table 9 indicates the forecasting accuracy of the best F-SARIMA models extracted from the model generation. As the F-SARIMA(1,1,1)(0,0,2) performs slightly better, the model comparison further considers the model in terms of predictive performance. Note that due to running costs, the rolling forecast range decreases from 52 time steps to 10 time steps. Therefore, the values differ strongly from the values given in the SARIMA comparison. For the overall comparison, the rolling forecast ranges will be the same for each compared method.

All in all, the models for each seasonally transformed time series tend to perform equally well. Though the theoretical approach seems to work best for the untransformed time series, model generation managed to find the best models for all time series, which resulted in three very comparable models. When looking further behind the commas, the untransformed time series in combination with SARIMA(0,0,1)(0,1,1) works best and therefore, further comparisons use this particular SARIMA model. Also, as other methods apply Fourier as well, the best F-SARIMA model is extracted from the model generation and the overall model comparison analyses the models in section 4.5.

Time Series	Model	AIC	BIC
UTS	SARIMA(0,0,1)(0,1,1)	4725	4737
SDec1	SARIMA(0,0,1)(0,1,1)	4731	4743
SDec2	SARIMA(0,0,1)(0,1,1)	4725	4737

Table 7: Best SARIMA models - AIC and BIC

Time Series	Model	RMSE	MAE	MPE	MAPE	Theil's U
UTS	SARIMA(0,0,1)(0,1,1)	832728	622625	-0.29	5.11	0.55
SDec1	SARIMA(0,0,1)(0,0,1)	977884	709455	-0.90	6.00	0.65
SDec2	SARIMA(0,0,1)(0,0,1)	876931	633619	-0.98	5.35	0.58
UTS	SARIMA(0,0,1)(0,0,2)	1097892	732046	-0.73	6.37	0.70
UTS	SARIMA(1,1,1)(0,1,2)	1109579	734512	-1.14	6.43	0.70
UTS	SARIMA(0,1,2)(0,1,1)	837572	627561	-0.30	5.15	0.55
SDec1	SARIMA(0,0,1)(0,0,2)	969101	717619	-0.75	6.04	0.64
SDec1	SARIMA(0,1,2)(0,0,2)	966723	715706	-0.75	6.02	0.64
SDec1	SARIMA(0,0,1)(0,1,1)	832238	626276	-0.26	5.12	0.55
SDec1	SARIMA(0,1,2)(0,1,1)	834646	628855	-0.24	5.14	0.55
SDec2	SARIMA(1,0,2)(0,0,1)	876268	638322	-0.98	5.37	0.58
SDec2	SARIMA(2,1,2)(0,0,1)	875958	639271	-0.97	5.38	0.58
SDec2	SARIMA(0,0,1)(0,1,1)	832727	622625	-0.29	5.11	0.55
SDec2	SARIMA(0,1,2)(0,1,1)	837572	627561	-0.30	5.15	0.55

Table 8: Forecasting Accuracy SARIMA models - For 52 time steps

Model	RMSE	MAE	MPE	MAPE	Theil's U
F-SARIMA(1,0,2)(0,0,2)	422293	380435	0.34	3.02	0.40
F-SARIMA(1,1,1)(0,0,2)	421129	379320	0.33	3.01	0.40

Table 9: Forecasting Accuracy F-SARIMA models - For 10 time steps

4.3 Optimal Random Forest model

For the formation of random forests, one has to select features for investigating patterns. To decide which variables are best for the given time series case, the random forest forms a Fourier feature with a seasonal lag. As derived from the variable importance plot in Figure 13, the seasonal lag and the seasonal period of 52 are strong indicators for the ordered quantity. Therefore, these are the features of the random forest. Note that the trend already has been extracted from the time series used in the random forest.

Besides, feature selection, one has to decide The lag size, one of the parameters, is set to 1. Through trial and error, the lag is set to 1, and therefore, there possibly still is room for improvement on the lag size. Besides lag size, model generation thoroughly investigates the different values for node size and node split, MAPE calculations captured the accuracy of the predictions of the variable combinations. Table 4.3 represents the results, in which node split ranges between 2-9 and node size ranges between 2-8. The random forest used in this paper works with adding errors to time series data points for recalculation of new trees. Since the errors are random, each calculation for a random forest will differ in the outcome. Therefore, to obtain the best value for node size and node split, the columns and rows are analysed separately in a search for constant, low values. As one can extract from Table 4.3, $nodesplit = 7$ has low constant performance, having values between 5.99-6.02, with two outliers to 6.9 and 5.95 - which also is the second lowest value. For node size, constant low values are achieved best at $nodesize = 6$, at which the function varies between 5.98 and 6.03. Therefore, $nodesplit = 7$ and $nodesize = 6$ are utilized as hyperparameters for the random forest function. Though there are deviations in choosing different values for node split and node size, the deviations are minimal, which is why the choice of the parameter will have a minimum impact on the improvement of the model.

Note that a node size and node split not are the actual values in the size of nodes and the number of splits. For node split, a higher number indicates a higher number of splits but not *exactly* the number of splits, and for node size, a larger node size causes *smaller trees* to be grown instead of larger ones - as one might have expected. Instead, the represented values are utilised as direct input for the randomForest function and thereby are relevant for the paper.

The optimisation of the hyperparameters formulates the best random forest model. Its predictive performance is presented in Table 11.

Node Size (nodesize) \ Node Splits (mtry)	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
2	6.14	6.14	6.10	6.09	6.09	6.09	6.15	6.19
3	6.07	6.04	6.00	6.07	6.03	6.02	6.04	6.08
4	6.05	6.03	6.07	6.00	6.02	6.01	5.99	6.06
5	6.03	6.06	6.04	6.04	6.03	**5.95**	5.98	6.03
6	6.08	6.03	5.99	6.02	6.00	6.02	6.01	5.98
7	6.04	6.11	6.05	6.06	6.05	6.02	6.07	6.01
8	6.04	6.06	6.02	5.97	6.00	5.99	6.03	**5.94**

Table 10: MAPE for different hyperparameter values Node Splits & Node Size

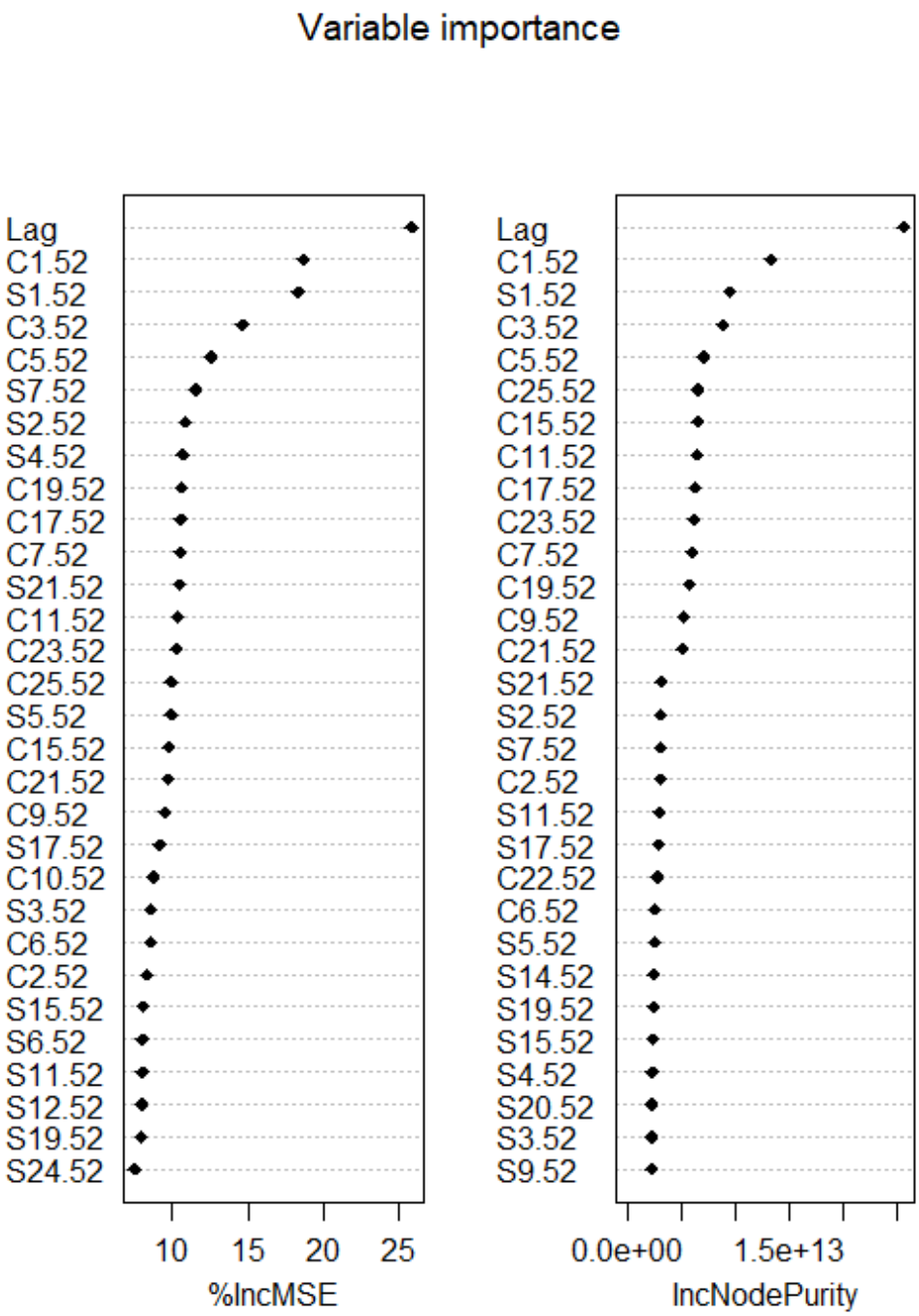

Figure 13: Variable importance random forest

Model	mtry	nodesize	RMSE	MAE	MPE	MAPE	Theil's U
Random forest	7	6	973845.9	699383.1	-1.92	5.98	0.63

Table 11: Forecasting Accuracy Random Forest

4.4 Optimal State-space model

The thesis investigates 60 state-space model specifications in the search for a model which fits the data. The formed state-space models utilise the polynomial, seasonal or Fourier, and ARMA components. The Gibbs sampler, maximum likelihood estimation (MLE), or a combination of both estimate the hyperparameters of the specified models. Also, the input differs in the utilised time series, which are the original time series, seasonally differenced or decomposed time series, and the original normalised time series. The variations of components, estimations, and time series resulted in state-space models of which the hyperparameter estimations did not seem to converge. In other words, the estimation functions did not manage to fit the model components to the given time series. This section describes what model variations, estimation variations, and variations of time series have been tested. Afterwards, the section finalises by identification of the most promising model in terms of convergence for the given time series. In the discussion at section 5.3.1, the state-space model is included in future work and methods to make the state-space model function for the given data case.

By the hand of the best state-space model in terms of convergence, the issues found in state-space models are described and explained. The hyperparameter estimation is executed on a normalised time series through MLE and Gibbs sampling to get a complete image of estimation methods. The utilised components in the state-space model follow the theoretical approach and cover patterns identified in the other applied models. The theoretical approach resulted in the application of a polynomial function for slight trend corrections, a seasonal component seasonal corrections, and an ARMA component to cover the pattern throughout the time series. The ARMA component follows the patterns found in the SARIMA model, resulting in an ARMA(0,1) model with an MA coefficient which is the same as estimated through the Arima function – 0.2. When executing the MLE, the function indicates a converged hyperparameter estimation implying the hyperparameter estimation should be correct. For verification, Table 12 gives the results of no estimation, the MLE, and Gibbs sampling estimation for a polynomial, seasonal and ARMA (0, 1) state-space model with a normalised time series. As extracted from the table, no estimation performs better than estimating the components through either Gibbs sampling or the MLE, and thereby the estimation methods tend to a non-convergent estimation of the components.

For further analysis of the estimation methods, Figures 14 and 15 support the Gibbs sampling method investigation. The Gibbs sampling of the hyperparameters should be a horizontal flat line around the optimal value for hyperparameters. As one can observe in both figures, this is not the case for the Gibbs sampler executed in this paper, which indicates the hyperparameter estimations used for the observation variance and hidden state variance are incorrect. Note that the hidden state variance estimations include the estimation of 54 variables used in the components.

Further analysis of the MLE could have been obtained by implementing the likelihood estimator. The outcome of the MLE indicates convergence. Nevertheless, due to the accuracy and outcomes of the other estimation method, there is a strong incentive that convergence is not the case. Therefore, it could be interesting to implement the estimator to plot the outcome of the estimation for different parameter inputs in order to get a grasp on happens within the function. The Future Work section includes this extension.

Note that the example described above is the example which seemed to tend most to convergence and thereby has the best convergence performance of all over 60 tested model specifications. Thereby, the other functions seem to be further away from convergent hyperparameters. Since none of the models has the optimal hyperparameters, statements about optimal state-space models are impossible to produce. Also, a comparison between state-space models would be unfair as a model which converges faster is more likely to have improved performance. In practice, these models could perform better and thus is not a correct indication of models' performance. Though correctly estimated models are missing, the 'luckily' found state-space model through not estimating the hyperparameters at all can be seen as an indication for the potential of state-space models.

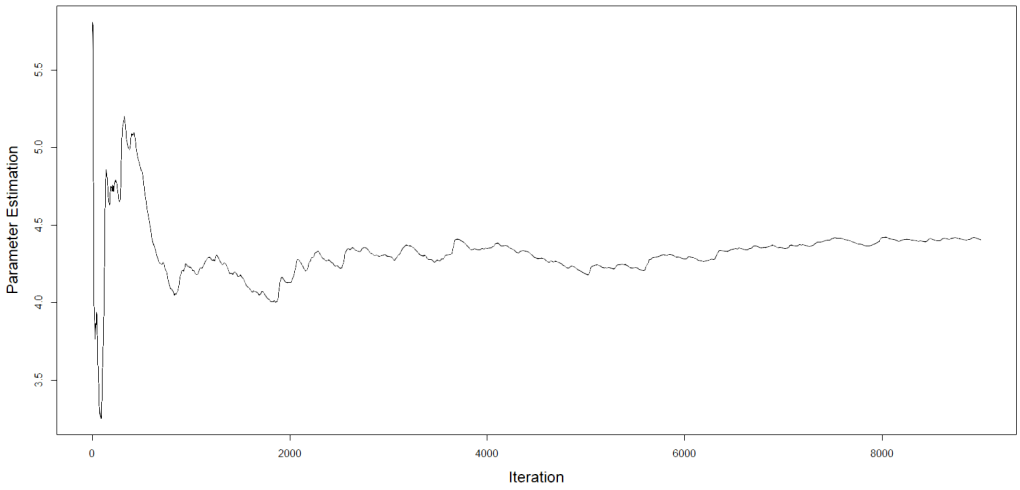

Figure 14: Gibbs sampler - dV - Observation Variance Estimation

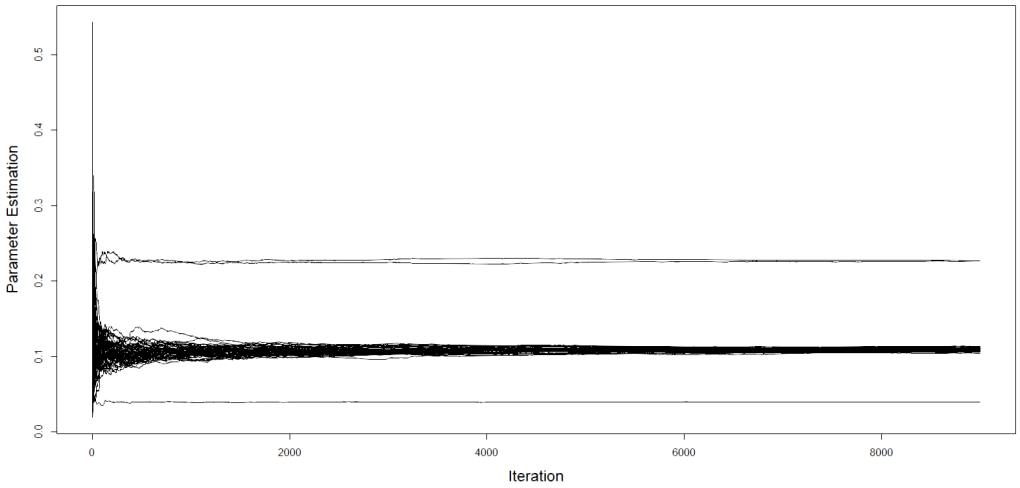

Figure 15: Gibbs sampler - dW - Diagonal elements of the system variance estimations

Estimation	RMSE	MAE	MPE	MAPE	Theil's U
No estimation	862538	667883	0.02	5.36	0.55
Maximum likelihood estimation	904900	681704	1.81	5.45	0.59
Gibbs sampler	1071414	815198	1.64	6.22	0.69

Table 12: Forecasting Accuracy SARIMA models

4.5 Models' Comparison

This section analyses the predictive performance of all optimal models. Table 13 presents the accuracy measures for ten time steps of each best model. Since the F-SARIMA can only calculate to ten time steps due to computational performance, the table expresses the ten time step accuracy measures of the other models as well for comparison.

As one can perceive, the accuracy metrics for ten time steps are far more accurate than the accuracy metrics for 52 time steps. Most likely, this is due to a pattern in the data in the second half of the validation set, which one can hardly perceive in the previous years of the training set. As the first ten time steps exclude the deviating time series points in the second half of the validation set, the models only investigate the pattern which looks like the patterns in previous years, which results in a more accurate forecast in general.

Comparing the methods, the SARIMA model performs best in terms of MAE and MAPE metrics. However, Theil's U, MPE, and RMSE of F-SARIMA have the best values compared to the other functions. This indicates SARIMA performs better in absolute deviation from the actual values, whereas F-SARIMA performs best in terms of an equal deviation, either positive or negative. Figure 16 confirms that by revealing the F-SARIMA model has a more peaked forecast around the validation time series, whereas the other two forecasts have fewer peaks. What is interesting to see in the random forest is that the Theil's U of random forest is equal to SARIMA and the MPE performs best. However, the other accuracy metrics perform far worse than the other forecasting models. As one can observe in the figure, the first forecast of random forest is far off, which decreases the performance of RMSE, MAE and MAPE and increases the MPE.

All in all, the SARIMA performs best in absolute deviation of the validation time series. F-SARIMA performs well in both absolute and average deviation, whereas random forest performs best in average deviation. Note that random forest most likely performs best on the metrics due to a strongly deviating starting point. SARIMA and F-SARIMA perform both well, and one can argue that the usage of both models depends on the forecasting goal. Where SARIMA is more balanced in its predictions, F-SARIMA has stronger peaks. If one wants peaked indications of the next week, one could use F-SARIMA. Though if one wants a stable forecast for the next week, one could use SARIMA.

Model	RMSE	MAE	MPE	MAPE	Theil's U
SARIMA(0,0,1)(0,1,1)	466039	365833	1.85	2.88	0.46
F-SARIMA(1,1,1)(0,0,2)	421129	379320	0.33	3.01	0.40
Random forest	584070	460596	0.14	3.83	0.46

Table 13: Forecasting accuracy best methods

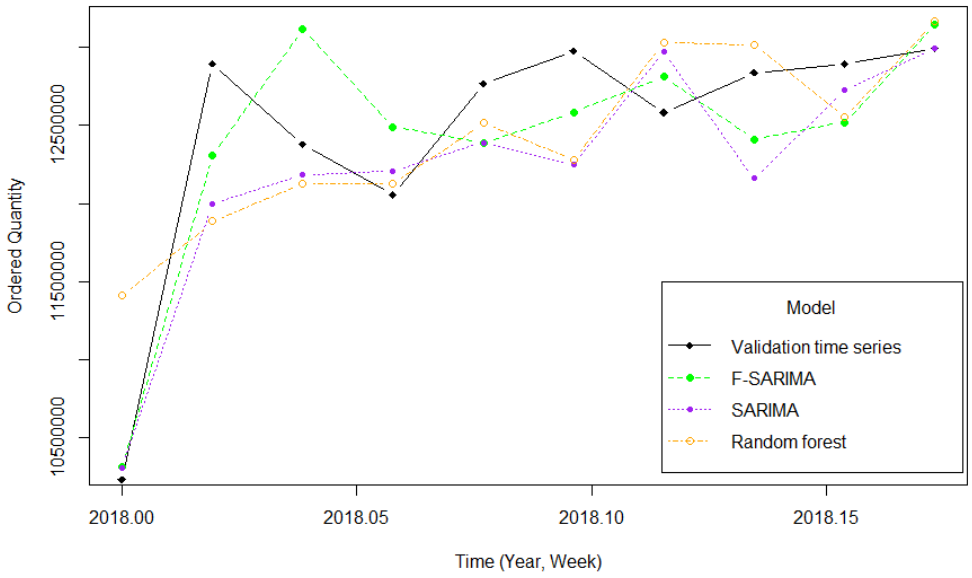

Figure 16: Three best models - forecast comparison

5 Discussion

5.1 Predictive Performance

The results section gives insights into the predictive performance of SARIMA, state-space models, and random forest for the utilised time series. It is clear SARIMA proofs to be most capable of producing forecasts: SARIMA outperforms the other methods on all five metrics and thereby performs significantly better. Three different variants of SARIMA - caused by different seasonal corrections - perform equally well. The variants of SARIMA performing best do not always have the lowest AIC and BIC values: applying more forms of differencing seems to have a positive impact on decreasing the information criteria. At the same time, the forecasts might be negatively influenced by applying more forms of differencing. Therefore, a comparison of only AIC and BIC values can border finding the optimal SARIMA model. The application of Fourier in SARIMA does not seem to have a positive impact on the performance of the algorithm compared to the other seasonal transformations. The best Fourier model performs worse on all accuracy metrics than SARIMA, which implies that Fourier transformations do not have a positive impact as a tool for seasonal corrections on the given time series.

Nevertheless, F-SARIMA still performs well compared to the random forest, which could be since SARIMA specialises in univariate time series [44]. In contrast, the random forest could perform better on multivariate data sets [40]. Also, the application of Fourier seasonality within the random forest might have the same, suboptimal effect as in the F-SARIMA case - compared to the SARIMA model. Therefore, comparing the random forest function to the F-SARIMA function is an equal comparison. Still, the F-SARIMA model performs better than the random forest model on every predictive performance aspect. Applying seasonal decomposition or seasonal differencing could improve the random forest forecasts in the given time series.

5.2 Application in-practice

Implementation Complexity

The implementation of the models employs the R language and its libraries for simplifying the implementation process. However, not all libraries are as easy to implement, and some models need additional code to make it work in the presented cases.

The most accessible model for implementing is the SARIMA model. Through the package by Hyndman, it is relatively easy to get a grasp on the processes necessary for modelling and forecasting the time series. The simple formation of models makes the learning curve sharper, as one can quickly identify and adapt SARIMA models if the previous case seems suboptimal. Additionally, one can find a broad literature base to find support once issues arise. In the case of applying model generation or rolling forecasts, the extensions of code mainly concern the formation of loops, which is not complex for a developer. However, the code might cause some issues when implemented by a user unfamiliar with R or programming languages in general.

The state-space models also are quite unsophisticated to implement: the structure of each component is clear, and one does not necessarily need to know the ins and outs of the model itself. However, the literature base on hyperparameter formation is not extensive in the specific context of time series analysis. Besides, if one has issues with the package, most of the relevant sources available are written by Petris and Petrone themselves - designers of the package. Though their quality-of-work is well, issues arising from standard usage are hard to cover by only two writers. Albeit the formation of the model itself is simple and intuitive, the support through literature is less extensive, which makes the implementation more sophisticated than the SARIMA principle.

The random forest needs pre-processing for time series and the translation of the time series to matrices involves quite some transitions, especially when one is applying trends and seasonal corrections. The utilised library for random forests is simple to use and has an extensive literature base to solve possible issues. The time series application has limited though extensive literature. It is doable if one understands how to implement R. Implementation of model generation and rolling forecasts are as straightforward as the implementation of the principles at SARIMA. However, the code is, compared to the other two methods, quite extensive and inexperienced users of R should beware of becoming overwhelmed by the lines of code.

All in all, SARIMA is the most convenient method for implementation due to the clean methods provided in the libraries and the extensive literature base to investigate once one faces issues. The components structure of the state-space model feels intuitive and clarifies the formation of the model in terms of basic components. Besides, the implementation of the hyperparameter estimation methods is straightforward. However, support on a *correct* execution of the model through a correct hyperparameter and components structure is limited, which can make it a hassle to implement the method successfully. Though random forest needs a relatively long function to operate, the code and principles implemented are understandable and documented well in literature. In terms of programming experience needed to implement the functions correctly, SARIMA is the simplest to implement, followed by random forest for time series and lastly, the state-space models. Note that outcomes on only one univariate time series form this judgement. Successful implementation of state-space models through an accurate hyperparameter estimation by having a different time series could have caused state-space models to be easier to implement than random forests for time series.

Understanding of models' principles

Understanding of the models is the foundation for model composition and parameter choices and is - in most cases - a necessity for successful implementations. Complex models are more likely to form a boundary in the usage of the models in practice. Therefore, the models should be easy to understand while still assessing well predictive performance. The following two perspectives enlight each model: the *ease* of understanding in general, and the *necessity* of understanding for the implementation of the models.

The ease of understanding the SARIMA model is simple: one has an excellent chance to find out each components' function in the model. As the seasonal components use the principles of the original ARIMA components, one needs to understand the autoregressive, integrated (differencing), and moving average components as central principles. Besides, the translation of the ARIMA components to Seasonal ARIMA components through different lags selection is also necessary for understanding the model. As the underlying principles are intuitive and straightforward, the underlying model is easy to understand if one wants to.

However, this is not a necessity for applying the model. For model composition and parameter choices at SARIMA, one can follow the two strategies mentioned in this paper: either the theoretical strategy or the model generation strategy. Both strategies differ in the level of understanding necessary for an optimal model formation. For the theoretical approach, one will need to get a grasp on the interpretation of ACF and PACF plots to choose the order of each parameter. Also, recognition of patterns in both the original time series and the underlying models' residuals is essential for correcting the order of each parameter and optimising the models. Note that only interpretation of the plots is necessary - not necessarily the understanding of the underlying principles - which makes the process of choosing the orders of parameters simple if one has an

overview to do so. A simple step-by-step execution, as done in this paper, lead to one of the optimal SARIMA models. Thereby, for this time series, finding the optimal SARIMA model through theory did not require much understanding of the SARIMA models. An approach which does not need any understanding at all is the formation of the models through model generation. If the user follows the methods used in this paper, only computational power and understanding of R are necessary for the formation of the optimal SARIMA model.

Random forest requires an understanding of the principles for choosing accurate values for the number of node splits, depth of the decision trees, and the number of trees to generate. The principles behind Random Forest have extensive explanations through literature. Also, one can visualise the random forest, which makes the model and its use of decision trees simple to understand. However, if one is diving deeper into the material, the available methods of bagging and boosting for optimising the random forest can be confusing due to the high number of variations in literature.

Nevertheless, the core principles of the random forest make the method understandable, intuitive and applicable in a quick manner. Besides the core principles of random forest, application of random forests on time series does require an understanding of the formation of matrices as it uses the theory to translate time series and its seasonal components to the data form random forest can handle. As random forests only need initialisation of two parameters which seem to have a limited impact on the performance of the model, the random forest is scalable to other time series: the model learns the variables it utilises in forecasting itself based on the given time series. However, the formation of the variables can also be a downside of a random forest. The model only returns the trees without an explanation. The unexplained trees make the process of finding out the main variables random forest bases its judgement upon difficult. Thereby, the random forest variables selection can be a black-box for the user, which causes a lack of understanding in the specified underlying model. Subsequently, it is harder for the user to slightly adapt or optimise the model by-hand since each time series has its specification of variables.

The state-space model is a complex model which uses matrices and normally distributed errors in two main equations to form the model. Though the principles and components used in state-space models, such as Gaussian errors and the Hidden-Markov Chain, are understandable, the implementation of several components can be hard to grasp. Every variable will need its specific position in a variable matrix for a correct implementation, which can confuse the view of what happens in the underlying model formation. Due to limited information in literature for time series applications and the utilised library, understanding the actual workings of the combinations of components for the state-space model - such as the polynomial, seasonal, and ARMA component - can be a hassle. Estimating the components is an integral part in forming the model, as the estimations shape the components to the time series used in the specific case. Once one tries to implement the hyperparameter estimation in the given setting, quite some issues can arise of which the literature base is limited. In this paper, issues emerged in the form of a seemingly non-convergent maximum likelihood estimation which was said to be convergent in the output. The future work section further investigates solvations of the issue. The other estimation method - Gibbs sampling - could not converge the estimations for hyperparameters either. Though the issue of convergence might be harder than average on the utilised time series, the fact that there is a risk for an intensive hyperparameter estimation makes the model less attractive and harder to interpret results.

Computational Costs

The efficiency of a model can make an impact on the applicability of the models in-practice, which makes it a relevant factor to enlighten. Since other applications might involve long-term forecasts and the formation of models and its parameters differ in running costs in terms of time, computational costs can be an important aspect for deciding whether or not to use a model. Note that this paper only uses indications of time for models running on a Windows system at which other small applications ran parallel. Also, the length of a time series and the complexity of a time series can hugely affect the computational costs necessary for finding the optimal model. Nevertheless, regarding the time indications, every model has run at least 30 times which stabilised the running time costs for each model. The given running periods are rough estimations of the actual running time on the utilised Windows machine for the specific time series.

The formation of a SARIMA model is efficient takes seconds to minutes for the used time series, depending on the complexity of the model. When one knows about applying the theoretical approach, the SARIMA model needs to be generated around six times at maximum, resulting in a total running time of 15 minutes when having a computationally complex SARIMA model. Thereby, the SARIMA model is very efficient and is suitable in quickly changing environments. However, if one wants to fit the best model through model generation, the calculation times increase significantly to 2 hours and 15 minutes, which makes it harder to apply on, e.g. real-time data. Nevertheless, by planning model generation once a week and utilise the best model throughout the rest of the week, the model can still be used frequently. Since the underlying history of data is unlikely to change drastically in most cases, a less frequent model generation should not be a problem.

Fourier SARIMA models are more computationally expensive than SARIMA, and when running through model generation, it can take ten to fifteen hours to find the optimal F-SARIMA model. Computationally advanced F-SARIMA functions, with high orders of SARIMA parameters and a high order of K, took roughly 1 hour and 30 minutes to execute. A computationally simple F-SARIMA model takes roughly 2 minutes to execute. The Fourier component thereby strongly affects the efficiency of the SARIMA model, which might affect the usage in-practice. It still is possible to plan the F-SARIMA model generation function over-night on a daily basis. However, it is advisable to do so every week, especially considering a possible longer execution at different time series.

The random forest function is quite efficient and only needs about five minutes to execute for the given time series. Thereby, the usability of the model in-practice in terms of computational costs is comparable with the SARIMA model. Pro of applying random forests over the other models is that the random forest directly fits the newly generated decision trees to the newly available data, resulting in a fit which always suits the newest time series.

The execution of state-space models itself is quick, and if the hyperparameters are known, it only takes seconds to generate the model and the forecast. However, the initialisation of hyperparameters has proven to be a time-intensive practice. If one decides to utilise seasonal components, the number of hyperparameters to estimate equals a seasonal period minus one. So, instead of estimating three variables at applying the polynomial and ARMA components, the function has to estimate 54 for weekly data with a seasonal trend over one year. Note that if one manages to find the actual hyperparameter estimations for a given time series, one can reuse the values as a starting point in optimising the state-space models to future time series extensions. So, theoretically, the efficiency of the full calculation - which has taken 30 hours in this paper without the desired result - can be made highly more efficient by already having the right hyperparameters for a time series in the past. Due to non-convergent hyperparameters

in this paper, it is difficult to give an impression of computational costs for estimating and re-estimating the hyperparameters. Thereby, in the future, this could be considered once having a fully functioning, optimal state-space model. Concluding, the state-space model tends to be interesting for real-time time series analysis, though this paper does not have the proof to give definite statements on this topic.

5.3 Future Work

5.3.1 State-space models

The hyperparameter estimation for the state-space models tested in this paper did not converge. Consequently, future work would be to apply several methods for a convergent hyperparameter estimation. The estimation functions used in this paper, which are the Gibbs sampler and maximum likelihood estimation, could run for longer timespans to make them converge. However, for a broad comparison of state-space models having efficient estimation functions are important for higher velocity in model comparisons. Reaching a higher efficiency in estimation could be done through further investigating combinations of both the MLE and Gibbs sampler, where the MLE could provide the starting point for the Gibbs sampler. Hyperparameter starting points close to the actual values decrease the iterations necessary for the Gibbs sampler to converge and thereby decreases running costs of the sampler. To further utilise and improve the starting point through the MLE, one could implement the MLE by hand and see what values for hyperparameters perform well as an initial starting point. Also, one can plot the output in graphs to see what is going on in the likelihood estimation. Once the MLE works well, the output could be a starting point for the Gibbs sampler for the final convergence of hyperparameters.

Once one manages to optimise the estimations of hyperparameters, further investigation of the coefficients of the ARMA components and the choice for the order of ARMA could be the next step to find out what model suits the time series best. Estimating the coefficients for ARMA could be done by the MLE. Afterwards, the true values for the hyperparameters could be estimated to obtain a fully fitted model. Having several convergent models makes them comparable, and thereby this process should be repeated for several orders of ARMA and several coefficients for ARMA to find the model fitting the underlying time series. Afterwards, one can compare the model specifications through the accuracy measures used in this paper. By applying this structured comparison to all kinds of state-space models, one can form optimisation of the models as structured as the SARIMA models' optimisation given in this paper. Though this is likely to be a computationally expensive task, the pros of state-space models mentioned in the computational costs section might make it worth it to initialise and utilise the best state-space model. Besides, when executing this task in a structured manner as described here, possible improvements for the efficiency of the model generation might come up as well, which could decrease the high computational costs. Improvements could be in the form of first estimating components separately and combining those afterwards, or by having a stationary time series as input to save the computational costs caused by the polynomial and seasonal components. Note that if one chooses to apply seasonally and trends correction outside the state-space model, one might not employ the full capacity of the state-space estimations. Therefore, the advice is to implement the seasonality and trend transformations as a component of the state-space estimation - as one can observe the full potential of state-space models in this particular setting.

Lastly, one could investigate an extension of the ARMA components with the seasonal ARMA components as it seems to fit the utilised time series well and the extension could cause a broader set of time series to fit in the model. Current scientific work does not yet apply the seasonal ARMA components for state-space models. Suggested is to

apply the Seasonal ARMA components either through the inclusion of the hyperparameters in the matrices – resulting in a new state-space model component – or by applying the seasonal ARMA transformations on forehand through utilising the residuals in state-space modelling. Though advanced, the first option would be the most interesting case to investigate as one utilises the full capacity of state-space modelling.

5.3.2 Multivariate Time Series

This paper identifies the predictive behaviour of univariate time series - a set of data points with a time variable and a continuous variable. Multivariate time series have several continuous variables in a time series structure and thereby can form a more comprehensive view on the patterns throughout the data to utilise in forecasting. In SARIMA, application of multivariate models is possible, though it does not necessarily improve predictive performance. It would be interesting to see whether an optimised SARIMA model for the given multivariate time series would lead to an improvement. The forecast package by Hyndman already provides the functionality of multivariate time series, and thereby the boundary should be low in applying the multivariate time series, once applicable. Ditto for multivariate time series as input for random forests, which will require a new data translation from the variable into the matrix. The favour of random forests over SARIMA is that the implementation should be easier and the formation of the model is likely to be more simple than SARIMA models, as SARIMA models require more understanding or computationally extensive model generation algorithms. However, one would need an understanding of R to implement random forests in the given setting.

For the state-space models applied in this paper, the case is a bit harder. Besides the issues of convergence of the parameter estimation, the utilised dlm package can only handle univariate time series. Consequently, this would imply that one will have to implement a state-space model from scratch, which will be a time-extensive task and requires a profound understanding of state-space models. Nevertheless, the extension by multivariate time series for state-space models proposed in this paper could be exciting as a successful implementation of state-space models might improve the current state-of-art methods used.

5.3.3 Hierarchical Time Series

Another time series extension which could be interesting to investigate in future work is a hierarchical structure of time series. In hierarchical structures, a specified variable groups the time series into several univariate time series at which models run separately. In the case of the utilised time series, the time series could be grouped by the customer to see whether customers show the same ordering patterns throughout the four years. If so, one could model highly predictable customers first, and one could group unpredictable customers to specialise in each customer group and improve the overall predictions.

Shifting the focus to the models utilised in this paper, one could apply all three models to this structure of time series. The random forest seems to be the easiest to apply, as the random forest will take care of variable definitions itself. SARIMA will either require high computational power or a high amount of time and understanding of SARIMA to design a SARIMA model for each time series. For state-space models, the initialisation for one univariate time series already takes hours. The initialisation for multiple univariate time series repeats that process resulting in an inefficient model formation. Due to the high computational costs, which are costly in terms of time, the state-space model does not seem to be the ideal model for hierarchical time series.

6 Conclusion

Demand Forecasting performs best at the utilization of the traditional SARIMA model in terms of predictive performance compared to random forests in the given univariate time series. Though state-space models do have the potential to outperform the SARIMA model, the models applied and estimated in this paper do not give enough evidence to support the hypothesis. In practice, the SARIMA model seems to be the most appropriate model for a single univariate time series to use due to the low computational costs, the ease of understanding of its principles, and the wide implementation support in the literature. However, for wide applicability of the model over multiple univariate time series, the random forest is an efficient alternative for SARIMA models due to its scalability property which makes implementation efficient. The state-space models are the least efficient, hardest to master, and is least supported throughout literature. However, the state-space models do reveal a high potential through its predictive performance. Due to the relative novelty of state-space models for time series analysis, the state-space models do have the potential to outperform the traditional SARIMA and random forest models on predictive performance once the model has a more comprehensive literature base. However, one should consider whether the possible improvement in prediction accuracy is worth the investment in both the formation of the model as the computational power necessary to execute the function.

Concluding, the non-convergent state-space model results in an inability to compare SARIMA and state-space models on predictive performance. However, convergent state-space ARMA estimations do reveal the potential to outperform the SARIMA models, though it will be at the expenses of practical difficulties. The SARIMA model outperforms the random forest in predictive performance for the given time series. Still, the random forest for time series does have potential due to its efficient scalability to other time series.

Appendices

Appendix I - Packages for implementation

Functionality	Package	Function	Reference
KPSS test	urca	ur.kpss	[32]
ADF test	aTSA	adf.test	[33]

Table 14: Stationarity packages and functions

Functionality	Package	Function	Reference
Fourier	forecast	fourier	[19]
Differencing	Stats	diff	[34]
Decomposition	Stats	decompose / stl	[34]

Table 15: Seasonality packages and functions

Functionality	Package	Function	Reference
SARIMA	forecast	Arima	[19]
ACF/PACF	Stats	acf / pacf	[34]
AIC/BIC	Stats	AIC / BIC	[34]
Forecasting	forecast	forecast	[19]
Accuracy measures	forecast	accuracy	[19]

Table 16: SARIMA packages and functions

Functionality	Package	Function	Reference
Polynomial component	dlm	dlmModPoly	[31]
Seasonal component	dlm	dlmModSeas	[31]
ARMA component	dlm	dlmModARMA	[31]
Gibbs sampler	dlm	dlmGibbsDIG	[31]
Maximum likelihood estimation	dlm	dlmMLE	[31]
Kalman filter	dlm	dlmFilter	[31]
Forecasting	dlm	dlmForecast	[31]

Table 17: State-space model packages and functions

Functionality	Package	Function	Reference
Random forest	randomForest	randomForest	[26]
Decomposition	Stats	stl	[34]
Trends forecast	forecast	auto.arima	[19]
MAPE	forecast	accuracy	[19]

Table 18: Random forest packages and functions

Appendix II - ACF and PACF plots

Residuals different time series SARIMA(0,0,0)(0,0,0)

Section 4.2.1: Utilized for defining order of P / Q

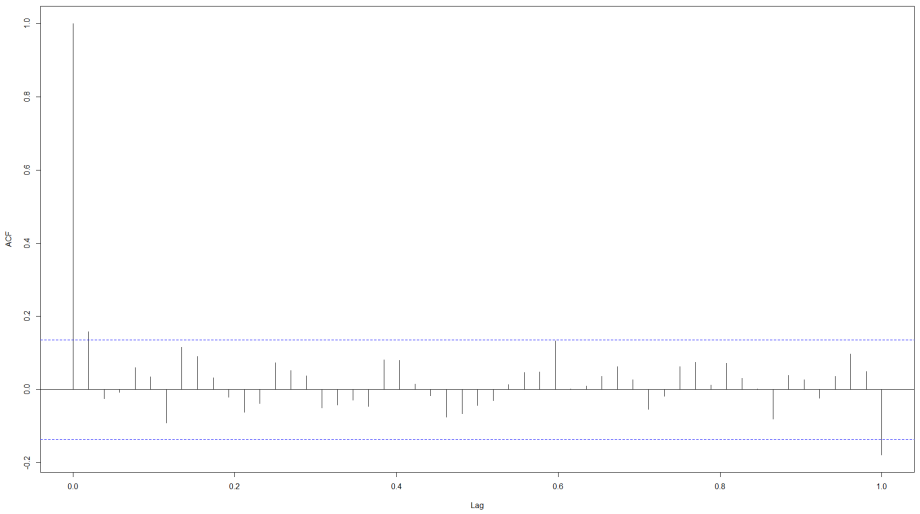

Figure 17: ACF - SDec1 Time Series

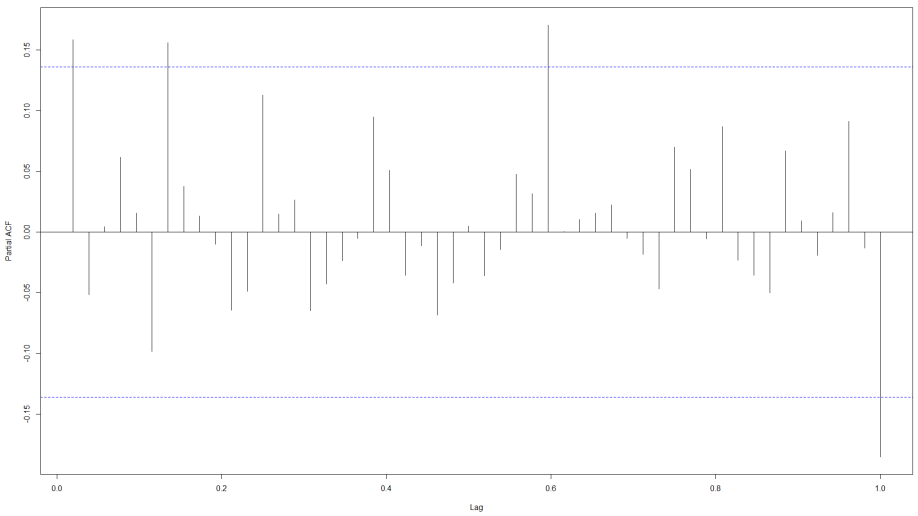

Figure 18: PACF - SDec1 Time Series

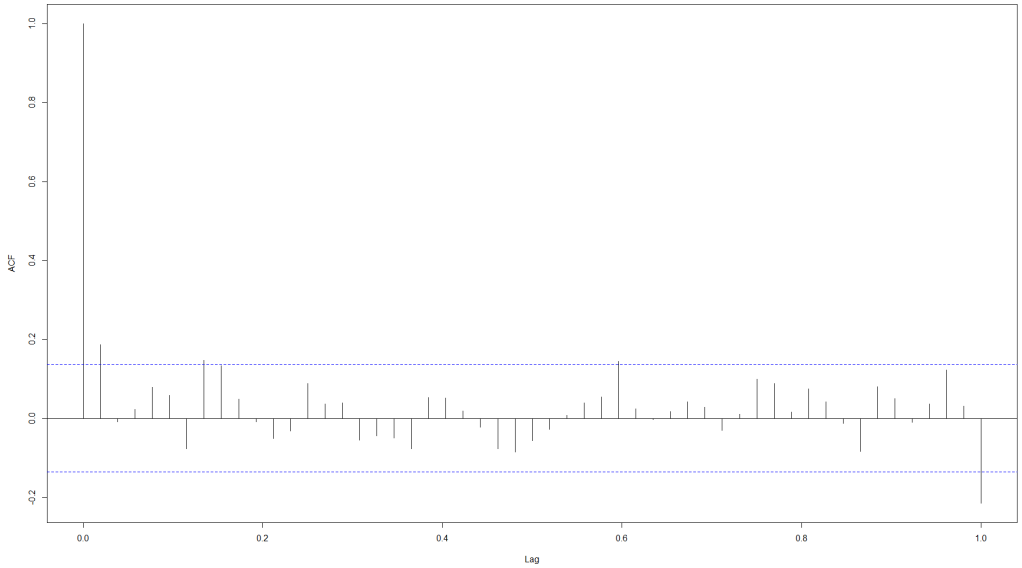

Figure 19: ACF - SDec2 Time Series

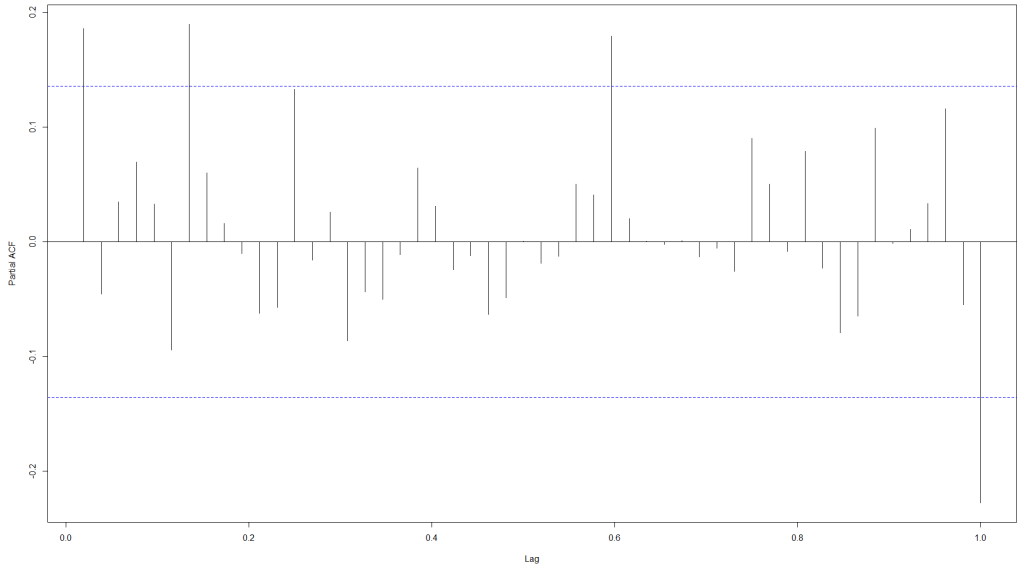

Figure 20: PACF - SDec2 Time Series

Residuals different time series SARIMA(0,0,0)(0,0,1)

Section 4.2.1: Utilized for defining order of p / q

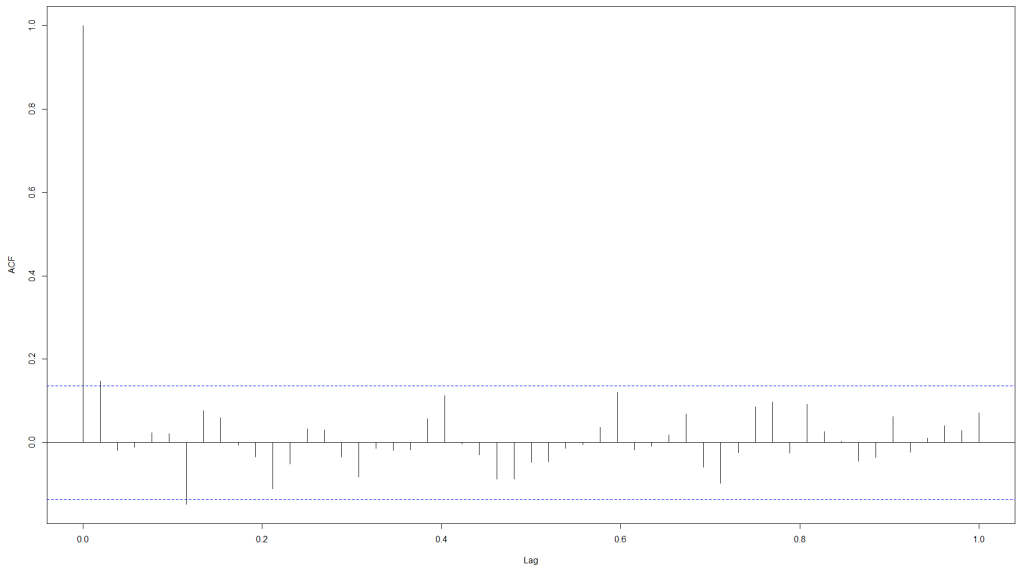

Figure 21: ACF - SDec1 Time Series, Q = 1

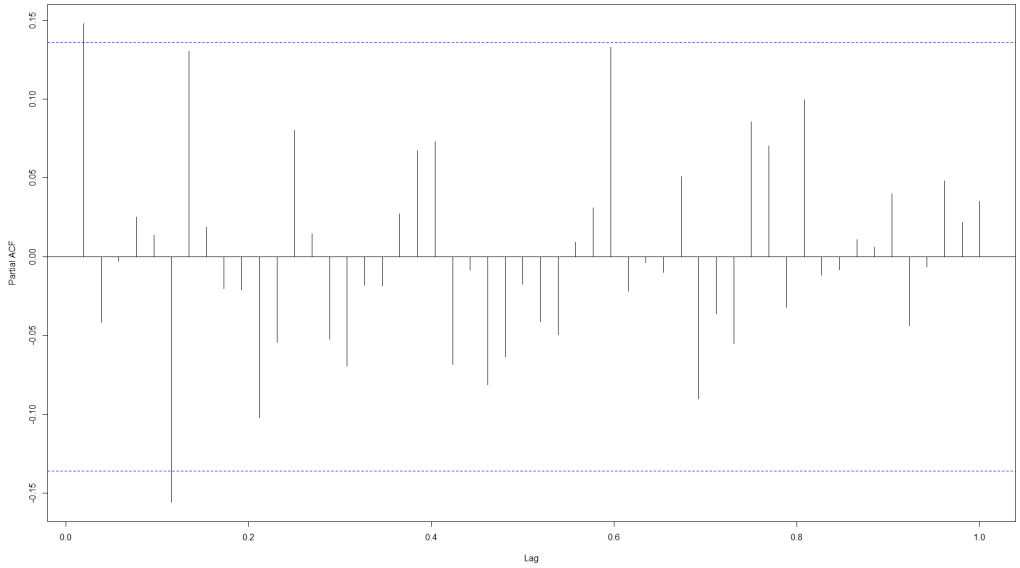

Figure 22: PACF - SDec1 Time Series, Q = 1

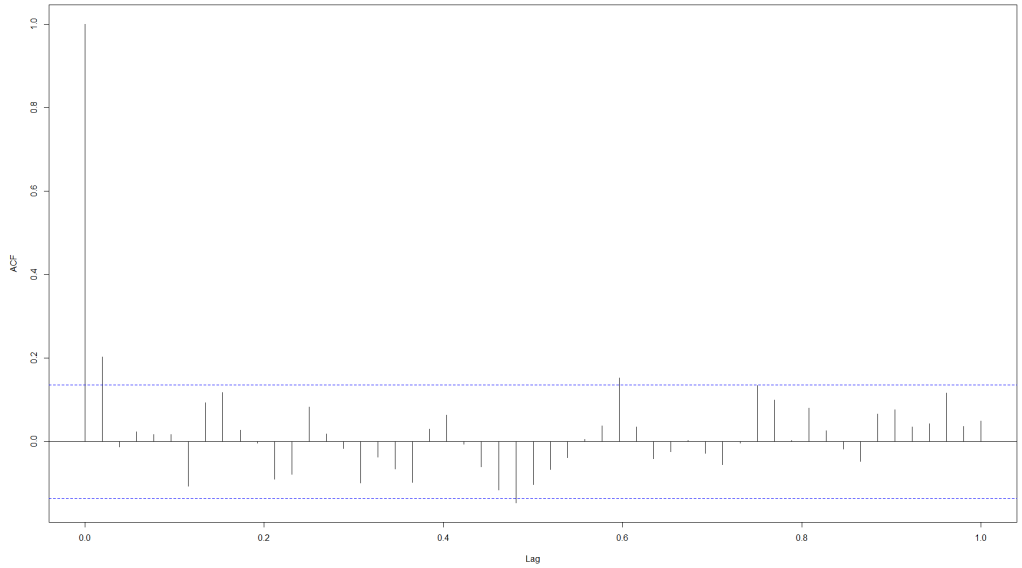

Figure 23: ACF - SDec2 Time Series, Q = 1

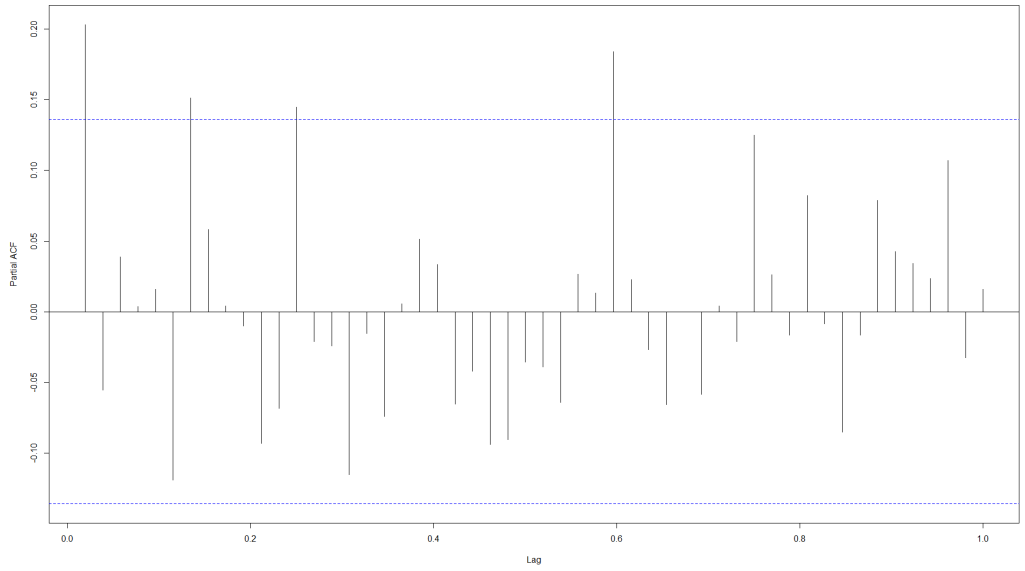

Figure 24: PACF - SDec2 Time Series, Q = 1

Residuals different time series SARIMA(0,0,1)(0,0,1)

Section 4.2.1: Final residuals of each theoretically based model

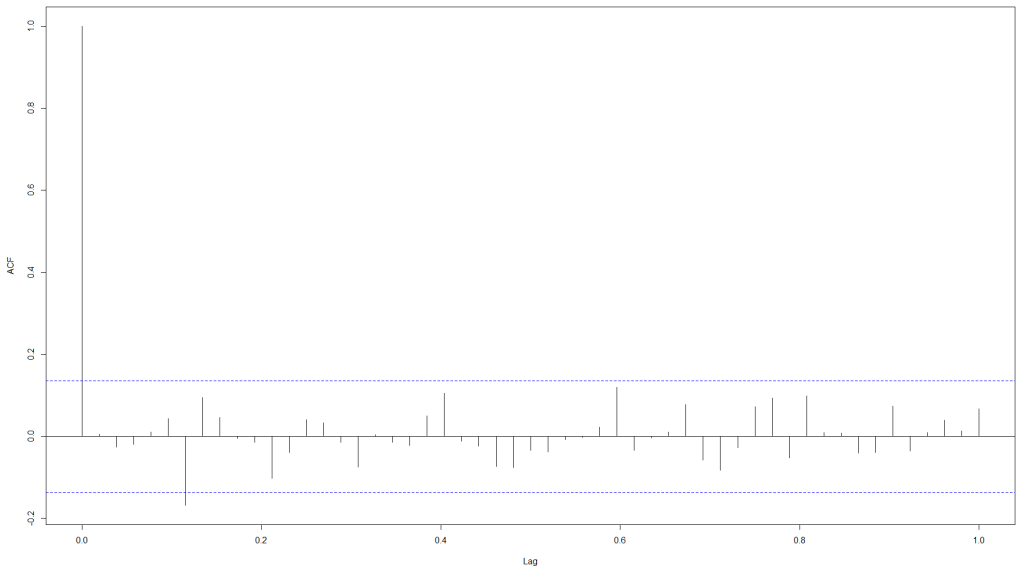

Figure 25: ACF - SDec1 Time Series SARIMA(0,0,1)(0,0,1)

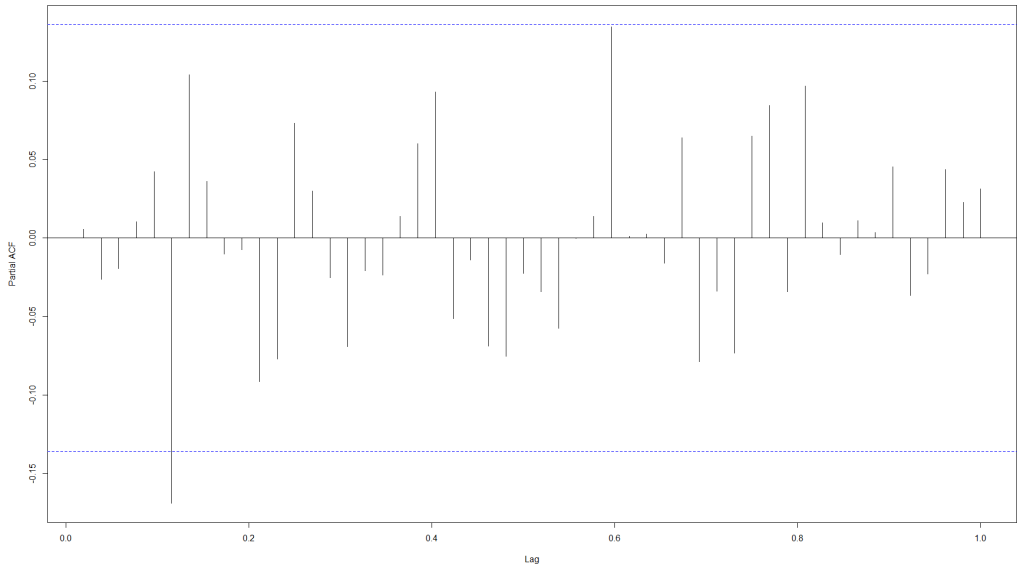

Figure 26: PACF - SDec1 Time Series SARIMA(0,0,1)(0,0,1)

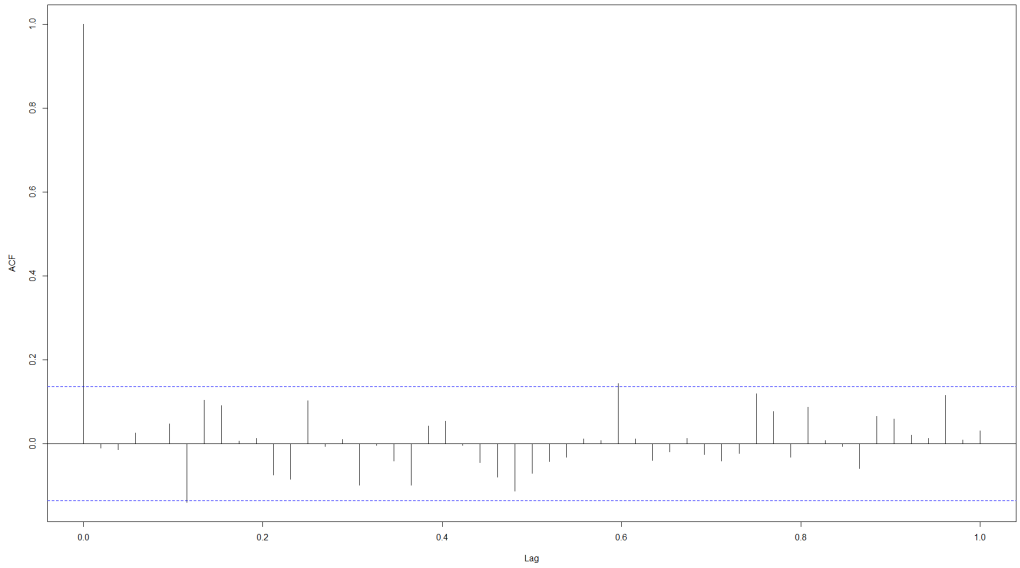

Figure 27: ACF - SDec2 Time Series SARIMA(0,0,1)(0,0,1)

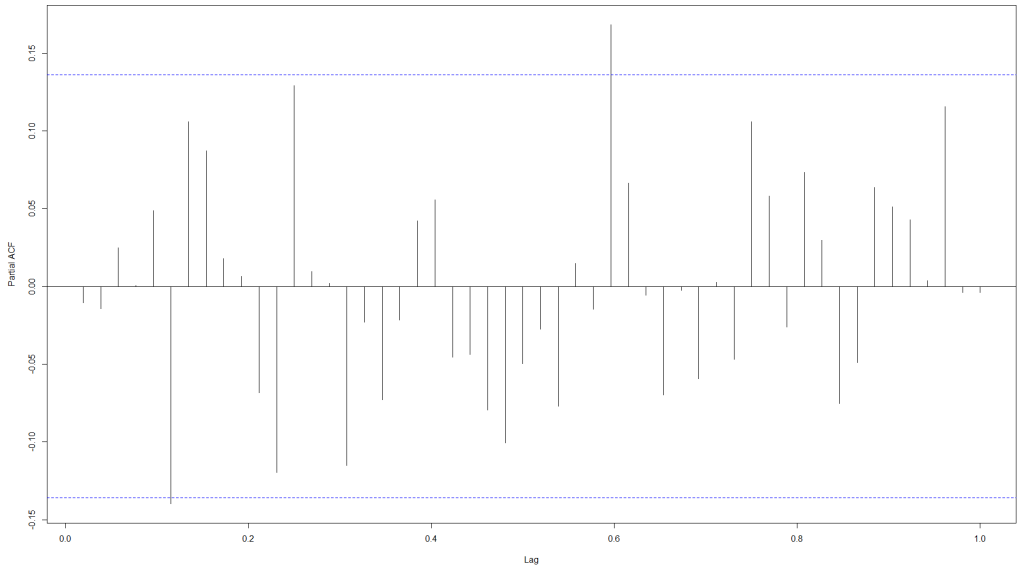

Figure 28: PACF - SDec2 Time Series SARIMA(0,0,1)(0,0,1)

Appendix III - AIC and BIC values

Original Time series

Section 4.2.1: Utilized in Model Generation SARIMA

AIC	BIC	AICC	p	d	q	P	D	Q
6336	6356	6336	0	0	1	0	0	2
6336	6356	6336	1	0	0	0	0	2
6337	6363	6337	1	0	2	0	0	2
6338	6368	6339	2	0	2	0	0	2
6338	6361	6338	1	0	1	0	0	2
6338	6361	6338	0	0	2	0	0	2
6338	6361	6338	2	0	0	0	0	2
6339	6366	6340	2	0	1	0	0	2
6344	6361	6345	0	0	0	0	0	2
6349	6372	6349	1	0	2	0	0	1
6349	6365	6349	1	0	0	0	0	1
6349	6366	6349	0	0	1	0	0	1
6350	6376	6350	2	0	2	0	0	1
6351	6371	6351	1	0	1	0	0	1
6351	6371	6351	2	0	0	0	0	1
6351	6371	6351	0	0	2	0	0	1
6353	6376	6353	2	0	1	0	0	1
6356	6370	6357	0	0	0	0	0	1
6379	6396	6379	1	0	1	0	0	0
6380	6401	6381	1	0	2	0	0	0
6381	6401	6381	2	0	1	0	0	0
6382	6405	6383	2	0	2	0	0	0
6386	6399	6386	1	0	0	0	0	0
6386	6402	6386	2	0	0	0	0	0
6389	6405	6389	0	0	2	0	0	0
6389	6402	6389	0	0	1	0	0	0
6402	6412	6402	0	0	0	0	0	0

Table 19: SARIMA Model Generation - UTS - Category 1: d = 0, D = 0

AIC	BIC	AICC	p	d	q	P	D	Q
6310	6330	6311	1	1	1	0	0	2
6310	6330	6311	0	1	2	0	0	2
6312	6336	6313	1	1	2	0	0	2
6313	6336	6314	2	1	1	0	0	2
6313	6330	6314	0	1	1	0	0	2
6314	6340	6315	2	1	2	0	0	2
6324	6344	6325	2	1	1	0	0	1
6324	6338	6325	0	1	1	0	0	1
6325	6348	6325	2	1	2	0	0	1
6325	6341	6325	0	1	2	0	0	1
6325	6342	6325	1	1	1	0	0	1
6326	6346	6326	1	1	2	0	0	1
6343	6363	6343	2	1	0	0	0	2
6354	6364	6354	0	1	1	0	0	0
6356	6369	6356	0	1	2	0	0	0
6356	6376	6356	2	1	2	0	0	0
6356	6369	6356	1	1	1	0	0	0
6356	6373	6356	2	1	0	0	0	1
6356	6373	6357	2	1	1	0	0	0
6357	6374	6357	1	1	2	0	0	0
6361	6378	6361	1	1	0	0	0	2
6376	6390	6377	1	1	0	0	0	1
6379	6392	6379	2	1	0	0	0	0
6386	6399	6386	0	1	0	0	0	2
6396	6406	6396	1	1	0	0	0	0
6402	6412	6402	0	1	0	0	0	1
6425	6431	6425	0	1	0	0	0	0

Table 20: SARIMA Model Generation - UTS - Category 2: d = 1, D = 0

AIC	BIC	AICC	p	d	q	P	D	Q
4725	4737	4725	0	0	1	0	1	1
4725	4738	4726	1	0	0	0	1	1
4727	4742	4727	2	0	0	0	1	1
4727	4742	4727	0	0	2	0	1	1
4727	4742	4727	1	0	1	0	1	1
4727	4742	4727	0	0	1	0	1	2
4727	4743	4728	1	0	0	0	1	2
4728	4737	4728	0	0	0	0	1	1
4729	4750	4729	2	0	2	0	1	1
4729	4747	4729	2	0	0	0	1	2
4729	4747	4729	0	0	2	0	1	2
4729	4747	4729	2	0	1	0	1	1
4729	4747	4729	1	0	1	0	1	2
4729	4747	4729	1	0	2	0	1	1
4730	4742	4730	0	0	0	0	1	2
4731	4755	4732	2	0	2	0	1	2
4731	4752	4731	2	0	1	0	1	2
4731	4752	4732	1	0	2	0	1	2
4754	4773	4755	2	0	2	0	1	0
4757	4766	4757	0	0	1	0	1	0
4757	4770	4758	2	0	0	0	1	0
4758	4767	4758	1	0	0	0	1	0
4758	4770	4758	0	0	2	0	1	0
4758	4771	4759	1	0	1	0	1	0
4759	4765	4759	0	0	0	0	1	0
4759	4775	4760	2	0	1	0	1	0
4760	4775	4760	1	0	2	0	1	0

Table 21: SARIMA Model Generation - UTS - Category 3: d = 0, D = 1

AIC	BIC	AICC	p	d	q	P	D	Q
4701	4713	4701	0	1	2	0	1	1
4702	4714	4702	1	1	1	0	1	1
4702	4717	4702	1	1	2	0	1	1
4703	4718	4703	2	1	1	0	1	1
4703	4718	4703	0	1	2	0	1	2
4704	4719	4704	1	1	1	0	1	2
4704	4722	4704	1	1	2	0	1	2
4704	4713	4704	0	1	1	0	1	1
4705	4723	4705	2	1	1	0	1	2
4705	4723	4705	2	1	2	0	1	1
4706	4718	4706	0	1	1	0	1	2
4707	4728	4708	2	1	2	0	1	2
4731	4744	4732	2	1	1	0	1	0
4732	4741	4732	0	1	2	0	1	0
4732	4745	4733	1	1	2	0	1	0
4732	4742	4733	1	1	1	0	1	0
4733	4739	4733	0	1	1	0	1	0
4733	4749	4734	2	1	2	0	1	0
4737	4749	4737	2	1	0	0	1	1
4739	4754	4739	2	1	0	0	1	2
4753	4762	4753	1	1	0	0	1	1
4754	4767	4755	1	1	0	0	1	2
4772	4782	4773	2	1	0	0	1	0
4773	4779	4773	0	1	0	0	1	1
4773	4783	4774	0	1	0	0	1	2
4791	4797	4791	1	1	0	0	1	0
4809	4812	4809	0	1	0	0	1	0

Table 22: SARIMA Model Generation - UTS - Category 4: d = 1, D = 1

Seasonally Decomposed Time series

Section 4.2.1: Utilized in Model Generation SARIMA

AIC	BIC	AICC	p	d	q	P	D	Q
6146	6166	6147	0	0	1	0	0	2
6146	6166	6147	1	0	0	0	0	2
6147	6177	6148	2	0	2	0	0	2
6148	6165	6148	0	0	1	0	0	1
6148	6171	6149	2	0	0	0	0	2
6148	6172	6149	0	0	2	0	0	2
6148	6172	6149	1	0	1	0	0	2
6148	6165	6149	1	0	0	0	0	1
6149	6165	6149	0	0	0	0	0	2
6150	6170	6150	2	0	0	0	0	1
6150	6170	6150	0	0	2	0	0	1
6150	6170	6150	1	0	1	0	0	1
6150	6177	6151	2	0	1	0	0	2
6150	6177	6151	1	0	2	0	0	2
6151	6164	6151	0	0	0	0	0	1
6151	6174	6151	2	0	1	0	0	1
6152	6175	6152	1	0	2	0	0	1
6153	6179	6153	2	0	2	0	0	1
6171	6184	6171	0	0	1	0	0	0
6172	6185	6172	1	0	0	0	0	0
6172	6195	6173	2	0	2	0	0	0
6173	6189	6173	2	0	0	0	0	0
6173	6189	6173	0	0	2	0	0	0
6173	6190	6173	1	0	1	0	0	0
6174	6195	6175	2	0	1	0	0	0
6175	6185	6175	0	0	0	0	0	0
6175	6195	6175	1	0	2	0	0	0

Table 23: SARIMA Model Generation - SDec1 - Category 1: d = 0, D = 0

AIC	BIC	AICC	p	d	q	P	D	Q
6124	6144	6124	0	1	2	0	0	2
6124	6144	6124	1	1	1	0	0	2
6125	6149	6126	2	1	2	0	0	1
6126	6142	6126	0	1	2	0	0	1
6126	6149	6126	2	1	1	0	0	2
6126	6149	6126	1	1	2	0	0	2
6126	6142	6126	1	1	1	0	0	1
6126	6143	6127	0	1	1	0	0	2
6127	6147	6128	2	1	1	0	0	1
6128	6148	6128	1	1	2	0	0	1
6128	6154	6128	2	1	2	0	0	2
6129	6142	6129	0	1	1	0	0	1
6147	6161	6148	0	1	2	0	0	0
6148	6161	6148	1	1	1	0	0	0
6149	6166	6149	2	1	1	0	0	0
6149	6166	6150	1	1	2	0	0	0
6151	6171	6151	2	1	2	0	0	0
6151	6161	6151	0	1	1	0	0	0
6174	6194	6175	2	1	0	0	0	2
6175	6192	6175	2	1	0	0	0	1
6190	6207	6190	1	1	0	0	0	2
6192	6205	6192	1	1	0	0	0	1
6200	6214	6200	2	1	0	0	0	0
6218	6232	6218	0	1	0	0	0	2
6219	6229	6219	1	1	0	0	0	0
6220	6230	6220	0	1	0	0	0	1
6244	6251	6244	0	1	0	0	0	0

Table 24: SARIMA Model Generation - SDec1 - Category 2: d = 1, D = 0

AIC	BIC	AICC	p	d	q	P	D	Q
4731	4743	4731	0	0	1	0	1	1
4731	4744	4732	1	0	0	0	1	1
4732	4747	4732	2	0	0	0	1	1
4732	4747	4733	0	0	2	0	1	1
4732	4748	4733	1	0	1	0	1	1
4733	4754	4733	2	0	2	0	1	1
4733	4748	4733	0	0	1	0	1	2
4733	4742	4733	0	0	0	0	1	1
4733	4749	4734	1	0	0	0	1	2
4734	4752	4734	2	0	0	0	1	2
4734	4752	4735	2	0	1	0	1	1
4734	4752	4735	1	0	2	0	1	1
4734	4752	4735	0	0	2	0	1	2
4734	4753	4735	1	0	1	0	1	2
4734	4759	4735	2	0	2	0	1	2
4735	4747	4735	0	0	0	0	1	2
4736	4757	4737	2	0	1	0	1	2
4736	4757	4737	1	0	2	0	1	2
4761	4779	4761	2	0	2	0	1	0
4763	4775	4763	2	0	0	0	1	0
4763	4773	4764	0	0	1	0	1	0
4764	4776	4764	0	0	2	0	1	0
4764	4773	4764	1	0	0	0	1	0
4764	4770	4764	0	0	0	0	1	0
4765	4777	4765	1	0	1	0	1	0
4765	4780	4765	2	0	1	0	1	0
4766	4781	4766	1	0	2	0	1	0

Table 25: SARIMA Model Generation - SDec1 - Category 3: d = 0, D = 1

AIC	BIC	AICC	p	d	q	P	D	Q
4707	4719	4707	0	1	2	0	1	1
4707	4722	4708	1	1	2	0	1	1
4708	4720	4708	1	1	1	0	1	1
4708	4723	4708	2	1	1	0	1	1
4709	4718	4709	0	1	1	0	1	1
4709	4724	4709	0	1	2	0	1	2
4709	4727	4710	1	1	2	0	1	2
4709	4725	4710	1	1	1	0	1	2
4710	4728	4710	2	1	1	0	1	2
4710	4728	4711	2	1	2	0	1	1
4710	4723	4711	0	1	1	0	1	2
4712	4733	4713	2	1	2	0	1	2
4737	4749	4737	2	1	1	0	1	0
4738	4747	4738	0	1	2	0	1	0
4738	4750	4739	1	1	2	0	1	0
4739	4745	4739	0	1	1	0	1	0
4739	4748	4739	1	1	1	0	1	0
4739	4754	4739	2	1	2	0	1	0
4745	4758	4746	2	1	0	0	1	1
4747	4763	4748	2	1	0	0	1	2
4761	4770	4761	1	1	0	0	1	1
4763	4775	4763	1	1	0	0	1	2
4779	4786	4780	0	1	0	0	1	1
4781	4790	4781	0	1	0	0	1	2
4781	4790	4781	2	1	0	0	1	0
4798	4804	4798	1	1	0	0	1	0
4815	4818	4815	0	1	0	0	1	0

Table 26: SARIMA Model Generation - SDec1 - Category 4: d = 1, D = 1

Seasonally Decomposed by STL Time series

Section 4.2.1: Utilized in Model Generation SARIMA

AIC	BIC	AICC	p	d	q	P	D	Q
6080	6107	6081	1	0	2	0	0	2
6080	6107	6081	2	0	1	0	0	2
6080	6104	6081	1	0	2	0	0	1
6080	6104	6081	2	0	1	0	0	1
6082	6102	6082	0	0	1	0	0	2
6082	6112	6083	2	0	2	0	0	2
6082	6109	6083	2	0	2	0	0	1
6082	6099	6083	0	0	1	0	0	1
6083	6103	6083	1	0	0	0	0	2
6083	6100	6084	1	0	0	0	0	1
6084	6107	6084	1	0	1	0	0	2
6084	6107	6084	0	0	2	0	0	2
6084	6107	6084	2	0	0	0	0	2
6084	6104	6085	1	0	1	0	0	1
6084	6104	6085	0	0	2	0	0	1
6084	6105	6085	2	0	0	0	0	1
6088	6104	6088	0	0	0	0	0	2
6091	6104	6091	0	0	0	0	0	1
6141	6155	6142	0	0	1	0	0	0
6142	6155	6142	1	0	0	0	0	0
6143	6166	6144	2	0	2	0	0	0
6143	6160	6143	2	0	0	0	0	0
6143	6160	6143	0	0	2	0	0	0
6143	6160	6144	1	0	1	0	0	0
6145	6165	6146	2	0	1	0	0	0
6145	6165	6146	1	0	2	0	0	0
6146	6156	6146	0	0	0	0	0	0

Table 27: SARIMA Model Generation - SDec2 - Category 1: d = 0, D = 0

AIC	BIC	AICC	p	d	q	P	D	Q
6058	6085	6059	2	1	2	0	0	2
6058	6082	6059	2	1	2	0	0	1
6060	6080	6060	0	1	2	0	0	2
6060	6083	6061	1	1	2	0	0	2
6060	6080	6061	1	1	1	0	0	2
6060	6077	6061	0	1	2	0	0	1
6061	6081	6062	1	1	2	0	0	1
6061	6078	6062	1	1	1	0	0	1
6062	6085	6062	2	1	1	0	0	2
6063	6083	6063	2	1	1	0	0	1
6066	6082	6066	0	1	1	0	0	2
6069	6082	6069	0	1	1	0	0	1
6107	6124	6108	2	1	0	0	0	1
6108	6128	6108	2	1	0	0	0	2
6118	6131	6118	0	1	2	0	0	0
6119	6132	6119	1	1	1	0	0	0
6120	6136	6120	2	1	1	0	0	0
6120	6137	6120	1	1	2	0	0	0
6122	6142	6123	2	1	2	0	0	0
6123	6133	6123	0	1	1	0	0	0
6129	6142	6129	1	1	0	0	0	1
6129	6146	6130	1	1	0	0	0	2
6154	6164	6154	0	1	0	0	0	1
6156	6169	6156	0	1	0	0	0	2
6168	6181	6168	2	1	0	0	0	0
6189	6199	6190	1	1	0	0	0	0
6217	6224	6217	0	1	0	0	0	0

Table 28: SARIMA Model Generation - SDec2 - Category 2: d = 1, D = 0

AIC	BIC	AICC	p	d	q	P	D	Q
4725	4737	4725	0	0	1	0	1	1
4725	4738	4726	1	0	0	0	1	1
4727	4742	4727	2	0	0	0	1	1
4727	4742	4727	0	0	2	0	1	1
4727	4742	4727	1	0	1	0	1	1
4727	4742	4727	0	0	1	0	1	2
4727	4743	4728	1	0	0	0	1	2
4728	4737	4728	0	0	0	0	1	1
4729	4750	4729	2	0	2	0	1	1
4729	4747	4729	2	0	0	0	1	2
4729	4747	4729	0	0	2	0	1	2
4729	4747	4729	2	0	1	0	1	1
4729	4747	4729	1	0	1	0	1	2
4729	4747	4729	1	0	2	0	1	1
4730	4742	4730	0	0	0	0	1	2
4731	4755	4732	2	0	2	0	1	2
4731	4752	4731	2	0	1	0	1	2
4731	4752	4732	1	0	2	0	1	2
4754	4773	4755	2	0	2	0	1	0
4757	4766	4757	0	0	1	0	1	0
4757	4770	4758	2	0	0	0	1	0
4758	4767	4758	1	0	0	0	1	0
4758	4770	4758	0	0	2	0	1	0
4758	4771	4759	1	0	1	0	1	0
4759	4765	4759	0	0	0	0	1	0
4759	4775	4760	2	0	1	0	1	0
4760	4775	4760	1	0	2	0	1	0

Table 29: SARIMA Model Generation - SDec2 - Category 3: d = 0, D = 1

AIC	BIC	AICC	p	d	q	P	D	Q
4701	4713	4701	0	1	2	0	1	1
4702	4714	4702	1	1	1	0	1	1
4702	4717	4702	1	1	2	0	1	1
4703	4718	4703	2	1	1	0	1	1
4703	4718	4703	0	1	2	0	1	2
4704	4719	4704	1	1	1	0	1	2
4704	4722	4704	1	1	2	0	1	2
4704	4713	4704	0	1	1	0	1	1
4705	4723	4705	2	1	1	0	1	2
4705	4723	4705	2	1	2	0	1	1
4706	4718	4706	0	1	1	0	1	2
4707	4728	4708	2	1	2	0	1	2
4731	4744	4732	2	1	1	0	1	0
4732	4741	4732	0	1	2	0	1	0
4732	4745	4733	1	1	2	0	1	0
4732	4742	4733	1	1	1	0	1	0
4733	4739	4733	0	1	1	0	1	0
4733	4749	4734	2	1	2	0	1	0
4737	4749	4737	2	1	0	0	1	1
4739	4754	4739	2	1	0	0	1	2
4753	4762	4753	1	1	0	0	1	1
4754	4767	4755	1	1	0	0	1	2
4772	4782	4773	2	1	0	0	1	0
4773	4779	4773	0	1	0	0	1	1
4773	4783	4774	0	1	0	0	1	2
4791	4797	4791	1	1	0	0	1	0
4809	4812	4809	0	1	0	0	1	0

Table 30: SARIMA Model Generation - SDec2 - Category 4: d = 1, D = 1

Appendix IV - F-SARIMA Model Generation

Parameter determination for K

Section 4.2.1: Utilized in Model Generation F-SARIMA

AIC	BIC	AICC	K
7963.899	7981.703	7964.136	1
7956.879	7981.804	7957.324	2
7959.731	7991.777	7960.451	3
7951.287	7990.454	7952.351	4
7949.091	7995.380	7950.571	5
7949.272	8002.683	7951.240	6
7930.353	7990.884	7932.882	7
7922.036	7989.689	7925.203	8
7919.400	7994.174	7923.282	9
7917.535	7999.431	7922.213	10
7912.179	8001.196	7917.735	11
7895.320	7991.459	7901.837	12
7888.136	7991.396	7895.701	13
7875.050	7985.431	7883.752	14
7866.909	7984.411	7876.838	15
7866.647	7991.271	7877.897	16
7856.460	7988.205	7869.126	17
7845.105	7983.972	7859.287	18
7842.125	7988.113	7857.923	19
7833.488	7986.597	7851.006	20
7824.013	7984.244	7843.359	21
7825.979	7993.331	7847.262	22
7822.260	7996.733	7845.593	23
7817.644	7999.239	7843.144	24
7814.288	8003.004	7842.075	25
7807.028	7999.305	7836.004	26

Table 31: F-SARIMA Model Generation - K-value = 26

Parameter determination SARIMA at Fourier K = 26

Section 4.2.1: Utilized in Model Generation F-SARIMA

AIC	BIC	AICC	p	d	q	P	D	Q
7725.945	7932.464	7759.994	1	0	1	0	0	2
7726.779	7936.859	7762.179	2	0	1	0	0	2
7726.783	7936.864	7762.183	1	0	2	0	0	2
7728.776	7942.416	7765.559	2	0	2	0	0	2
7729.849	7932.807	7762.581	0	0	1	0	0	2
7729.954	7932.913	7762.687	1	0	1	0	0	1
7730.711	7933.670	7763.444	1	0	0	0	0	2
7730.985	7937.505	7765.035	0	0	2	0	0	2
7731.522	7938.041	7765.571	2	0	1	0	0	1
7731.531	7938.050	7765.581	1	0	2	0	0	1
7731.563	7938.083	7765.613	2	0	0	0	0	2
7733.905	7943.985	7769.305	2	0	2	0	0	1
7734.518	7933.916	7765.966	0	0	0	0	0	2
7735.430	7934.828	7766.878	0	0	1	0	0	1
7736.118	7935.516	7767.566	1	0	0	0	0	1
7736.267	7939.226	7769.000	0	0	2	0	0	1
7736.895	7939.854	7769.628	2	0	0	0	0	1
7738.472	7934.310	7768.668	0	0	0	0	0	1
7796.764	7999.723	7829.497	1	0	2	0	0	0
7796.865	7999.824	7829.598	2	0	1	0	0	0
7797.623	7997.021	7829.071	1	0	1	0	0	0
7798.455	8004.974	7832.505	2	0	2	0	0	0
7799.072	7994.909	7829.268	0	0	1	0	0	0
7800.352	7996.190	7830.549	1	0	0	0	0	0
7800.424	7999.822	7831.872	0	0	2	0	0	0
7801.082	8000.480	7832.530	2	0	0	0	0	0
7807.028	7999.305	7836.004	0	0	0	0	0	0

Table 32: F-SARIMA Model Generation - Category 1: d = 0, D = 0

AIC	BIC	AICC	p	d	q	P	D	Q
7704.358	7910.654	7738.578	1	1	2	0	0	2
7705.037	7914.889	7740.614	2	1	2	0	0	2
7708.039	7910.778	7740.935	0	1	2	0	0	2
7708.673	7911.413	7741.569	1	1	2	0	0	1
7708.891	7911.630	7741.786	1	1	1	0	0	2
7709.875	7916.171	7744.095	2	1	1	0	0	2
7710.152	7916.448	7744.372	2	1	2	0	0	1
7713.031	7912.214	7744.635	0	1	1	0	0	2
7713.984	7913.167	7745.588	0	1	2	0	0	1
7714.671	7913.853	7746.275	1	1	1	0	0	1
7715.584	7918.323	7748.479	2	1	1	0	0	1
7717.302	7912.928	7747.647	0	1	1	0	0	1
7762.800	7965.539	7795.695	2	1	0	0	0	2
7771.667	7970.850	7803.271	2	1	0	0	0	1
7773.488	7976.227	7806.384	2	1	2	0	0	0
7775.667	7971.292	7806.011	0	1	2	0	0	0
7776.711	7975.894	7808.315	1	1	2	0	0	0
7776.917	7972.542	7807.261	1	1	1	0	0	0
7777.786	7976.968	7809.390	2	1	1	0	0	0
7783.975	7976.044	7813.093	0	1	1	0	0	0
7795.249	7994.431	7826.853	1	1	0	0	0	2
7801.938	7997.563	7832.283	1	1	0	0	0	1
7828.475	8024.101	7858.820	2	1	0	0	0	0
7831.903	8027.529	7862.248	0	1	0	0	0	2
7839.373	8031.441	7868.490	0	1	0	0	0	1
7863.125	8055.194	7892.242	1	1	0	0	0	0
7896.708	8085.220	7924.630	0	1	0	0	0	0

Table 33: F-SARIMA Model Generation - Category 2: d = 1, D = 0

Appendix V - Code rolling forecast, model generation & random forest

```r
# alldata:      training and validation time series combined
# dataint:      last index training data
# lengthtest:   number of rolling forecasts
# o:            order of the (p, d, q) parameters
# s:            order of the (P, D, Q) parameters
roll_arima <- function(alldata, dataint, lengthtest, o, s) {
  results <- data.frame()

  for(i in 0:lengthtest) {
    print(i)
    ma <- Arima(ts(alldata[1:(dataint+i)], start = c(2014, 1), frequency =
     52), order = o, seasonal = s, include.drift = TRUE)

    pred <- forecast(ma, h = 1)
    pred$index <- (dataint + i)
    print(pred)
    results <- rbind(results, as.data.frame(pred))
  }

  return(results)
}
```

Listing 1: SARIMA rolling forecast

```r
# alldata:      training and validation time series combined
# dataint:      last index training data
# lengthtest:   number of rolling forecasts
# o:            order of the (p, d, q) parameters
# s:            order of the (P, D, Q) parameters
# k:            Fourier seasonality period
roll_arima_fourier <- function(alldata, dataint, lengthtest, o, s, k) {
  results <- data.frame()

  for(i in 0:lengthtest) {
    print(i)
    ma <- Arima(ts(alldata[1:(dataint+i)], start = c(2014, 1), frequency =
     52), xreg = fourier(ts(alldata[1:(dataint+i)], start = c(2014, 1),
    frequency = 52), K = k), order = o, seasonal = s, include.drift = TRUE
    )

    pred <- forecast(ma, xreg = fourier(ts(alldata[1:(dataint+i)], start =
     c(2014, 1), frequency = 52),
        K = k, h = 1), h = 1)

    pred$index <- (dataint + i)
    print(pred)
    results <- rbind(results, as.data.frame(pred))
  }

  return(results)
}
```

Listing 2: F-SARIMA rolling forecast

```r
# data:          training time series
# maxp:          maximum order of p for iterations
# d:             order of differencing
# maxq:          maximum order of q for iterations
# maxsp:         maximum order of P for iterations
# sd             order of seasonal differencing
# maxsq:         maximum order of Q for iterations
# drift:         application of drift: TRUE / FALSE
model_generator_sarima <- function(data, maxp, d, maxq, maxsp, sd, maxsq,
    drift) {
  results = data.frame()
    for (sp in 0:maxsp) {
      for (sq in 0:maxsq) {
        for (p in 0:maxp) {
          for (q in 0:maxq) {
            o <- c(p, d, q)
            s <- c(sp, sd, sq)
            print(o)
            print(s)
            am <- forecast::Arima(data, order = o, seasonal = s,
                include.drift = drift)
            results <- rbind(results, c(am$aic, am$bic, am$aicc, o, s))
          }
        }
      }
    }
  return(results)
}
```

Listing 3: SARIMA model generator

```r
# data:          training time series
# maxk:          maximum value for k for Fourier
# drift:         application of drift: TRUE / FALSE
model_generator_farima_k <- function(data, maxk, drift) {
  results = data.frame()
  for (k in 1:maxk) {
    o <- c(0, 0, 0)
    s <- c(0, 0, 0)
    print(o)
    print(s)
    am <- forecast::Arima(data, order = o, seasonal = s, xreg = forecast::
    fourier(data, K = k), include.drift = drift)
    results <- rbind(results, c(am$aic, am$bic, am$aicc, o, s, k))
  }
  return(results)
}
```

Listing 4: F-SARIMA model generator - k value

```
# data:          training time series
# maxp:          maximum order of p for iterations
# d:             order of differencing
# maxq:          maximum order of q for iterations
# maxsp:         maximum order of P for iterations
# sd             order of seasonal differencing
# maxsq:         maximum order of Q for iterations
# k:             value for k of Fourier
# drift:         application of drift: TRUE / FALSE
model_generator_farima_arima <- function(data, maxp, d, maxq, maxsp, sd, maxsq, k, drift) {
  results = data.frame()
  for (sp in 0:maxsp) {
    for (sq in 0:maxsq) {
      for (p in 0:maxp) {
        for (q in 0:maxq) {
          o <- c(p, d, q)
          s <- c(sp, sd, sq)
          print(o)
          print(s)
          am <- forecast::Arima(data, order = o, seasonal = s, xreg = forecast::fourier(data, K = k), include.drift = drift)
          results <- rbind(results, c(am$aic, am$bic, am$aicc, o, s, k))
        }
      }
    }
  }
  return(results)
}
```

Listing 5: F-SARIMA model generator - SARIMA parameter values

```r
# dataset:      training and validation time series
# dataint:      final index of training time series
# lengthtest:   number of time data points to predict
# mtrypar:      list of node split parameter values to iterate through
# nodesizepar:  list of node size  parameter values to iterate through
RFRandomForest_pm_estimation <- function(dataset, dataint, lengthtest, mtrypar, nodesizepar) {
  trend_forx <- data.frame()
  accuracy.matrix <- matrix(0, nrow = length(mtrypar), ncol = length(nodesizepar))
  row.names(accuracy.matrix) <- mtrypar
  colnames(accuracy.matrix) <- nodesizepar

  for(m in mtrypar) {
    for(n in nodesizepar) {
      print(m)
      print(n)
      results <- data.frame()
      for(i in 0:lengthtest) {
        datacase <- ts(dataset[1:(dataint+i)], start = c(2014, 1), frequency = 52)

        fuurx <- fourier(datacase, K = 26)
        decomp_tsx <- stl(datacase, s.window = "periodic", robust = TRUE)
        trend_partx <- ts(decomp_tsx$time.series[,2])
        trend_fitx <- auto.arima(trend_partx)
        trend_forx <- rbind(trend_forx, forecast(trend_fitx, 1)$mean)

        lag_orderx = 1
        periodx = 52
        Nx <- length(datacase)
        windowx <- (Nx / 52) - lag_orderx
        new_loadx <- rowSums(decomp_tsx$time.series[, c(1,3)])
        lag_seasx <- decomp_tsx$time.series[1:(periodx*windowx), 1]

        matrix_trainx <- data.table(Load = tail(new_loadx, windowx*periodx),
                                    fuurx[(periodx + 1):Nx,],
                                    Lag = lag_seasx)
        rf_modelx <- randomForest(Load ~., data = data.frame(matrix_trainx),
                                  ntree = 1000, nodesize = n, mtry = m,
                                  importance = TRUE)

        test_lagx <- decomp_tsx$time.series[((periodx*windowx)):Nx, 1]
        fuur_testx <- fourier(datacase, K = 26, h = 1)
        matrix_testx <- data.table(fuur_testx, Lag = test_lagx)

        trend_forx_selector = i + 1
        for_rfx <- predict(rf_modelx, data.frame(matrix_testx)) + trend_forx[trend_forx_selector, ]
        results <- rbind(results, for_rfx[1])
      }
      accuracy.matrix[as.character(m), as.character(n)] <- accuracy(ts(results, start = c(2018, 1), frequency = 52), vts)[5]
      print(accuracy.matrix[as.character(m), as.character(n)])
    }
  }
  return(accuracy.matrix)
}
```

Listing 6: Random forest parameter estimation

```r
# dataset:      training and validation time series
# dataint:      final index of training time series
# lengthtest:   number of time data points to predict
# m:            node split parameter value
# n:            node size  parameter value
RFRandomForest_forecast <- function(dataset, dataint, lengthtest, m, n) {

  trend_forx <- data.frame()
  results <- data.frame()


  for(i in 0:lengthtest) {
    print(i)
    datacase <- ts(dataset[1:(dataint+i)], start = c(2014, 1), frequency =
     52)

    fuurx <- fourier(datacase, K = 26)
    decomp_tsx <- stl(datacase, s.window = "periodic", robust = TRUE)

    trend_partx <- ts(decomp_tsx$time.series[,2])
    trend_fitx <- auto.arima(trend_partx)
    trend_forx <- rbind(trend_forx, forecast(trend_fitx, 1)$mean)

    lag_orderx = 1
    periodx = 52
    Nx <- length(datacase)
    windowx <- (Nx / 52) - lag_orderx

    new_loadx <- rowSums(decomp_tsx$time.series[, c(1,3)])
    lag_seasx <- decomp_tsx$time.series[1:(periodx*windowx), 1]

    matrix_trainx <- data.table(Load = tail(new_loadx, windowx*periodx),
                                fuurx[(periodx + 1):Nx,],
                                Lag = lag_seasx)

    rf_modelx <- randomForest(Load ~., data = data.frame(matrix_trainx),
                              ntree = 1000, nodesize = n, mtry = m,
                              importance = TRUE)

    test_lagx <- decomp_tsx$time.series[((periodx*windowx)):Nx, 1]
    fuur_testx <- fourier(datacase, K = 26, h = 1)
    matrix_testx <- data.table(fuur_testx,
                               Lag = test_lagx)

    trend_forx_selector = i + 1
    for_rfx <- predict(rf_modelx, data.frame(matrix_testx)) + trend_forx[
    trend_forx_selector, ]
    results <- rbind(results, for_rfx[1])
  }
  return(results)
}
```

Listing 7: Random forest rolling forecast

References

[1] J. Scott Armstrong and Robert Fildes. Correspondence on the selection of error measures for comparisons among forecasting methods. *Journal of Forecasting*, 14(1):67–71, 1995.

[2] J.Scott Armstrong and Fred Collopy. Error measures for generalizing about forecasting methods: Empirical comparisons. *International Journal of Forecasting*, 8(1):69 – 80, 1992.

[3] Atilla Aslanargun, Mammadagha Mammadov, Berna Yazici, and Senay Yolacan. Comparison of ARIMA, neural networks and hybrid models in time series: tourist arrival forecasting. *Journal of Statistical Computation and Simulation*, 77(1):29–53, 2007.

[4] N de P Barbosa, E da S Christo, and KA Costa. Demand forecasting for production planning in a food company. *ARPN Journal of Engineering and Applied Sciences*, 10(16):7137–7141, 2015.

[5] George Edward Pelham Box and Gwilym Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Inc., San Francisco, CA, USA, 1990.

[6] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[7] P.J. Brockwell and R.A. Davis. *Introduction to Time Series and Forecasting*. Springer Texts in Statistics. Springer International Publishing, 2016.

[8] Wojciech W. Charemza and Ewa M. Syczewska. Joint application of the Dickey-Fuller and KPSS tests. *Economics Letters*, 61(1):17 – 21, 1998.

[9] D. Commenges, A. Sayyareh, L. Letenneur, J. Guedj, and A. Bar-Hen. Estimating a difference of kullback–leibler risks using a normalized difference of aic. *Ann. Appl. Stat.*, 2(3):1123–1142, 09 2008.

[10] Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, Aug 2000.

[11] Philip Doganis, Alex Alexandridis, Panagiotis Patrinos, and Haralambos Sarimveis. Time series sales forecasting for short shelf-life food products based on artificial neural networks and evolutionary computing. *Journal of Food Engineering*, 75(2):196 – 204, 2006.

[12] Jamal Fattah, Latifa Ezzine, Zineb Aman, Haj El Moussami, and Abdeslam Lachhab. Forecasting of demand using arima model. *International Journal of Engineering Business Management*, 10:1847979018808673, 2018.

[13] R A Fildes and J K Ord. *Forecasting competitions - their role in improving forecasting practice and research*, pages 322–353. Blackwell, 2002.

[14] Andrea Fumi, Arianna Pepe, Laura Scarabotti, and Massimiliano M. Schiraldi. Fourier analysis for demand forecasting in a fashion company. *International Journal of Engineering Business Management*, 5:30, 2013.

[15] Charles J. Geyer. Practical markov chain monte carlo. *Statistical Science*, 7(4):473–483, 1992.

[16] Isabelle Guyon. A scaling law for the validation-set training-set size ratio. In *AT & T Bell Laboratories*, 1997.

[17] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. Horowitz, and William J. Dally. Eie: Efficient inference engine on compressed deep neural network. In *Proceedings of the 43rd International Symposium on Computer Architecture*, ISCA '16, page 243–254. IEEE Press, 2016.

[18] Jouni Helske. Computational efficiency comparison of MCMC algorithms for non-Gaussian state space models, 06 2017.

[19] Rob Hyndman, George Athanasopoulos, Christoph Bergmeir, Gabriel Caceres, Leanne Chhay, Mitchell O'Hara-Wild, Fotios Petropoulos, Slava Razbash, Earo Wang, and Farah Yasmeen. *forecast: Forecasting functions for time series and linear models*, 2019. R package version 8.9.

[20] Rob J. Hyndman and Anne B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679 – 688, 2006.

[21] Michael J. Kane, Natalie Price, Matthew Scotch, and Peter Rabinowitz. Comparison of arima and random forest time series models for prediction of avian influenza h5n1 outbreaks. *BMC Bioinformatics*, 15(1), 8 2014.

[22] Piotr Keblowski and Aleksander Welfe. The ADF–KPSS test of the joint confirmation hypothesis of unit autoregressive root. *Economics Letters*, 85(2):257 – 263, 2004.

[23] Ummul Khair, Hasanul Fahmi, Sarudin Al Hakim, and Robbi Rahim. Forecasting Error Calculation with Mean Absolute Deviation and Mean Absolute Percentage Error. *Journal of Physics: Conference Series*, 930:012002, dec 2017.

[24] T. H. Lai. Time series analysis univariate and multivariate methods : William W.S. Wei, (Addison-Wesley, Reading, MA, 1990). *International Journal of Forecasting*, 7(3):389–390, November 1991.

[25] Colin D Lewis. Chapter 2 - Short-term forecasting for stationary demand situations. In *Demand Forecasting and Inventory Control*, pages 21 – 43. Butterworth-Heinemann, Boston, 1997.

[26] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[27] J. Z. Ma and Li Deng. Efficient decoding strategies for conversational speech recognition using a constrained nonlinear state-space model. *IEEE Transactions on Speech and Audio Processing*, 11(6):590–602, Nov 2003.

[28] K. R. Müller, A. J. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. Predicting time series with support vector machines. In Wulfram Gerstner, Alain Germond, Martin Hasler, and Jean-Daniel Nicoud, editors, *Artificial Neural Networks — ICANN'97*, pages 999–1004, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[29] Serena Ng and Pierre Perron. Unit root tests in arma models with data-dependent methods for the selection of the truncation lag. *Journal of the American Statistical Association*, 90(429):268–281, 1995.

[30] Liam Paninski, Yashar Ahmadian, Daniel Gil Ferreira, Shinsuke Koyama, Kamiar Rahnama Rad, Michael Vidne, Joshua T. Vogelstein, and Wei Wu. A new look at state-space models for neural data. *Journal of Computational Neuroscience*, 29:107–126, 2009.

[31] Giovanni Petris, Sonia Petrone, and Patrizia Campagnoli. *Dynamic Linear Models with R*. useR! Springer-Verlag, New York, 2009.

[32] B. Pfaff. *Analysis of Integrated and Cointegrated Time Series with R*. Springer, New York, second edition, 2008. ISBN 0-387-27960-1.

[33] Debin Qiu. *Alternative Time Series Analysis*, 2015.

[34] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2018. ISBN 3-900051-07-0.

[35] L. Rabiner and B. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, Jan 1986.

[36] W. D. Ray. Time series models, second edition: Andrew C. Harvey, 1993, (Harvester-Wheatsheaf, New York) xviii + 308 pp., isbn 0-7450-1200-0. *International Journal of Forecasting*, 9(4):582–583, 1993.

[37] R. Roesser. A discrete state-space model for linear image processing. *IEEE Transactions on Automatic Control*, 20(1):1–10, February 1975.

[38] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. *Ann. Statist.*, 26(5):1651–1686, 10 1998.

[39] G. William Schwert. Tests for Unit Roots: A Monte Carlo Investigation. *Journal of Business & Economic Statistics*, 7(2):147–159, 1989.

[40] Mark R. Segal and Yuanyuan Xiao. Multivariate random forests. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, 1:80–87, 2011.

[41] A. Seghouane and S. Amari. The AIC Criterion and Symmetrizing the Kullback–Leibler divergence. *IEEE Transactions on Neural Networks*, 18(1):97–106, Jan 2007.

[42] Christopher A. Sims. Seasonality in regression. *Journal of the American Statistical Association*, 69(347):618–626, 1974.

[43] George Tauchen. Finite state markov-chain approximations to univariate and vector autoregressions. *Economics Letters*, 20(2):177 – 181, 1986.

[44] Efthymia V. Tsitsika, Christos D. Maravelias, and John Haralabous. Modeling and forecasting pelagic fish production using univariate and multivariate arima models. *Fisheries Science*, 73(5):979–988, Sep 2007.

[45] Cyril Voyant, Gilles Notton, Soteris Kalogirou, Marie-Laure Nivet, Christophe Paoli, Fabrice Motte, and Alexis Fouilloy. Machine learning methods for solar radiation forecasting: A review. *Renewable Energy*, 105:569 – 582, 2017.

[46] Stephen F. Witt and Christine A. Witt. Tourism forecasting: Error magnitude, direction of change error, and trend change error. *Journal of Travel Research*, 30(2):26–33, 1991.

[47] Yong Yu, Tsan-Ming Choi, and Chi-Leung Hui. An intelligent fast sales forecasting model for fashion products. *Expert Syst. Appl.*, 38(6):7373–7379, June 2011.

[48] Eric Zivot and Jiahui Wang. *Unit Root Tests*, pages 111–139. Springer New York, New York, NY, 2006.

Abbreviations

ACF Autocorrelation Function. 7–9, 15, 21, 36

ADF Augmented Dickey-Fuller. 3–5, 14, 20

AIC Akaike Information Criterion. 8, 15, 25–27, 35

AR Autoregressive. 6–10

BIC Bayesian Information Criterion. 8, 15, 25–27, 35

I Integrated. 6

KPSS Kwiatkowski–Phillips–Schmidt–Shin. 3–5, 14, 20

MA Moving Average. 6–8, 10, 31

MAE Mean Absolute Error. 13, 33

MAPE Mean Absolute Percentage Error. 13, 17, 29, 33

ME Mean Error. 13

MLE maximum likelihood estimation. 16, 31, 39

MPE Mean Percentage Error. 13, 33

MSE Mean Square Error. 13

PACF Partial-Autocorrelation Function. 7–9, 15, 21, 36

RMSE Root Mean Squared Error. 13, 33