October 31st, 2023
University of Twente
Enschede

Master Thesis - Pattern Analysis of Time Series Data

K.J. Rijnbergen

Supervisors University of Twente:
dr. M.C. van der Heijden
dr. D. Guericke
Supervisors Slimstock:
S. Pauly

Master Thesis Industrial Engineering and Management

UNIVERSITY OF TWENTE

This page is intentionally left blank.

Executive Summary

Introduction

Slimstock and many of its clients struggle with identification and forecasting of seasonal time series. These are time series that show structural fluctuations over time that occur due to influences of natural forces and human behaviour. Therefore, the objective of this research is as follows: *"Select the best method to classify and forecast future demand of different types of stock keeping units (SKUs)."*

Context analysis

In the current situation, our relevant SKUs are first classified into one of four classes based on the presence of a trend and seasonality. Different SKU types use different models. Stationary SKUs, which are SKUs with no trend nor seasonality use simple exponential smoothing. SKUs that also have a trend use double exponential smoothing. In case a SKU has seasonality present, it uses triple exponential smoothing, where a trend is modeled only in case it is significant.

Literature

Although our research only has a single objective, we need to solve two problems to reach our goal. Firstly, we have a classification problem. Classification of time series has been widely studied in the past, but despite all these efforts there is no general consensus on the best method. In our literature studies, we find two potentially suitable tests for seasonality, being a standard Fisher's F-test and the Friedman test. For trend classification, we find linear regression generally works well.

The second problem we have is selection of suitable forecasting models. During our literature research, we found three different types of methods, being statistical methods, regression methods, and machine learning methods. In our research, we will mostly focus on two types of methods. Firstly, the currently used statistical methods and some extensions and adaptations of these methods. Secondly, we look into Prophet, which is a regression model developed by Facebook.

Approach

The first step in our research is is evaluating the performance that we obtain by using the classification method and forecasting models that are currently used by Slimstock. The current classification method uses a Fisher's F-test and linear regression, with a trend first approach. We evaluate the performance based on a short, medium, and long cover period of 3, 7, and 21 days respectively. Classification is done on a monthly level.

After evaluating the performance of the current classification method, we evaluate the performance of alternative classification methods using the same forecasting models. Our evaluated alternative classification methods use a seasonality-first approach instead of a trend-first approach, a Friedman test instead of an F-test, or a combination of both for a total of three alternatives to the current method. Comparing the current method to explored alternatives, we find that the current method works best for all of our cover periods.

We then extend the classification method to include weekly and daily seasonality. This allows for SKUs to have multiple levels of seasonality, which allows us to extend the current models to use nested seasonality. Nested seasonality allows us to detect patterns over the weeks in a year and over the days in a week which we would otherwise not be able to detect.

Our other extensions are a Fourier transform used to ensure seasonality progresses smoothly over the year, and an adaptation to updating of model parameters that occur when we start to use multiple levels of seasonality.

Lastly, Prophet. Prophet is a fully automatic regression model that breaks down time series into three components, being trend, seasonality, and holidays and events. It uses a piecewise trend model for its trend, a Fourier series for seasonality, and regression for holiday and event effects.

Results

As mentioned, we found that linear regression for trend in combination with an F-test for seasonality was the best method for classifying our time series. We then made a further distinction between our different time series. During our experiments we find that including daily seasonality in our current models leads to a significant improvement in performance for short cover times. Using a cover time of 3 days, we find SKUs that were classified as having a daily seasonality and no trend see and improvement of 11.47%. SKUs that have a daily seasonality and a trend see an improvement of 12.82%. Note that improvement is based on root mean squared error, which is directly linked to our safety stocks. As these two classes cover 14.02% and 33.89% of our total SKUs respectively, we conclude that this extension is certainly worthwhile. Some other SKU types may also benefit from this change but we do not have sufficient evidence this benefit is statistically significant. Application of our Fourier transform leads to a significantly lowered RMSE for our daily-weekly-monthly SKUs without a trend for long cover periods (improvement: 15.41%). The same applies to daily-weekly-monthly SKUs with a trend, though here we find the improvement occurs for all cover periods (short: 8.18%, medium: 12.89%, long: 17.17%). These classes cover 1.43% and 3.05% of our SKUs so this change is much less impactful.

Moreover, we found that switching to Prophet appears to work very well for multiple classes of SKUs, with the model being particularly effective for shorter cover periods and more complex time series. Switching to Prophet where applicable leads to improvements in RMSE of at least 11.17% for short cover periods, 1.77% for medium cover periods, and 3.07% for long cover periods. Note that some SKUs may benefit more than others.

Recommendations

Based on our findings, we recommend Slimstock to implement daily seasonality in their classification and forecasting as it was able to improve short term forecasts by a significant margin based on RMSE. We also recommend implementation of the Fourier extension as it yields significant improvements in performance with very little effort. Implementing Prophet for suitable classes of SKUs leads to even better performance, though this change is much more rigorous and thus will take a lot longer to develop.

Acknowledgements

Successfully conducting a research and writing this thesis at Slimstock marks the end of my master's program at the University of Twente. Handing in this thesis is a milestone I have worked up to for the past seven and a half years. Although this is my largest solo project to date, I could not have done it alone. Before proceeding to the main body of this report, I would like to thank some people without whom I could not have made it.

First of all, I would like to thank my university supervisors, dr. M.C. van der Heijden and dr. D. Guericke, for their enthusiastic guidance and valuable feedback.

Secondly, I would like to thank my colleagues at Slimstock, my research team, and particularly, my company supervisor S. Pauly. Thank you for your help, feedback, support, and conviviality.

Thirdly, I would like to thank my family. Thank you for being there for me at all times.

Lastly, I would like to thank my friends and dining club. You have been there both when I needed it and when I wanted it. Hanging out with you is always an enjoyable experience. I know I can count on you, and thank you for never letting me down.

During my life as a student, which took a little over seven years, I learned a lot. Not only did I improve on my analytical, teamworking, and critical thinking skills, I grew as a person. Yes, I faced difficulties at times, and no, it has not always been fun. But my comfort zone expanded. I could have never imagined getting to where I am now, and I am very proud of it. I am so proud of all the things I have seen, experienced, and felt, as well as all the friends I made along the way.

I am very proud of this research. Not only does finishing this research mark the end of the current era, it also marks the beginning of a new one. And I am looking forward to it. The past years have been absolutely amazing, and as you will see in this research, history has a tendency to repeat.

Contents

List of Figures

List of Tables

List of Abbreviations

MAD - Mean Absolute Deviation. Identical to MAE.
MAE - Mean Absolute Error. Identical to MAD.
MAPE - Mean Absolute Percentage Error.
MASE - Mean Absolute Squared Error.
MSE - Mean Squared Error.
RMSE - Root Mean Squared Error.
RMSSE - Root Mean Squared Scaled Error.
N/A - Not Applicable
SKU - Stock Keeping Unit
sMAPE/SMAPE - symmetric Mean Absolute Percentage Error.
WRMSSE - Weighted Root Mean Squared Scaled Error.

Glossary

AIC - Akaike Information Criterion. A measure that evaluates goodness-of-fit.
Fisher's F-Test / F-test / Fisher test - A statistical test on variance.
Forecast - An estimation on how much demand we will occur during our cover time
Fourier Series - Expansion of a function or time series into a sum of sines and cosines.
Fourier Transform - A mathematical algorithm that breaks down a signal into its frequency components. Returns a Fourier series.
Friedman test / Friedman-Chi-Squared test - Statistical test on variance.
Level - The expected demand at the current time T, before seasonal factors.
Low-pass filter - A filter that can be applied to a Fourier series. Application of a low-pass filter sets high frequencies to 0 whilst keeping low frequencies unchainged.
p-value - Probability of making a Type I error.
Seasonality - The phenomenon of having more/less demand for a SKU due to being at a certain time of the week/month/year.
Stockout - Having no inventory on hand.
Trend - The expected increase or decrease in demand over time.
Type I Error - Rejecting the Null hypothesis in case it is, in fact, true.
t-test / Student's t-test - Statistical test on the difference between two means.

Nomenclature

a - level
b - trend
D_t - Demand at time t
F_t - Demand forecast for time t
h - Forecasting horizon
n - Number of observations
S_t - Seasonality for time t
t - time t
α - Polysemous. Refers to either Level smoothing factor, or to the significance level of a statistical test.
β - Trend smoothing factor
$\sum_{i=1}^{n} X_i$ - Shorthand notation for $X_1 + X_2 + ... + X_{n-1} + X_n$.
$x.yEz$ - Scientific notation for $x.y * 10^z$. Example: $1.23E4 = 1.23 \cdot 10^4 = 1230$.

1 Introduction

The research for this thesis is on the topic of forecasting demand of products in the maturity phase of their product life cycle. This research was done as the final part of my master's program in Industrial Engineering and Management at the University of Twente in Enschede, Netherlands. The research itself is conducted at Slimstock B.V. in Deventer, Netherlands. Data for this research have been obtained from one of Slimstock's customers. The origin of the data is kept confidential, and any data present in the report is obfuscated such that no information on the customer can be extracted.

This chapter starts off by providing a short description about Slimstock in Section 1.1. Then, in Section 1.2, we motivate our research by identifying our core problem and explaining the problem context. In the following section, Section 1.3, we describe the problem statement, where we describe the ideal scenario, the actual scenario, and a general direction for our research such that we can get closer to the ideal scenario. The research goal and deliverables are then covered in Section 1.4. We finish off this chapter by providing the research questions in Section 1.5.

1.1 Concise Company Description

Inventory management is a core part of a lot of businesses, but it is also very complex. Because of this complexity, businesses oftentimes outsource the task of managing their inventory to other companies, such as (Slimstock, 2022).

Slimstock, founded in 1993, is a company that provides services in the form of inventory management. With over 1300 clients in 23 countries, they are the market leader in the field of demand forecasting, planning, and inventory/supply chain management. Using their own software package, Slim4, Slimstock maximizes performance. Clients and experts work together on minimising inventory levels and maximising service levels, such that costs are kept low and customer satisfaction is as high as possible. In short, Slimstock helps their clients get the right inventory, at the right place, at the right time.

1.2 Research Motivation

In this section, we identify our core problem. We also provide a brief introduction to the core problem by making a general analysis as of why this problem exists and why it is so difficult to solve.

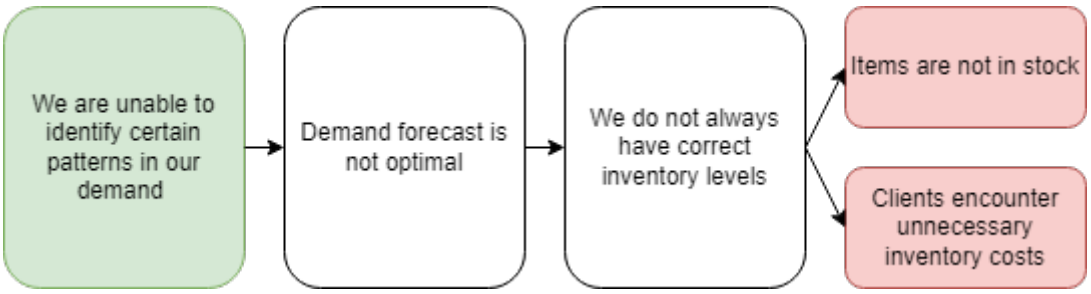

Figure 1.1: Problem Cluster.

Keeping inventory is part of nature. Humans have goods in their house, squirrels collect acorns for winter, and companies keep inventory such that they can sell their products to their customers. Knowing how much inventory to keep can be a difficult problem. Having too little inventory leads

to stockouts whilst having too much inventory leads to high costs. In this research, we are going to investigate methods to optimise inventory levels.

We first make a problem cluster to identify our core problem, which describes what exactly needs to be improved. This problem cluster is depicted in Figure 1.1 above.

As can be seen in the problem cluster, we identify a sub-optimal forecast as a big contributor to our problem. Therefore, we are going to look at improving the forecast. In order to improve the forecast, we look at patterns that exist on a smaller scale than the patterns we are detecting now. In addition, we explore alternative forecasting algorithms.

Basics of Forecasting

Without going into the details of the functioning of forecasting models, most forecasts are made like this:

$$F_t = (a + b_t) \cdot S_t$$

Where a is the level of today (which is the base value we would expect without seasonal effects or growth), b is the long-term average growth of the value, also known as the trend, and S_t is a multiplier based on our the present seasonality at time t.

When looking at this basic model, we notice that a good forecast is dependent on estimating three factors well. These are the level, the trend, and seasonal factors.

Many of Slimstock's clients struggle to identify patterns that are present in demand over time. In general, products that have a more stable demand (e.g. toothpaste) are easier to forecast than products that are rising or declining in popularity, meaning they have a trend (an example we came up with is induction stoves, though finding examples for this class is difficult as products only keep rising/declining in popularity for a shorter period..

The most difficult products to forecast are the products that are constantly undergoing changes in demand as well as being affected by seasonality. A good example of a product in this class would be petrol. Petrol has a very dynamic trend as demand greatly varies over time, rising in popularity as more people get access to cars and start driving more, and declining in demand as less people drive less and/or switch to electric engines. In addition to having a very dynamic trend, petrol is also exposed to seasonal effects, having increased demand in summer and decreased demand in winter. In addition to being exposed to trending and seasonal effects, demand of petrol is also highly affected by cyclical effects such as economical cycles.

Core Problem

We identify the inability to identify certain patterns in our demand to be our core problem. This inability to identify patterns leads to suboptimal forecasts, which in turn leads to suboptimal inventory, which has its consequences as explained in the next paragraph.

Consequences

Ideally, we would always have exactly as much of a product as we need. In case we are going to sell 7 bottles of water before we get our next replenishment, we would want to have exactly 7 bottles on hand. That way, we do not have to use more storage space than necessary, and we do not lose out on sales because we did not have an item our customer wanted. In reality though, knowing how much of a given product we will need is extremely difficult, which leads to either

reduced revenue because of lost sales in case we have too little inventory, or increased inventory costs in case we have too much inventory.

1.3 Problem Statement and Solution Directions

In this section, we explain how this problem has emerged and what impact the problem has. After introducing the problem, we describe the basic idea of how we intend to solve the problem in our research.

1.3.1 The Early Stages of Slim4 and Origin of the Problem

When Slimstock was founded in 1993, its main customers were wholesalers and industry. These customers are customers that generally have lots of space to store their products in their very large warehouses and production facilities. This combined with the fact that logistics in the previous century were nowhere as developed as they are today meant that Slimstock's main customers had a fairly simple strategy. Regularly but infrequently order large amounts and store them in their gigantic warehouses until they are needed. Oftentimes, these companies had massive replenishment orders coming in about once a month. For Slimstock, this meant inventory was managed on a monthly basis and in turn, forecasts were made on a monthly basis.

With the rise of Slimstock, Slimstock has attracted many customers in various different industries and sectors, such as supermarkets and pharmacies. These are generally customers that have little storage space, a greater number of stock keeping units (SKUs), and lower demands per SKU. In addition, some customers have highly perishable goods. This all leads to the very different inventory management strategy and ordering policy of ordering smaller lot sizes more frequently, and this is exactly where our core problem has started to emerge.

We hypothesize we are currently unable to extract certain demand patterns because Slim4 restricts seasonality extraction to monthly. In our research, we investigate whether the usage of different binning periods can possibly assist in revealing seasonal patterns in our data. In addition, we explore some more modern alternatives to the forecasting methods and algorithms that have been the backbone of Slim4 forecasting since the early days of Slim4 up until now.

1.3.2 Ideal Scenario

As mentioned before, we want to better identify repeating patterns for SKUs where we suspect such a pattern may be present. This means we will have to select a suitable method for classifying our different classes of SKUs, as well as selecting the best methods to develop forecasts for these SKUs.

1.3.3 Solution Directions

Our research has two main objectives. The first objective of our research is selecting a suitable method to classify SKUs by demand pattern (which includes classification of trend and classification of seasonality). The second objective of our research is selecting the best algorithms/forecasting methods to forecast demand for each of our SKUs.

Selection of Classification method

To select our classification method, we investigate the current classification method (which is a Fisher's F-test combined with linear regression), and compare it to alternatives found in literature.

Classification methods explored in our research will always consist of a statistically based test in combination with regression.

Selection of Forecasting Methods

To select the best forecasting method for each of our SKU types, our research will include exploration of a combination of numerical algorithms and machine learning. Although prior research has shown that numerical algorithms vastly outperform more advanced methods such as machine learning, especially when seasonality and/or other patterns are present (Makridakis et al., 2018), we will not completely disregard machine learning methods. The early stage of the research (specifically, our literature research) covers a very wide range of forecasting methods, ranging from very simple regression and smoothing algorithms to very complex machine learning algorithms.

1.4 Research Goal

The goal of our research is fairly straightforward. Firstly, we identify the best way to classify our SKUs based on the presence of trend and seasonality. Secondly, we select the best method to forecast these SKUs, depending on their characteristics. The objective of the research is formulated as follows:

- *"Select the best method to classify and forecast future demand of our different types of stock keeping units."*

Deliverables of the research will be a thesis report, as well as the source code of all algorithms and machine learning models used in our research. The application will mainly be written using Python. In addition to Python, which is used for processing the data and running the actual models, we will use Excel for model validation and model development. Any useful Excel files will also be delivered along with the report.

1.5 Research Questions & Approach

Now that we have made a problem statement and research goal, we are going to look into solving the problem. To solve the problem, we develop a set of research questions. When answered, these research questions will lead us to a solution to the problem. We start off by posing our main research question.

- *Main Question: "How can we better detect patterns in our time series and improve our forecasts?"*

To answer our main question, we compose a set of smaller research questions. We divide these research questions into four types. These are: preparatory questions, questions on the theoretical background, experimental questions, and roundup questions.

We first cover the preparatory questions. These questions cover the current modus operandi, and the general workings of Slim4.

Our first research question is as follows:

- *Research question 1: "How does Slim4 currently make its forecasts?"*

In order to know how to improve on the current situation, we first have to know the in and outs of the current situation. To answer this question, we break down this research question into a set of sub-questions. These are deployed below. We will answer these questions by talking to colleagues at Slimstock, and through the internal training on Slim4.

- *Sub-question A: "Which types of SKU use which forecasting model?"*
- *Sub-question B: "What are the relevant parameters in these models?"*
- *Sub-question C: "How does Slim4 measure the performance of used forecasting models?"*
- *Sub-question D: "What is the performance of these models?"*
- *Sub-question E: "What are the advantages of these models?"*
- *Sub-question F: "What are the disadvantages of these models?"*
- *Sub-question G: "Which assumptions are made when applying these models?"*

Secondly, the theoretical background. After diving deeper into the current way of working and the abilities and limitations of Slim4, we are going to conduct a literature study. In this literature study, we explore solutions and ideas to problems similar to ours. This leads us to our third research question:

- *Research question 2: "What can we learn from literature on pattern recognition and forecasting?"*

We break this down into the following sub-questions:

- *Sub-question A: "How can we classify the presence of trends and seasonal effects in demand of products?"*
- *Sub-question B: "Which forecasting models are available for forecasting demand for different SKU types?"*
- *Sub-question C: "How can we measure performance of used forecasting models?"*

Now that we have a solid idea of the theory behind our problem, we will work on developing a model to solve our problem. After developing the model, we will also be researching model optimisation. In order to start developing our model, we first have to answer our next research question. This question is as follows:

- *Research question 3: "How can we evaluate our newly developed model?"*

We break this down into the following sub-questions:

- *Sub-question A: "How are the data collected and prepared?"*
- *Sub-question B: "How are we going to measure and express our model performance and select the best method?"*
- *Sub-question C: "How do we validate and verify our models?"*

After these questions have been answered, we can start on the design and implementation of our solution. To get a good starting point, we answer the following question:

- *Research question 4: "What will our models look like?"*

We break this down into the following sub-questions:

- *Sub-question A: "Which classification methods on trend and seasonality (from literature) are we going to compare in our research?"*
- *Sub-question B: "Which methods (from literature) are we going to be using as a starting point for our forecasting models?"*
- *Sub-question C: "What are the main parameters in these models?"*

- *Sub-question D: "How are we going to optimise our model parameters?"*
- *Sub-question E: "How are we going to modify these models such that we can improve their performance?"*

At this point in the research, we are going to do our implementation. Once implementation is done, we can proceed to the final step.

The final step is evaluation of performance of our models, our implications and findings. Although we will have a finalized model at this point, we want to explain why the model performs the way it does. In addition, we look into the pros and cons, recommendations, and directions for further research. This leads us to our next and final research question:

- *Research question 5: "What are our findings?"*

Breaking this down:

- *Sub-question A: "What is the best classification method?"*
- *Sub-question B: "What is the best forecasting model for each of our SKU types?"*
- *Sub-question C: "What is the performance of the new method?"*
- *Sub-question D: "What are the advantages and disadvantages of newly found models?"*
- *Sub-question E: "Based on performance, advantages, and disadvantages, what can we recommend to our client?"*
- *Sub-question F: "Similarly, what are our recommendations to Slimstock?"*

1.6 Research Scope

The research will be done using a combination of numerical algorithms and machine learning methods. We are going to look at established models such as simple exponential smoothing, double exponential smoothing, and the Holt-Winters triple exponential smoothing model. We will explore the use of numerical methods such as the Fourier transform, as well as more advanced models such as Prophet.

Our research will not include pure machine learning models such as deep neural networks or reinforcement learning. Prior research shows models that are combinations of statistical/numerical models and regression models outperform pure machine learning methods (Makridakis et al., 2020), and thus we will not put an emphasis on pure machine learning models.

We will be limiting our scope to the items classified as Frequent, Normal, and Lumpy as specified by Slim4. More on this can be found in Section 2.1.

In addition to limiting ourselves to Frequent, Normal, and Lumpy items as specified by Slim4, we will only research products that are in the maturity phase of their product life cycles. These are products that have been on the market for a while, and thus their demand patterns have had the time to stabilise. These products are also still actively sold, and thus we expect their demands in the near future to follow the same distribution as demands in the recent past. More on product life cycles can be found in (Rink and Swan, 1979).

Data used in our research originates from a single customer of Slimstock. This means the data originates from a single field, meaning results found in our research may not necessarily be applicable to any field. The developed method can be applied to any data set that has

sufficient history such that we can evaluate the performance of the models in different fields. As some methods used in our research require at least four years worth of data, the methods and models are not applicable to emerging markets.

2 Context Analysis

We start off by analysing the current situation. We answer our first research question:

- *Research question 1: "How does Slim4 currently make its forecasts?"*

2.1 Types of SKU and Their Forecasting Models

The first step in finding out how forecasts are currently made is finding out which models are used for which types of SKU. This means we have to find the answer to the following research question:

- *Sub-question A: "Which types of SKU use which forecasting model?"*

To answer this question, we first have to find out how Slim4 currently distinguishes its products. We obtain this information from the internal training on Slim4.

Slim4 currently makes a distinction between five main classes of Slim4-controlled items. These classes are: Frequent, Normal, Lumpy, Irregular, and Slow. These five classes and their characteristics are shown in Table 2.1 below. (Slimstock, 2022)

Table 2.1: Demand Patterns

Demand class	Forecast each month	Months w/o demand taken into account	Forecasting based on forecasted orders	Forecasting method
Frequent	Yes	N/A	No	Exponential Smoothing
Normal	Yes	N/A	No	Exponential Smoothing
Lumpy	Yes	Yes	No	Exponential Smoothing
Irregular	No	No	Yes	Croston's Method
Slow	No	No	Yes	Croston's Method

We will now explain the basic properties of these classes. As our research only covers Frequent, Normal, and Lumpy items, we will not go into too much detail on Irregular and Slow items. That being said, we think it is important there are classes other than Frequent, Normal, and Slow.

2.1.1 Frequent, Normal, and Lumpy Items

Frequent, Normal, and Lumpy items are items that have a demand greater than 0 in most periods. The class "Frequent" is the class that is assigned to the most stable of items. These are items that are ordered every month or nearly every month. "Normal" items are similar to "Frequent" items, but they are ordered a bit less often and thus we have fewer data points. "Lumpy" items have more months with demand than months without demand, but do not have demand in every month. Slim4 uses a couple more characteristics for this classification. These characteristics are kept confidential, but it is sufficient to know that Frequent, Normal, and Lumpy items are items that are relatively stable over time. As can be seen in Table 2.1, "Frequent", "Normal", and "Lumpy" items are forecasted using exponential smoothing. (Slimstock, 2022)

Exponential Smoothing

Items that are forecasted by exponential smoothing use either single, double, or triple exponential smoothing. In single exponential smoothing, we make a forecast based on past observations,

putting a greater emphasis on more recent observations. Single exponential smoothing only smooths the level of the observations by a single parameter α, and does not take trend or seasonality into account. Double exponential smoothing works in a similar manner to single exponential smoothing, but it also takes a trend into account. Parameters for double exponential smoothing are α for the level and β for trend (Holt, 2004), (Silver et al., 2016).

Triple exponential smoothing is the most advanced of the three, taking into account the level, trend, and seasonalities present in our time series. Level and trend are calculated as before. Seasonality is a bit more complex, and is only calculated once we have sufficient evidence to assume the time series have a seasonality present. To test for this, we use a statistical Fisher's F-test. This test is explained in Appendix A. Triple exponential smoothing has two variants, being one that includes and one that excludes the modeling of a trend. Note that, from now on, "exponential smoothing" can refer to any level of exponential smoothing. In case we need to be more specific, the level is specified.

We dive deeper into exponential smoothing in Section 2.2, 3.2.1, 6.3, and Appendix C, F and G.

2.1.2 Irregular Items and Slow Items

"Irregular" items are items that typically have high spikes and many periods of no demand. For these SKUs, we do not make a forecast for each month. Instead, we use Croston's forecasting method, meaning we make a forecast based on expected future orders. To do this, we use historical transaction data instead of aggregated demand data. "Slow" items are like irregular items, but demands are low when compared to irregular items. (Slimstock, 2022)

2.2 Model Parameters

Now that we know which models are used for our SKUs, we further look into the exact workings of these models. More specifically, we are going to look at the relevant parameters in these models, answering the following sub-question of our research:

- *Sub-question B: "What are the relevant parameters in these models?"*

Slim4 uses a single value for its smoothing parameters for all of its customers. The exact value is kept confidential, but the value is in line with values proposed in (Silver et al., 2016). Models use either one or two smoothing parameters, being just α for level or α and β for level and trend. Seasonality is never smoothed, just recalculated.

2.2.1 Single Exponential Smoothing

Single exponential smoothing uses a single parameter, which is α. α is used to smoothen the level.

2.2.2 Double Exponential Smoothing

Double exponential smoothing uses two parameters, being α for level and β for trend. Slim4 uses the same value for both of these parameters (i.e. $\alpha = \beta$).

2.2.3 Triple Exponential Smoothing (Holt-Winters)

Triple exponential smoothing, also known as Holt-Winters is a well-established method for "Forecasting seasonals and trends by exponentially weighted moving averages" (Holt, 2004).

We explain traditional triple exponential smoothing in detail in Appendix C, but at it's core, triple exponential smoothing is exponential smoothing that is applied three times.

In triple exponential smoothing, we apply exponential smoothing once on the level of the time series, once on the trend, and once on the seasonal factors. Each of these smoothing procedures has its own parameter, being α for the level smoothing factor, β for the trend smoothing factor, and γ for the seasonal smoothing factor. These parameters represent the weight of the most recent data point, meaning older data have a weight of (1 - α) for level, (1 - β) for trend, and (1 - γ) for seasonal factors (Silver et al., 2016), (Holt, 2004).

At Slimstock, only level and trend are smoothened as before. Seasonality is never smoothed, just recalculated. Slim4 uses values that are in line with values proposed in (Silver et al., 2016). (Slimstock, 2022)

As mentioned before, forecasts are made according to the following formula.

$$F_t = (a + b_t) \cdot S_t$$

Where a is the level, b is the trend, and S_t is seasonality at time t. Note that, when forecasting for the longer term, trend is usually dampened (meaning we decrease (if trend is positive)/increase (if trend is negative) our value for trend when we forecast farther into the future). This is done as trends often flatten out in the long term, converging to a value of zero (Gardner, 1985). Note that this dampening is never done by slim4 automatically. Instead, this is manually done by the consultant.

2.3 Performance Measures

Now that we know which parameters are relevant in our models, we can move on to the next research question, which covers measuring of performance. The question is as follows:

- *Sub-question C: "How does Slim4 measure the performance of used forecasting models?"*

No matter which model we use, a forecast will never be perfect 100% of the time. In statistics, we make a distinction between two sources of imperfection, being error and bias. We briefly clarify the difference between those two.

Error

We define error as uncontrolled variation that causes our prediction to be off. Error is a metric that indicates the degree to which our forecast is correct (where lower errors mean better forecasts). It is the part of imperfection that occurs due to chance alone.

Bias

When forecasting multiple data points, we have an error for each individual data point. In case this series of errors structurally deviates from zero, we say our forecast is biased. In case our forecast is structurally higher than reality, the mean of our error is greater than zero and thus we have a positive bias. In case our forecasts are structurally lower than reality, our mean error is smaller than zero, we have a negative bias. In both cases, our forecast contains systematic errors, which we would like to be as close to zero as possible.

2.3.1 Implementation in Slim4 and Automatic Re-Initialization of Forecast Parameters

As of currently, Slim4 does not measure the Error of the forecasts it makes. For our research, this means we will have to implement the current model and measure forecasting performance ourselves. This is covered in Section 6.1.

Contrary to the error of the forecasts, which is not measured, slim4 does measure the bias of the forecast. In slim4, this is called the tracking signal. When actual demand significantly deviates from the forecast for multiple months, Slim4 generates a structural exception. In case Slim4 generates such an exception, the SKU should be reviewed urgently.

Note that there are two levels of structural exceptions in Slim4, being the tracking signal warning and the tracking signal critical. In case the tracking signal is of level warning, the review is left to be evaluated by the consultant. In case the tracking signal reaches the level of critical, Slim4 automatically re-initializes the item, updating its parameters and forecasts. Values as to when the tracking signal reaches levels warning and critical are kept confidential. (Slimstock, 2022)

2.4 Model Performance

As Slim4 does not currently measure the error of our forecasts, we do not have immediate access to the values of our performance measures. This makes answering our following research question, which is as follows:

- *Sub-question D: "What is the performance of these models?"*

A bit more difficult, as we have to measure baseline performance ourselves. This is done in Section 6.1. In our research, we decide to use three different metrics for measuring the error of a model. These metrics are: Normalised RMSE, average SMAPE, and normalised MAD. More on these metrics is found in Section 3.3.

2.5 Model Advantages

Now that we have covered used models and their performances on our data set, we can answer our next research question. This question is as follows:

- *Sub-question E: "What are the advantages of these models?"*

We find single, double, and triple exponential smoothing have three main advantages. Firstly, exponential smoothing has declining weights on older input data. This is desirable, as newer data are usually more representative of reality. Secondly, exponential smoothing is very easy to compute and thus does not require much computational power. Lastly, we need minimum input data. Triple exponential smoothing requires at least two years worth of data for determining seasonality and seasonal factors. Single and double exponential smoothing can be done with only a handful of data points.

2.6 Model Disadvantages

In addition to advantages, each model will also have its disadvantages. These are explored in our sixth research question.

- *Sub-question F: "What are the disadvantages of these models?"*

Despite being a well-established forecasting model, exponential smoothing models also have their drawbacks. We identify fitting to noise to be the main downside of these models.

When determining the smoothing factors for our exponential smoothing models, we have to make a tradeoff between responsiveness and stability of our models. We can use high smoothing values to put more emphasis on recent data, but this comes at a cost. When we put a lot of emphasis on our recent data, the models become less stable as it is more sensitive to recent changes in the underlying data, even if these changes are not representative to the current environment. This results us fitting the model to random fluctuations and noise rather than fitting a model that closely represents reality. The opposite also holds. When we use very low values for our smoothing parameters, we will have a very stable model that is not susceptible to noise. However, this model will likely perform poorly as it is unable to capture the changes that happen in our environment and thus should be taken into account. (McClain and Thomas, 1973)

In addition, due to the nature of the workings of these numerical models, exponential smoothing does not take any information other than historical demand as its input. This means that any known information about the future is not and cannot be taken into account. As explained in Appendix C, it is possible to improve the quality of exponential smoothing forecasts by including known information about the future. This is confirmed by (Chatfield and Yar, 1988), who were able to do this in their research.

2.7 Model Assumptions

Lastly, model assumption. We answer the last sub-question of this chapter.

- *Sub-question G: "Which assumptions are made when applying these models?"*

Single, double, and triple exponential smoothing make some basic assumptions, with the main assumption being that history will repeat itself in the future. We do not necessarily see this as a bad thing as repeatability of the past is why forecasting even works in the first place. The second assumption is that, by exponentially smoothing rather than averaging our model parameters, we assume more resent data is more representative for the future than older data. This also seems reasonable under most circumstances.

2.8 Conclusion

To summarise, Slim4 makes its forecasts using either exponential smoothing or Croston's method. For the SKUs that are relevant in our research, Slim4 uses exponential smoothing only.

The level of exponential smoothing is determined based on a Fisher's F-test, and linear regression. The Fisher's F-test is used to determine whether or not seasonality is present. Linear regression is used to examine the presence of a trend.

SKUs that have neither seasonality nor a trend use classical single exponential smoothing. For SKUs that have a trend but no seasonality, double exponential smoothing is used. SKUs that have a seasonality present use a modified version of triple exponential smoothing. This modification comes in the form of in/exclusion of a trend, and a modification to the updating of seasonality. In Slim4, seasonality is not smoothened. Instead, Slim4 recalculates seasonality whenever needed.

3 Literature Review

This chapter consists of a literature study on time series forecasting for inventory management. We aim to provide a comprehensive overview of existing literature on the subject, covering both commonly used methods as well as modern alternatives to these methods. In doing so, we answer our second research question.

- *Research question 2: "What can we learn from literature on pattern recognition and forecasting?"*

Note that our literature studies covers many methods that we did not end up using in our experiments or final model. Some of these methods were immediately discarded based on literature itself, and some were later discarded for other reasons. These reasons include, but are not limited to: insufficient access to data needed by the model, high computational cost for customers if implemented (and thus infeasible in practice), and extreme computational cost for researching purposes to the point where the research would be infeasible. Models that were covered in our literature research, but have not ended up in our experiments and final model, can be found in Appendices J and K.

First, we explore the difference between stationary, trending, and seasonal time series. Next, we research regularity and periodicity of our time series, exploring different methods to identify the presence of seasonality and different ways to extract the underlying seasonal factors. Finally, we discuss several forecasting methods, algorithms, and regressors that can be used to make good forecasts. We will explore both statistical as well as machine-learning based models. We finish this chapter by discussing relevant performance metrics.

We begin by answering our first sub-question. This question is as follows:

- *Sub-question A: "How can we classify the presence of trends and seasonal effects in demand of products?"*

3.1 Classification of Different Types of SKUs

We first look at (Silver et al., 2016). In (Silver et al., 2016), time series are broken down into five components, being the level (a), trend (b), seasonal variations (S), cyclical movements (C), and noise (ϵ).

Starting off with the simplest form of time series, which are time series that only have a level. These are time series with a constant demand over time. A level is part of all of our time series.

The next component is the trend. The trend is the component that is responsible for the growth of decline of demand over time.

Thirdly, we have seasonal variations. These are variations over time that occur because of natural forces such as the weather, or from human behaviour, such as increased sales during the holiday season. Seasonal factors are regular fluctuations in sales, repeating every period with similar intensities and timings.

Cyclical variations are variations over time that occur as a result of changes in the economical environment and fundamental business changes over time. They behave similarly to seasonal effects, but they are much more irregular, and do not necessarily repeat every year. (ARSHAM, 1985)

Lastly, the elephant in the room: noise. Noise is the inexplicable part of the time series, and thus it has no analytical formula. Noise is oftentimes assumed to be normally distributed around 0.

In our research, we focus on just four of these five effects, being the level, trend, seasonal factors, and noise. Focusing on these four effects yields four different types of time series, being as follows:

- Time series that consist of just level + noise (Stationary time series)
- Time series that consist of level + trend + noise (Trending time series)
- Time series that consist of level + seasonals + noise (Seasonal nontrending time series)
- Time series that consist of level + trend + seasonals + noise (Seasonal trending time series)

These four types of time series are illustrated in Figure 3.1 on the next page.

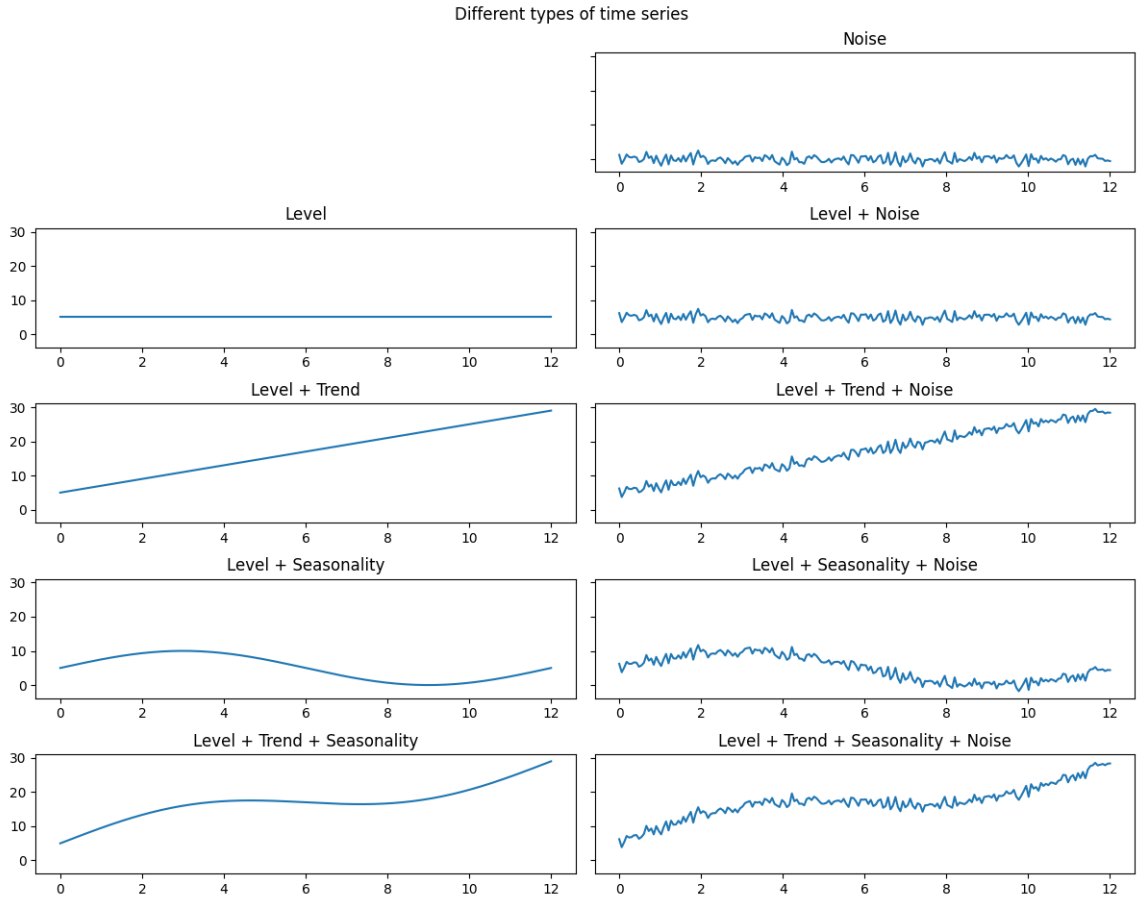

Figure 3.1: Different Types of Time Series

Due to the fundamental differences in these time series, we may need to use a different model for each type of SKU. Therefore, we need to find a method to classify the presence of trend and seasonality in our time series.

3.1.1 Classification of Trends

One of the distinctions we need to make is based on the presence of a trend (Hyndman and Athanasopoulos, 2018). There are many different available methods available for checking the presence of a trend, but one of the most common methods is ordinary least squares (OLS), which is a method of performing linear regression (Hess et al., 2001) (Silver et al., 2016). Using OLS, we can extract both the level and trend from our time series.

The equations for OLS are as follows:

$$\text{Trend} = \hat{b} = \frac{\sum_{t=1}^{n} t x_t - (\frac{n+1}{2}) \sum_{t=1}^{n} x_t}{n(n^2-1)/12} \tag{3.1}$$

$$\text{Level} = \hat{a} = \sum_{t=1}^{n} x_t/n - \hat{b}(n+1)/2 \tag{3.2}$$

For time series that only have a level (no noise), the level will be perfectly extracted, and the trend will be exactly zero. For time series that have a level and trend (again, no noise), OLS is also optimally able to extract level and trend. This is because, if a line is a straight line, all points in that line will fall on the regression line. Therefore, the vertical distance between each point on the line and and the regression line itself is equal to zero. In this case, we can conclude a time series is stationary if its trend is equal to zero. In case trend is not equal to zero, the time series is trending.

Once we introduce noise, however, we will no longer have the scenario where all points in the line fall on the regression line. Therefore, our error will no longer be equal to zero, and we will likely not be able to perfectly extract our level and trend. This also means that our previous method of checking whether a trend is exactly equal to zero is no longer sufficient for classification of a trend. Therefore, we need to test our trend for statistical significance of correlation. (Santer et al., 2000).

3.1.2 Classification of Seasonality

In addition to classification of trend, we also need to check whether or not a time series is seasonal. Currently, Slimstock uses a Fisher's F-test to perform this analysis. In this section, we explore a non-parametric alternative to this test.

Friedman Test

The Friedman test is a non-parametric method that tests whether or not samples are drawn from populations with equal medians. That is, we test whether or not each period in our timespan has approximately the same median, or if there are some periods that have a significantly lower/higher median. In case a period has a significantly lower or higher median than the other periods, it means the time series have significant present seasonal effects. Being a nonparametric test, the Friedman-Chi-Squared test does not require any distributional assumptions. (Friedman, 1937), (Friedman, 1940). A numerical example of the Friedman test is provided in Appendix B.

3.1.3 Combined Classification of Trend and Seasonality

When looking at just a single year worth of data, we notice that we cannot be sure as to why a SKU may have an increasing / decreasing demand over the course of a year. This is because this effect can have two sources, being seasonality and presence of a trend.

We construct an arbitrary time series, and provide its decomposition. This is depicted in Figure 3.2 on the next page. When looking at the figure, we notice that our seasonality component does not only have seasonality, it also has a trend. When we apply regression for trend first, we notice that this "trend" component of the seasonality is wrongly attributed to trend. In case we were to remove seasonality first, we would wrongly attribute our trend to seasonality. As we get access to multiple years of data, the impact of this decreases, but it never fully vanishes.

Ideally, we are able to assign the trend component to our trend, and the seasonality component to our seasonality, without getting these two mixed up. Given we have access to multiple years

worth of data (which, in fact, is required to be classified as a seasonal item), (Silver et al., 2016) find a way to do exactly this, making use of moving averages. A full explanation is provided in section 3.4.4.3 of (Silver et al., 2016), but what it ultimately boils down to is a removal of the trend effect, after which seasonal factors can be obtained. Then, the entire time series can be deseasonalised, and we can perform linear regression for our trend component. From now on, we refer to this as being the seasonality first approach.

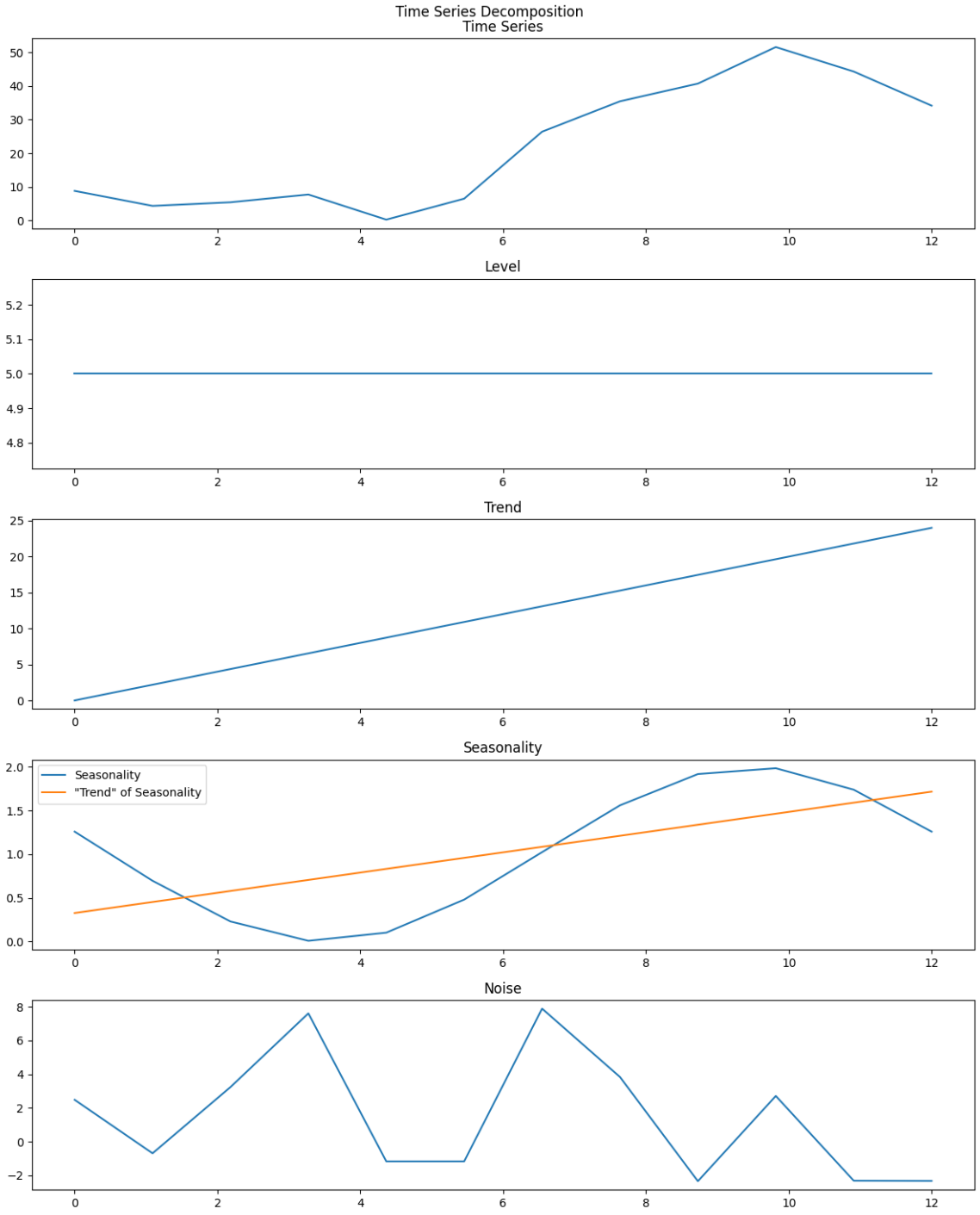

Figure 3.2: Time Series Decomposition

3.2 Forecasting Methods, Algorithms, and Regressors

Now that we have researched which methods can be used to classify various time series, we are going to look into the available forecasting models to forecast said time series. We answer our next sub-question.

- *Sub-question B: "Which forecasting models are available for forecasting demand for different SKU types?"*

To get a good idea of how much inventory to keep for the upcoming period, we need to make a forecast on how many units we are going to need. In order to make such a forecast, we need to select an appropriate forecasting model. There are many different types of forecasting models, some being fairly simple statistical models such as time series models, regression models, and exponential smoothing, and some being more complex methods such as neural networks and machine learning regressors. In this section, we will cover a wide variety of models by exploring their workings, strengths, and weaknesses. During the modeling phase of our research, we will come up with some ideas to use these models and improve our forecasts.

We start off with the statistical models. Statistical forecasting models have been around for a long time, and for good reason. They are oftentimes simple to compute, and they tend to work fairly well. In fact, statistical models such as Holt's method (dating back as early as 1957 (Holt, 2004)) have dominated for a very long time, with statistical methods still beating out Machine-Learning models in the M3 competition (Makridakis et al., 2018). Only in recent years have hybrid models and pure machine learning models started to arise, with 2020's M4 competition (Makridakis et al., 2020) featuring hybrid models and 2022's M5 competition featuring mostly machine learning models among top performers (Makridakis et al., 2022). Despite most state-of-the-art models being machine learning models, statistical methods have not disappeared. Many machine learning models still benefit from statistical preprocessing such as trend removal and deseasonalisation (Makridakis et al., 2020), (Zhang and Qi, 2005).

3.2.1 Statistical Models

There are two widely applied statistical models in the field of time series forecasting. These are exponential smoothing and its extensions, and ARIMA and its extensions. We will briefly cover these models here. More details can be found in Appendices C and D.

Simple Exponential smoothing

Simple exponential smoothing (SES) is a simple forecasting model suitable for time series that do not have a trend or seasonality present. The forecasting formula for SES is as follows:

$$F_t = a + \epsilon_t \tag{3.3}$$

Simple exponential smoothing only forecasts based on level a. This level is updated every period. We use smoothing parameter α to update the level. Greater values for α correspond to greater updating (Silver et al., 2016).

Double Exponential Smoothing

Double exponential smoothing is an extension to simple exponential smoothing, including one additional parameter β for a present trend. Double exponential smoothing forecasts based on the

following formula:

$$F_t = a + bt + \epsilon_t \tag{3.4}$$

Where b is the trend. This trend is updated in a similar manner to the level as covered before. (Silver et al., 2016).

Holt-Winters Exponential Smoothing

In addition to a trend and a level, some time series also have a seasonality present. Seasonality at a given time t is indicated by S_t. This yields the following forecasting formula:

$$F_t = (a + bt) \cdot S_t + \epsilon_t \tag{3.5}$$

Seasonality is smoothened with parameter γ in a similar manner to level and trend (Silver et al., 2016). More on Holt-Winters exponential smoothing can be found in Appendix C.

Auto Regressive Moving Average methods (ARMA Family)

The main alternatives to exponential smoothing models are models of the ARMA family, most notably ARIMA and seasonal ARIMA. We explore these models as described in (George E. P. Box, 1994) and (Wan Ahmad and Ahmad, 2013). Models belonging to the ARMA class consist of two main parts, being an autoregressive (AR) part, and a moving average (MA) part. ARMA models are always based on a predetermined number of past observations for both the AR and the MA part. This number of observations is referred to as the order.

The autoregressive part of the model models the value of a time series as a linear combination of its past values. The $AR(p)$ model is denoted by:

$$F_t = \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t \tag{3.6}$$

Where p is the number of significant time periods, X_{t-i} is the value at period $t - i$, and ϕ_i is some scalar dependent on the significance of the period.

The moving average part of the model models the value of the time series based on past forecast errors. It has order q. We define the $MA(q)$ model as follows:

$$X_t = \mu + \sum_{i=1}^{q} (\phi_i \cdot \epsilon_{t-i}) + \epsilon_t \tag{3.7}$$

Where μ is the level. Other parameters are as before.

Combining these AR and MA models, we get the following $ARMA(p, q)$ model:

$$X_t = \mu + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{i=1}^{q} (\phi_i \cdot \epsilon_{t-i}) + \epsilon_t \tag{3.8}$$

Note that this model only works for stationary time series. We can extend the ARMA model to the ARIMA model, which first differentiates non-stationary time series by order d in order to make them stationary. We can then further extend the model to also allow for seasonality. This seasonality extension is commonly referred to as seasonal ARIMA. More on ARMA models and

their extensions can be found in Appendix D.

Fourier Transforms

Fourier analysis is a mathematical technique that is often used in signal processing and time series analysis, although it is not very common in demand forecasting. Despite this, prior research shows potential to the application of FFTs for forecasting highly seasonal SKUs. The paper by (Musbah and El-Hawary, 2019) proposes an improvement to the seasonal ARIMA model by extracting seasonal factors using the Fast Fourier Transform algorithm. The Fast Fourier Transform (FFT) is an algorithm that can transform a time series into its frequency components, all of which correspond to certain fluctuations over time. In case a pattern is present, some frequency components will be dominant. After filtering out dominated frequency components, which we assume to be mostly noise, we can apply the inverse Fourier transform (IFFT) to obtain the pattern that is present in our time series.

(Fumi et al., 2013) go even further in applying the Fourier transform for forecasting, proposing a simple FFT forecasting model for forecasting demand in the fashion industry, which is an industry that often has uncertain products, consisting of short-lifespan products that are often highly influenced by seasonal effects, promotional effects, and social factors among others. Despite being a field that appears to be highly unpredictable, the Fourier transforms yielded large improvements over moving averages and exponential smoothing.

3.2.2 Machine Learning Models

Machine learning models have seen a sharp rise in the field of demand forecasting over the last couple of years. Starting with a transition from purely statistical models to hybrid models (Makridakis et al., 2018), (Makridakis et al., 2020). The era of hybrid models did not last for long, however, as the transition from statistical models to hybrid models was quickly followed up by the transition to pure machine learning models (Makridakis et al., 2022).

In 2018, about five years ago, dominant machine learning models were Bayesian Neural Networks (BNN), Multilayer Perceptrons (MLP) and Support Vector Regressors (SVR) (Makridakis et al., 2018), though none of these models managed to outperform statistical models. Despite not being as performant in (Makridakis et al., 2018), the Long Short-Term Memory (LSTM) recurrent neural network saw some action as an alternative to ARIMA in 2018 and 2019 (Siami-Namini et al., 2018), (Siami-Namini and Namin, 2018), (Siami-Namini et al., 2019) (Abbasimehr et al., 2020), though this seemed to be short-lived as it had already fallen out of flavour and showed no presence in 2020's M4 competition (Makridakis et al., 2020).

Gradient Boosters

Gradient boosting algorithms are very popular machine learning algorithms due to their high performance and versatility. They perform well in a wide variety of tasks, including time series forecasting as shown by their dominance in 2022's M5 forecasting competition (Makridakis et al., 2022). Gradient boosting is a ensemble machine learning method that is highly suitable for classification and regression. Being an ensemble method, gradient boosting can be used to create well-performing models through combining several weak learners, typically coming in the form of decision trees (classification) and regression trees (regression) (Bentéjac et al., 2021). In the case of time series forecasting, we are dealing with the latter.

Gradient boosting starts off by a single value or regression tree, known as the base learner. Oftentimes, this single value of regression tree will not perform well. To improve performance

of the model, we extend the model by creating an additional regression tree based on the errors made by our previous regression trees. The new tree improves the performance of the model by reducing error through minimisation of a loss function. After adding the newly constructed tree, we can update our predictions and recompute errors. We iteratively improve the model by adding more trees until we reach the maximum number of trees. This iterative approach allows use to generate models tend to perform well and are relatively easy to train (Friedman, 2001), (Friedman et al., 2000), (Friedman, 2002).

During our literature study, we investigated the three most popular gradient boosting algorithms, being CatBoost (Prokhorenkova et al., 2018), (Dorogush et al., 2018), LightGBM (Ke et al., 2017), and XGBoost (Chen and Guestrin, 2016). Information on these algorithms is found in Appendix K. As we do not use them in our final model, they have been excluded from the main body of this report.

Prophet

Another candidate model is Prophet, which is a forecasting model developed by Facebook. In addition to the usual level and trend, Prophet is able to fit yearly, weekly, and daily seasonalities. Moreover, Prophet supports adjustment for holiday effects by default. It is able to handle missing data and outliers very well and thus does not need a lot of preprocessing (Taylor and Letham, 2018). Several studies have shown Prophet to outperform ARIMA and Seasonal ARIMA by a significant margin (Aditya Satrio et al., 2021), (Samal et al., 2019), (Yenidogan et al., 2018).

Prophet was designed with ease-of-use in mind. Although analysts and consultants often have a lot of knowledge in their field, this knowledge is not often paired with a deep understanding of time series and forecasting. Models like ARIMA often demand a deeper understanding of the workings of the models to work well, as the model is sensitive to parameters such as the order of differencing, order of moving average components, and the order of auto-regressive components. Prophet was designed to make forecasting scalable, accurate, and intuitive. (Taylor and Letham, 2018)

Prophet is a time series model based on decomposition of the time series (Harvey and Peters, 1990). The model has four components, being three components that we can model and a noise component, which is simply estimated by a normal distribution. The three modeled components are trend, seasonality, and holidays. (Taylor and Letham, 2018) provide the following equation:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t \tag{3.9}$$

Where $y(t)$ is the forecast, $g(t)$ is the trend function, $s(t)$ denotes seasonality, $h(t)$ represents holidays and events, and ϵ_t is noise. As we can see in Equation 3.9, Prophet uses an additive model by default, though we can change the model to use multiplicative seasonality by means of a log transform (Taylor and Letham, 2018). The additive model is similar to the generalized additive models as described in (Hastie and Tibshirani, 1987) and (Gardner, 1985).

Prophet allows us to fit the model for two types of trends, being the saturating growth model and the piecewise linear model. The saturating growth model is essentially an extended exponential growth model, as the model allows for exponential growth that stagnates once it starts to reach a certain level known as the carrying capacity. In its most basic form, the model

is denoted by:

$$g(t) = \frac{C}{1 + exp(-k(t - m))} \tag{3.10}$$

With C being the carrying capacity, k being the growth rate, and m denoting the offset. As mentioned in Appendix C, our research does not cover SKUs that are in the exponential growth/decline phase of their product life cycle, and thus we will not explore the saturating growth model in more depth than we have done so far.

The type of trend we will likely encounter is the piecewise linear trend. The piecewise trend is a model that allows us to use a linear trend with changepoints. This means we model the trend as a constant rate of growth for each time period, and allow for this rate of growth to change at each of our changepoints. The number of changepoints can be estimated by the analyst or automatically selected by Prophet (Taylor and Letham, 2018). In our research, we let Prophet determine the number of changepoints.

Allowing these growth rate changes and trend reversals introduces some uncertainty into the model as we cannot be certain about the changes the trend will undergo in the future. To overcome this uncertainty, (Taylor and Letham, 2018) make an assumption the trend will continue to change with similar magnitude and frequency. (Taylor and Letham, 2018)

Prophet relies on a Fourier series for modeling seasonalities and periodic effects (Harvey and Shephard, 1993). As Fourier series are dependent on the period we are analysing, choice of this parameter is important. Prophet uses a period $T = 365.25$ for yearly seasonality and $T = 7$ for weekly seasonality. Note that Prophet does not use monthly seasonalities by default. If desired, we can manually add custom seasonalities to Prophet and include monthly seasonality despite this not being allowed by means of a Fourier transform (Taylor and Letham, 2018).

Using all components of the Fourier series often causes the model include noise and thus overfit. To overcome this, we can use a filter known as a low-pass frequency filter. A low-pass frequency filter is a filter that truncates the Fourier series at a certain frequency known as the cutoff frequency. That is, we modify the Fourier series such that only fluctuations over longer timespans remain. This ensures the modeled seasonality only contains fluctuations that slowly and steadily occur over time. Fluctuations that occur rapidly over time are assumed to be noise. An example of how a low-pass filter works can be found in Appendix E.2.2. By default, Prophet uses a cutoff frequency of 10 for its yearly seasonality. This means that we allow the Fourier series to contain at most 10 components. These are a level component (which is a sine that has a frequency of 0), and sines with frequencies of 1 Hz to 9 Hz. A single Hz corresponds to one dip and one peak per year. For daily seasonality, Prophet uses a cutoff frequency of 3. One can choose to manually alter these parameters, or automatically change them based on the Akaike Information Criterion (AIC) (Stoica and Selen, 2004). (Taylor and Letham, 2018). In our research, we use the default settings as provided by Prophet.

Inclusion of holidays and events is an area where Prophet may have an advantage over traditional statistical models. Holidays and events often change demand for certain SKUs around their date of occurrence, and Prophet allows us to model them and include these changes in our forecast. Prophet models the effects of holidays using regression. This also includes a window around the holiday/event itself as demand on these days are oftentimes also affected by the holiday. (Taylor and Letham, 2018).

Generalized additive models such as prophet have some practical advantages when compared to generative models such as ARIMA. Firstly, the model is able to fit very fast by using an optimization algorithm known as L-BFGS (Byrd et al., 1995). L-BFGS is an optimisation algorithm that works on minimising the error function by iteratively adjusting input parameters. Because Prophet essentially models the problem as a curve-fitting regression problem instead of solving the problem by means of ensemble methods such as CatBoost, we face less computational costs for making our forecast. Secondly, the components of the model are very intuitive and thus easy to interpret and evaluate for our analysts. Consultants in the field of inventory management often know a lot more about seasonality than they do about measures such as autocorrelation, which inherently makes the additive Prophet model more intuitive than the generative ARIMA model. This allows the analyst/consultant to better estimate seasonality and trend over multiple periods. (Taylor and Letham, 2018)

3.3 Performance Measures

Having researched a wide range of classification methods and forecasting algorithms, we can proceed on to the last part of the literature studies, which covers measuring of performance. We answer the third sub-question, which is as follows:

- *Sub-question C: "How can we measure performance of used forecasting models?"*

There are two important variables in measuring performance of our forecasts. These are the accuracy of the forecast, and the bias of the forecast. In this section, we cover metrics that can be used to numerically express and compare these variables.

Accuracy

We need to measure their performance by means of performance measures for two main reasons. First, "historical performance of a forecasting process helps us create the description of future demand to support resource allocation decisions" (Silver et al., 2016). Second, by keeping track of forecasting performance, we can potentially find ways to improve our forecasts (Silver et al., 2016).

Forecast performance is commonly evaluated by accuracy measures (De Gooijer and Hyndman, 2006), (Makridakis et al., 2022). Some commonly used performance measures are: MSE, RMSE, MAD/MAE, MAPE, sMAPE (De Gooijer and Hyndman, 2006), (Silver et al., 2016). In the M5 accuracy competition, MASE (Hyndman and Koehler, 2006), RMSSE (Hyndman and Koehler, 2006) and WRMSSE (Makridakis et al., 2022) are brought up as additional performance measures. For all performance measures covered here, lower scores mean lower forecasting errors and thus more accurate forecasts. Covered performance measures, definitions, and potential remarks are provided in Table 3.1 on the next page.

Choosing a suitable forecast metric depends heavily on the cost of forecast error, which is the type of error that is least desirable. In addition, when comparing performance across different time series, we commonly use percentage errors as they are scale-independent. (Hyndman and Koehler, 2006). Depending on the case, we may need different performance measures.

Bias

We track the bias of our forecasts such that we can know whether we are consistently underforecasting or overforecasting. (Trigg, 1964) propose tracking the bias by means of a

Table 3.1: Performance Measures

Measure	Definition	Comments
MAD/MAE	$MAD/MAE = \frac{1}{n}\sum_{t=1}^{n}\lvert D_t - F_t\rvert$	Forecasts are medians of future distributions. For Normal distributions: $\hat{\sigma_1} = \sqrt{\frac{\pi}{2}} \cdot MAD \approx 1.25 \cdot MAD$
MAPE	$MAPE = 100\% \cdot \frac{1}{n}\sum_{t=1}^{n}\lvert\frac{D_t - F_t}{D_t}\rvert$	Not useful for low demand, undefined for $D_t = 0$
MASE	$MASE = \frac{\frac{1}{n}\sum_{t=1}^{n}\lvert D_t - F_t\rvert}{\frac{1}{n-1}\sum_{t=2}^{n}\lvert D_t - F_t\rvert}$	MASE is the ratio of current MAD to prior MAD
MSE	$MSE = \frac{1}{n}\sum_{t=1}^{n}(D_t - F_t)^2$	$\hat{\sigma_1} = \sqrt{MSE}$
RMSE	$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(D_t - F_t)^2}$	$RMSE = \hat{\sigma_1} = \sqrt{MSE}$
RMSSE	$RMSSE = \sqrt{\frac{\frac{1}{h}\sum_{t=n+1}^{n+h}(D_t - F_t)^2}{\frac{1}{n-1}\sum_{t=2}^{n}(D_t - D_{t-1})^2}}$	Only computed for products that are actively sold.
sMAPE	$sMAPE = 100\% \cdot \frac{1}{n}\sum_{t=1}^{n}\frac{\lvert D_T - F_t\rvert}{(\frac{\lvert D_t\rvert + \lvert F_t\rvert}{2})}$	Like MAPE, but symmetric unlike MAPE, has an upper and lower bound.
WRMSSE	$WRMSSE = \sum_{x=1}^{X} w_x \cdot RMSSE$	Weighted RMSSE. X denotes number of SKUs, w_x is weight of SKU x

tracking signal, defined as follows:

$$\text{Smoothed error} = (1-\alpha)\text{ previous smoothed error} + \alpha\text{ latest error} \tag{3.11}$$

$$\text{M.A.D.} = (1-\alpha)\text{ previous M.A.D.} + \alpha\text{ latest absolute error} \tag{3.12}$$

$$\text{Tracking signal} = \frac{\text{Smoothed error}}{\text{M.A.D.}} \tag{3.13}$$

Using this system, a tracking signal of 0 implies no bias. Values close to -1 means we are consistently overforecasting, whilst values close to 1 mean we are underforecasting.

3.3.1 Conclusion

Based on our literature research, we see value in extending the seasonal models to include multiple levels of seasonality. In our research, we will investigate the effects of including weekly and daily seasonality in addition to the currently used monthly seasonality.

In addition, we find inspiration in the use of Fourier transforms. (Musbah and El-Hawary, 2019) propose an improvement to the seasonal ARIMA model by using Fourier transforms. In our research, we investigate whether triple exponential smoothing can also benefit from seasonal extraction and processing using Fourier transforms.

Lastly, we find that LightGBM and Prophet are candidate alternatives to exponential smoothing. Both of these algorithms have proven to perform well based on the M5 accuracy competition (Makridakis et al., 2022) and we will consider both of these models during our experimental design.

4 Methodology

In this chapter, we explain the methodology we use in our research. We first explain how the data used in our research are collected and preprocessed. Then, we explain the strategy used to evaluate model performance and select the best model. Lastly, we explain how our models are validated. We recite our research third question for clarity,

- *Research question 3: "How can we evaluate our newly developed model?"*

4.1 Data Collection & Preparation

To answer our fourth research question, we first answer our first sub-question:

- *Sub-question A: "How are the data collected and prepared?"*

Data used in the research have been collected from one of Slimstock's customers, whom will be kept confidential in this report. Data have been provided in the form of a plain text file that includes all historical transactions from May 1st, 2018 to May 10th, 2023. The data set includes over 100.000.000 transactions of over 50.000 different stock keeping units.

In addition to transaction data, we also have access to the demand classifications as made specified by Slim4. These are the classifications covered in Section 2.1.

We preprocess the data by aggregating all transactions for each product on a daily level. Then, we filter out all SKUs that are not in demand class frequent, normal, or lumpy. This data is then cast to a tabular format.

Once we have this tabular format, we filter out all SKUs that are not in the maturity phase of their product life cycle. Although we do not have exact classifications on which products are in the maturity phase of their product life cycles, we use a heuristic. This means that, to remove SKUs that have been introduced somewhere during the five years of our data, we remove all SKUs that do not have any demand in the first 10 days (being between 01-05-2018 and 10-05-2018). To remove SKUs that are no longer sold, we remove all SKUs that are in our data, but not in our list of currently managed products as obtained from Slim4.

Once we have completed this procedure, we are left with a very large table of 9.419 rows (one for each product) and 1826 columns (one column for each day). This table will be used as an input for our models.

4.2 Model Performance

Now that our data have been prepared, we proceed on to answering our second sub-question:

- *Sub-question B: "How are we going to measure and express our model performance and select the best method?"*

Following from our literature on performance metrics, combined with the fact that we now have more insights into our data, we can choose a set of performance metrics that will be used to evaluate and express model performance. Chosen metrics will be used to select the best models later in our research.

Cost of Forecasting Error

As mentioned before, the best metric to evaluate and compare performances depends on the cost of the forecasting error. Although we use data from a single customer in our research, the

research aims to be applicable to as many similar fields as possible. In addition, we do not have any information on which types of errors are more costly for our customer.

Selected Error Metrics

Because of this, we have chosen to use four different performance metrics, three of which give an indication of accuracy. These three metrics are: RMSE, sMAPE, and MAD.

We have chosen to use these metrics for several reasons. Having researched the metrics of: MAD/MAE, MAPE, MASE, MSE, RMSE, RMSSE, sMAPE, and WRMSSE, we selected our three metrics for the following reasons.

Firstly, MAD/MAE. We have chosen to use MAD as one of our performance metrics as mean absolute deviation gives a good indication of the average magnitude of our forecasts, although it does not measure costs in any way.

Our second metric of choice is RMSE. RMSE, or root mean squared error, is a suitable metric for measuring the accuracy of our forecasts, especially when larger deviations are punished more heavily. Note that RMSE is often used in determining the level of safety stocks.

Lastly, we have selected sMAPE. sMAPE, sMAPE, like MAPE, gives an indication of the percentage error of the forecast. Higher percentages mean larger errors. Note that, because sMAPE is symmetric, values for sMAPE range from 0% to 200%.

Excluded Metrics

In addition to our chosen metrics, we also researched some metrics that we decided not to use. We will briefly explain why we have excluded these measures.

Firstly, MAPE. MAPE was not selected because it is undefined in cases where demand is equal to 0. Instead of MAPE, sMAPE is used, which solves this problem.

Secondly, MASE. MASE is a scaled variant of MAD. Because we are going to scale our time series for evaluation of performance manually for the purpose of fairly comparing our other performance metrics, MASE is not included. MAD is included instead.

Thridly, MSE. MSE has been excluded because its more common counterpart, RMSE, has been used.

Fourthly, RMSSE. Although RMSSE is an interesting metric for sure, we note that its main advantage lies in the fact that it only computes products that are actively sold. In our research, all products are actively sold and thus this metric appears to be obsolete. Note that RMSSE is a variant of MASE, which in turn is a variant of MAD.

Lastly, WRMSSE. WRMSSE has been excluded from our research for the same reasons as RMSSE. In addition, WRMSSE is a scaled version of RMSSE. As we do not have access to the weights of our time series, this metric is not feasible.

Bias

As the main aim of the research is the development of an accurate forecasting method, we focus mostly on measures of accuracy. That being said, bias of a model is still evaluated as a tiebreaker. Bias of models is also evaluated in Slim4 by default. The metric is used to indicate which forecasts perform poorly (i.e. which time series see an unexpected change in demand), allowing the consultant to take action immediately.

Normalisation

Our research uses many different time series, some of which are of larger magnitude than others. To prevent the time series of larger magnitude from having a greater impact on our performance metrics, all time series have been normalised such that the maximum demand is equal to one. This is done by dividing every time series by its own maximum value. Doing so ensures all time series are considered to be equally important in our comparisons.

4.2.1 Selection of the best Models

The aim of this research is to find methods that are significantly better than the current method. We make this assessment mainly based on our three accuracy-based performance metrics.

Simply looking at the values of our performance metrics for the old and new models is not enough. Although a lower number for our RMSE is better, we cannot directly conclude the model with the lower number is always the better model.

We can confidently conclude a model is better than another model when two requirements are met. Firstly, the difference has to be sufficiently large. Secondly, the difference has to be present consistently. This way, we know the difference in performance is not due to chance alone.

To check whether a model is better than another model, we use a statistical test that checks whether both of these requirements are present. Specifically, we use a Student's t-test on the difference between two means.

Our research use a two-tailed test rather than a one-tailed test because we are interested in finding whether the mean values of our performance metrics are statistically different. Although the use of a one-tailed test is better for checking whether or not a model is significantly better than another model, this will give us some problems later on. This is because it is possible for a model to be better on a certain performance metric, and worse on another performance metric (although by a smaller margin). In these cases, we have to evaluate both if a model is significantly better and if it is significantly worse.

Domination, Multiple winners, and Finding the best Method

The use of multiple performance metrics has its advantages and disadvantages. Although the use of three performance metrics gives more information on performance than the use of a single metric would, this also means we can end up in difficult situations.

Selection of the best model when comparing two models is easiest in case one of the models is clearly better than the other model on all of our performance metrics. The better model scores significantly lower on all of our accuracy measures as confirmed by our t-test, so that model is the better model. Extending this, a model is also better than another model in case at least one of its performance metric scores is significantly better, and the other ones are a tie.

The most difficult choices need to be made in case a model is better on a certain performance metric, and worse on another. In this case, we have to test for two things. Firstly, we have to test whether the performance on the better metric is significantly better, and if the performance on the worse metric is significantly worse. In case the better performance is significantly better, and the differences in performance on the other metric is not statistically significant, it is still possible to pick our best model. In case both tests return significant differences, we cannot make a decision and we have to re-evaluate our metrics and method.

It can occur that we find multiple methods that are significantly better than the current method, though the difference between the two found alternative methods is not significant. In that case,

we make our decision based on the p-value of our t-test. The model that has the lower p-value is selected as the better model. In case neither of the models have lower p-values than the other model on all performance metrics, we make our decision based on root mean squared error. This is because a lower value for root mean squared error is directly linked to lower safety stocks.

Decision Making Regarding Model Selection

When comparing two distinct models, one of the two models will be better than the other unless their performance is exactly identical, which is rarely the case. In theory, we can always find the better model by providing both models with an infinite number of samples, and seeing which one performs best. In reality though, we do not have an infinite number of samples we can use to evaluate our model. This can lead to either of four scenarios:

1. The current method is significantly better based on our t-test.
2. The alternative method is significantly better based on our t-test.
3. The current method is better based on performance metrics, but the difference is not significant based on our t-test.
4. The alternative method is better based on performance metrics, but the difference is not significant based on our t-test.

Deciding on a model in scenarios 1 and 2 is trivial. We select the model that is significantly better. Making a decision in the third scenario is also fairly easy, as we can stick to the current method, and either refine the third method or evaluate it using more samples (or a combination of both). The fourth and last scenario is the scenario where risk-takers may opt for the alternative method, whereas the risk-averse may stick to the current method. In case we have multiple models that are significantly better than the current method, we pick the best one based on RMSE.

This leads us to two different ideologies we will use for selection of our best forecasting models. We refer to these as the conservative approach, and the progressive approach.

Conservative Approach

The conservative approach is the approach that likes to stick to the current method unless another method is proven to be better. This means that, when making the decision to change from one approach to another approach, the decision maker can be fairly certain of a net gain in performance.

Progressive Approach

The progressive approach is focused on making improvements at all cost. Like the conservative approach, the progressive approach always favours a model over another model in case its performance is significantly better. Unlike the conservative approach, however, the progressive approach will always select the model that is best based on performance metrics, even if the difference is not proven to be statistically significant.

4.3 Validation & Verification of the Models

Ultimately, a correct model has two requirements. Firstly, a model has to be specified to complete the task we want it to perform using a suitable method. Secondly, the model has to be implemented such that it performs as specified.

Checking whether the first requirement of a model is met is known as validation. When validating a model, we check whether a method (e.g. exponential smoothing) is suitable for solving the problem we attempt to solver (i.e. forecasting demand).

The second requirement is related to the output and implementation of the model. We check whether the model output is correct. This is known as verification.

This leads us to the following research question:

- *Sub-question C: "How do we validate and verify our models?"*

4.3.1 Validation

All methods and algorithms we use in our studies are established models that have appeared in prior research. We refer to literature on these models for the verification of our methods. For our used methods, literature is provided in Chapter 3, and in the appendices that cover these models. These are Appendix G and Appendix I.

4.3.2 Verification

Verification of the models is fairly straightforward. After implementing a model, we look at examples that are present in literature, and check whether our model outputs the same results. Validation for our models is mainly done using (Silver et al., 2016) and (Facebook, 2023).

Our research also includes adaptations of the model as specified by Slimstock. These models are not present in literature. Therefore, these models have been verified internally by Slimstock.

4.4 Notes on Implementation

All of our models are evaluated using the general-purpose programming language Python, version 3.11.5. An implementation report is found in Appendix G and Appendix I.

5 Experimental Design

Now that we know how our models will be evaluated, we move on to selection and specification of our models. We will do this by answering our fourth research question.

- *Research question45: "What will our models look like?"*

5.1 Classification Methods

Our first sub-question is as follows:

- *Sub-question A: "Which classification methods on trend and seasonality (from literature) are we going to compare in our research?"*

We have decided to use linear regression for classification of trends. We will research the Fisher's F-test and the Friedman-Chi-Squared test for classification of seasonalities. More information on the Fisher's F-test can be found in Appendix A. More information on the Friedman test can be found in Chapter 3.1.2. A numerical example can be found in Appendix B.

5.2 Forecasting Models

- *Sub-question B: "Which methods (from literature) are we going to be using as a starting point for our forecasting models?"*

Our literature research yielded three potentially suitable types of models, being exponential smoothing methods, regression methods, and machine learning methods. We have several types of models in each of these classes.

For exponential smoothing methods, we have single exponential smoothing, double exponential smoothing, triple exponential smoothing, double nested exponential smoothing, and triple nested exponential smoothing. In addition to models found in literature, we also have triple exponential smoothing as specified by Slimstock. For regression methods, we have found ARIMA and Prophet. For machine learning methods, we have found LightGBM, CatBoost, and XGBoost.

5.2.1 Exponential Smoothing Methods

We first explain the workings of exponential smoothing as implemented in our research. All models are based on the same methodology, though some models have one or multiple extensions depending on their complexity. The exact specification of all of our exponential smoothing models is provided in Appendix G.

At its core, all of our models have three phases. These phases are: the initialisation phase, the updating phase, and the forecasting + updating phase. Note that our research uses 5 years worth of data. We explain the full procedure in the following paragraphs.

Initialisation phase

The initialisation is the first step in all of our models. In this phase, we use the first 3 years of data (in our case, data from 11-05-2018 to 10-05-2020) to get an initial estimate of the parameters in our model. For stationary time series, this is just the level. For trending time series, this is level and trend. Seasonal time series have level and seasonality parameters. Seasonal time series that also have a trend use all parameters, being level, trend, and seasonality.

Updating Phase

The updating procedure is a natural extension of exponential smoothing (Silver et al., 2016). The updating phase serves two main purposes. Firstly, updating a model removes the need to initialise it again every time we need to make a forecast. This reduces computational cost significantly. Secondly, the updating phase allows us to assign exponentially more weight to more recent observations, with higher smoothing factors increasing this effect. An example of the updating procedure is found in Section G.1.1, where model 1 (SES) is validated. In our models, level is always updated. If present, trend is also updated. Seasonalities are never updated. Instead of updating seasonalities, they are recalculated. This is done as by specifications of Slim4.

Our research does not change this specification, and thus all of our models use re-computation of seasonality rather than smoothing of seasonality. Seasonalities are updated at the beginning of a new period, not halfway during a period (i.e. we only update seasonality for January once we need to calculate a forecast for a given day in January). In practice, for Months and Weeks, this comes down to updating seasonalities once a year. For daily patterns, we decide to update once a year as well. Although we can theoretically update every week, we have chosen not to do so for computational reasons. This may lead to slightly different results compared to weekly updating, but as our experiments are not computationally feasible without this simplification, we have opted for this approach. We estimate effects of this simplification are very minor if at all significant. In our research, updating the models is done over the course of a year, using data from 11-05-2020 to 10-05-2021.

Forecasting + Updating Phase

Once we have performed initialisation and updating as described before, we are going to make forecasts. These forecasts will be used to evaluate the performance of our models. Evaluation of the forecasting performance is done over the course of a year.

Slim4 creates forecasts for 1 period ahead every time we need to make a forecast. This means we are always using the latest information available, without using information about the future. Once we have made the forecasts, we let time pass, meaning the model is updated on a daily basis. Once we need to make a new forecast, we again forecast 1 period ahead. The entire procedure of forecasting and updating is done until we have made forecasts for an entire year.

Our research uses 3 cover periods, being short (3 days), medium (7 days), and long (21 days). This allows us to make some computational optimisations regarding our forecasts. These optimisations are due to the fact that we can make forecasts simultaneously, and do not have to run the model three times. We cover the entire procedure in Appendix G.1.1 which covers the verification of model 1 (SES).

5.2.2 Regression Methods

Our literature research yielded two different regression methods, being ARMA models, and Prophet. In our model design, we opted to select Prophet as a candidate and discarding all models of the ARMA family.

As covered in Appendix D, ARMA models have been an alternative to exponential smoothing models, and although various research has been conducted on finding which of the two performs better, no general consensus has been found. We have discarded the ARMA models as researching them did not align with the interests of our stakeholders. In addition, both exponential smoothing and ARMA models have been shown to be outperformed by Prophet (Aditya Satrio et al., 2021), (Samal et al., 2019), (Yenidogan et al., 2018).

As we just mentioned, Prophet, which is our other regression model, has been shown to outperform exponential smoothing and ARMA methods. In addition, it is easy to implement and use. Therefore, we have decided to include Prophet in our research.

5.2.3 Machine Learning Methods

Although machine learning methods have shown to perform very well in the M5 accuracy competition (Makridakis et al., 2022), we have dropped them from our research at a fairly early stage. This is mainly because, upon further research, we found these algorithms rely heavily on external features such as historical sales price data, which we do not have access to. Using the model without access to these features delivers performance nowhere near the performance of the other models we research. Being a machine-learning model, performance is heavily dependent on hyperparameter tuning, which takes a very long time. Because of these reasons, we decided machine learning methods in the form of gradient boosters are beyond the scope of this research. Despite exclusion of these methods for now, in case Slimstock can get more access to data in the future it may be worthwhile looking into those methods again.

5.3 Parameters

Now that we have selected our forecasting models, being exponential smoothing models and Prophet, we are going to look at the relevant parameters in these models. This answers our next sub-question:

- *Sub-question C: "What are the main parameters in these models?"*

5.3.1 Exponential Smoothing Parameters

Exponential smoothing as we are going to implement in our research has a single parameter α for single exponential smoothing, and two parameters, being α and β for double exponential smoothing. In addition to these parameters, model performance is affected by the duration of the initialisation period and the updating period. (Silver et al., 2016) suggest an initialisation period of at least 4 years and an updating period of a single year or more. Having access to 4 years of data for initialisation, we decide the best we can do is three years for initialisation and a single year for updating the model. Once this is completed, the model can continue to be used using just forecasting and updating. Exact workings and implementation of the model is covered in Appendix G.

5.3.2 Prophet Parameters

Although the prophet model has a large number of parameters that can be adjusted in our code, Facebook does an excellent job highlighting the most important parameters for us. These parameters and their effects are as follows:

1. Changepoint prior scale. This parameter is responsible for the flexibility of the trend. This parameter is essentially a lasso penalty. Values that are too low cause the trend to underfit, meaning changes in trend are considered to be noise. Values that are too high cause the trend to overfit Facebook (2023).

2. Seasonality prior scale. This parameter is responsible for the magnitude of seasonal effects. This parameter is essentially a ridge penalty. Higher values mean there is less regularisation. Lower values apply more regularisation. (Facebook, 2023) note that the default value of 10 oftentimes works well as there is inherent regularisation because of the way seasonality

is modeled. The use of a Fourier series combined with a low-pass filter effectively reduces overfitting by itself, and this additional regularisation is oftentimes not needed.

3. Holidays prior scale. This parameter is responsible for the magnitude of the effect of holidays on our model. Like seasonality prior scale, this parameter applies a ridge penalty. (Facebook, 2023) note that the default value of 10 (very little regularisation) usually works well if we have multiple observations of each of our holidays. In case we have fewer observations, we can decrease the value of this parameter to apply more regularisation such that we do not overfit to our holiday effects.
4. Seasonality mode. This parameter is used to specify whether the model used is an additive model or a multiplicative model.

5.4 Parameter Optimisation

Now that we have selected our models and identified the most important parameters, we are going to look into optimisation of the models. Hence, our next research question is as follows:

- *Sub-question D: "How are we going to optimise our model parameters?"*

We start off by covering the optimisation of parameters for the default model and its derivations, being the exponential smoothing models. Then, we will look into optimisation of Prophet.

Optimisation of Exponential Smoothing Parameters

Slim4 uses a single value for its updating parameters α and β. As of now, updating is done on a monthly basis. Slimstock wishes to transition to the use of daily models, which are also used in our research. Therefore, we have to find new values for α and β. As we explore in Appendix C, (Silver et al., 2016) explore various ranges of values for different updating periods. For our research, we decide to go with $\alpha = \beta = 0.01$ daily. We have chosen to use this value as this is the lower end of the range defined in (Silver et al., 2016). We use the lower end of the range because the models have been modified to use a higher updating frequency than before. Higher updating frequencies work better with lower smoothing factors. As by specifications of Slim4, smoothing parameter γ for seasonality, is not used.

Although optimal values for our smoothing parameters may vary depending on the environment, Slim4 does not optimise its smoothing parameters due to computational constraints. Therefore, our research uses values as recommended by literature.

Parameter Optimisation of Prophet

Like exponential smoothing, if implemented at all, Prophet will not be optimised for individual scenarios due to computational constraints. Therefore, we have to propose a suitable range of values that can be used by Slimstock in case Prophet ends up being implemented in Slim4.

To perform this optimisation, we can choose for either manual or automatic optimization. We have chosen to automatically optimise our model parameters as this can be done without supervision, meaning we can perform a greater number of experiments. This leaves us the options of grid search and random search. We have chosen to perform a random search as random search has been proven to be more efficient than grid search (Bergstra and Bengio, 2012). This is because we search a larger (albeit less promising) solution space.

As found in Research Question 5.3.2, Facebook suggests to restrict tuning of the model to just four parameters. These parameters, as well as their value ranges, can be found in Table 5.1 below.

We optimise Prophet based on its score for RMSE over our full dataset. To do so, we run 500 iterations of random search, using 4 years worth of data for initialisation, and 1 year for evaluation. We then calculate the RMSE of all of our individual SKUs, and validate whether or not the difference between models is statistically significant using a two-tailed t-test.

Table 5.1: Prophet Parameters

Parameter Name	Functionality	Suggested Range	Recommended
changepoint prior scale	Adjusts flexibility of the trend	[0.001, 0.5]	0.05
seasonality prior scale	Adjusts flexibility of the seasonality	[0.01, 10]	10
holidays prior scale	Adjusts flexibility of the holiday effects	[0.01, 10]	10
seasonality mode	Switches between additive and multiplicative seasonality effects	Additive/ Multiplicative	Additive

After performing 500 iterations of random search, we notice that the performance of our model is mostly dependent on a single parameter: seasonality mode. The additive model was significantly better in all iterations, no matter the configuration of the other parameters. Use of a two-tailed t-test with $\alpha = 0.05$ leads to the conclusion that the default value for seasonality mode, which is additive, yields significantly better performance than its alternative, being multiplicative seasonality. This holds true even when comparing the worst additive model to the best multiplicative model.

The difference in performance between the best and worst performing additive models is not statistically significant. This leads us to conclude that the effect of the other three parameters is very minor if at all present and thus we do not optimise these parameters any further. In our research, we use the default values as recommended by Facebook. These values are found in Table 5.1 above.

5.5 Modifications to Existing Models

In addition to researching Prophet, which is an alternative of exponential smoothing, we explore different alternatives in the form of modifications to the current models. In this section, we will elaborate on these modifications and answer our research question of:

- *Sub-question E: "How are we going to modify these models such that we can improve their performance?"*

We first provide a quick overview of all the starting points for forecasting models that will be part of our research. We also provide information on which SKU types for which these SKUs are suitable.

Table 5.2: Starting Forecasting Models

Model	SKU Types
Single Exponential Smoothing	Stationary SKUs
Double Exponential Smoothing	Trending SKUs
Triple Exponential Smoothing without Trend	Seasonal SKUs
Triple Exponential Smoothing with Trend	Trending Seasonal SKUs
Prophet	Any

We are going to modify two of these models, being the triple exponential smoothing models. We are going to modify these models in two different ways, which will be covered shortly. We will also explore models that use both of these modifications.

5.5.1 Triple Exponential Smoothing with Nested Seasonality

As covered in our literature studies in Chapter 3, the use of nested seasonalities is a natural extension to traditional exponential smoothing. Instead of using just monthly seasonalities, we are also going to classify SKUs based the presence of seasonality between weeks. In addition, we do the same for seasonality over the days within a week. Seasonality will only be extracted in case it is deemed significant by our test as we find by answering research question 5.A. This leads to fourteen additional variations of triple exponential smoothing. These fourteen variations come from the 7 possible combinations of seasonality and the presence/absence of a trend. As each of our seasonalities is either present or absent, we find $2^3 = 8$ combinations. One of these is the combination where all seasonality is absent, leaving 7 possible different levels of seasonality. Including/excluding the presence of a trend yields another 7 combinations.

5.5.2 Fourier Transforms

We hypothesise that seasonality over the year is not nearly as erratic as numerical analysis of limited historical data may make it seem to be. Under the assumption that most products smoothly transition between periods of increased demand and periods of decreased demand, we know that seasonality should always be either at a local extrema (i.e. $S_{t+1} < S_t$ and $S_{t+1} < S_{t+2}$ for a low seasonality, or $S_{t+1} > S_t$ and $S_{t+1} > S_{t+2}$ for high seasonality), or in a phase where it transitions (i.e. $S_t < S_{t+1} < S_{t+2}$ or $S_t > S_{t+1} > S_{t+2}$). In addition, local extrema should always be preceded and followed by a period of transition. Periods of transition may be preceded and followed by either a local extrema or another period of transition (if the latter, we consider these combined periods to be a single period). These requirements ensure seasonal transitions occur smoothly.

We illustrate our point in Figure 5.1 below. Observed demand is depicted by the blue line.

Looking at the blue line, we see demand of the product is approximately periodic in 6 months, meaning the seasonal pattern repeats approximately every 6 months, or twice a year. This means we should have two periods of high seasonality, two periods of low seasonality, and four periods of transition. More specifically, we expect the following pattern: Transition > Max > Transition > Min > Transition > Max > Transition > Min, to repeat.

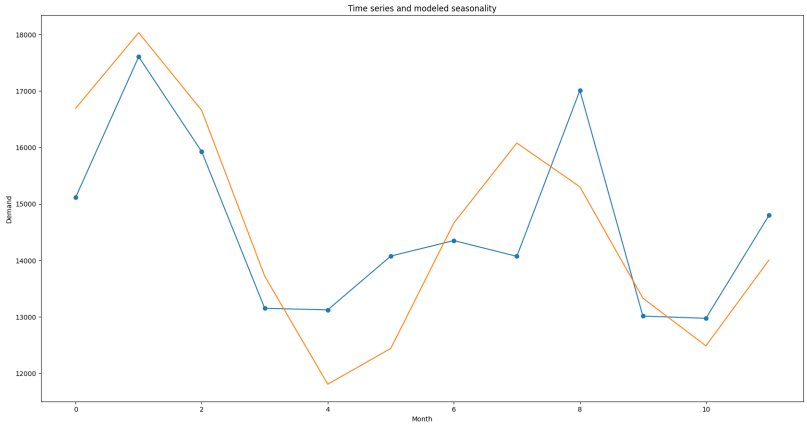

Figure 5.1: Observed Demand and Smoothened Seasonality

We now look at the blue line, and the constraints we proposed earlier. We see that the blue line follows the following pattern: Transition > Max > Transition > Transition > Min > Transition > Max > Min > Max > Transition > Min. This violates our hypothesis of smooth transitions of seasonality.

Multiplicative Fourier Seasonalities

As covered in Section 3.2.1 of Chapter 3, (Musbah and El-Hawary, 2019) were able to improve the performance of ARIMA models by extracting seasonal factors by means of a Fourier transform rather than using the traditional method. Prophet takes a similar approach for modeling seasonality (Taylor and Letham, 2018).

This leads us to a simple extension of the exponential smoothing models. After regular extraction of seasonalities as specified by Slim4, we apply a Fourier transform to the seasonal factors. This is done in an attempt to smoothen the transitions of seasonality. Full implementation is covered in G.5.

An example is provided in Figure 5.1 above. Applying a Fourier transform and some filtering and scaling to our previously covered blue line as described in Appendix E, we are able to extract a much smoother pattern for seasonalities.

In our research, we implement the Fourier Transform as described in Appendix G.5. We use a single Fourier transform for our monthly seasonal factors, combined with a low-pass frequency filter of cuttoff frequency 10. We use a cutoff frequency of 10 as (Taylor and Letham, 2018) have found this works well for the modeling of seasonality in Prophet.

5.5.3 Combination

A logical further extension is the combination of these models. As we expect the majority of noise to be present in the detection of monthly seasonal patterns, we apply the modifications on a monthly level only. We apply the same transformation as before.

5.5.4 Prophet

Prophet has a single advantage over our other methods, which comes in the form of holiday support. Prophet allows us to include the modeling of holidays, and thus we will also test the effect this has on our performance.

5.6 Conclusion

Our experiments are divided into parts, being the classification part and the forecasting part. During the classification part, we are going to evaluate the overall forecasting performance of four different classifiers. This is done by dividing the data set into four classes, all of which currently use a different exponential smoothing model. Division of the data set will be different for each classifier. We will select the classifier that results in the best division based on overall performance to be the best classifier.

In the second part of our experiments, we extend the classification to include daily and weekly seasonality. This results in sixteen classes of SKUs rather than the four classes we identified before. For each of these sixteen classes, we evaluate a variety of forecasting models. We evaluate the performance of our forecasting models based on accuracy, using a t-test to check whether or not the newly researched model is better than the current model.

6 Results

In this chapter, we provide the results of our experiments. We start off by an analysis of the current performance, after which we proceed to analysing the performance of our alternative classifiers. After selecting the best classification method, we introduce classification for multiple levels of seasonality, also known as nested seasonality.

In the following sections, we analyse the performance of nested triple exponential smoothing, and a slightly modified version of triple exponential smoothing that uses a different way to update the level parameter. Furthermore, we provide the results of our experiments on the use of a Fourier series for seasonality processing. We do this for both the current methods and for the nested triple exponential smoothing adaptation.

After performing our analysis on exponential smoothing methods, we proceed to the experiments of our alternative forecasting model: Prophet. We evaluate the performance of this model for our various types of SKUs over our different cover periods. Lastly, we will be looking at the difference in forecasting performance in case holidays are included or excluded in the model.

Note that our research uses 9419 time series of five years each. These are all time series that have been classified as "Frequent", "Normal", or "Lumpy" by the specifications of Slim4. We further classify these time series based on the presence of seasonality and trend.

Analyses in this chapter are as concise and objective as possible. Interpretation, conclusion, and discussion of the results are found in Chapter 7.

6.1 Determining a Baseline

We start off our experiments by evaluating performance of the current method. We do this, such that we can compare it to our alternative methods, and determine whether or not the alternative method is significantly better than the current method.

The current method consists of two phases, being classification and forecasting. The current classification method is as follows:

1. Classify whether or not the time series has a trend. This is done using linear regression. In our research, we use a significance level of $\alpha = 0.05$ to do this.
2. If the time series has a significant trend, detrend the time series.
3. Classify the presence of seasonality based on the F-test.

Now, if neither trend nor seasonality was found, use single exponential smoothing. If a significant trend is present but seasonality is not, use double exponential smoothing. If seasonality is present but a trend is not, use triple exponential smoothing without a trend. If both trend and seasonality are present, use traditional triple exponential smoothing.

We evaluate the performance of the current method over three different cover times, being short (3 days), medium (7 days) and long (21 days). We do this based on the 1 period ahead forecast. We will refer to these performances as our baseline performance. Results are provided in Table 6.1.

Table 6.1: Performance of the Current Method for Each of our Cover Times

SKU types	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias
Short (3 days)	0.188	74.8	0.14	-0.003
Medium (7 days)	0.282	50.87	0.214	-0.013
Long (21 days)	0.652	37.6	0.515	-0.029

6.2 Alternative Classification Methods

Slimstock currently use a combination of linear regression and a Fisher's F-test for classifying its SKUs using a trend-first approach. In our research, we explore three alternatives to this approach. These are a Fisher's F-test using a seasonality first approach, a Friedman test using a trend first approach, and a Friedman test using a seasonality first approach. Note that the seasonality first approach removes seasonality before trend regression regardless of significance. Significance levels are $\alpha = 0.05$ for both trend regression and seasonality classification. Results of the classification process are provided in Appendix M. Full performance figures of the classification can be found in Appendix 7. We provide the most important results in Table 6.2, 6.3, and 6.4.

Table 6.2: Performance of each of our Classifiers for Short Cover Times

Classifier	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias
F, Trend first	0.188	74.8	0.14	-0.003
F, Seasonality first	0.188	74.79	0.141	-0.005
Friedman, Trend first	0.2	75.97	0.15	0.028
Friedman, Seasonality first	0.201	75.99	0.151	0.028

Table 6.3: Performance of each of our Classifiers for Medium Cover Times

Classifier	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias
F, Trend first	0.282	50.87	0.214	-0.013
F, Seasonality first	0.283	50.85	0.216	-0.015
Friedman, Trend first	0.32	52.82	0.243	0.035
Friedman, Seasonality first	0.322	52.91	0.245	0.041

Table 6.4: Performance of each of our Classifiers for Long Cover Times

Classifier	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias
F, Trend first	0.652	37.6	0.515	-0.029
F, Seasonality first	0.656	37.57	0.518	-0.03
Friedman, Trend first	0.792	40.23	0.619	0.044
Friedman, Seasonality first	0.801	40.36	0.624	0.058

6.2.1 Conclusion on the Best Classification Method

Now that we have evaluated the model performances for each of our test and each of our cover periods, we can compare the results and find which classification method works best. We start off by comparing the classifiers for our short cover period, for which the results are provided in Table 6.2.

For our short cover period, we see that both variants of the F-test have very similar performance, being close enough for us to consider their performances to be equal. The same holds for the Friedman tests. We notice that none of the investigated alternatives perform significantly better than the current method, and thus we conclude the current method is optimal. Analysing Table 6.3, we come to the same conclusion for our medium cover times. From Table 6.4, we conclude that this also holds for our long cover times. To summarize, the F-test, trend first approach seems to be the best classifier in our research. We make the assumption the best test for monthly seasonality will also be the best test for other levels of seasonality, and thus we use this classification method for the rest of our experiments.

6.3 Nested Classification of SKUs

Slimstock currently only uses monthly seasonality for classification. A possible way of improving the quality of our forecasts is the use of nested seasonality as described in (Taylor, 2003b) and (Taylor, 2010). In this section, we reclassify our SKUs using nested seasonality. In the following sections, we conduct experiments using this new classification.

Currently, Slimstock classifies its SKUs based on the presence of trend and monthly seasonality only. In our research, we extend this method by including classification based on daily and weekly seasonality in addition to monthly seasonality and trend. In our classification, we use the F-test, trend first approach. Our new classification is provided in Table 6.5 below.

Table 6.5: Nested Classification of SKUs

SKU Class	SKUs in Class
Stationary SKUs: Nonseasonal SKUs without a trend	894
Daily SKUs without a trend	1 283
Weekly SKUs without a trend	24
Monthly SKUs without a trend	5
Daily-weekly SKUs without a trend	140
Daily-monthly SKUs without a trend	25
Weekly-monthly SKUs without a trend	17
Daily-weekly-monthly SKUs without a trend	131
Trending SKUs: Nonseasonal SKUs with a trend	3 071
Daily SKUs with a trend	3 101
Weekly SKUs with a trend	67
Monthly SKUs with a trend	15
Daily-weekly SKUs with a trend	294
Daily-monthly SKUs with a trend	40
Weekly-monthly SKUs with a trend	33
Daily-weekly-monthly SKUs with a trend	279

Analysing our newly made classification, we see the number of SKUs in each class varies a lot. Our biggest classes of SKUs are the daily SKUs without a trend, nonseasonal SKUs with a trend, and daily SKUs with a trend. We also note that we have a handful of classes that are very small.

Intuitively, the biggest gains can be made in the larger classes of SKUs. Improving performance of a lot of SKUs can lead to significant improvements across the board, even if the improvement is relatively small.

In this research, we compare different models for each of our classes separately. We use a t-test to compare the difference between two means. It is important to note that this test essentially tests for two criteria at the same time. Firstly, the difference between our two groups has to be sufficiently large. Secondly, the difference has to be consistently present over a large number of samples.

For our research, this means that it will be difficult to come to conclusions for our smaller classes. While we will oftentimes be fairly certain about which model performs best for classes that have a large number of SKUs, making this decision is much more difficult for our smaller classes. This leads us to a limitation related to the data used in our research, more on this later in Section 7.9.

In the following sections, we will go over each of our alternative models. We will only cover the most significant results in this chapter. Overall impact of the changes is found in Chapter 7. All

intermediate results for our experiments are reported in the corresponding appendices.

6.4 Effect of Inclusion of Nested Seasonality for Triple Exponential Smoothing

The first alternative forecasting model we explore is nested triple exponential smoothing. Full experiment results are found in Appendix O, where we also provide a brief analysis on each of our experiments. We now go over our most significant findings.

To begin, we note that our stationary SKUs, monthly seasonal SKUs without a trend, trending SKUs, and monthly seasonal SKUs with a trend have not had their models changed and are thus not affected. We can be fairly brief in analysing the results of our experiments on the other classes. Based on our experiments, we find that nested triple exponential smoothing is never better than the current method. In fact, the current method is always significantly better on at least one of our performance metrics, even for our smaller classes.

6.5 Effect of Inclusion of Nested Seasonality for Triple Exponential Smoothing: Modified version.

We note that the performance of nested triple exponential smoothing is extremely poor when compared to regular triple exponential smoothing in all cases.

Upon further investigation of the issue, we notice that triple exponential smoothing has a fundamental flaw that is very difficult to overcome. The problem lies in the updating of the level, which is done according to the following equation:

$$\hat{a}_t = \alpha D_t/(\hat{F}_{1,t-P1}) + (1-\alpha)(\hat{a}_{t-1}) \tag{6.1}$$

The updating of the level includes a division by the seasonality of the current period, which, if 0, is undefined. This causes our model to break. Slim4 currently solves this issue by implementing a minimum seasonal factor, such that this division is no longer undefined. Note that, using the current method (in which we only use monthly seasonality), this does not occur often in practice.

During our research, we conducted some experiments on a good value to use for our minimum seasonal factor. We ran various experiments with values ranging anywhere from 0.01 to 0.2. Deciding on a suitable value is a trade-off between sensitivity of the level update (where lower values will cause the level to update by a large amount, in fact, this update is often undesirably large for very low values) and creation of accurate forecasts, which prefers this minimum value to be as close to 0 as possible as higher values cause high values for our forecast even if seasonality of the current period is, in fact, 0. For the current method, we found a value of 0.01 to work fairly well.

Problems start to arise once we allow for the nesting of seasonality, where level is updated according to the following equation:

$$\hat{a}_t = \alpha D_t/(\hat{F}_{1,t-P1}\hat{F}_{2,t-P2}\hat{F}_{3,t-P3}) + (1-\alpha)(\hat{a}_{t-1}) \tag{6.2}$$

In case we have historically not seen demand on a given day, week, and month, we note that

all of our seasonal factors are, in fact, 0. As an example, we look at a situation where we have never had demand occur on January, week 1, on Saturdays. Using a minimum seasonality of 0.01, this leads to a seasonal factor of 0.000001 (1.0E-6) for this day. In case demand occurs on that day in our recent data, even though it does not occur in our historical data, this means the first part of our updating equation will evaluate to one million in case demand is a single unit. This ultimately means our level gets increased by 10 000 when we use a smoothing factor of $\alpha = 0.01$ (like we do in our research).

Although many of our SKUs will not have this problem, we note our dataset contains a handful of outliers which cause very poor overall performance. In an attempt to fix the fundamental flaws of triple exponential smoothing, we run experiments that exclude deseasonalisation in the updating of our level. The new updating equation is now as follows:

$$\hat{a}_t = \alpha D_t + (1 - \alpha)(\hat{a}_{t-1}) \tag{6.3}$$

Results of Experiments

After implementing this change, we run our new experiments. Results of these experiments are provided in Appendix P, where we again provide a brief analysis for each class. In this section, we go over our most significant findings. Classes that are not covered in this section were either negatively affected by the new model, or the difference in performance was not statistically significant.

Daily Seasonal SKUs without a trend
Analysing our daily seasonal SKUs without a trend, which is a class consisting of 1 283 SKUs. Analysing Table 6.6, we notice that for short cover times, the new model is significantly better based on RMSE and MAD. Performance based on Bias is significantly worse, but the difference is minimal at just 2.78%. RMSE of the new model is 11.48% better. MAD is 16.18% better.

Table 6.6: Effects of changing from SES to TES Nested Adjusted for Daily Seasonal SKUs for Short Cover Times

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	SES	0.183	77.54	0.136	**-0.036**
	TES Nested Adjusted	**0.162**	**74.2**	**0.114**	-0.037
p-value	-	**0.0**	0.07	**0.0**	**0.0**

Note that our medium and long cover times are not affected by the presence of daily seasonality as these cover times are integer multiples of 7, which is the number of days in a week.

Daily-weekly SKUs without a trend
The next class for which we find significant results is the class that covers Daily-weekly SKUs without a trend. This is a class consisting of 140 SKUs. Results of our experiments are provided in Table 6.7 below.

Table 6.7: Effects of changing from SES to TES Nested Adjusted

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	SES	0.204	72.11	0.157	**-0.065**
	TES Nested Adjusted	**0.19**	**69.63**	**0.136**	0.076
p-value	-	0.144	0.67	**0.006**	**0.0**
Medium	SES	**0.273**	**45.38**	**0.211**	-0.095
	TES Nested Adjusted	0.319	47.68	0.243	**0.007**
p-value	-	**0.009**	0.63	**0.022**	**0.0**
Long	SES	**0.606**	**31.42**	**0.486**	-0.111
	TES Nested Adjusted	0.723	33.15	0.566	**0.108**
p-value	-	**0.027**	0.65	0.056	**0.006**

We note that, for short cover times, the adjusted nested model sees improvements to MAD at the cost of bias. For medium and long cover times, single exponential smoothing is better based on our accuracy metrics.

Daily seasonal SKUs with a trend
The next class for which we find the new model to be significantly better is the class of daily seasonal SKUs with a present trend. This is a very large class, consisting our 3101 SKUs. Note that only short cover times are affected.

Table 6.8: Effects of changing from DES to TES Trending Nested Adjusted

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	DES	0.195	67.65	0.148	0.016
	TES Trending Nested Adjusted	**0.17**	**63.51**	**0.123**	**0.011**
p-value	-	**0.0**	**0.0**	**0.0**	**0.0**

Looking at Table 6.8, we see that the new model is significantly better based on all of our performance metrics. RMSE is improved by 12.82%. sMAPE is improved by 6.12%. MAD is improved by 16.89%. Bias is improved by 31.25%.

Daily-weekly seasonal SKUs with a trend
The class containing Daily-weekly seasonal SKUs is a medium sized class at just 294 SKUs. Results of our experiments are provided in Table 6.9 below.

Table 6.9: Effects of changing from DES to TES Trending Nested Adjusted

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	DES	0.219	61.78	0.17	**0.046**
	TES Trending Nested Adjusted	**0.213**	**58.91**	**0.155**	0.183
p-value	-	0.443	0.44	**0.022**	**0.0**
Medium	DES	**0.317**	**42.49**	**0.245**	**0.045**
	TES Trending Nested Adjusted	0.381	44.88	0.29	0.207
p-value	-	**0.0**	0.5	**0.0**	**0.0**
Long	DES	**0.766**	**34.68**	**0.616**	**0.043**
	TES Trending Nested Adjusted	0.941	36.88	0.736	0.234
p-value	-	**0.0**	0.51	**0.002**	**0.0**

As was the case for daily-weekly SKUs without a trend, we notice that the nested triple exponential smoothing model is better than the current model based on short term MAD. For medium and long cover periods, the current model remains the better model.

6.5.1 Conclusion on Nested Triple Exponential Smoothing

In this section, we analysed the impact of inclusion of nested seasonalities for triple exponential smoothing and the modifications made to the updating of our level parameter. Firstly, we note that adjustments made to the updating of the level parameter are necessary for nested triple exponential smoothing to work. Secondly, Looking at our most significant findings, we see that inclusion of daily seasonality yields significant short-term improvements over the current method. We also see some short term improvements to the models that include daily-weekly seasonality, though these improvements are a lot smaller than the improvements obtained by including daily seasonality alone. We hypothesise that inclusion of daily seasonality is beneficial, while inclusion of weekly seasonality is not. This hypothesis is confirmed when looking at the performance differences that occur when including just weekly seasonality. We therefore conclude that it is best to classify our SKUs based on monthly and daily seasonality rather than just monthly seasonality. This improvement is most significant for short cover times. For longer cover times that are not a multiple of 7 days, the improvement will be present, but to a much lesser extent.

6.6 Fourier Seasonalities

As covered in Section 5.5.2, we run experiments that use a Fourier transform to smoothen our monthly seasonality extraction. For now, this only affects SKUs that have a monthly seasonality present. Full experiment results are provided in Appendix Q. In this section, we analyse the classes of SKUs for which performance between the current model and the Fourier adaptation are statistically significant.

Daily-weekly-monthly SKUs without a trend
The first class of SKUs for which we find the difference in performance between the current model and the Fourier adapted model is significant is the class of daily-weekly-monthly SKUs without a trend. This class contains 131 SKUs.

Table 6.10: Effects of changing from TES to TES Fourier

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES	0.214	73.22	0.158	-0.117
	TES Fourier	**0.202**	**73.05**	**0.151**	**-0.045**
p-value	-	0.302	0.98	0.493	0.43
Medium	TES	0.34	50.45	0.248	-0.132
	TES Fourier	**0.304**	**50.44**	**0.227**	**-0.034**
p-value	-	0.072	1.0	0.153	0.348
Long	TES	0.863	**38.98**	0.639	-0.175
	TES Fourier	**0.73**	39.37	**0.56**	**-0.085**
p-value	-	**0.035**	0.94	0.094	**0.049**

Analysing the results as provided in Table 6.10 above, we find that the Fourier adaptation of the model is significantly better than the current model for long cover periods. RMSE of the adaptated model is 15.41% better. Bias is 51.43% better. Difference in sMAPE and MAD is not statistically significant.

For our medium cover times, we see that the p-value of our test on RMSE is just 0.072, which is very close to the threshold value of 0.05. We suspect the adaptation is also better for medium cover periods, but we need more data to back up this claim.

Daily-weekly-monthly SKUs with a trend
Daily-weekly-monthly SKUs with a trend is a class consisting of 279 SKUs. For this class, we also find statistically significant differences in performance between the current model and the Fourier adaptation.

Table 6.11: Effects of changing from TES Trending to TES Trending Fourier

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES Trending	0.22	**76.22**	0.157	**-0.052**
	TES Trending Fourier	**0.202**	77.1	**0.148**	0.177
p-value	-	**0.04**	0.8	0.201	**0.013**
Medium	TES Trending	0.388	**53.42**	0.274	**-0.077**
	TES Trending Fourier	**0.338**	55.39	**0.253**	0.21
p-value	-	**0.008**	0.55	0.101	0.106
Long	TES Trending	1.06	**42.91**	0.754	**-0.086**
	TES Trending Fourier	**0.878**	45.59	**0.678**	0.211
p-value	-	**0.003**	0.38	0.069	0.531

Analysing the results in Table 6.11, we again see the Fourier adaptation outperform the current model by a significant margin based on RMSE. For our short cover periods, this difference is 8.18%. For medium cover periods, the difference is 12.89%. Long cover periods see the greatest benefit at 17.17%.

6.6.1 Conclusion on Fourier Series for Seasonality

In this section, we analysed the difference in performance between the current model and the Fourier adaptation of the model. We find that daily-weekly-monthly seasonal SKUs, both with and without a trend, see a significant benefit to long term RMSE. Daily-weekly-monthly SKUs with a trend also see this improvement for short and medium cover periods.

Other affected SKU classes are monthly SKUs, daily-monthly SKUs, and weekly-monthly SKUs, all with and without a trend. These six classes of SKUs are all very small and thus we are not able to come to a significant conclusion in our research. Although we do not have sufficient data to back up a meaningful claim on these classes, we note that accuracy of the Fourier adaptation is better for five of these six classes. The exception to this is the class of monthly seasonal SKUs without a trend, is a class that consists of just 5 SKUs. This leads us to believe the Fourier adaptation of the model has a positive effect in most cases.

6.7 Nested Seasonality with Fourier Adjustments

In the previous sections, we explored both inclusion of nested seasonality as well as the proposed Fourier adaptation of the current implementation of triple exponential smoothing. We found both of these adjustments lead to improvements in some classes. Full results of our experiments, as well as in-depth analyses of experiments, are found in Appendix R. In this section, we analyse our most significant results.

After combining the modifications made to adjusted nested triple exponential smoothing and the application of a Fourier transform, we see that eight of our sixteen classes get a change to their

models. These are the eight classes that include monthly seasonality. Analysing our eight classes, we find that the combined model yields significantly better performance than the current model for two of our eight classes. These are the daily-weekly-monthly SKUs without a trend, and the daily-monthly SKUs with a trend.

Daily-weekly-monthly SKUs without a trend
The combined model shows a significant short-term improvement for daily-weekly-monthly SKUs without a trend, which is a class that consists of 131 SKUs.

Table 6.12: Effects of changing from TES to TES Nested Fourier

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES	0.214	73.22	0.158	-0.117
	TES Nested Fourier	**0.199**	**70.6**	**0.136**	**0.052**
p-value	-	0.188	0.65	**0.019**	0.314

We previously saw neither nesting nor Fourier filtering was able to significantly perform the performance of the model. Now, when combining these two methods, we actually start to reap the benefits. Short term MAD was significantly improved, making the nested TES Fourier model the better model for short cover periods. MAD gets improved by 13.92%. RMSE also sees an improvement of 7.01%, but this difference is not statistically significant.

Daily-monthly SKUs with a trend
Daily-monthly SKUs with a trend is a class that contains 40 SKUs. Despite containing just 40 SKUs, we found that the difference in performance between the two models was statistically significant for both short and long cover periods. Difference in performance of the two models was not statistically significant for medium cover periods.

Table 6.13: Effects of changing from TES Trending to TES Trending Nested Fourier

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES Trending	0.186	90.73	0.136	-0.076
	TES Trending Nested Fourier	**0.154**	**87.27**	**0.11**	**0.057**
p-value	-	0.055	0.71	**0.036**	0.871
Long	TES Trending	0.764	40.81	0.543	**-0.069**
	TES Trending Nested Fourier	**0.531**	**35.55**	**0.409**	0.117
p-value	-	**0.038**	0.3	**0.042**	0.524

Like found when analysing the performance of this model on daily-weekly-monthly seasonal SKUs without a trend, we find that our combined model produces significantly better results than the current model, even though neither adaptation of the model was significantly better by itself. For short cover periods, this is solely based on MAD. For long cover periods, this conclusion is based on RMSE and MAD. Short cover times see an improvement of 17.20% to RMSE, though this difference was not statistically significant. MAD sees a significant improvement of 19.12%. For long cover times, RMSE and MAD see significant improvements of 30.50% and 24.68% respectively.

6.7.1 Conclusion on Nested Seasonality with Fourier Adjustments

Analysing the results of our experiments on the combined model, we find that, when combined to the current model, the combined model shows a significant improvement in short-term forecasts

for daily-weekly-monthly SKUs without a trend. The combined model also shows improvements for both short and long cover periods for daily-monthly SKUs with a trend. This is interesting, as neither of our two adaptations was able to significantly improve the performance by itself. Based on our experiments, we conclude that combining nesting and the Fourier adaptation of the model is beneficial to performance for short cover time forecasts for daily-weekly-monthly SKUs without a trend, and for short and long cover period forecasts for daily-monthly SKUs with a trend.

6.8 Alternative Forecasting Method: Prophet

In addition to exponential smoothing models and covered adaptations to these models, we explore the Prophet forecasting model. As covered in Chapter 3.2.2 is a regression-based forecasting model. We run experiments on the performance of Prophet compared to performance of the models currently used by Slimstock. Full experiment results as well as analyses are provided in Appendix S.

We start off this section by mentioning Prophet performs extremely well for most of our SKU classes, where performance is either significantly better or not significantly worse for fourteen of our sixteen classes. For ten of those fourteen classes, Prophet performs significantly better based on accuracy for at least one of our three cover periods. In this section, we provide an analysis on our four largest SKU classes, being stationary SKUs, trending SKUs, and daily SKUs with and without a trend. This includes the two classes for which Prophet is significantly worse than the current model, being stationary SKUs and daily seasonal SKUs without a trend.

Stationary SKUs
Starting off with our first class of SKUs, being stationary SKUs. Being a class containing 894 SKUs, we have quite a lot of data to back up our claim.

Table 6.14: Effects of changing from SES to Prophet

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	SES	0.158	94.6	0.112	**-0.017**
	Prophet	**0.151**	**94.1**	**0.108**	-0.12
p-value	-	0.096	0.83	0.151	**0.0**
Medium	SES	**0.241**	**66.03**	**0.18**	**-0.011**
	Prophet	0.25	68.13	0.19	-0.151
p-value	-	0.166	0.32	**0.033**	**0.0**
Long	SES	**0.519**	**44.9**	**0.411**	**-0.011**
	Prophet	0.564	47.7	0.449	-0.209
p-value	-	**0.008**	0.1	**0.006**	**0.0**

Analysing the results in Table 6.14 above, we find that Prophet has better short-term performance based on accuracy, but this difference is not statistically significant. For medium and long cover periods, SES is superior to Prophet for this class of SKUs.

Daily SKUs without a trend
The second class of SKUs we cover is the class of daily SKUs without a trend. This class is a fairly large class, containing 1 283 SKUs.

Table 6.15: Effects of changing from SES to Prophet

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	SES	0.183	77.54	0.136	**-0.036**
	Prophet	**0.166**	**74.93**	**0.12**	-0.111
p-value	-	**0.0**	0.16	**0.0**	**0.0**
Medium	SES	**0.26**	**49.33**	**0.195**	**-0.045**
	Prophet	0.27	50.55	0.206	-0.147
p-value	-	0.082	0.44	**0.005**	**0.0**
Long	SES	**0.542**	**33.2**	**0.423**	**-0.056**
	Prophet	0.586	34.89	0.461	-0.194
p-value	-	**0.002**	0.2	**0.0**	**0.0**

Analysing Table 6.15, we find that Prophet significantly outperforms the current method for short cover times based on RMSE and MAD. For medium and long cover times, we find the opposite holds true, with the current method outperforming Prophet. We conclude Prophet is best for short cover times, and SES is better for medium and long cover times.

Trending SKUs without seasonality
The next class we cover is the class of trending SKUs without seasonality. At 3 071 items, this class makes up almost a third of our total.

Table 6.16: Effects of changing from DES to Prophet

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	DES	0.183	75.14	0.137	**0.006**
	Prophet	**0.167**	**72.26**	**0.123**	-0.169
p-value	-	**0.0**	**0.02**	**0.0**	**0.0**
Medium	DES	0.283	52.58	0.217	**-0.008**
	Prophet	**0.281**	**52.44**	**0.217**	-0.213
p-value	-	0.703	0.9	0.868	**0.0**
Long	DES	**0.671**	39.92	**0.538**	**-0.031**
	Prophet	0.675	**39.81**	0.538	-0.273
p-value	-	0.716	0.91	0.992	**0.0**

Analysing Table 6.16 above, we find that Prophet significantly outperforms the current method for short cover times based on accuracy. Difference in performance for medium and long forecasting horizons is not statistically significant.

Daily SKUs with a trend
The last class we cover is our largest class of SKUs. At 3 101 items, daily SKUs with a trend make up almost a third of our total.

Table 6.17: Effects of changing from DES to Prophet

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	DES	0.195	67.65	0.148	**0.016**
	Prophet	**0.172**	**63.69**	**0.126**	-0.141
p-value	-	**0.0**	**0.0**	**0.0**	**0.0**
Medium	DES	**0.283**	**45.21**	**0.217**	**0.006**
	Prophet	0.285	45.25	0.219	-0.177
p-value	-	0.514	0.97	0.354	**0.0**
Long	DES	**0.652**	34.18	**0.519**	**-0.013**
	Prophet	0.665	**34.11**	0.524	-0.22
p-value	-	0.185	0.94	0.507	**0.0**

For this class of SKUs, we again see Prophet outperform the current method for our short cover time. Difference in performance for medium and long cover times is statistically insignificant.

6.8.1 Conclusion on Performance of Prophet Compared to the Current Method

To summarise, we find that Prophet is a very good method for creating short-term forecasts, significantly outperforming the current method for our three largest SKU classes. Medium and long term forecasting performance of the model varies across the board. An analysis for each of our individual SKU classes is found in Appendix S.

6.9 Effects of Holidays on performance of Prophet

Another advantage Prophet has over the other methods is the fact that Prophet has access to information on holidays. In this section, we research the difference in model performances when holidays are excluded and when they are included. Note that we only use Dutch holidays as the data used in our research have been provided by a Dutch company. In this experiment, we do not perform any form of classification. Instead, we use all of our data to evaluate the performance of our two different models (being Prophet with holidays and Prophet without holidays). Results of our experiments are provided in Table T.1, T.2, and T.3 of Appendix T.

Analysing our results, we find that our data does not provide enough evidence to conclude there is a significant difference between the performance of the two models. This holds for all three of our cover periods. We select Prophet without holidays as our best model as it has slightly (though not significantly) better bias than its holiday-adjusted counterpart. In addition, it is also the more generalisable model as it is not bound to a single country.

7 Findings, Conclusion, and Discussion

In this chapter, we finalise our research. We start off by discussing our findings. We look at our best found classification and forecasting methods, their associated performances, and the advantages and disadvantages of these methods. After discussing our findings, we discuss implications and recommendations of our research for both Slimstock and our client.

After discussing our findings and the implications and recommendations for our stakeholders, we compare our research to prior research. Next, we look at the limitations of our research. We also discuss scientific relevance. We wrap up our research by providing our recommendations for further research.

7.1 Findings

We start off by answering research question 5, discussing our findings. Reciting:

- *Research question 5: "What are our findings?"*

The findings in our research are twofold, consisting of findings on classification of time series and findings on time series forecasting. We first cover classification.

7.1.1 Best Classification Method

In the first step of our research, we investigated several classification methods. This allows us to answer the first sub-question:

- *Sub-question A: "What is the best classification method?*

This question was covered in Section 6.2.1. In our experiments, we find the F-test, trend first approach yields the best performance. Performance for the F-test, seasonality first approach is nearly identical. As the trend-first approach requires less computation, we consider it to be better than the seasonality first approach. We find that using a Friedman test instead of a more traditional F-test does not result in better performance. Experiments were executed on a monthly aggregated level. We later assume that the F-test, which works best for this level of aggregation, can be applied to other levels of seasonality without loss of generality.

Making this assumption allows us to classify our SKUs into more specific classes. In addition to monthly seasonality, we can now identify weekly and daily seasonality. These two new levels of seasonality allow use to further tailor the forecasting models to our needs. In our later experiments, we found that including the presence of weekly seasonality in our forecasting models does not result in better performance. Including daily seasonality results in significant improvements for SKUs that have structural fluctuations in demand over the week. Note that classification on a weekly seasonality not being beneficial is a retrospective finding. The remainder of our experiments therefore do include the possibility of weekly seasonality.

To conclude, we can best classify seasonality in our time series on a monthly and daily level. To do so, we use a Fisher's F-test as described in Appendix A. Classification of trend is best done using linear regression.

7.1.2 Best Forecasting Models

Although the use of a single forecasting model would reduce the need for classification, we find that a combination of different forecasting models for different types of SKUs yields better forecasting performance than the use of a single method. Now that we have found our best classification

method, we can select the best forecasting models for our different SKU classes under our two different ideologies. By doing so, we answer our second sub-question, which is as follows:

- *Sub-question B: "What is the best forecasting model for each of our SKU types?"*

Our research identifies sixteen classes of SKUs. In this section, we use a shorthand notation to describe these classes. This notation is as follows: (T: (y/n), D: (y/n), W: (y/n), M: (y/n)), where T is trend, D denotes daily seasonality, W is for weekly, and M is for monthly. y is for yes (inclusion), and n is for no (exclusion). We select our best models as described in section 4.2.1. Model selection is done independently for our three different forecasting horizons. We look at the most significant improvements in this section. For a full analysis and results, we refer to Appendix U.

Best Models for Short Cover Times

For short cover times, we see a lot of changes for both a progressive and a conservative ideology. Under a progressive ideology, we find that all of our SKU classes benefit from a change to Prophet. Under a conservative ideology, we see nine of our sixteen classes have their models switch to Prophet. Overall improvements based on RMSE are about 12% for the progressive ideology and 11% for a conservative ideology.

Looking at our four largest SKU classes in Table 7.1 below, we see some very interesting things. Firstly, the class of stationary SKUs sees an improvement of 4.43% based on RMSE, but this change is not statistically significant. Classes two and nine see an improvement of around 9%. For class 10, which is the class of daily seasonal SKUs with a trend, Prophet is very good, but it is not actually the best option. For this class, we see an improvement of a little under 12% when using Prophet and a little under 13% when using the level-updating-adjusted variant of nested triple exponential smoothing.

Best Models for Medium Cover Times

For medium cover periods, we see that improvements are of a much smaller magnitude. Whereas we previously saw improvements of around 10%, improvements for medium cover times are approximately 2.5% for a progressive ideology and approximately 1.8% for a conservative ideology.

Again looking at our four largest SKU classes in Table 7.2 below, we see that classes 1, 2, and 9, being stationaries and daily seasonal items with an without a trend, do not benefit from a model change at all and thus these classes are not improved. Trending SKUs without seasonality see a small improvement from changing to Prophet, but the difference is not statistically significant. Performance gains for this class actually originate from our smaller SKU classes switching to Prophet as shown in Appendix U.

Best Models for Long Cover Times

As seen in Table 7.3, none of our four largest classes benefit from a model change for long cover times. Despite the fact that our largest four classes do not see any benefit, switching from the current model to Prophet does yield improved performance. This performance improvement is around 3.4% under a progressive ideology and around 3.1% under a conservative ideology. Changes originate from our smaller classes of SKUs as shown in Appendix U.

Conclusion on the Best Models

To conclude, we find that the best way to forecast our different SKUs is to use a combination of currently used methods and Prophet, with Prophet being more prevalent when forecasting on a

shorter horizon. The biggest performance gains are made on a short forecasting horizon, though longer cover periods also benefit from the new models to some extent.

There is one exception to this, being daily seasonal SKUs with a trend on a short cover period. These SKUs are best forecasted using the adjusted version of nested triple exponential smoothing. As this is the only situation in which a not yet implemented model is the best model, and using Prophet instead of exponential smoothing yields very similar performance, we conclude that using Prophet here, despite not being optimal, is also a good option.

Table 7.1: Short Cover Periods: Changes for our Largest Classes

SKU types	Current Model	Alternative Model	Number of SKUs	RMSE Improvement (%)
1: (T: n, D: n, W: n, M: n)	SES	SES ——— Prophet	894	0 ——— 4.43
2: (T: n, D: y, W: n, M: n)	SES	Prophet	1 283	9.29
9: (T: y, D: n, W: n, M: n)	DES	Prophet	3 071	8.74
10: (T: y, D: y, W: n, M: n)	DES	Prophet ——— Trending TES Nested Adjusted	3 101	11.79 ——— 12.82

Table 7.2: Medium Cover Periods: Changes for our Largest Classes

SKU types	Current Model	Alternative Model	Number of SKUs	RMSE Improvement (%)
1: (T: n, D: n, W: n, M: n)	SES	SES	894	0.00
2: (T: n, D: y, W: n, M: n)	SES	SES	1 283	0.00
9: (T: y, D: n, W: n, M: n)	DES	DES ——— Prophet	3 071	0 ——— 0.71
10: (T: y, D: y, W: n, M: n)	DES	DES	3 101	0.00

Table 7.3: Long Cover Periods: Changes for our Largest Classes

SKU types	Current Model	Alternative Model	Number of SKUs	RMSE Improvement (%)
1: (T: n, D: n, W: n, M: n)	SES	SES	894	0.00
2: (T: n, D: y, W: n, M: n)	SES	SES	1 283	0.00
9: (T: y, D: n, W: n, M: n)	DES	DES	3 071	0.00
10: (T: y, D: y, W: n, M: n)	DES	DES	3 101	0.00

7.2 Performance of the New Models

Now that we have identified the best classification method and forecasting models, we take a more detailed look at our overall performance gains. We answer our third sub-question.

- *Sub-question C: "What is the performance of the new method?"*

As covered in the previous question and Appendix U, the new method is most beneficial for short cover periods, where we the new method is better than the current method by a little over 10% based on RMSE. We analyse the effects on overall performance in Table 7.4 below. Looking at Table 7.4, we see that accuracy is improved at the cost of bias. Whereas Bias of the current method is very close to zero, we see that the Bias of the new models is slightly negative. This means that the new models structurally over-forecast by a slight margin.

Table 7.4: Performance of the Different Approaches for Short Cover Times

Method	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias
Current Approach	0.188	74.8	0.14	-0.003
New Approach: Conservative	0.167	71.58	0.122	-0.082
Performance Gain (+) / Loss (-)	+ 11.17%	+ 4.30%	+12.86%	- 2633.33%
New Approach: Progressive	0.166	71.49	0.121	-0.094
Performance Gain (+) / Loss (-)	+ 11.70%	+ 4.43%	+ 13.57%	- 3000.33%

When moving on to the medium cover times (Table 7.5), we see that although performance gains found here are still significant and very welcome, these gains are nowhere as large when compared

to short-term gains found earlier. Despite gains not being anywhere as large as they were before, we are fairly certain they are present based on our statistical t-tests as covered in Chapter 6. When looking at bias we again find that although the percentage difference is certainly large, the absolute difference is fairly minor.

Table 7.5: Performance of the Different Approaches for Medium Cover Times

Method	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias
Current Approach	0.282	50.87	0.214	-0.013
New Approach: Conservative	0.277	50.72	0.211	-0.02
Performance Gain (+) / Loss (-)	+ 1.78%	+ 0.29%	+ 1.40%	- 53.85%
New Approach: Progressive	0.275	50.59	0.210	-0.09
Performance Gain (+) / Loss (-)	+ 2.48%	+ 0.55%	+ 1.87%	- 592.31%

Looking at our long cover times (Table 7.6), we again see that performance of the new model is significantly better than the old model. With gains in RMSE being a little over three percent, we think our new method provides our stakeholders with a relatively small though very welcome improvement. Looking at bias, we find that bias is worse than it was before, though absolute difference of bias is fairly small.

Table 7.6: Performance of the Different Approaches for Long Cover Times

Method	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias
Current Approach	0.652	37.6	0.515	-0.029
New Approach: Conservative	0.632	37.35	0.49	-0.031
Performance Gain (+) / Loss (-)	+ 3.07%	+ 0.66%	+ 4.85%	- 6.90%
New Approach: Progressive	0.630	37.29	0.500	-0.038
Performance Gain (+) / Loss (-)	+ 3.37%	+ 0.82%	2.91%	- 31.03%

Summarising, we see that the new method improves short term performance by approximately 11-12% and medium to long term performance by approximately 2-3%. Differences in performance of the progressive and conservative approach are very small.

7.3 Advantages and Disadvantages of the Newly Found Models

Now that we know more about the effects of adopting a new model, we find ourselves asking the very difficult question of: Is this model worth implementing?

To answer this question, merely looking at numbers does not suffice. Even though a method may be better than another method based on numerical analysis, we always need to look at the bigger picture. To do so, we are going to cover the advantages and disadvantages of our new method compared to the current method. This answers the next sub-question of:

- *Sub-question D: "What are the advantages and disadvantages of newly found models?"*

Firstly, the advantage of the new models. The main advantage of the new models is significantly improved performance based on accuracy. This allows for a reduction in safety stock and, in turn, safety stock costs.

Then, the disadvantages of the models. We identify two disadvantages that come with our new models.

Firstly, maintenance and development. With the new method adding both a new classification level and a new forecasting algorithm, the system has become more complex and thus more difficult to maintain and develop.

Secondly, computation. The new method is more computationally complex than the method that is currently used in Slim4. This means that forecasting will take longer to complete, and customers may need to invest in better hardware to make up for this.

7.4 Recommendations for our Client

Having analysed performance, advantages, and disadvantages of the new method, we can proceed to the recommendations for our client, answering the following sub-question:

- *Sub-question E: "Based on performance, advantages, and disadvantages, what can we recommend to our client?"*

With our client being a customer of Slimstock, this research provides very little direct results until Slimstock finishes implementation of the models. Despite this, we can provide them with one single recommendation, which is related to the presence of daily seasonality.

We notice that our client's portfolio contains many SKUs that have daily seasonality present. This means they are heavily exposed to the effects of daily seasonality, which is not currently supported by Slim4. We recommend our client to incorporate processing of daily seasonality as soon as Slimstock adds this functionality.

7.5 Recommendations for Slimstock

With Slimstock as an intermediary, we provide more recommendations to them than to our client. We answer our penultimate sub-question:

- *Sub-question F: "Similarly, what are our recommendations to Slimstock?"*

Firstly, we recommend to include seasonality classification on a daily level. With 4 384 of our client's 9 419 SKUs having daily seasonality, this is a distinction worth making. This is most beneficial on a short forecasting horizon. Contrarily to daily seasonality, inclusion of weekly

seasonality does not seem to be a great contributor to forecasting performance. None of the models that include weekly seasonality do any better than their counterparts and thus we do not recommend them.

Secondly, Prophet appears to be a much more accurate forecasting algorithm than exponential smoothing for many classes of SKUs. We recommend Slimstock to include it in their software.

Modifying the current software to use Prophet instead of exponential smoothing requires a lot of time and effort, making the transition more of a long-term plan. In our research, we researched modeling of seasonality using a Fourier transform, which is similar to the way Prophet models its seasonality. Our experiments (Appendix Q) show that inclusion of a Fourier transform allows us to significantly improve forecasting performance with very little effort. We therefore recommend Slimstock to include this modification before fully transitioning to our new models.

7.6 Conclusion

Now that all of our experiments are done and results have been thoroughly analysed, we can answer our main research question, which is as follows:

- *Main Question: "How can we better detect patterns in our time series and improve our forecasts?"*

Based on our experiments and analysis of results, we conclude the quality of our forecasts can be improved by making two changes. Firstly, adding support for daily seasonality. By including daily seasonality, we can model and predict demand fluctuations on an intraweek level, which in turn vastly improves the quality of our short-term forecasts. Secondly, changing some of our forecasting algorithms. Although exponential smoothing remains a good method for forecasting on longer time horizons, recent developments in the field of time series forecasting have provided us with more modern algorithms such as Prophet. Our research shows Prophet is very good at forecasting all types of SKU on a short horizon. In addition, Prophet is very good at forecasting complex time series on medium and long horizons, outperforming exponential smoothing methods in some of our experiments. For simpler time series on longer time horizons, however, exponential smoothing remains the best method.

7.7 Discussion

This leaves us with just the discussion. Some of the results of our experiments contradict our expectations based on prior research on the same topics. We start by investigating why this is the case, analysing the differences between our research and the relevant studies in detail.

7.8 Comparison to Prior Research

During our research, we investigated the effects of including different levels seasonality for triple exponential smoothing. We investigated three levels of seasonality, being monthly, weekly, and daily. We also investigate combining multiple levels of seasonality.

Combination of multiple levels of seasonality is commonly referred to as nested seasonality. During our literature studies, we found two papers on nested seasonality for triple exponential smoothing. The first paper: "Short-term electricity demand forecasting using double seasonal exponential smoothing.", covers forecasting using two levels of seasonality (Taylor, 2003b). The second paper we used is called "Triple seasonal methods for short-term electricity demand forecasting.". This paper covers forecasting using three levels of seasonality (Taylor, 2010).

In both of these papers, including multiple levels of seasonality positively affected forecasting performance. In our research, we found that neither double nor triple seasonal methods were able to outperform the current method. In fact, using multiple levels of seasonality oftentimes caused more harm than good. Despite both double and triple seasonal exponential smoothing having their place at times, we note that there are some fundamental differences between demand forecasting for supply chains as in our research, and demand forecasting for electricity as in (Taylor, 2003b) and (Taylor, 2010).

Firstly, (Taylor, 2003b) and (Taylor, 2010) research forecasting on a much shorter term, with their forecasting period being anywhere from half an hour to a day ahead instead of the multiple day and / or multiple week forecasts that are required for demand forecasting in supply chains.

Secondly, (Taylor, 2003b) and (Taylor, 2010) make use of different levels of seasonality. Instead of daily, weekly, and monthly seasonality, (Taylor, 2010) makes use of intraday (hourly), intraweek (daily), and intrayear (weekly) levels. Although we do not recommend the use of weekly seasonality for demand forecasting in supply chains, we can see why it does work in other fields.

The main reason we expect the use of weekly seasonality in demand forecasting has a negative impact whilst it has a positive impact in short-term electricity forecasting is mainly related to human behaviour. We find energy consumption is naturally linked to the levels of seasonality as used in (Taylor, 2010). With society generally consuming more energy in winter than we do in summer, during weekdays, and during waking hours of the day, this makes a lot of sense.

For demand forecasting, however, the use of weekly seasonality introduces a problem, being misalignment of impact caused by social constructs. Whereas energy consumption is mostly linked to natural forces, which are generally fairly smooth over time, human behaviour can be fairly dynamic. Some examples of this are : Christmas, which does not always occur in the same week, holidays, which start and end at a different time of year each year, and external events, which can influence our short term buying behaviour way more than it influences energy consumption. It would be possible to extend the triple exponential smoothing models to include additional seasonal factors for special events, but this is beyond the scope of our research.

The last problem that we identify emerges when looking at demand forecasting for supply chains instead of forecasting for energy consumption has to do with deseasonalisation. For energy consumption, we note that seasonality is never zero. When we look at the following equation:

$$\hat{a}_t = \alpha D_t / (\hat{S}_{1,t-P1} S_{2,t-P2} S_{3,t-P3}) + (1-\alpha)(\hat{a}_{t-1}) \tag{7.1}$$

This means that $\alpha D_t / (\hat{S}_{1,t-P1} S_{2,t-P2} S_{3,t-P3})$ is never undefined. Not only is it never undefined, we also note that, in energy forecasting, none of our seasonal factors ever get remotely close to zero. This means that, in addition to never being undefined, this part of the equation will never blow up like we had occur during our research.

7.9 Limitations

Inventory management and forecasting are both extremely complex topics, and objectively researching these topics is extremely difficult if at all possible. Therefore, we have to make some simplifications and assumptions, which leads to our research having a couple limitations. In addition, external effects have introduced some limitations to our research that we have not been able to overcome.

Firstly, as our research focuses on improving detection and forecasting of seasonal SKUs, we have limited ourselves to items that are sold throughout most of the year as specified by Slim4. These are the SKUs that have a demand occur in most months of the year. This means our conclusions do not necessarily apply to SKUs that have mostly intermediate demand or demand due to one-off events.

Secondly, our research focuses on products that are in the maturity phase of their product life cycles. This decision has been made as we limited the scope of our research to numerical methods, which are applicable under the assumption that history repeats itself to some extent. Limiting our research to SKUs that are in the maturity phase of their product life cycles ensures their demand patterns have had time to stabilise as much as possible.

Thirdly, we have analysed all of our SKUs over three different cover periods, being short (3 days), medium (7 days) and long (21 days). Although we would ideally want to analyse SKUs depending on their forecasting frequency and timing (i.e. grouping together all trending daily-monthly seasonal SKUs that are reviewed every Monday and Thursday), this would lead to a major problem. Introducing so many different groups will mean each group is extremely small. This means that we will not have sufficient samples to perform our statistical analysis. Adding onto this, cover periods are oftentimes not static. Depending on the forecasting frequency and timing of a single SKU, there may be multiple cover times present. An SKU that is reviewed on Mondays and Thursdays has review intervals of 3 and 4 days respectively. Even when assuming that lead time is static at a single week, this SKU will have two different cover times. As we did not have access to the review timings of each SKU during our research, this simplification is compulsory, and even if we did, this simplification is likely to have still been required due to reasons mentioned before.

Fourthly, data used in our research have been collected between May 1st of 2018 and May 10Th of 2023. During this period, COVID-19 has had a very large effect on the supply chains and demands of many companies, and we have very good reason to believe this has also been the case for Company X, whose data was used to conduct this analysis. With demand of many products from many companies all over the world being highly affected, analysis of time series has become very difficult. For Company X, we estimate that COVID-19 has had a negative influence on demand, especially during 2020 and the beginning of 2021 when measures such as working from home, quarantine, bans on large gatherings, and curfews were prevalent. As these measures were later lifted and / or loosened, this means models that assign more weight to more recent observations (such as exponential smoothing models that use very high smoothing factors) are favoured. This is also one of the main reasons our research does not entail extensive research on the optimisation of smoothing factors for exponential smoothing. The other reason we do not optimise our smoothing factors to fit our data is because this is also not done by Slimstock per specifications of Slim4.

Lastly, our research makes an assumption regarding classification. In finding the best classification method, we only investigated performance of classification on a monthly level of aggregation. We make the assumption that the found method is also best for identifying the presence of seasonality on different levels of aggregation. This assumption had to be made to reduce the required number of experiments for our research.

7.10 Scientific Contribution

This research contributes to the scientific database in multiple ways. Firstly, our research proposes an improvement to triple exponential smoothing by means of a Fourier transform. Although not always significant, we saw the application of a Fourier transform often had a positive influence on

model accuracy.

Secondly, our research highlights the fundamental flaws of exponential smoothing for demand forecasting in supply chains. Although the use of double and triple nested triple exponential smoothing has a positive impact on forecasting electricity consumption, which is the field researched in the papers that cover the nesting of seasonality, our research has shown this is not the case for demand forecasting in supply chains. We identified two reasons for this, being the problems related to deseasonalisation used for updating the level of our time series, and the problem of misalignment of weekly seasonalities. In addition to highlighting these issues, our research has provided a more effective approach to the first of these two problems.

7.11 Recommendation for Further Research

During both our literature research as well as during our experimentation we found several interesting topics that may be worth investigating in the future. We identify four research directions to be particularly interesting.

Firstly, as mentioned during our literature review in Chapter 3, machine learning forecasting methods such as gradient boosters have been able to outperform numerical methods in the M5 competition. Despite this, we have not researched them extensively for computational reasons as well as data-related reasons. Machine learning algorithms have an advantage over time series forecasting algorithms we used in our in the sense that they allow for input of more features such as sales price. Slimstock does not currently collect these data, but starting collection of this data may mean machine learning methods are worth revisiting.

Secondly, updating of Prophet rather than refitting. With Prophet showing significant improvements to accuracy of our forecasts, we recommend implementing it despite its high computational cost. As mentioned in Appendix I, Prophet does have a way of reducing this computational cost by eliminating the need to refit the model every time we need to make a forecast. Instead, the model can be updated by means of a warm start. As mentioned in documentation (Facebook, 2023), this may yield sub-optimal results, but we are uncertain of the extent to which this occurs. Therefore, we recommend investigation of the difference between refitting and the use of warm starts for this model.

Thirdly, we have a recommendation that is related to our fourth limitation. Whereas traditional models such as exponential smoothing have no way to circumvent the influence of external events such as COVID-19, Prophet actually allows us to account for events that we do not expect to repeat in the near future. This can be done by modeling the periods of lockdown and other measures as a one-off holiday. We are interested in seeing the effect this has on our model performance and consider this to be suitable topic for further research.

Fourthly, significance of both trend and seasonal classification. Our research uses a significance level of $\alpha = 0.05$ for both trend detection as well as seasonal classification. Although $\alpha = 0.05$ is fairly standard, it may not be optimal. Investigation of different significance levels may lead to better classification of our time series.

References

Abbasimehr, H., Shabani, M., and Yousefi, M. (2020). An optimized model using lstm network for demand forecasting. *Computers & industrial engineering*, 143:106435.

Abraham, B. and Ledolter, J. (1986). Forecast functions implied by autoregressive integrated moving average models and other related forecast procedures. *International Statistical Review/Revue Internationale de Statistique*, pages 51–66.

Aditya Satrio, C. B., Darmawan, W., Nadia, B. U., and Hanafiah, N. (2021). Time series analysis and forecasting of coronavirus disease in indonesia using arima model and prophet. volume 179, page 524 – 532. Cited by: 56; All Open Access, Bronze Open Access.

Al Daoud, E. (2019). Comparison between xgboost, lightgbm and catboost using a home credit dataset. *International Journal of Computer and Information Engineering*, 13(1):6–10.

Alemu, A. B., Parakash Raju, U. J., Seid, A. M., and Damtie, B. (2023). Comparative study of seasonal autoregressive integrated moving average and holt–winters modeling for forecasting monthly ground-level ozone. *AIP Advances*, 13(3):035303.

ARSHAM, H. (1985). Seasonal and cyclic forecasting for the small. *American Journal of Small Business*, 9(4):47.

Bentéjac, C., Csörgő, A., and Martínez-Muñoz, G. (2021). A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*, 54:1937–1967.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2).

Brockwell, P. and Davis, R. (2009). *Time Series: Theory and Methods*. Springer Series in Statistics. Springer New York.

Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5):1190–1208.

Chatfield, C. and Yar, M. (1988). Holt-winters forecasting: some practical issues. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 37(2):129–140.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Dabral, P. and Murry, M. Z. (2017). Modelling and forecasting of rainfall time series using sarima. *Environmental Processes*, 4(2):399–419.

De Gooijer, J. G. and Hyndman, R. J. (2006). 25 years of time series forecasting. *International Journal of Forecasting*, 22(3):443 – 473. Cited by: 1007; All Open Access, Green Open Access.

Dorogush, A. V., Ershov, V., and Gulin, A. (2018). Catboost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*.

Edwards, J. (1961). The recognition and estimation of cyclic trends. *Annals of human genetics*, 25(1):83–87.

Facebook (2023). *Facebook documentation: Prophet*.

Ferov, M. and Modrỳ, M. (2016). Enhancing lambdamart using oblivious trees. *arXiv preprint arXiv:1609.05610*.

Freedman, L. (1979). The use of a kolmogorov–smirnov type statistic in testing hypotheses about seasonal variation. *Journal of Epidemiology & Community Health*, 33(3):223–228.

Friedman, J., Hastie, T., and Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, 32(200):675–701.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The annals of mathematical statistics*, 11(1):86–92.

Fumi, A., Pepe, A., Scarabotti, L., and Schiraldi, M. M. (2013). Fourier analysis for demand forecasting in a fashion company. *International Journal of Engineering Business Management*, 5(SPL.ISSUE). Cited by: 46; All Open Access, Gold Open Access, Green Open Access.

Gardner, E. S. (1985). Exponential smoothing: The state of the art. *Journal of Forecasting*, 4(1):1 – 28. Cited by: 699.

Gardner, E. S. (2006). Exponential smoothing: The state of the art—part ii. *International Journal of Forecasting*, 22(4):637–666.

George E. P. Box, Gwilym M. Jenkins, G. C. R. (1994). *Time Series Analysis: Forecasting and Control*. Forecasting & control. Prentice-Hall, 3rd edition.

Gubner, J. A. (2006). *Probability and random processes for electrical and computer engineers*. Cambridge University Press, 1 edition.

Hagan, M. T. and Behr, S. M. (1987). The time series approach to short term load forecasting. *IEEE Transactions on Power Systems*, 2(3):785–791.

Harvey, A. (1990). *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press.

Harvey, A. C. (1984). A unified view of statistical forecasting procedures. *Journal of forecasting*, 3(3):245–275.

Harvey, A. C. and Peters, S. (1990). Estimation procedures for structural time series models. *Journal of forecasting*, 9(2):89–108.

Harvey, A. C. and Shephard, N. (1993). 10 structural time series models.

Hastie, T. and Tibshirani, R. (1987). Generalized additive models: some applications. *Journal of the American Statistical Association*, 82(398):371–386.

Hess, A., Iyer, H., and Malm, W. (2001). Linear trend analysis: a comparison of methods. *Atmospheric Environment*, 35(30):5211–5222. Visibility, Aerosol and Atmospheric Optics.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

Holland, P. W. (1986). Statistics and causal inference. *Journal of the American Statistical Association*, 81(396):945 – 960. Cited by: 2897.

Holt, C. C. (2004). Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1):5–10.

Hyndman, R. and Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts.

Hyndman, R. J. and Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679 – 688. Cited by: 2739; All Open Access, Green Open Access.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. volume 2017-December, page 3147 – 3155. Cited by: 3378.

Makridakis, S. and Hibon, M. (1997). Arma models and the box–jenkins methodology. *Journal of forecasting*, 16(3):147–163.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2018). Statistical and machine learning forecasting methods: Concerns and ways forward. *PLoS ONE*, 13(3). Cited by: 518; All Open Access, Gold Open Access, Green Open Access.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2020). The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54 – 74. Cited by: 284; All Open Access, Hybrid Gold Open Access.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2022). M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4):1346 – 1364. Cited by: 31; All Open Access, Hybrid Gold Open Access.

McClain, J. O. and Thomas, L. J. (1973). Response-variance tradeoffs in adaptive forecasting. *Operations Research*, 21(2):554–568.

McDonald, G. C. (2009). Ridge regression. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(1):93–100.

Mitchell, B. (1971). A comparison of chi-square and kolmogorov-smirnov tests. *Area*, pages 237–241.

Musbah, H. and El-Hawary, M. (2019). Sarima model forecasting of short-term electrical load data augmented by fast fourier transform seasonality detection. In *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, pages 1–4.

Park, K. I. (2018). Fundamentals of probability and stochastic processes with applications to communications.

Park, M. Y. and Hastie, T. (2007). L1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(4):659–677.

Pegels, C. C. (1969). Exponential forecasting: Some new variations. *Management Science*, pages 311–315.

Pincus, S. M. (1991). Approximate entropy as a measure of system complexity. *Proceedings of the National Academy of Sciences of the United States of America*, 88(6):2297 – 2301. Cited by: 4517; All Open Access, Bronze Open Access, Green Open Access.

Pincus, S. M., Gladstone, I. M., and Ehrenkranz, R. A. (1991). A regularity statistic for medical data analysis. *Journal of clinical monitoring*, 7:335–345.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., and Gulin, A. (2018). Catboost: Unbiased boosting with categorical features. volume 2018-December, page 6638 – 6648. Cited by: 763.

Rink, D. R. and Swan, J. E. (1979). Product life cycle research: A literature review. *Journal of Business Research*, 7(3):219–242.

Rubin, D. B. (1981). The bayesian bootstrap. *The annals of statistics*, pages 130–134.

Samal, K. K. R., Babu, K. S., Das, S. K., and Acharaya, A. (2019). Time series based air pollution forecasting using sarima and prophet model. page 80 – 85. Cited by: 50.

Santer, B. D., Wigley, T., Boyle, J., Gaffen, D. J., Hnilo, J., Nychka, D., Parker, D., and Taylor, K. (2000). Statistical significance of trends and trend differences in layer-average atmospheric temperature time series. *Journal of Geophysical Research: Atmospheres*, 105(D6):7337–7356.

Sarmukaddam, S. and Rao, S. (1987). A comparison of statistical tests for seasonality. *Nimhans Journal*, 5:53–58.

Schober, P. and Schwarte, L. A. (2018). Correlation coefficients: Appropriate use and interpretation. *Anesthesia and Analgesia*, 126(5):1763 – 1768. Cited by: 2561; All Open Access, Hybrid Gold Open Access.

Shumway, R. H. and Stoffer, D. S. (2017). *ARIMA Models*, pages 75–163. Springer International Publishing, Cham.

Siami-Namini, S. and Namin, A. S. (2018). Forecasting economics and financial time series: Arima vs. lstm. *arXiv preprint arXiv:1803.06386*.

Siami-Namini, S., Tavakoli, N., and Namin, A. S. (2018). A comparison of arima and lstm in forecasting time series. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)*, pages 1394–1401. IEEE.

Siami-Namini, S., Tavakoli, N., and Namin, A. S. (2019). The performance of lstm and bilstm in forecasting time series. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 3285–3292. IEEE.

Silver, E. A., Pyke, D. F., and Thomas, D. J. (2016). *Inventory and Production Management in Supply Chains*. CRC Press, 4 edition.

Slimstock (2022). Experts in inventory management solutions.

Stoica, P. and Selen, Y. (2004). Model-order selection: a review of information criterion rules. *IEEE Signal Processing Magazine*, 21(4):36–47.

Stolwijk, A., Straatman, H., and Zielhuis, G. (1999). Studying seasonality by using sine and cosine functions in regression analysis. *Journal of Epidemiology & Community Health*, 53(4):235–238.

Taylor, J. W. (2003a). Exponential smoothing with a damped multiplicative trend. *International journal of Forecasting*, 19(4):715–725.

Taylor, J. W. (2003b). Short-term electricity demand forecasting using double seasonal exponential smoothing. *Journal of the Operational Research Society*, 54(8):799–805.

Taylor, J. W. (2010). Triple seasonal methods for short-term electricity demand forecasting. *European Journal of Operational Research*, 204(1):139–152.

Taylor, S. J. and Letham, B. (2018). Forecasting at scale. *American Statistician*, 72(1):37 – 45. Cited by: 714.

Trigg, D. (1964). Monitoring a forecasting system. *OR*, 15(3):271–274.

Valipour, M. (2015). Long-term runoff study using sarima and arima models in the united states. *Meteorological Applications*, 22(3):592–598.

Wan Ahmad, W. K. A. and Ahmad, S. (2013). Arima model and exponential smoothing method: A comparison. In *AIP Conference Proceedings*, volume 1522, pages 1312–1321. American Institute of Physics.

Winters, P. R. (1960). Forecasting sales by exponentially weighted moving averages. *Management science*, 6(3):324–342.

Yenidogan, I., Cayir, A., Kozan, O., Dag, T., and Arslan, C. (2018). Bitcoin forecasting using arima and prophet. page 621 – 624. Cited by: 40.

Zhang, G. P. and Qi, M. (2005). Neural network forecasting for seasonal and trend time series. *European Journal of Operational Research*, 160(2):501 – 514. Cited by: 619.

Appendices

A Fisher's F-test

As of currently, Slimstock uses a slight modification of the Fisher's F-test to classify seasonality of their SKUs. As we do not have access to these modifications and these modifications are to be kept confidential, we use the default Fisher's F-test in our research. In the following paragraphs, we describe the workings of this F-Test.

Fisher's F-Test tests whether seasonality is present in a time series. The test evaluates whether the two time series are sufficiently correlated. In addition, the test also tests whether the differences between our time buckets are significant enough to apply a seasonal pattern.

Fisher's F-test takes in two time series as well as our parameter alpha as an input. The two time series represent our aggregated demand data. Time series 1 represents our most recent year of data, and time series 2 represents our second most recent year of data. Parameter alpha is our significance level. In our research, the Fisher's F-test, as well as all other statistical tests, use $\alpha = 0.05$.

We explain the test using an example. Our example uses quarterly aggregated demands.

Table A.1: Example input data for our F-Test

Time Series	**Q1**	**Q2**	**Q3**	**Q4**
Year 1	100	200	100	50
Year 2	80	100	100	40
Average	90	150	100	45

Firstly, the test first calculates the arithmetic average of each of our time buckets. Our aggregated demand data, as well as these averages, are provided in Table A.1 above.

After computing the averages of our time buckets, we compute the average of these averages. We get $\frac{90+150+100+45}{4} = 96.25$. This is our double average.

Now that we know our double average, we compute two values, being the variance within our groups, and the variance between our groups.

First, the variance within our groups. Variance within our groups is defined as the sum of the squared differences between our double average and the actual data, for each time bucket. Thus, for year 1, this value is

$$\text{ss within (year 1)} = (100 - 90)^2 + (200 - 150)^2 + (100 - 100)^2 + (50 - 45)^2 = 2625$$

For year 2, this computation yields the same result as year 1 and 2 are symmetric around the average. Thus, ss within $= 2625 \cdot 2 = 5250$.

Second, the variance between groups. The variance between groups is computed by summing the squared differences between the averages and the double average. Hence,

$$\text{ss between} = (90 - 96.25)^2 + (150 - 96.25)^2 + (100 - 96.25)^2 + (45 - 96.25)^2 = 5568.75$$

We now compute 2 values, being MSb and MSw. To compute these values, we need to know our degrees of freedom. To compute MSb, we need the degrees of freedom related to the length of

our input data. Our degrees of freedom are length of the input data minus one. In our case, $DFb = 4 - 1 = 3$.

We can now compute MSb as $MSb = \frac{\text{ss between}}{\text{DFb}} = \frac{5568.75}{3} = 1856.25$.

Then, MSw. To compute MSw, we need our degrees of freedom DFw. DFw is defined as the length of our input series of year 2 minus one. In our case, $DFw = 4 - 1 = 3$. We can now compute MSw as $MSw = \frac{\text{ss within}}{DFw} = \frac{5250}{4} = 1312.50$.

Now that we know our values for MSb and MSw, we can compute the value of our F-Test. The value is defined as F-Test value $= \frac{MSb}{MSw}$.

In our case, F-Test value $= \frac{MSb}{MSw} = \frac{1856.25}{1312.50} \approx 1.414$

To check what this value implies, we need to find the critical value for our F-Test. This is determined by the inverse F-Distribution. In our case, this value is $F(x)^{-1}_{3,\,3,\,0.05} \approx 9.277$.

Since $1.414 < 9.277$, we conclude the data show sufficient evidence to assume a seasonal pattern is present.

B Friedman Test

One of the investigated methods for classification of seasonality is the Friedman test. The Friedman test is a nonparametric alternative to the currently used F-test. In the following paragraphs, we describe the workings of this Friedman test.

The Friedman test tests whether seasonality is present in a time series. It does so by checking if the months of low and high demand appear in the same (or a comparable) order over the years.

The Friedman test uses three years worth of data, aggregated to the level for which we are testing the seasonality. Parameter alpha is our significance level. In our research, the Friedman test, as well as all other statistical tests, uses $\alpha = 0.05$.

We explain the test using an example. Our example uses quarterly aggregated demands.

We first start off by providing our hypotheses.

H_0 : There is no significant difference between our four quarters. H_1 : There is a significant difference between our four quarters.

Table B.1: Example input data for our Friedman test.

Time Series	**Q1**	**Q2**	**Q3**	**Q4**
Year 1	100 -> 3	200 -> 1	130 -> 2	50 -> 4
Year 2	80 -> 3	100 -> 2	110 -> 1	40 -> 4
Year 3	120 -> 1	100 -> 2	30 -> 4	80 -> 3
Rank Sum	7	5	7	11

Firstly, the test first calculates evaluates the order of our time series by the year. The highest value gets rank 1, and the lowest value gets rank 4. Ranks are provided in Table B.1 above.

After evaluating the order within each year, we sum the ranks for each of our periods. We notice that Quarter 1 has a value of 7, quarter 2 has a value of 5, quarter 3 has a value of 7, and quarter 4 has a rank sum of 11.

We then compute the expectation of the number of ranks for a given period. That is, the value we would get for our rank sum in case there was no difference for all of our time points. In our example, this is as follows:

$$E(R) = \frac{N(k+1)}{2} = \frac{3(4+1)}{2} = 7.5 \tag{B.1}$$

Now, we calculate the value of our test statistic. This value is given by the following formula:

$$\chi_r^2 = \frac{12}{nk(k+1)} \sum R^2 - 3n(k+1) \tag{B.2}$$

For our example, this evaluates to:

$$\chi_r^2 = \frac{12}{3 \cdot 4(4+1)}(7^2 + 5^2 + 7^2 + 11^2) - 3 \cdot 3 \cdot (4+1) \tag{B.3}$$

$$\chi_r^2 = \frac{12 \cdot (5^2 + 2 \cdot 7^2 + 11^2)}{3 \cdot 4(4+1)} - 3 \cdot 3 \cdot (4+1) \tag{B.4}$$

$$\chi_r^2 = \frac{2928}{60} - 45 \tag{B.5}$$

$$\chi_r^2 = 3.8 \tag{B.6}$$

Now, we look up the critical value for our test statistic. Note that we have $df = 4 - 1 = 3$ degrees of freedom. For 3 degrees of freedom, using a significance level of $\alpha = 0.05$, this value is 7.815.

We can now check if the value of our test statistic is greater than our critical value. In our case, $3.8 < 7.815$, and thus we do not reject H_0.

Thus, we find the provided data does not provide sufficient evidence to conclude that there is a difference between our four groups, using a significance level of $\alpha = 0.05$. Therefore, we consider the time series to be nonseasonal.

C Holt-Winters Exponential Smoothing

Although we already covered the workings of the general Holt-Winters exponential smoothing method in Section 2.2.3, we would like to highlight the alternative to this model, the extensions that have been proposed over the years, as well as some problems related to parameter selection and other shortcomings of the model.

Firstly, the alternatives to the model covered in Section 2.2.3. The model we covered in Section 2.2.3 is of the form:

$$F_t = (a + b_t) \cdot S_t \text{ (ATMS)} \tag{C.1}$$

Which is an additive trend multiplicative seasonality (ATMS) model. This model has three alternatives, the choice of which heavily depends on the scenario. The three alternatives are additive trend additive seasonality (ATAS), multiplicative trend additive seasonality (MTAS), and multiplicative trend multiplicative seasonality (MTMS) (Pegels, 1969). Note that, although (Pegels, 1969) proposes nine alternatives to the original exponential smoothing models, being all combinations of (no trend, additive trend, multiplicative trend) and (no seasonal effect, additive seasonal effect, and multiplicative seasonal effect), we only cover the four options that have both a trend and seasonality (either additive or multiplicative). This is done as the combinations that have either no trend, no seasonality, or both, are subsets of these four models (no trend models have parameter $t = 0$, whereas no seasonality models have $St = 1$ for multiplicative and $St = 0$ for additive).

$$F_t = (a + b_t) + S_t \quad \text{(ATAS)} \tag{C.2}$$
$$F_t = (a \cdot b_t) + S_t \quad \text{(MTAS)} \tag{C.3}$$
$$F_t = (a \cdot b_t) \cdot S_t \quad \text{(MTMS)} \tag{C.4}$$

Like mentioned, we think all of these models have their place. We make a distinction based on three basic generalizations provided in (Silver et al., 2016). These three generalizations are:

- Products have a limited lifespan, starting with a strong growth phase and ending with a phase of decline.
- Products (and their profits) increase substantially during their growth phase, stabilize the maturity stage, and decline towards the end of their life cycle.
- We need a different strategy for each stage. This includes inventory management and forecasting.

Firstly, the maturity stage. This is the stable stage of the product lifespan, and will oftentimes be the longest phase of them all. We consider two of the four models to be suitable for products in this stage. Considered models are ATMS and ATAS. We exclude MTAS and MTMS models as these cause exponential growth due to their b^t components, which is not present during this stage.

Products that are either in the phase in or phase out stage of their life cycles oftentimes show an exponential increase (phase in) and exponential decrease (phase out) in demand (with our trend dampening over time for both of these phases (Gardner, 1985)). Thus, MTAS and MTMS are suitable models for these phases. Models that include dampening of these trends are covered in (Gardner, 2006) and (Taylor, 2003a).

The best forecasts are made by combining statistical forecasting based on the past and known information about the future (Silver et al., 2016), the latter of which is not taken into

account by the Holt-Winters model. Since establishment of Holt's method in 1957 (Holt, 2004) and Winters' extensions in 1960 (Winters, 1960), researchers have attempted to solve some of these practical and theoretical problems. We first focus on issues related to parameter selection, initialization, and updating procedures.

Although a lot of research has been conducted on selection of smoothing parameters, a general consensus has not been reached. Selection of smoothing parameters will always depend on the degree to which the environment is dynamic, with more dynamic environments requiring more smoothing, whilst perfectly stationary environments perform best when our smoothing factors are 0 (Silver et al., 2016). Note that, for all smoothing parameters, the smoothing value should increase as the length of the updating periods increase. (Silver et al., 2016) recommend the following values: Level $\alpha = 0.01 - 0.30$, compromise 0.10 for simple exponential smoothing (no trend, no seasonals), and level and trend $\alpha = \beta = 0.01 - 0.30$, compromise 0.10 for a trend model. Triple exponential smoothing is a bit more complex, though (Silver et al., 2016) conduct some experiments based on $\alpha = 0.01 - 0.30$, compromise 0.10. Results are shown in Table C.1 below. Note that γ_{HW} should be significantly lower than α_{HW} for a stable result (McClain and Thomas, 1973). (Chatfield and Yar, 1988) recommend values of $\alpha = \beta = 0.4$ and $\gamma = 0.1$ which are in line with the stability requirement proposed by (McClain and Thomas, 1973) but violate the upper limit of $\beta = 0.176$ proposed by (Silver et al., 2016).

Table C.1: Smoothing Parameter Ranges as Explored by (Silver et al., 2016).

	Underlying α	α_{HW}	β_{HW}	γ_{HW}
Upper end	0.30	0.51	0.176	0.50
Reasonable	0.10	0.19	0.053	0.10
Lower end	0.01	0.02	0.005	0.05

These parameters are input parameters for the three update equations (Silver et al., 2016):

$$\hat{a}_t = \alpha_{HW}(x_t/\hat{F}_{t-P}) + (1 - \alpha_{HW})(\hat{a}_{t-1} + \hat{b}_{t-1}) \tag{C.5}$$

$$\hat{b}_t = \beta_{HW}(\hat{a}_t - \hat{a}_{t-1}) + (1 - \beta_{HW})\hat{b}_{t-1} \tag{C.6}$$

$$\hat{F}_t = \gamma_{HW}(x_t/\hat{a}_t) + (1 - \gamma_{HW})\hat{F}_{t-P} \tag{C.7}$$

Before we jump to conclusions on the optimal choices and ranges for smoothing parameters, we would like to mention the downside of any exponential smoothing model. The main downside of any exponential smoothing model is the fact that we always need to make a trade-off between responsiveness of the model and variance of the model. When updating any value in our model, we can never be sure we are updating because of a shift in the level/trend/seasonality in our model, or if we are updating because of random fluctuations. This is extensively covered in (McClain and Thomas, 1973).

Regarding the optimal value ranges of our parameters, we conclude $\alpha = 0.01 - 0.30$, with a reasonable value being 0.10 for monthly works well for single exponential smoothing. Having values of $\alpha > 0.30$ will often imply we are ignoring a present trend/seasonality and thus we selected the wrong model. We agree with (Silver et al., 2016) on double exponential smoothing, where $\alpha = \beta = 0.01 - 0.30$, with a reasonable value being 0.10 for monthly are proposed. Note that deviating in this range may not always have an effect as double exponential smoothing parameters are symmetric, meaning $\alpha = a$ and $\beta = b$ is equivalent to $\alpha = b$ and $\beta = a$ (McClain and Thomas, 1973). Triple exponential smoothing is most complex as it requires most parameters and there is no general consensus on optimal values and value ranges. Although seasonality

extraction is a large part of our research, we have decided selection of parameters is largely beyond the scope of our research for two main reasons. Firstly, we may not have sufficient data to include smoothing for all cases. Implementation of smoothing requires an initialization over an x number of periods first. (Silver et al., 2016) recommend at least 4, ideally 5, and sometimes 6 periods for their initialization, though we have not found any scientific sources to back this claim. Secondly, optimisation of smoothing parameters is computationally costly, and it is not performed in slim4 for that reason. Therefore, we select smoothing parameters that should work well according to (Silver et al., 2016).

We now look at optimality of the models. Simple exponential smoothing and double exponential smoothing are known to be optimal state-space models (meaning we do not include knowledge about the future) for the no trend (simple) and linear trend (double) models (Abraham and Ledolter, 1986) (Chatfield and Yar, 1988). For triple exponential smoothing, we can model trend and seasonality in a similar way to the additive Holt-Winters model and optimally update it by means of the Kalman filter (Chatfield and Yar, 1988) (Harvey, 1984) (Harvey, 1990).

It is possible to manually improve the Holt-Winters method by including knowledge about the future. As (Silver et al., 2016) mention, the best forecasts are often made using a combination of numerical analysis and a prediction of what will happen in the near future. (Chatfield and Yar, 1988) successfully improve the quality of the Holt-Winters forecast by including subjective modifications, but this is not an automatic process. Due to the sheer scale of forecasts that need to be made by our problem owner Slimstock, we have excluded manual methods.

D Auto Regressive Moving Average methods (ARMA Family)

A popular though more complex alternative to exponential smoothing methods are methods belonging to the ARMA class. ARMA models are models that combine Auto Regressive (AR) models and Moving Average (MA) models to make their forecast. In this part of our literature review, we first research the AR and MA models. Then, we explore how we can combine these models into the ARMA models. After we have researched the basic ARMA model, we look at its most common extensions, being ARIMA for non-stationary models, and seasonal ARIMA for non-stationary time series with seasonalities.

Auto Regressive (AR) models

First, we explore the auto regressive models as described in (George E. P. Box, 1994) and (Wan Ahmad and Ahmad, 2013). Auto regressive forecasting models are simple forecasting models that make their forecasts based on the assumption that the future values of the time series are a linear combination of its past values plus some random component ϵ_t. We denote the auto regressive model by $AR(p)$, where p is the order of the model. The order of the model describes the number of significant components taken into account.

We define the $AR(p)$ model as follows:

$$X_t = \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t \tag{D.1}$$

The simplest AR model possible is the $AR(0)$ model. This model corresponds to a process where past values do not significantly contribute to the future values and thus the process only consists of white noise. In essence, this model models a process that does not show a significant pattern. We define the $AR(0)$ model as follows:

$$X_t = \epsilon_t \tag{D.2}$$

Once we get to an order of 1 or higher, the model starts to become useful. Suppose we have a time series in which the future value is dependent on only the current value. In this case, we are dealing with an AR(1) model.

$$X_t = \phi_1 X_{t-1} + \epsilon_t \tag{D.3}$$

Here, we can see the value of our forecast is equal to the current value multiplied by some factor plus some noise term. For very simple, stationary time series, this is a very naive though realistic model. In the case of demand forecasting, we assume demand is nonnegative and thus we only forecast values greater than or equal to 0. For the $AR(1)$ model, we only consider nonnegative values of ϕ_1. Starting from degree 2, we start considering negative values of ϕ_i, though we cannot have the case where all values of ϕ_i are smaller than 0, as we do not want to make negative forecasts.

The order of the auto regressive model is often determined based on the PACF (as explained in Section J.4.3). The more significant terms we have, the higher the order of the AR model.

Moving Average (MA) models

Now, on to the Moving Average (MA) models. Again, we explore the MA model as described in (George E. P. Box, 1994) and (Wan Ahmad and Ahmad, 2013). The MA model is another very simple model that forecasts future values based on past forecast errors ϵ_{t-i} and some random component ϵ_t. Just like the $AR(p)$ model, which has order p, we denote the MA model by its order q. We define the $MA(q)$ model as follows:

$$X_t = \mu + \sum_{i=1}^{q} (\phi_i \cdot \epsilon_{t-i}) + \epsilon_t \tag{D.4}$$

The $MA(q)$ model functions in a similar way as the AR(p) model, except the model fits based on errors instead of past observations. Just like the $AR(p)$ model, we can use correlation metrics to determine the order of the the $MA(q)$ model. The relevant correlation metric for the $MA(q)$ model is ACF, as explained in Section J.4.2.

The Auto Regressive Moving Average model (ARMA)

The ARMA model is a forecasting model that combines previously described AR(p) and MA(q) models into the ARMA(p,q) model, defined as follows:

$$X_t = \mu + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{i=1}^{q} (\phi_i \cdot \epsilon_{t-i}) + \epsilon_t \tag{D.5}$$

The ARMA model serves as an improvement to both the AR and MA models by taking both the autocorrelation and the moving average of our time series into account (Silver et al., 2016), (George E. P. Box, 1994), (Wan Ahmad and Ahmad, 2013), (Makridakis and Hibon, 1997). Just like the individual $AR(p)$ and $MA(q)$ models, we can use the PACF and ACF for determining the order of the model. Alternatively, we can use the Akaike Information Criterion (AIC) (Stoica and Selen, 2004) for finding p and q (Brockwell and Davis, 2009). Note that most stationary time series can be modeled by ARMA models that have $p + q = 2$ (Silver et al., 2016). Models belonging to the ARMA class are often used as an alternative to exponential smoothing, though this requires some extensions for non-stationarity (ARIMA) and seasonality (seasonal ARIMA). These extensions are covered in the next two paragraphs.

Auto Regressive Integrated Moving Average (ARIMA)

As mentioned, the ARMA model only applies to stationary time series. ARIMA is a generalisation to the ARMA model, allowing for differencing of the time series in order to obtain stationarity. Common notation for the model is $ARIMA(p, d, q)$, where d is the order of differentiation. p and q are defined as before (Wan Ahmad and Ahmad, 2013), (George E. P. Box, 1994), (Shumway and Stoffer, 2017).

Seasonal Auto Regressive Integrated Moving Average (Seasonal ARIMA)

Although the ARIMA model allows for a bit more flexibility than the ARMA model, it is not able to model seasonality. The Seasonal ARIMA model is an extension to the ARIMA model that features an additional part to account for seasonality. It is denoted as $ARIMA(p, d, q)(P, D, Q)_m$, where p,d,q are as before. P,D,Q are the seasonal counterparts to p,d,q. Lastly we have our new parameter m, which indicates the number of seasons per year

(Hyndman and Athanasopoulos, 2018). The Seasonal ARIMA model works well in environments where seasonalities are present, though performance rapidly goes down if we do not select a suitable value for our parameter m (Valipour, 2015), (Dabral and Murry, 2017).

Comparison between Exponential Smoothing and ARMA Models

Based on literature alone it is difficult to say whether Exponential Smoothing or ARMA models are better suited for forecasting retail demand data. (Alemu et al., 2023) finds that exponential smoothing outperforms seasonal ARIMA models in less variable environments whilst seasonal ARIMA performs better in highly erratic environments, though we have to note that the study was conducted in a field that is fundamentally different from retail. (Wan Ahmad and Ahmad, 2013) come to the same conclusion in their research though this study was also conducted in another field. In addition, (Wan Ahmad and Ahmad, 2013) find seasonal ARIMA performs better over the long horizon when compared to exponential smoothing.

It is interesting to note that both (Wan Ahmad and Ahmad, 2013) and (Alemu et al., 2023) base their conclusions on the metrics RMSE/MSE (use of either of these metrics will lead to the same conclusion), MAD, and MAPE. As explained in Section 3.3, selection of the best model is highly dependent on the cost of error unless either of the models strictly dominates all other candidates. Therefore, we cannot come to a conclusion based on these two studies alone.

E Fourier Analysis

In this appendix, we describe the mathematics and logic behind the Fourier transform. The Fourier transform is used in the Fourier variants of exponential smoothing and nested exponential smoothing, as well as in the Prophet algorithm also covered in our research. The Fourier transform is used in these models for the purpose of modeling seasonality.

To support our explanations, we use part of an example data set. This data set is hourly energy consumption from PJM Interconnection LLC (PJM), which is a regional transmission organization (RTO) in the United States (source: https://www.kaggle.com/datasets/robikscube/hourly-energy-consumption) Figures in Table E.1 below are demand values at 0:00 on the first day of each month. The Fourier transform can only be used in case the data has no trend. For time series that have a trend, the algorithms handle trend removal themselves.

Table E.1: Monthly Demand Data Example

Date	**Value**
01/01/2010	15116
01/02/2010	17604
01/03/2010	15925
01/04/2010	13150
01/05/2010	13124
01/06/2010	14076
01/07/2010	14350
01/08/2010	14071
01/09/2010	17008
01/10/2010	13014
01/11/2010	12974
01/12/2010	14796
01/01/2011	13694
01/02/2011	17479
01/03/2011	15845
01/04/2011	15699
01/05/2011	12121
01/06/2011	17811
01/07/2011	15669
01/08/2011	16872
01/09/2011	15940
01/10/2011	12998
01/11/2011	14817
01/12/2011	16718

E.1 Fourier Series

The Fourier Transform is a linear invertible mathematical transformation that allows us to analyse discrete time signals in the frequency domain. The algorithm transforms a time series into a series of sinusoids, all having their own magnitudes and frequencies. Analysis of a signal in the frequency can allow us to detect patterns that would be very difficult to detect in the time domain. The general form of the Fourier Transform is continuous, but it also has a discrete time counterpart known as the Discrete Fourier Transform (DFT). In our research, we make use of the

DFT and its inversion (IDFT).

The Fourier Transform uses complex numbers. A key property of these complex numbers is Euler's formula, given by:

$$e^{i\xi} = cos(\xi) + isin(\xi) \tag{E.1}$$

We also make use of the fact that we can rewrite any complex number of the form $a + bi$ to the form $Ae^{i\xi}$, where A is the magnitude, ξ is the angle, and i denotes the imaginary unit $\sqrt{-1}$. Also note that any real number of the form a can be written as a complex number $a + bi$, where a number is real if and only if $b = 0$.

Now, using complex input sequence $\{x_n\} = x_0, x_1, ..., x_{N-1}$, we can use the DFT to generate our output sequence $\{X_n\} = X_0, X_1, ..., X_{N-1}$. The DFT is defined as follows:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N}kn} \tag{E.2}$$

Suppose we use the first part of our time series:

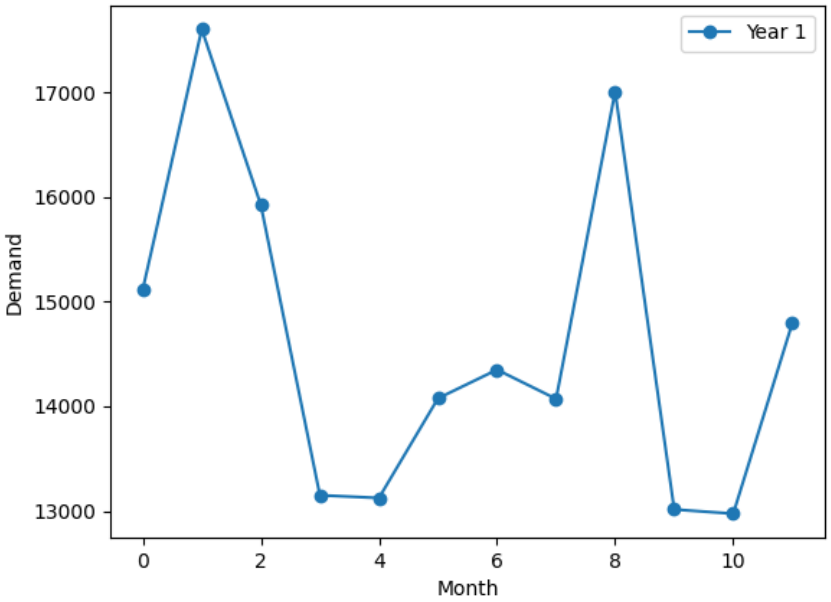

Figure E.1: Time series.

Given by the following data:

Table E.2: Monthly Demand Data Example

Date	Value
01/01/2010	15116
01/02/2010	17604
01/03/2010	15925
01/04/2010	13150
01/05/2010	13124
01/06/2010	14076
01/07/2010	14350
01/08/2010	14071
01/09/2010	17008
01/10/2010	13014
01/11/2010	12974
01/12/2010	14796

We now apply the Discrete Fourier Transform to this. This yields the following:

Table E.3: Discrete Fourier Transform of Monthly Demand Data Example

Term	Real Part	Imaginary Part
0	175208	+ 0i
1	3832.71	- 734.50i
2	4060	- 8346.75i
3	1999	- 2677i
4	-4159	+ 3491.81i
5	-3533.71	- 2350.50i
6	1786	+ 0i
7	-3533.71	+ 2350.50i
8	-4159	- 3491.81i
9	1999	+ 2677i
10	4060	+ 8346.75i
11	3832.71	+ 734.50i

Although these numbers are still a perfect representation of our time series, it doesn't tell us much as this representation is not intuitive. We can use the power spectral density (PSD) to get a more intuitive idea of what is going on. The power spectal density shows the power of each frequency in our time series. The PSD of our signal is depicted in Figure E.2 below. Note that the y axis is logarithmic for viewing purposes.

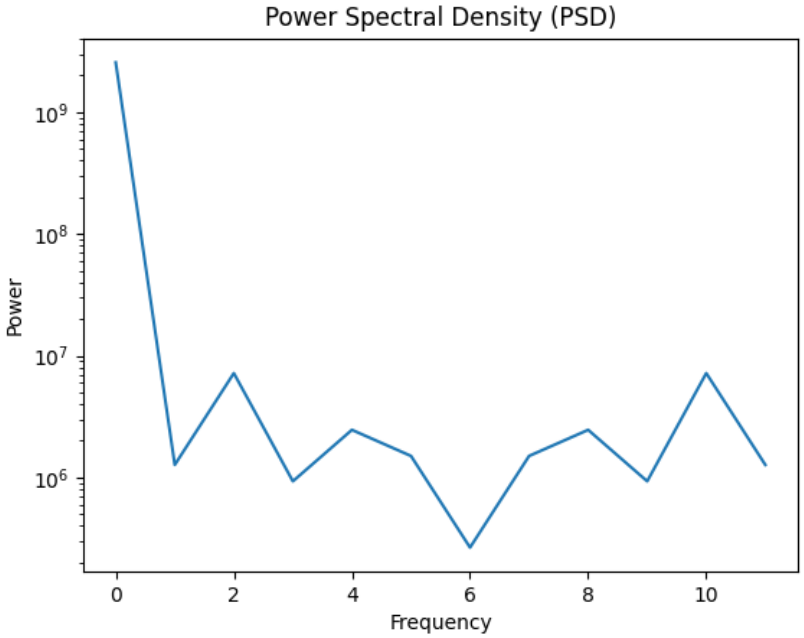

Figure E.2: Power Spectral Density of our Time Series.

E.2 Filtering

Now that we have extracted the frequency components from our time series, we can choose to use just the terms we desire. The operation of picking out just the desired terms of our Power Spectrum is known as filtering. We make a distinction between two types of filters, being power filters and frequency filters. We will start off by briefly describing the power filter, though this filter is not used in our research. The power filter is an alternative to the later explained frequency filter, which is the filter we ended up using in our research.

E.2.1 Power Filters

We can see the most powerful term is a sinusoid with frequency 0. This is a constant that corresponds to the level. Other very significant terms appear to be the sinusoids with frequencies 2 and 10. We can model our seasonalities using just these three terms (0, 2, 10). This yields the following Fourier series and modeled seasonality:

Table E.4: High-Power filtered Discrete Fourier Transform of Monthly Demand Data Example

Term	Real Part	Imaginary Part
0	175208	+ 0i
1	0	+ 0i
2	4060	- 8346.75i
3	0	+ 0i
4	0	+ 0i
5	0	+ 0i
6	0	+ 0i
7	0	+ 0i
8	0	+ 0i
9	0	+ 0i
10	4060	+ 8346.75i
11	0	+ 0i

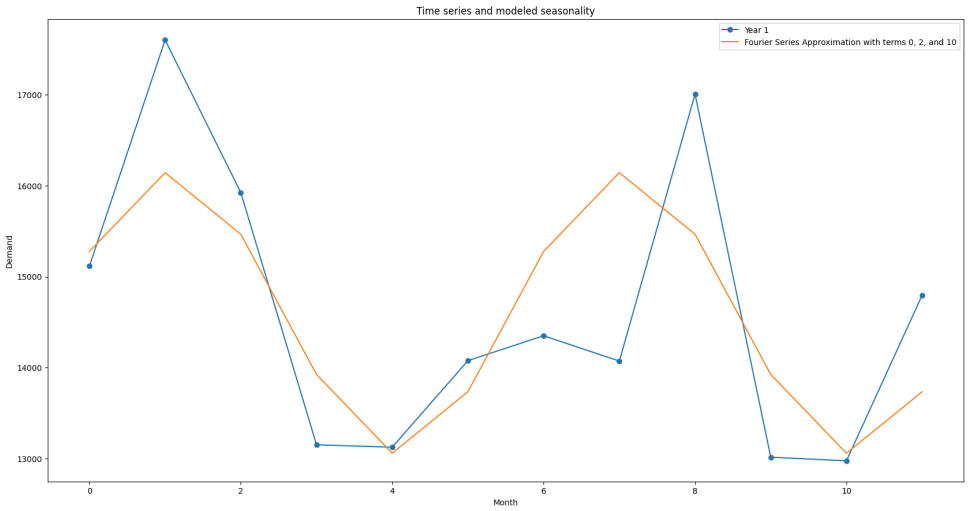

Figure E.3: Modeled seasonality using 3 terms (0, 2, 10).

Whilst the modeled seasonality has significant visible deviations from our original time series, we note that inclusion of just 3 Fourier terms is able to capture the biggest cyclical effects. It also filters out effects that could be noise (such as the dip in the month having index 7 (August)).

Instead of picking the terms that are very powerful, we can also eliminate the terms that are not very powerful. In our case, this is the term with a frequency of 6 as seen in Figure E.2. Removing just this term yields the following Fourier series and modeled seasonality:

Table E.5: Low-Power Filtered Discrete Fourier Transform of Monthly Demand Data Example

Term	Real Part	Imaginary Part
0	175208	+ 0i
1	3832.71	- 734.50i
2	4060	- 8346.75i
3	1999	- 2677i
4	-4159	+ 3491.81i
5	-3533.71	- 2350.50i
6	0	+ 0i
7	-3533.71	+ 2350.50i
8	-4159	- 3491.81i
9	1999	+ 2677i
10	4060	+ 8346.75i
11	3832.71	+ 734.50i

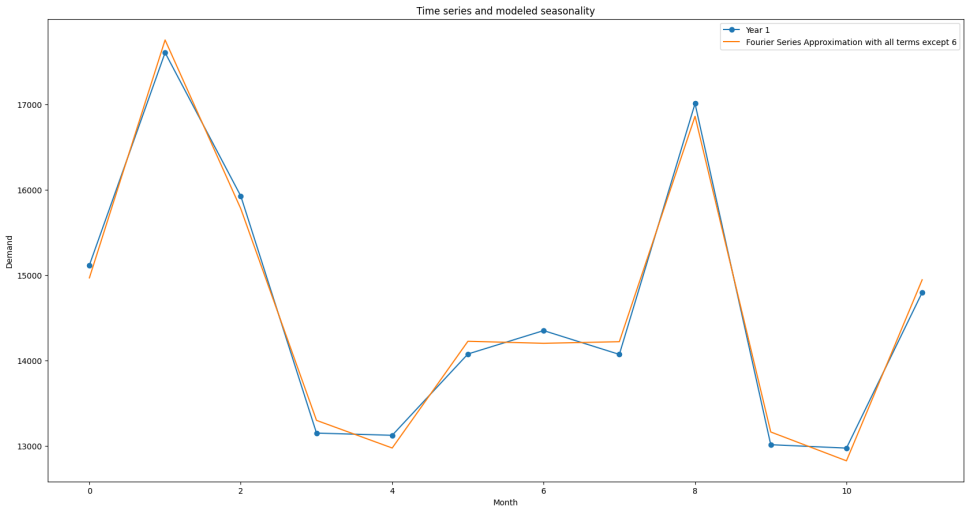

Figure E.4: Modeled seasonality using all terms except 6.

Using all but our most significant of terms, we notice the modeled seasonality is very close to the original time series. The main differences appear to be the modeling of lower seasonality in months indexed 4 and 10 (May and November), and the flattening during summer months indexed 5 to 7 (June, July, and August). This removes the dip in August, which is likely noise. Despite being frequently used in the field of signal processing (for example, to remove annoying sharp and high-pitched tones in audio), we have no reason to believe frequencies other than high frequencies are responsible for the noise present in our time series.

E.2.2 Frequency Filters

Contrary to power filters, which manually pick out a set of frequencies, we can make use of a frequency filter. Frequency filters are filters that only keep the frequencies within a certain range, setting all other values to zero. A low pass filter only passes lower frequencies (e.g. a low-pass 3 filter only passes 0, 1, and 2), and a high pass filter only passes high frequencies (e.g. high pass 3 passes 9, 10, 11). We illustrate the low pass filter in Table E.6 and Figures E.5 and E.6 below. Filters can be combined to filter out certain ranges of frequencies, but this is not done in our

research as we hypothesise the low and medium frequencies are responsible for the macroscopic effects we wish to extract and use for modeling seasonality.

Table E.6: Discrete Fourier Transform of Monthly Demand Data Low Pass Example

Term	Real Part	Imaginary Part
0	175208	+ 0i
1	3832.71	- 734.50i
2	4060	- 8346.75i
3	0	+ 0i
4	0	+ 0i
5	0	+ 0i
6	0	+ 0i
7	0	+ 0i
8	0	+ 0i
9	0	+ 0i
10	0	+ 0i
11	0	+ 01

Now, this yields the following Power Spectrum:

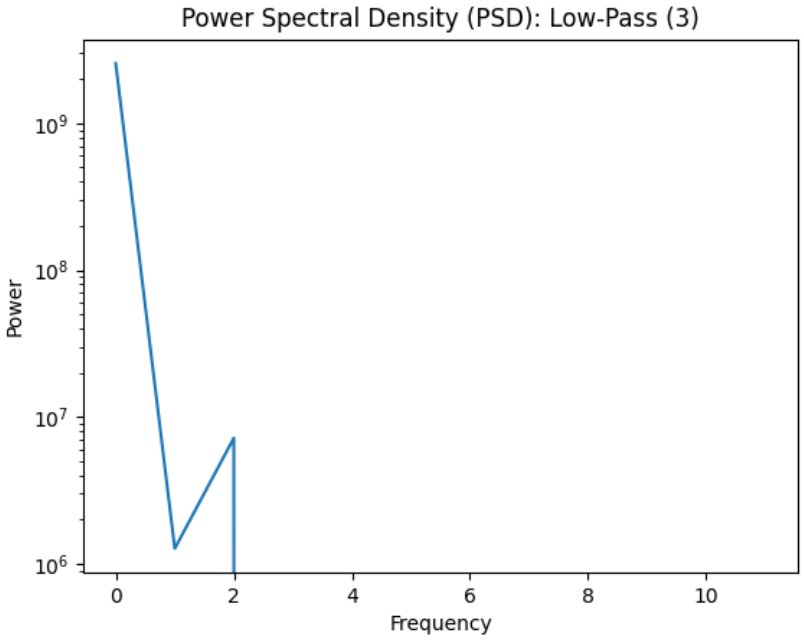

Figure E.5: Low-Pass Power Spectrum

And leads to the following modeled seasonality:

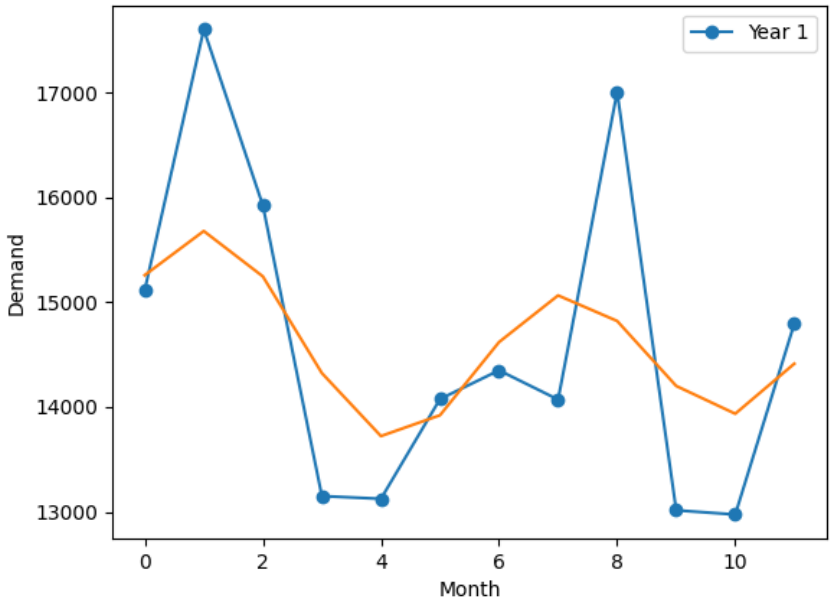

Figure E.6: Time series and Seasonality using a Low-Pass (3) filter

As we can see, the seasonality we modeled using just 3 terms is able to capture the longer cycles (level, yearly and half-yearly), but the magnitude of the effect appears to be very small. This is because the frequencies responsible for our fluctuations do not have enough energy to represent the entire time series. The following section proposes a solution to this problem, though this solution is not used in our research. The solution used in our research is covered in Section G.5.

E.2.3 Scaling

As depicted in Figure E.6, the effects of seasonality are very small. This is because we lose energy in our power spectrum when we apply any filter. This was most notable when we applied the low-pass filter. This is due to the removal of frequencies that carried a lot of power, and we did not do anything to compensate for this.

To overcome this issue, we need to scale all of our terms except for the first term, which is the level (we do not scale the level as we wish to keep this the same) by a certain value. We determine this value to be the ratio between the energy present in the frequencies of the power spectrum before and after the low-pass filter was applied (note that the frequency 0 is excluded as this is, again, the level).

The total energy of all nonzero frequencies in Figure E.2 is approximately 26 940 037. In the filtered power spectrum, this is approximately 8 448 417. This gives us a ratio of $\frac{26\ 940\ 037}{8\ 448\ 417} \approx 3.18$. Scaling all Fourier terms except our first by this number yields the following Fourier series and modeled seasonality:

Table E.7: Scaled Discrete Fourier Transform of Monthly Demand Data Low Pass Example

Term	Real Part	Imaginary Part
0	175208	+ 0i
1	12188.00	- 2335.70i
2	12910.80	- 26542.67i
3	0	+ 0i
4	0	+ 0i
5	0	+ 0i
6	0	+ 0i
7	0	+ 0i
8	0	+ 0i
9	0	+ 0i
10	0	+ 0i
11	0	+ 01

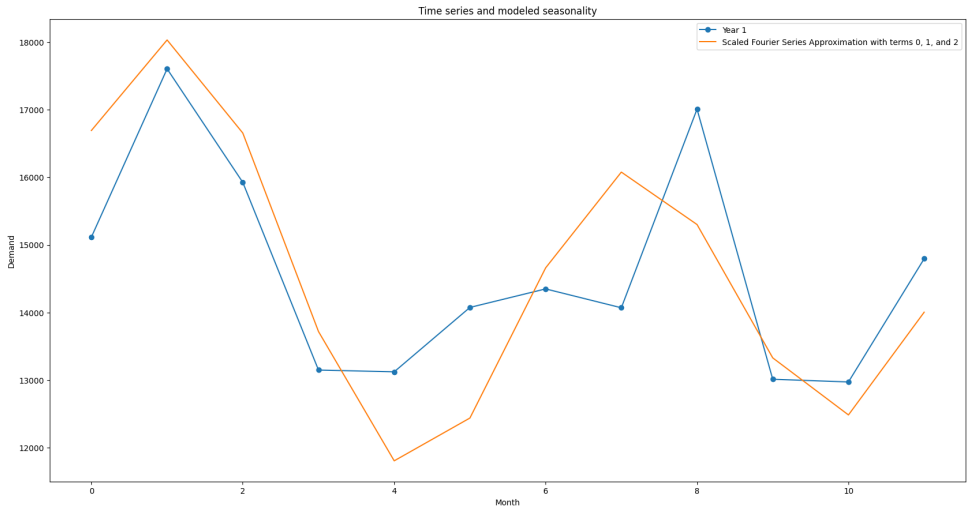

Figure E.7: Time series and Scaled Seasonality using a Low-Pass (3) filter

We note that the modeled seasonality now has comparable amplitudes to the original time series.

E.3 Inversion

When we applied the Fourier Transform to our time series, we ended up with a Fourier series that perfectly represents the original signal. We then applied operations such as filtering and scaling to our Fourier series. To go from the modified Fourier series to our modified time series, we need to invert the Fourier transform. This is done using the Inverse Discrete Fourier Transform (IDFT), which functions similarly to the Discrete Fourier Transform as provided before.

Now, using complex input sequence $\{X_n\} = X_0, X_1, ..., X_{N-1}$, we can use the IDFT to generate our output sequence $\{x_n\} = x_0, x_1, ..., x_{N-1}$. The IDFT is defined as follows:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot e^{\frac{i2\pi}{N} kn} \tag{E.3}$$

E.4 Computation and the Fast Fourier Transform

The DFT and IDFT are both algorithms that run in $O(N^2)$ time, where N is the length of our time series. For both the DTF and the IDFT, we can reduce this computational complexity to $O(NlogN)$ using an algorithm known as the Fast Fourier Transform (FFT). The FFT makes use of symmetry present in both the DFT and IDFT. The FFT is simply a way to efficiently compute the DFT, and we use the terms DFT and FFT interchangeably from now on. We will use IFFT and IDFT to refer to the Inverse DFT. The code used in our research uses the more efficient variants of the algorithm.

F Multiplicative Models and Nested Seasonalities

Our forecasting models all use a level, an additive trend, and either a multiplicative or an additive seasonality. Multiplicative forecasts are made based on the following equation:

$$F_t = (a + bt) \cdot St \tag{F.1}$$

Seasonality usually comes in the form of a monthly, weekly, or daily seasonality. Some SKUs, however, may face multiple seasonal patterns. These SKUs can have different demands depending on the week of the year, but also different demands depending on the day of the week.

To account for the presence of multiple seasonal patterns, we have two possible solutions. We will cover both of these briefly in the following paragraphs.

The first solution is the use of 365 seasonal indices, one for every day of the year. This solution maintains the form of Equation F.1. Maintaining this form comes at a cost, however. As we have 365 seasonal indices, and also 365 data points per year, we have a 1:1 ratio of parameters: data. We improve on this ratio by adding more data, but even when we use five years worth of data (which is a lot), we only have a 1:5 ratio of parameters:data.

The second solution is adaptation of our models to use nested seasonal factors. Instead of having a seasonal factor for every day of the year, we now have a factor for every day of the week, week of the year, and month of the year. This results in a total of $7 + 53 + 12 = 72$ parameters, which is a lot less than 365. By using nested seasonal factors, we are able to update each of our seasonals much more frequently. Now, our days of the week will have 52/53 (depending on the year) data points, our weeks will have 7 data points. Each of our months has a variable number of data points depending on the length of the month.

F.1 Nesting

For our research, we have chosen to use individual indices for days, weeks, and months. This approach is similar to the approach described in (Taylor, 2010). For both seasonality extraction and deseasonalisation, we process our time series in the order of month -> week -> day. We need to process from largest to smallest time frame to obtain the correct seasonal factors. We use this approach for Holt-Winters, and also for deseasonalisation of time series in case we use the Seasonality first, Trend second approach for classification of our time series.

G Implementation and Verification of Single, Double, and Triple Exponential Smoothing

This appendix describes the implementation of used single, double, and triple exponential smoothing models, as well as the modifications made. This implementation is written using Python version 3.11.5 and has the following external dependencies:

- numpy 1.24.2
- pandas 1.5.3
- tabulate 0.9.0
- scipy 1.10.1
- scikit-learn 1.2.2

Our research includes the following exponential smoothing models:

1. Single Exponential Smoothing (Currently used)
2. Double Exponential Smoothing (Currently used)
3. Triple Exponential Smoothing with trend, monthly seasonality only (Currently used in a modified variant)
4. Triple Exponential Smoothing without trend, monthly seasonality only (Currently used in a modified variant)
5. Triple Exponential Smoothing with trend, Fourier-processed monthly seasonality only (Alternative to the current method)
6. Triple Exponential Smoothing without trend, Fourier-processed monthly seasonality only (Alternative to the current method)
7. Triple Exponential Smoothing with trend, nested seasonality (Alternative to the current method)
8. Triple Exponential Smoothing without trend, nested seasonality (Alternative to the current method)
9. Triple Exponential Smoothing with trend, Fourier-processed nested seasonality (Alternative to the current method)
10. Triple Exponential Smoothing without trend, Fourier-processed nested seasonality (Alternative to the current method)

G.1 Single Exponential Smoothing

Single exponential smoothing is our simplest forecasting model, using just a level. We use the definition from (Silver et al., 2016). Demand is assumed to be distributed according to the following formula:

$$x_t = a_t + \epsilon_t \tag{G.1}$$

As mentioned before, initial level for each SKU is equal to the average value of the time series belonging to that SKU. After this, level is updated daily, according to the following formula:

$$\hat{a}_t = \alpha D_t + (1 - \alpha)\hat{a}_{t-1} \tag{G.2}$$

G.1.1 Single Exponential Smoothing: Verification

Suppose we have the following time series:

Table G.1: SES: Verification

Day	Value
1	1
2	2
3	1
4	2
5	1
6	2
7	1
8	2
9	1
10	2

We now validate the implementation of the model using the following parameters: Initialisation period: 4 days Updating period: 2 days Forecasting + Updating period: 4 days alpha: 0.2.

For our short cover time, we use 1 day here. Medium cover time is 2 days. Long cover time is 3 days.

Initialisation: Initialisation of this model is done by averaging out the first 4 observations. Manual computation yields an initial level of level $= \frac{1+2+1+2}{4} = 1.5$.

Updating: After performing 4 periods worth of initialisation, we perform 2 days of updating. This is done as follows: On day 5, our level was 1.5. Actual demand is 1. At the end of day 5, we compute the level for day 6. We obtain: $\text{level}_6 = (1 - 0.2) * 1.5 + 0.2 * 1 = 1.4$. On day 6, our level was 1.4. Actual demand is 2. At the end of day 6, we compute the level for day 7. We obtain: $\text{level}_7 = (1 - 0.2) * 1.4 + 0.2 * 2 = 1.52$.

Forecasting + Updating: Now, at the end of day 6, we need to make forecasts for all three of our time periods. This means we forecast 1 day ahead for short cover times, 2 days ahead for medium cover times, and 3 days ahead for long cover times. We forecast the value of 1.52, as this is our most recent level.

We now update the model to compute the level of day 8. At the end of day 7, we need to make forecasts for our short cover period, as day 7 is a review day. Note that we do not need to forecast for our medium and long cover periods, as day 7 is not a review day. Updating the level as before yields $\text{level}_8 = (1 - 0.2) * 1.52 + 0.2 * 1 = 1.416$. We forecast this value for day 8 for our short cover period.

At the end of day 8, we compute the level of day 9 as we need to make make forecast for the short and medium cover times.

Following this updating and forecasting procedure should yield the following output:

Table G.2: SES: Verification Output

Day	**7**	**8**	**9**	**10**
Cover time: 1 day	1.52	1.416	1.5328	1.42624
Cover time: 2 days	1.52	1.52	1.5328	1.5328
Cover time: 3 days	1.52	1.52	1.52	1.42624

Which is in line with our Python model. We conclude SES is correctly implemented.

G.2 Double Exponential Smoothing

Double exponential smoothing is fairly similar to single exponential smoothing, with the difference being the introduction of a trend parameter. We again use the definition provided in (Silver et al., 2016). Demand is assumed to be distributed according to the following formula:

$$x_t = a_t + bt + \epsilon_t \tag{G.3}$$

With updating equations:

$$\hat{a}_t = \alpha_{HW} x_t + (1 - \alpha_{HW})(\hat{a}_{t-1} + \hat{b}_{t-1}) \tag{G.4}$$

$$\hat{b}_t = \beta_{HW}(\hat{a}_t - \hat{a}_{t-1}) + (1 - \beta_{HW})\hat{b}_{t-1} \tag{G.5}$$

Initialisation of a and b is done using linear regression. Updating is done in a similar manner as described before.

We implement Double Exponential Smoothing, and validate the implementation using Table 3.3 of (Silver et al., 2016). Our model outputs the correct values, so we conclude the implementation is correct.

G.3 Triple Exponential Smoothing Monthly Only (Holt-Winters)

Our implementation of the Holt-Winters triple exponential smoothing model is designed along the specifications of slim4. This means we use the exponential smoothing model where level and trend are smoothed, and seasonality is recalculated. Initialisation of the model is done using the seasonality-first, level + trend second methodology as described in (Silver et al., 2016). Note that we use full seasons for initialisation.

For Triple Exponential Smoothing, demand is assumed to be distributed according to the following formula:

$$D_t = (a + bt)F_{1,t} + \epsilon_t \tag{G.6}$$

As Slim4 currently only uses monthly seasonality, we only have a single level of seasonality, which si monthly seasonality. This means we have 12 seasonal factors, one for each month. Seasonal factors are normalised such that their sum is 12. Seasonal factors are used for both forecasting, as well as updating. Updating our level and trend requires deseasonalisation of actual demand, which is done using the seasonal factor corresponding to the current month.

This makes our updating equations:

$$\hat{a}_t = \alpha D_t/(\hat{S}_{1,t-P1}) + (1-\alpha)(\hat{a}_{t-1} + \hat{b}_{t-1}) \tag{G.7}$$

$$\hat{b}_t = \beta(\hat{a}_t - \hat{a}_{t-1}) + (1-\beta)\hat{b}_{t-1} \tag{G.8}$$

Note that $\alpha D_t/(\hat{S}_{1,t-P1})$ is undefined in case $\hat{S}_{1,t-P1} = 0$. In case $\alpha D_t = 0$, demand is 0, so we consider deseasonalised demand to be 0 as well, even though $\frac{0}{0}$ is undefined. This makes the assumption seasonality is constant in a sense that, in case demand in a given month is historically 0 (meaning we have a computed seasonality of 0), the product has not been sold on that month in the past and will not be sold in this month in the future. We consider this assumption to be acceptable as a historic seasonality of 0 implies a forecast of 0 anyway, and in case the actual demand in that month is not 0, the seasonal factor for next year will not be 0 due to the re-computation of seasonality in Slim4.

Lastly, we forecast according to the equation:

$$\hat{D}_{t,t+\tau} = (\hat{a}_t + \hat{b}_t\tau)\hat{S}_{1,t-P1} \tag{G.9}$$

We have implemented and validated the initialisation of the model along specification of Section 3.4.4.3 of (Silver et al., 2016). After implementing this model (which is a quarterly model), we have made a model that implements the extension to daily forecasts using monthly seasonality in MS Excel. This model has been validated by Slimstock. Lastly, we validate the total model implementation in Python using a synthetically generated data set. The model is able to extract the parameters used to generate the data set within margin of error.

G.4 Triple Exponential Smoothing Monthly Only, No Trend (Holt-Winters)

Implementation and verification of Triple Exponential Smoothing using monthly seasonality and no trend is almost identical to the model that does include a trend. Omitting the trend yields the following new equation for updating:

$$\hat{a}_t = \alpha D_t/(\hat{S}_{1,t-P1}) + (1-\alpha)(\hat{a}_{t-1}) \tag{G.10}$$

$$\tag{G.11}$$

And forecasting is done according to:

Lastly, we forecast according to the equation:

$$\hat{D}_{t,t+\tau} = (\hat{a}_t)\hat{S}_{1,t-P1} \tag{G.12}$$

G.5 Triple Exponential Smoothing with trend, Fourier-processed Monthly Seasonality

This model is an extension to Triple Exponential Smoothing with Trend as described in Section G.3. The models and implementation is identical, except for a small modification to the seasonal factors in forecasting (note that these are not the seasonal factors used for deseasonalisation, that part remains unchanged).

The modification is as follows: Currently, the model stores our 12 seasonal factors in a numpy array (which we can treat as a list of numbers here). Our first value is the value for January, the second value is the value for February, and so on. In the modification, we apply a Fourier transform (as explained in Section E, then a low-pass frequency filter (as explained in Section E.2.2, which filters out the high frequencies. We use a value of 10 as our cutoff frequency, this is in line with (Taylor and Letham, 2018). Finally, we invert the filtered transform (as explained in Section E.3).

In code, this looks as follows:

```
MONTHS_IN_YEAR = 12

# Seasonal factors are stored in the array monthly_seasonal_factors
# Apply Fourier Transform
fft_array = numpy.fft.fft(monthly_seasonal_factors)

# Set filtering frequency
cutoff_frequency = 10

# Filter out the high frequencies by setting them to 0
for idx in range(cutoff_frequency, MONTHS_IN_YEAR):
    fft_array[idx] = 0

# Inversion
monthly_seasonal_factors = numpy.fft.ifft(fft_array).real

# Normalisation
monthly_seasonal_factors = (monthly_seasonal_factors \
/ numpy.sum(monthly_seasonal_factors)) * MONTHS_IN_YEAR
```

The application, filtering, and inversion of the Fourier Transform has been validated and is correctly implemented.

G.6 Triple Exponential Smoothing without trend, Fourier-processed Monthly Seasonality

This model is a simplification of the previous model, as the model is identical except for the fact that this model does not include a trend. Omission of the trend is described in Section G.4.

G.7 Triple Exponential Smoothing with trend, nested seasonality

As described in Appendix F, which in turn is based on (Taylor, 2010) , we can extend Triple Exponential Smoothing to include nesting of seasonalities. The model functions in a similar manner to Triple Exponential Smoothing with trend, monthly seasonality only (which is described in Section G.3), except we iteratively remove seasonal effects in case they are present. We start off by removing monthly seasonal effects in case monthly seasonal effects are present. Then, we test for weekly seasonal effects using the F-test, trend first approach. In case weekly seasonality is present, we remove this as well. Lastly, we do the same for daily seasonality. In case any of our seasonality levels is not present, we do not modify the time series (which is a computational shortcut of deseasonalisation using seasonal factors of 1).

G.8 Triple Exponential Smoothing without trend, nested seasonality

This model is identical to nested Triple Exponential Smoothing with trend, except the trend is not modeled. Again, omission of the trend is described in Section G.4.

G.9 Triple Exponential Smoothing with Trend, Fourier-processed Nested Seasonality

This model is a combination of Triple Exponential Smoothing with trend, nested seasonality as explained in Section G.7 and the Fourier processing of seasonality as described in Section G.5. We only apply the Fourier processing to our monthly seasonal factors, meaning only SKUs that have monthly, monthly-weekly, monthly-daily, or monthly-weekly-daily seasonality are affected.

G.10 Triple Exponential Smoothing without Trend, Fourier-processed Nested Seasonality

This model is identical to our previous model, except trend is not modeled as explained in Section G.4.

H Performance of Classifiers for our Time Series

This appendix contains the full performance evaluation tables of our experiments on finding the best classifier for trend and seasonality.

H.1 F-test, Trend First (Current Method)

Table H.1: Performance of the Current Method for Short Cover Times

SKU types	Method	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias	Weight
Stationaries	SES	0.175	83.94	0.128	-0.031	$\frac{2341}{9419}$
Trending	DES	0.19	71.07	0.144	0.013	$\frac{6533}{9419}$
Seasonal Nontrending	TES no trend	0.204	79.34	0.149	-0.108	$\frac{178}{9419}$
Seasonal Trending	TES with trend	0.215	80.71	0.153	-0.058	$\frac{367}{9419}$
Total	-	0.188	74.8	0.14	-0.003	1

Medium cover times give the following results:

Table H.2: Performance of the Current Method for Medium Cover Times

SKU types	Method	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias	Weight
Stationaries	SES	0.254	55.69	0.19	-0.035	$\frac{2341}{9419}$
Trending	DES	0.285	48.75	0.219	0.002	$\frac{6533}{9419}$
Seasonal Nontrending	TES no trend	0.326	53.66	0.236	-0.128	$\frac{178}{9419}$
Seasonal Trending	TES with trend	0.38	56.39	0.268	-0.084	$\frac{367}{9419}$
Total	-	0.282	50.87	0.214	-0.013	1

And lastly, long cover times give the following results:

Table H.3: Performance of the Current Method for Long Cover Times

SKU types	Method	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias	Weight
Stationaries	SES	0.538	37.76	0.424	-0.042	$\frac{2341}{9419}$
Trending	DES	0.667	37.1	0.534	-0.018	$\frac{6533}{9419}$
Seasonal Nontrending	TES no trend	0.814	40.57	0.6	-0.149	$\frac{178}{9419}$
Seasonal Trending	TES with trend	1.027	44.15	0.728	-0.083	$\frac{367}{9419}$
Total	-	0.652	37.6	0.515	-0.029	1

To summarise, we have the following performances for our short, medium, and long cover times:

Table H.4: Performance of the Current Method for Each of our Cover Times

SKU types	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias
Short (3 days)	0.188	74.8	0.14	-0.003
Medium (7 days)	0.282	50.87	0.214	-0.013
Long (21 days)	0.652	37.6	0.515	-0.029

H.2 F-test, Seasonality First

Table H.5: Performance of the F-test, Seasonality First Method for Short Cover Times

SKU types	Method	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias	Weight
Stationaries	SES	0.174	84.47	0.128	-0.041	$\frac{2710}{9419}$
Trending	DES	0.191	70.11	0.145	0.015	$\frac{6205}{9419}$
Seasonal Nontrending	TES no trend	0.175	89.1	0.126	-0.106	$\frac{194}{9419}$
Seasonal Trending	TES with trend	0.263	74.91	0.189	-0.029	$\frac{310}{9419}$
Total	-	0.188	74.79	0.141	-0.005	1

Table H.6: Performance of the F-test, Seasonality First Method for Medium Cover Times

SKU types	Method	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias	Weight
Stationaries	SES	0.252	55.82	0.189	-0.045	$\frac{2710}{9419}$
Trending	DES	0.286	48.17	0.22	0.004	$\frac{6205}{9419}$
Seasonal Nontrending	TES no trend	0.281	58.3	0.282	-0.147	$\frac{194}{9419}$
Seasonal Trending	TES with trend	0.486	56.29	0.345	-0.038	$\frac{310}{9419}$
Total	-	0.283	50.85	0.216	-0.015	1

Table H.7: Performance of the F-test, Seasonality First Method for Long Cover Times

SKU types	Method	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias	Weight
Stationaries	SES	0.538	38.19	0.423	-0.056	$\frac{2710}{9419}$
Trending	DES	0.671	36.74	0.537	-0.014	$\frac{6205}{9419}$
Seasonal Nontrending	TES no trend	0.667	40.54	0.488	-0.174	$\frac{194}{9419}$
Seasonal Trending	TES with trend	1.378	46.98	0.978	-0.025	$\frac{310}{9419}$
Total	-	0.656	37.57	0.518	-0.03	1

Table H.8: Performance of the F-test, Seasonality First Method for Each of our Cover Times

SKU types	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias
Short (3 days)	0.188	74.79	0.141	-0.005
Medium (7 days)	0.283	50.85	0.216	-0.015
Long (21 days)	0.656	37.57	0.518	-0.03

H.3 Friedman-Chi-Squared-test, Trend First

Table H.9: Performance of the Friedman-Chi-Squared-test, Trend First Method for Short Cover Times

SKU types	Method	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias	Weight
Stationaries	SES	0.171	89.33	0.125	-0.034	$\frac{1598}{9419}$
Trending	DES	0.19	76.14	0.143	0.033	$\frac{3363}{9419}$
Seasonal Nontrending	TES no trend	0.201	75.19	0.149	0.001	$\frac{921}{9419}$
Seasonal Trending	TES with trend	0.223	69.98	0.169	0.057	$\frac{3537}{9419}$
Total	-	0.2	75.97	0.15	0.028	1

Table H.10: Performance of the Friedman-Chi-Squared-test, Trend First Method for Medium Cover Times

SKU types	Method	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias	Weight
Stationaries	SES	0.253	59.49	0.192	-0.042	$\frac{1598}{9419}$
Trending	DES	0.291	53.04	0.225	0.031	$\frac{3363}{9419}$
Seasonal Nontrending	TES no trend	0.31	51.58	0.231	0.022	$\frac{921}{9419}$
Seasonal Trending	TES with trend	0.38	49.93	0.287	0.078	$\frac{3537}{9419}$
Total	-	0.32	52.82	0.243	0.035	1

Table H.11: Performance of the Friedman-Chi-Squared-test, Trend First Method for Long Cover Times

SKU types	Method	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias	Weight
Stationaries	SES	0.541	39.79	0.429	-0.055	$\frac{1598}{9419}$
Trending	DES	0.685	39.98	0.552	0.015	$\frac{3363}{9419}$
Seasonal Nontrending	TES no trend	0.744	38.87	0.572	0.052	$\frac{921}{9419}$
Seasonal Trending	TES with trend	1.02	41.01	0.781	0.114	$\frac{3537}{9419}$
Total	-	0.792	40.23	0.619	0.044	1

Table H.12: Performance of the Friedman-Chi-Squared-test, Trend First Method for Each of our Cover Times

SKU types	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias
Short (3 days)	0.2	75.97	0.15	0.028
Medium (7 days)	0.32	52.82	0.243	0.035
Long (21 days)	0.792	40.23	0.619	0.044

H.4 Friedman-Chi-Squared-test, Seasonality First

Table H.13: Performance of the Friedman-Chi-Squared-test, Seasonality First Method for Short Cover Times

SKU types	Method	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias	Weight
Stationaries	SES	0.17	90.79	0.125	-0.039	$\frac{1601}{9419}$
Trending	DES	0.199	72.74	0.15	0.049	$\frac{1844}{9419}$
Seasonal Nontrending	TES no trend	0.201	71.87	0.15	0.017	$\frac{1862}{9419}$
Seasonal Trending	TES with trend	0.213	73.54	0.161	0.05	$\frac{4112}{9419}$
Total	-	0.201	75.99	0.151	0.028	1

Table H.14: Performance of the Friedman-Chi-Squared-test, Seasonality First Method for Medium Cover Times

SKU types	Method	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias	Weight
Stationaries	SES	0.253	60.59	0.192	-0.048	$\frac{1601}{9419}$
Trending	DES	0.313	51.57	0.244	0.051	$\frac{1844}{9419}$
Seasonal Nontrending	TES no trend	0.319	50.04	0.24	0.052	$\frac{1862}{9419}$
Seasonal Trending	TES with trend	0.355	51.82	0.268	0.066	$\frac{4112}{9419}$
Total	-	0.322	52.91	0.245	0.041	1

Table H.15: Performance of the Friedman-Chi-Squared-test, Seasonality First Method for Long Cover Times

SKU types	Method	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias	Weight
Stationaries	SES	0.54	40.43	0.428	-0.066	$\frac{1601}{9419}$
Trending	DES	0.752	39.52	0.612	0.04	$\frac{1844}{9419}$
Seasonal Nontrending	TES no trend	0.782	38.9	0.605	0.095	$\frac{1862}{9419}$
Seasonal Trending	TES with trend	0.934	41.36	0.714	0.098	$\frac{4112}{9419}$
Total	-	0.801	40.36	0.624	0.058	1

Table H.16: Performance of the Friedman-Chi-Squared-test, Seasonality First Method for Each of our Cover Times

SKU types	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias
Short (3 days)	0.201	75.99	0.151	0.028
Medium (7 days)	0.322	52.91	0.245	0.041
Long (21 days)	0.801	40.36	0.624	0.058

I Implementation of Prophet

This appendix describes the implementation of used single, double, and triple exponential smoothing models, as well as the modifications made to support paralellisation. This implementation is written using Python version 3.11.5 and has the following external dependencies:

- numpy 1.24.2
- pandas 1.5.3
- tabulate 0.9.0
- prophet 1.1.2
- holidays 0.22
- cmdstanpy 1.1.0

Our research covers the default model of Prophet as well as Prophet with holiday support.

I.1 Implementation of Individual Experiments

Implementation of Prophet for a single time series is fairly simple, as described in the official documentation (Facebook, 2023). In our research, we make forecasts for a single cover period at a time. For the first cover period, this means we initialise the model and fit it to the first four years worth of data. We then forecast ahead for our three different forecasting horizons. Once this is done, we skip forward to the next day on which forecasts need to be made. Contrary to exponential smoothing methods, which are updated every day, Prophet needs to be refitted on our entire dataset.

Although it is possible to refit every day, we note that we only need to refit on days that are multiples of any of our cover periods. This means that, for our cover periods of 3, 7, and 21 days respectively, we need to refit on day 0, 3, 6, 7, 9, 12, 14, 15, 18, 21, 24, ... , 364, for a total of 157 unique fits per time series. The first fit uses the first 4 years worth of data. The second fit uses the first 4 years + 3 days, the third uses the first 4 years + 6 days, and so on, all the way to our last fit on day 364, which uses almost 5 years worth of data.

I.1.1 Complexity

Performing initialisation, fitting, and forecasting of a single SKU takes approximately 100 seconds on the CPU our experiments were run on, which is an Intel(R) Xeon(R) Platinum 8352V CPU @ 2.10GHz. Although this would be doable for a single SKU, fitting 9419 SKUs as we have in our experiments, would take approximately 11 days in this way. As we need to run these experiments twice (once with holidays and once without holidays), we get a total runtime of approximately 22 days.

As a runtime of 11 days for each of these experiments is too long, we have two options, being either parallelisation, or updating of the models rather than refitting. Updating of models rather than refitting is possible as described in the documentation of Prophet. As mentioned in the documentation of Prophet, updating of models rather than refitting can lead to worse performance, especially if a lot of data is added. In addition, the number of changepoints (which are data points where a change in either seasonality or trend is detected) cannot change from model to model, or an error will be raised and thus the experiment will terminate. Facebook expects about a 5X speedup when using updating (so-called "Warm starts") rather than refitting. This would reduce

our runtime from approximately 22 days to approximately 4.4 days, which is a nice speedup, but still results in very long runtimes.

Although updating of models rather than refits would certainly be a good recommendation for further research in case the Prophet model turns out to yield the best performance, our research uses a different approach to reduce runtime for several reasons.

Firstly, updating models rather than refitting can lead to a suboptimal fit. Although the process may be faster, we are looking for optimal performance in our research to make the best comparison.

Secondly, as the addition of new data might raise an error rather than finish the experiment, we are forced to refit in case an error is thrown anyway. Refitting skus from scratch sometimes and updating some other times results in unpredictable behaviour in our experiments, which is not desired.

Thirdly, although the reduction from approximately 22 days to approximately 4.4 days is nice, we still think 4.4 days is too slow. Although an actual implementation of the model would only need a single fit rather than 157 fits for every time we need to forecast, the forecasting procedure will take multiple hours if a customer needs to fit thousands of time series each day (which is not as uncommon as one may expect).

I.1.2 Parallelisation

This leads us to the other approach, which is parallelisation. Instead of fitting our SKUs sequentially, we can fit multiple at the same time. Our implementation breaks down our data set into a configurable number of groups. In our case, we break down the dataset into 134 groups of approximately 70 SKUs each. Doing so results in a computational time of a little under 18000 seconds, or just under 5 hours. Note that the use of heavy parallelisation may require very expensive hardware, which not all of Slimstock's customers will have access to. Our research implements parallelisation using the multiprocessing library.

J Researched Unused Classification Methods

This appendix describes classification methods that were investigated in our literature study, but that have not been researched in more detail.

J.1 Chi-Square Goodness-of-Fit

For items where seasonality is not present, we expect to encounter a uniform distribution in demand after we have corrected for our trend. In cases where seasonality is present, this uniform distribution will not hold. We can test for the presence of seasonality by using a Pearson's Chi-Square test, with the Null hypothesis being that demand is evenly spread over time, where n is the number of periods. The Challenger hypothesis is then that demand is not evenly spread out over time. (Sarmukaddam and Rao, 1987), (Edwards, 1961). Despite its popularity, (Edwards, 1961) mention this test has its downfalls and may even be a very bad test for seasonality. This is evident from the fact that, despite there being n! ways to rearrange the time series, the Chi-Square test is not affected by any rearrangements we make.

J.2 Regression of Trigonometric Functions and Chi-Square Test

(Stolwijk et al., 1999) extend this Chi-Square approach by iteratively applying sine and cosine function by means of regression, evaluating the regressed curve at each time point and using these values as an input to the Chi-Square test. This is done until either incorporated functions are not a significant component (in which case seasonality is not present), or until the regressed functions and values are no longer rejected. Advantages of this approach are the fact that this test also extracts seasonals, as well as the fact that the Chi-Square test is now sensitive to the order of our time series.

J.3 Kolmogorov-Smirnov Goodness-of-Fit

(Freedman, 1979) propose the Kolmogorov-Smirnov Goodness-of-Fit test as an alternative to the Chi-Square test. (Mitchell, 1971) investigate the differences between the two tests and find that the Kolmogorov-Smirnov test is suitable when the assumptions of the Chi-Square test are not met. The main difference between the Chi-Square test and the Kolmogorov-Smirnov test are that the Chi-Square test is suitable for binned data, whereas the Kolmogorov-Smirnov test is more suited for continuously distributed data. As we are mainly dealing with aggregated data, we will exclude the Kolmogorov-Smirnov test from our research.

J.4 Correlation, Autocorrelation and Partial Autocorrelation

Another approach to checking the periodicity/cyclicity of a time series is the use of correlation metrics. We cover the most important ones here.

J.4.1 Correlation

In time series analysis, we often make use of a metric known as Pearson's Correlation Coefficient. It is a metric that describes the similarity between two different time series. Being defined as the

covariance of the two time series divided by the product of their variances, its value ranges from -1 to 1. A correlation of 1 means time series are perfectly correlated. That is, any increase/decrease in value present in A is also present in B. A correlation of 0 means the two time series are not related at all, meaning the changes in A and B are not related. A value of -1 means changes are inversely related, meaning an increase in A corresponds to a similar decrease in B and vice versa. (Schober and Schwarte, 2018), (Gubner, 2006)

J.4.2 Autocorrelation (ACF)

Autocorrelation, commonly referred to as the Autocorrelation Function or ACF, is a function that checks for the relation between a variable at time T and its k-lagged variant (where k is an integer representing the number of periods lagged). ACF includes both direct and indirect effects. Autocorrelation is computed in a similar manner to Pearson's Correlation Coefficient. Range and implications for its values are identical. (George E. P. Box, 1994), (Park, 2018).

J.4.3 Partial Autocorrelation (PACF)

Partial autocorrelation, like regular autocorrelation, is a metric that checks for similarity between a time series signal and a lagged version of itself. PACF is an extension to ACF, as PACF excludes the presence of indirect effects by means of regression. (Hagan and Behr, 1987).

J.4.4 Conclusion on Correlation, ACF, and PACF

Despite the fact that correlation does not imply causation and we have to ensure findings in this field are in line with the scientific method (Holland, 1986), we see value in the use of all of correlation metrics for time series forecasting. Although clustering is beyond the scope of our research, we see potential for the use of correlation for forecasting groups of SKUs. Like mentioned in (Silver et al., 2016), we can aggregate demands of similar SKUs in order to make better forecasts. A well known example of related SKUs is different types of ice cream. Suppose we have ice cream type A and B, but we have no historical data on B. We do have historical data on A. In case we assume A and B are indeed positively related, we can use seasonal parameters of A in order to make a forecast for B. This works because we assume A and B have a strong positive correlation.

In addition to strong positive correlation, we can make examples of strong negative correlations. In case product A is negatively correlated with product B, we have what is known as an inverse correlation. This means that, whenever SKU A sells a lot, SKU B sells very little. The other way around also applies. Examples of inversely correlated products are inflatable swimming pools and snow shovels.

We intuitively know that any time series that is perfectly correlated with a lagged version of itself is trivial to predict as the pattern that is present infinitely repeats. In case a time series is almost perfectly correlated with a lagged version of itself, our predictions will likely not be too far off (although they are no longer perfect). Note that ACF and PACF are both metrics that are often used as they are often used to determine the orders of the Auto Regressive Moving Average model (ARMA) and its extensions (e.g. ARIMA, sARIMA). This family of models is covered in Appendix D.

J.5 Approximate Entropy

Entropy is a scientific concept that relates to the degree of order that is present in a certain field or system. Greater entropy represents more chaos and thus less predictability. Entropy is measured in different ways depending on the field.

In time series analysis, we can use approximate entropy (ApEn) to indicate the regularity or predictability of our data. Approximate entropy assumes values ranging from 0 to infinity, where 0 indicates perfect predictability and higher values indicate more chaos.

Approximate entropy calculates the logarithm of conditional probability that two sub-sequences of size m are similar to a certain degree (this degree is determined by our tolerance parameter).

Approximate entropy has its advantages and disadvantages. Advantages of approximate entropy are its insensitivity to the length of the original time series, and its computational efficiency. Despite these advantages, approximate entropy can be a difficult metric to use. Although insensitive to the length of the original input series, approximate entropy is very sensitive to the used sub-sequence length m. Using an m that is too small (shorter than the length of the present pattern) means approximate entropy cannot identify the patterns that are present in our data, and thus the complexity of the time series is not properly assessed. On the other hand, using an m-value that is too large means the complexity of the time series is overestimated, as these sub-sequences can contain a lot of redundancy and noise. The same problems arise when our tolerance parameter r is not properly selected. Values of r that are too small will result in an inability to capture the complexity of the time series (under-fitting problems). Using values that are too large causes over-estimations of the complexity (over-fitting). Lastly, the metric assumes our input data to be stationary. This means that, for non-stationary data, we need to apply some transformations to our data before evaluating the metric. (Pincus et al., 1991), (Pincus, 1991).

K Researched Unused Forecasting Methods

K.1 LightGBM

LightGBM is an extension to traditional gradient boosting machines and is developed by Microsoft. LightGBM introduces several optimizations that makes it faster, more memory-efficient, and slightly more performant. We base our analysis of LightGBM on the paper written by (Ke et al., 2017).

The first optimisation regarding to speed comes from optimisations made in the sampling method. LightGBM introduces a new approach going by the name of Gradient-based One-Side Sampling (GOSS). GOSS is suitable for large or very large data sets and speeds up the learning process by focusing on samples that have larger gradients. Samples that have larger gradients allow for a greater reduction of error in each step without reducing accuracy of the model. Although we will eventually run into the vanishing gradient problem (Hochreiter, 1998) in case we end up using all samples in our data set, we will often reach our stopping criterion long before doing so and thus this method is an efficient way to speed up learning without compromising model performance.

The second optimisation regarding training efficiency comes in the form of memory optimisations. Using a method known as Exclusive Feature Bundling (EFB), LightGBM aggregates several features which allows for a reduction in dimensionality of the input data. This reduction in dimensionality makes the model less memory-intensive and improves computation speed.

The third optimization in training speed comes from early stopping. Once LightGBM fails to improve the performance of the model over several iterations as it is no longer able to construct trees that reduce model error, the training process terminates before reaching the maximum number of trees. This saves computation time and resources.

The last major optimization made is mostly related to model performance, although it may also speed up computation in some cases. Traditional GBMs expand the model by growing trees on a per-level basis, meaning the tree is constructed by adding one level of depth at a time. LightGBM takes a different approach, growing trees on a per-leaf basis. This allows us to expand the tree by growing the leaf that results in the greatest loss, resulting in a lower value to our objective loss function. This produces trees that have lower loss when compared to traditional gradient boosted trees, though this comes at the potential cost of generating more complex trees (i.e.g trees with greater depth).

In addition to these major optimizations, we can speed up the training process of LightGBM by parallelising the learning process. As LightGBM supports multi-core and GPU processing, the training time can be reduced at the cost of more resources.

K.2 CatBoost

An alternative to Microsoft's LightGBM is CatBoost, which is developed by Yandex. Like LightGBM, CatBoost builds upon the general idea of GBMs, introducing a set of optimisations. (Prokhorenkova et al., 2018), (Dorogush et al., 2018).

The first optimisation comes in the way the weak learners are constructed. CatBoost uses a type of decision tree known as an Oblivious decision tree (Ferov and Modrỳ, 2016), (Friedman et al., 2000). The oblivious decision tree uses a single splitting criterion for each level.

This results in more balanced trees that are less prone to overfitting and can be evaluated more efficiently. This optimisation introduces both increased performance and more efficient training.

Bayesian bootstrapping. CatBoost uses sampling with replacement, assigning a probability to each sample according to a Dirichlet distribution. This sampling method allows us to reduce overfitting by introducing randomness to the algorithm at the cost of increased computational cost. (Rubin, 1981)

Similar to LightGBM, another optimization comes in the form of parallelisation. Parallel processing by using multiple CPU or GPU cores allows for more rapid training at the cost of computational power.

K.3 XGBoost

The third and last extension to the traditional GBMs is XGBoost. XGBoost is an open-source machine learning algorithm that extends traditional GBMs both by improving system optimizations (e.g. parallelisation, caching, and optimizations to the way the data is structured in memory), and by introducing both an exact and approximate algorithm for splitting the input data. (Chen and Guestrin, 2016).

XGBoost introduces two algorithms for splitting the input data, one of which is exact and one of which is an approximation. Choice of which algorithm to use depends on the size of the data and available memory. In case the input data completely fits into memory, the exact algorithm is used. In cases where data is distributed among multiple machines (distributed computation is another system optimization made by XGBoost) or where not all of our data fit into memory, we have to use the approximate algorithm. Both algorithms use a histogram-based partitioning, basing their decision on information gain.

XGBoost deals with overfitting in two ways. Firstly, the generated trees have a maximum depth, which means the trees have to generalize and thus we get a model with a more generalized fit. Secondly, XGBoost uses a combination of Lasso (L1) (Park and Hastie, 2007) and Ridge (L2) (McDonald, 2009) regression for its regularization. This also prevents the model from overfitting.

Like LightGBM and CatBoost, XGBoost allows for early stopping once fitting more trees does not significantly improve performance. Other optimisations such as parallel processing and GPU processing are part of the beforementioned system optimisations. We do not cover system optimisations in detail as these are not relevant to the fundamental workings of the algorithm.

K.4 Conclusion on Gradient Boosters

(Bentéjac et al., 2021) and (Al Daoud, 2019) compare the three algorithms and find CatBoost performs best although difference in performance between the three models is not statistically significant. LightGBM was the fastest out of the three, whilst CatBoost was the slowest, leading us to believe LightGBM is most suitable for our task. This is in line with (Makridakis et al., 2022), where LightGBM came out on top.

L Breakdown of Performance of Conservative and Progressive Approach

L.1 Conservative Approach

Table L.1: Performance of the Conservative Approach for Short Cover Times

SKU types	Method	Normalised RMSE	Average SMAPE	Average Normalised MAD	Average Bias	Weight
1: (T: n, D: n, W: n, M: n)	SES	0.158	94.6	0.112	-0.017	$\frac{894}{9419}$
2: (T: n, D: y, W: n, M: n)	Prophet	0.166	74.93	0.12	-0.111	$\frac{1283}{9419}$
3: (T: n, D: n, W: y, M: n)	SES	0.182	98.41	0.135	-0.045	$\frac{24}{9419}$
4: (T: n, D: n, W: n, M: y)	TES	0.125	123.39	0.08	0.075	$\frac{5}{9419}$
5: (T: n, D: y, W: y, M: n)	Prophet	0.168	67.24	0.124	-0.134	$\frac{140}{9419}$
6: (T: n, D: y, W: n, M: y)	TES	0.175	94.4	0.13	-0.031	$\frac{25}{9419}$
7: (T: n, D: n, W: y, M: y)	TES	0.192	91.44	0.13	-0.208	$\frac{17}{9419}$
8: (T: n, D: y, W: y, M: y)	Prophet	0.165	67.82	0.119	-0.131	$\frac{131}{9419}$
9: (T: y, D: n, W: n, M: n)	Prophet	0.167	72.26	0.123	-0.169	$\frac{3071}{9419}$
10: (T: y, D: y, W: n, M: n)	TES Nested Adjusted	0.17	63.51	0.123	0.011	$\frac{3101}{9419}$
11: (T: y, D: n, W: y, M: n)	DES	0.183	83.5	0.135	0.073	$\frac{67}{9419}$
12: (T: y, D: n, W: n, M: y)	TES Trending	0.175	114.7	0.119	-0.043	$\frac{15}{9419}$
13: (T: y, D: y, W: y, M: n)	Prophet	0.182	56.0	0.134	-0.12	$\frac{294}{9419}$
14: (T: y, D: y, W: n, M: y)	Prophet	0.149	86.26	0.109	-0.06	$\frac{40}{9419}$
15: (T: y, D: n, W: y, M: y)	Prophet	0.166	86.49	0.115	-0.062	$\frac{33}{9419}$
16: (T: y, D: y, W: y, M: y)	Prophet	0.159	69.57	0.114	-0.199	$\frac{279}{9419}$
Total	-	0.167	71.58	0.122	-0.082	1

Table L.2: Performance of the Conservative Approach for Medium Cover Times

SKU types	Method	Normalised RMSE	Average SMAPE	Average Normalised MAD	Average Bias	Weight
1: (T: n, D: n, W: n, M: n)	SES	0.241	66.03	0.18	-0.011	$\frac{894}{9419}$
2: (T: n, D: y, W: n, M: n)	SES	0.26	49.33	0.195	-0.045	$\frac{1283}{9419}$
3: (T: n, D: n, W: y, M: n)	SES	0.3	70.82	0.234	-0.022	$\frac{24}{9419}$
4: (T: n, D: n, W: n, M: y)	TES	0.234	71.99	0.142	0.036	$\frac{5}{9419}$
5: (T: n, D: y, W: y, M: n)	SES	0.273	45.38	0.211	-0.095	$\frac{140}{9419}$
6: (T: n, D: y, W: n, M: y)	TES	0.261	58.45	0.195	-0.051	$\frac{25}{9419}$
7: (T: n, D: n, W: y, M: y)	TES	0.348	66.0	0.235	-0.259	$\frac{17}{9419}$
8: (T: n, D: y, W: y, M: y)	Prophet	0.275	48.42	0.207	-0.154	$\frac{131}{9419}$
9: (T: y, D: n, W: n, M: n)	DES	0.283	52.58	0.217	-0.008	$\frac{3071}{9419}$
10: (T: y, D: y, W: n, M: n)	DES	0.283	45.21	0.217	0.006	$\frac{3101}{9419}$
11: (T: y, D: n, W: y, M: n)	DES	0.316	64.13	0.248	0.089	$\frac{67}{9419}$
12: (T: y, D: n, W: n, M: y)	TES Trending	0.321	82.45	0.223	-0.02	$\frac{15}{9419}$
13: (T: y, D: y, W: y, M: n)	DES	0.317	42.49	0.245	0.045	$\frac{294}{9419}$
14: (T: y, D: y, W: n, M: y)	TES Trending	0.307	58.25	0.223	-0.106	$\frac{40}{9419}$
15: (T: y, D: n, W: y, M: y)	Prophet	0.281	64.17	0.209	-0.092	$\frac{33}{9419}$
16: (T: y, D: y, W: y, M: y)	Prophet	0.271	50.0	0.204	-0.259	$\frac{279}{9419}$
Total	-	0.277	50.72	0.211	-0.02	1

Table L.3: Performance of the Conservative Approach for Long Cover Times

SKU types	Method	Normalised RMSE	Average SMAPE	Average Normalised MAD	Average Bias	Weight
1: (T: n, D: n, W: n, M: n)	SES	0.519	44.9	0.411	-0.011	$\frac{894}{9419}$
2: (T: n, D: y, W: n, M: n)	SES	0.542	33.2	0.423	-0.056	$\frac{1283}{9419}$
3: (T: n, D: n, W: y, M: n)	SES	0.706	52.52	0.578	-0.031	$\frac{24}{9419}$
4: (T: n, D: n, W: n, M: y)	TES	0.603	48.35	0.355	0.091	$\frac{5}{9419}$
5: (T: n, D: y, W: y, M: n)	SES	0.606	31.42	0.486	-0.111	$\frac{140}{9419}$
6: (T: n, D: y, W: n, M: y)	TES	0.568	39.86	0.44	0.016	$\frac{25}{9419}$
7: (T: n, D: n, W: y, M: y)	TES	0.861	51.56	0.607	-0.265	$\frac{17}{9419}$
8: (T: n, D: y, W: y, M: y)	Prophet	0.636	36.54	0.488	-0.26	$\frac{131}{9419}$
9: (T: y, D: n, W: n, M: n)	DES	0.671	39.92	0.538	-0.031	$\frac{3071}{9419}$
10: (T: y, D: y, W: n, M: n)	DES	0.652	34.18	0.519	-0.013	$\frac{3101}{9419}$
11: (T: y, D: n, W: y, M: n)	Prophet	0.677	48.89	0.541	-0.264	$\frac{67}{9419}$
12: (T: y, D: n, W: n, M: y)	Prophet	0.431	50.91	0.328	0.052	$\frac{15}{9419}$
13: (T: y, D: y, W: y, M: n)	DES	0.766	34.68	0.616	0.043	$\frac{294}{9419}$
14: (T: y, D: y, W: n, M: y)	Prophet	0.515	34.7	0.401	-0.051	$\frac{40}{9419}$
15: (T: y, D: n, W: y, M: y)	Prophet	0.661	46.43	0.507	-0.112	$\frac{33}{9419}$
16: (T: y, D: y, W: y, M: y)	Prophet	0.635	38.45	0.49	-0.377	$\frac{279}{9419}$
Total	-	0.632	37.35	0.502	-0.031	1

L.2 Progressive Approach

Table L.4: Performance of the Progressive Approach for Short Cover Times

SKU types	Method	Normalised RMSE	Average SMAPE	Average Normalised MAD	Average Bias	Weight
1: (T: n, D: n, W: n, M: n)	Prophet	0.151	94.1	0.108	-0.12	$\frac{894}{9419}$
2: (T: n, D: y, W: n, M: n)	Prophet	0.166	74.93	0.12	-0.111	$\frac{1283}{9419}$
3: (T: n, D: n, W: y, M: n)	Prophet	0.169	97.42	0.124	-0.058	$\frac{24}{9419}$
4: (T: n, D: n, W: n, M: y)	Prophet	0.11	121.04	0.065	0.186	$\frac{5}{9419}$
5: (T: n, D: y, W: y, M: n)	Prophet	0.168	67.24	0.124	-0.134	$\frac{140}{9419}$
6: (T: n, D: y, W: n, M: y)	Prophet	0.159	91.37	0.113	-0.096	$\frac{25}{9419}$
7: (T: n, D: n, W: y, M: y)	Prophet	0.167	91.57	0.121	-0.294	$\frac{17}{9419}$
8: (T: n, D: y, W: y, M: y)	Prophet	0.165	67.82	0.119	-0.131	$\frac{131}{9419}$
9: (T: y, D: n, W: n, M: n)	Prophet	0.167	72.26	0.123	-0.169	$\frac{3071}{9419}$
10: (T: y, D: y, W: n, M: n)	TES Nested Adjusted	0.17	63.51	0.123	0.011	$\frac{3101}{9419}$
11: (T: y, D: n, W: y, M: n)	Prophet	0.166	79.67	0.119	-0.168	$\frac{67}{9419}$
12: (T: y, D: n, W: n, M: y)	Prophet	0.136	112.12	0.089	0.033	$\frac{15}{9419}$
13: (T: y, D: y, W: y, M: n)	Prophet	0.182	56.0	0.134	-0.12	$\frac{294}{9419}$
14: (T: y, D: y, W: n, M: y)	Prophet	0.149	86.26	0.109	-0.06	$\frac{40}{9419}$
15: (T: y, D: n, W: y, M: y)	Prophet	0.166	86.49	0.115	-0.062	$\frac{33}{9419}$
16: (T: y, D: y, W: y, M: y)	Prophet	0.159	69.57	0.114	-0.199	$\frac{279}{9419}$
Total	-	0.166	71.49	0.121	-0.094	1

Table L.5: Performance of the Progressive Approach for Medium Cover Times

SKU types	Method	Normalised RMSE	Average SMAPE	Average Normalised MAD	Average Bias	Weight
1: (T: n, D: n, W: n, M: n)	SES	0.241	66.03	0.18	-0.011	$\frac{894}{9419}$
2: (T: n, D: y, W: n, M: n)	SES	0.26	49.33	0.195	-0.045	$\frac{1283}{9419}$
3: (T: n, D: n, W: y, M: n)	Prophet	0.285	70.75	0.22	-0.031	$\frac{24}{9419}$
4: (T: n, D: n, W: n, M: y)	Prophet	0.193	69.19	0.116	0.178	$\frac{5}{9419}$
5: (T: n, D: y, W: y, M: n)	SES	0.273	45.38	0.211	-0.095	$\frac{140}{9419}$
6: (T: n, D: y, W: n, M: y)	Prophet	0.235	55.86	0.176	-0.096	$\frac{25}{9419}$
7: (T: n, D: n, W: y, M: y)	Prophet	0.293	67.61	0.224	-0.36	$\frac{17}{9419}$
8: (T: n, D: y, W: y, M: y)	Prophet	0.275	48.42	0.207	-0.154	$\frac{131}{9419}$
9: (T: y, D: n, W: n, M: n)	Prophet	0.281	52.44	0.217	-0.213	$\frac{3071}{9419}$
10: (T: y, D: y, W: n, M: n)	DES	0.283	45.21	0.217	0.006	$\frac{3101}{9419}$
11: (T: y, D: n, W: y, M: n)	Prophet	0.286	60.89	0.219	-0.208	$\frac{67}{9419}$
12: (T: y, D: n, W: n, M: y)	Prophet	0.22	76.15	0.158	0.068	$\frac{15}{9419}$
13: (T: y, D: y, W: y, M: n)	Prophet	0.306	41.37	0.234	-0.149	$\frac{294}{9419}$
14: (T: y, D: y, W: n, M: y)	Prophet	0.239	54.71	0.186	-0.077	$\frac{40}{9419}$
15: (T: y, D: n, W: y, M: y)	Prophet	0.281	64.17	0.209	-0.092	$\frac{33}{9419}$
16: (T: y, D: y, W: y, M: y)	Prophet	0.271	50.0	0.204	-0.259	$\frac{279}{9419}$
Total	-	0.275	50.59	0.210	-0.09	1

Table L.6: Performance of the Progressive Approach for Long Cover Times

SKU types	Method	Normalised RMSE	Average SMAPE	Average Normalised MAD	Average Bias	Weight
1: (T: n, D: n, W: n, M: n)	SES	0.519	44.9	0.411	-0.011	$\frac{894}{9419}$
2: (T: n, D: y, W: n, M: n)	SES	0.542	33.2	0.423	-0.056	$\frac{1283}{9419}$
3: (T: n, D: n, W: y, M: n)	Prophet	0.672	54.16	0.539	-0.025	$\frac{24}{9419}$
4: (T: n, D: n, W: n, M: y)	Prophet	0.466	42.49	0.265	0.274	$\frac{5}{9419}$
5: (T: n, D: y, W: y, M: n)	SES	0.606	31.42	0.486	-0.111	$\frac{140}{9419}$
6: (T: n, D: y, W: n, M: y)	Prophet	0.458	36.27	0.359	-0.091	$\frac{25}{9419}$
7: (T: n, D: n, W: y, M: y)	Prophet	0.685	52.51	0.558	-0.511	$\frac{17}{9419}$
8: (T: n, D: y, W: y, M: y)	Prophet	0.636	36.54	0.488	-0.26	$\frac{131}{9419}$
9: (T: y, D: n, W: n, M: n)	DES	0.671	39.92	0.538	-0.031	$\frac{3071}{9419}$
10: (T: y, D: y, W: n, M: n)	DES	0.652	34.18	0.519	-0.013	$\frac{3101}{9419}$
11: (T: y, D: n, W: y, M: n)	Prophet	0.677	48.89	0.541	-0.264	$\frac{67}{9419}$
12: (T: y, D: n, W: n, M: y)	Prophet	0.431	50.91	0.328	0.052	$\frac{15}{9419}$
13: (T: y, D: y, W: y, M: n)	Prophet	0.73	32.96	0.565	-0.171	$\frac{294}{9419}$
14: (T: y, D: y, W: n, M: y)	Prophet	0.515	34.7	0.401	-0.051	$\frac{40}{9419}$
15: (T: y, D: n, W: y, M: y)	Prophet	0.661	46.43	0.507	-0.112	$\frac{33}{9419}$
16: (T: y, D: y, W: y, M: y)	Prophet	0.635	38.45	0.49	-0.377	$\frac{279}{9419}$
Total	-	0.630	37.29	0.500	-0.038	1

M Classifications for each of our Different Methods

Table M.1: Number of SKUs by Seasonal Properties using the F-test.

	No Seasonality	**Monthly Seasonality**	**Total**
No Trend	2341	178	2519
Trend (Trend First)	6533	367	6900
Total	8874	545	9419

Table M.2: Number of SKUs by Seasonal Properties using the F-test, Seasonality First.

	No Seasonality	**Monthly Seasonality**	**Total**
No Trend	2710	194	2904
Trend (Seasonality First)	6205	310	6515
Total	8915	504	9419

Table M.3: Number of SKUs by Seasonal Properties using the Friedman-Chi-Squared-test, Trend First.

	No Seasonality	**Monthly Seasonality**	**Total**
No Trend	1598	921	2519
Trend (Trend First)	3363	3537	6900
Total	4961	4458	9419

Table M.4: Number of SKUs by Seasonal Properties using the Friedman-Chi-Squared-test, Trend First.

	No Seasonality	**Monthly Seasonality**	**Total**
No Trend	1601	1862	3463
Trend (Trend First)	1844	4112	5956
Total	3445	5974	9419

N SKU Classification using the F-test, Trend First Approach for Multiple Levels of Seasonality

Table N.1: Number of SKUs by Seasonal Properties using the F-test.

	No Seasonality	Daily Only	Weekly Only	Monthly Only	Daily + Weekly	Daily + Monthly	Weekly + Monthly	Daily, Weekly, and Monthly	Total
No Trend	894	1283	24	5	140	25	17	131	2519
Trend (Trend First)	3071	3101	67	15	294	40	33	279	6900
Total	3965	4384	91	20	434	65	50	410	9419

O Performance of Nested Exponential Smoothing Compared to the Current Method

Now, we look at each of our SKU classes except for Classes 1, 4, 9, and 12. Classes 1 and 9 do not have their models changed, and although classes 4 and 12 have techinically had their models changed, we note that this change has no effect as the changes made to the models only affect daily and weekly seasonalities, which are not present in SKUs within these classes.

Class 2: Daily seasonal SKUs without a trend. (1 283 items)

Table O.1: Effects of changing from SES to TES Nested

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	SES	**0.183**	**77.54**	**0.136**	**-0.036**
	TES Nested	0.396	91.93	0.304	-0.514
p-value	-	**0.0**	**0.0**	**0.0**	**0.0**
Medium	SES	**0.26**	**49.33**	**0.195**	**-0.045**
	TES Nested	0.809	72.12	0.667	-0.544
p-value	-	**0.0**	**0.0**	**0.0**	**0.0**
Long	SES	**0.542**	**33.2**	**0.423**	**-0.056**
	TES Nested	2.346	59.79	1.952	-0.557
p-value	-	**0.0**	**0.0**	**0.0**	**0.0**

We see that, for this class of SKUs, SES outperforms nested triple exponential smoothing by a significant margin based on all of our performance metrics. SES is the best model here.

Class 3: Weekly seasonal SKUs without a trend. (24 items)

Table O.2: Effects of changing from SES to TES Nested

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	SES	**0.182**	**98.41**	**0.135**	**-0.045**
	TES Nested	0.698	111.43	0.43	-0.546
p-value	-	**0.0**	0.35	**0.0**	**0.0**
Medium	SES	**0.3**	**70.82**	**0.234**	**-0.022**
	TES Nested	1.48	93.7	0.951	-0.592
p-value	-	**0.0**	0.09	**0.0**	**0.001**
Long	SES	**0.706**	**52.52**	**0.578**	**-0.031**
	TES Nested	4.191	84.8	2.859	-0.638
p-value	-	**0.0**	**0.01**	**0.0**	**0.003**

For this class, SES outperforms nested TES by a significant margin on all metrics except short and medium term sMAPE, where SES is still better but the difference is not statistically significant. We conclude SES is our best model here as well.

Class 5: Daily-weekly SKUs without a trend. (140 items)

Table O.3: Effects of changing from SES to TES Nested

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	SES	**0.204**	**72.11**	**0.157**	**-0.065**
	TES Nested	1.299	89.34	0.782	-0.693
p-value	-	**0.001**	**0.0**	**0.0**	**0.0**
Medium	SES	**0.273**	**45.38**	**0.211**	**-0.095**
	TES Nested	2.497	76.89	1.785	-0.744
p-value	-	**0.0**	**0.0**	**0.0**	**0.0**
Long	SES	**0.606**	**31.42**	**0.486**	**-0.111**
	TES Nested	7.274	70.66	5.524	-0.776
p-value	-	**0.0**	**0.0**	**0.0**	**0.0**

For Daily-weekly SKUs without a trend, SES outperforms nested TES by a significant margin on all performance metrics for all cover periods. SES is the superior model.

Class 6: Daily-monthly SKUs without a trend. (25 items)

Table O.4: Effects of changing from TES to TES Nested

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES	**0.175**	**94.4**	**0.13**	**-0.031**
	TES Nested	0.335	106.35	0.244	-0.367
p-value	-	**0.033**	0.28	**0.028**	**0.0**
Medium	TES	**0.261**	**58.45**	**0.195**	**-0.051**
	TES Nested	0.616	79.78	0.505	-0.412
p-value	-	**0.011**	0.05	**0.011**	**0.0**
Long	TES	**0.568**	**39.86**	**0.44**	**0.016**
	TES Nested	1.742	64.02	1.434	-0.379
p-value	-	**0.008**	**0.02**	**0.01**	**0.0**

Here, we note that TES is a better model than nested TES based on all performance metrics except for short and medium term sMAPE, where the difference found is not statistically significant.

Class 7: Weekly-monthly seasonal SKUs without a trend. (17 items)

Table O.5: Effects of changing from TES to TES Nested

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES	**0.192**	**91.44**	**0.13**	**-0.208**
	TES Nested	2.156	114.6	1.353	-0.68
p-value	-	**0.013**	0.21	**0.014**	**0.0**
Medium	TES	**0.348**	**66.0**	**0.235**	**-0.259**
	TES Nested	4.879	103.34	3.162	-0.718
p-value	-	**0.012**	**0.04**	**0.014**	**0.001**
Long	TES	**0.861**	**51.56**	**0.607**	**-0.265**
	TES Nested	1.457E1	99.77	9.93	-0.803
p-value	-	**0.012**	**0.01**	**0.013**	**0.0**

Once again, we see that nested TES gets significantly outperformed by regular TES on all metrics except for short term sMAPE, where the difference is not statistically significant.

Class 8: Daily-weekly-monthly seasonal SKUs without a trend. (131 items)

Table O.6: Effects of changing from TES to TES Nested

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES	**0.214**	**73.22**	**0.158**	**-0.117**
	TES Nested	1.575E1	91.61	6.184	-0.713
p-value	-	0.065	**0.0**	0.065	**0.0**
Medium	TES	**0.34**	**50.45**	**0.248**	**-0.132**
	TES Nested	3.179E1	79.84	1.462E1	-0.767
p-value	-	0.06	**0.0**	0.064	**0.0**
Long	TES	**0.863**	**38.98**	**0.639**	**-0.175**
	TES Nested	8.163E1	74.08	4.617E1	-0.786
p-value	-	0.06	**0.0**	0.069	**0.0**

For daily-weekly-monthly seasonal SKUs, TES significantly outperforms nested TES by a significant margin based on both sMAPE and Bias.

It is interesting to note that, although TES has better scores on both RMSE and MAD by several orders of magnitude, the difference between the two models is not statistically significant. This is likely because this difference is not a structural difference. Instead, most of this effect originates from a handful of outliers, which perform so poorly that overall performance on RMSE and MAD is tens to hundreds of times worse than the current method. We will see this phenomenon occur for all models that include multiple levels of seasonality, with the error exponentially compounding as more levels of seasonality come into play. We attempt to solve this in the next section, which is Section 6.5.

Class 10: Daily seasonal SKUs with a trend. (3 101 items)

Table O.7: Effects of changing from DES to TES Trending Nested

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	DES	**0.195**	**67.65**	**0.148**	**0.016**
	TES Trending Nested	0.537	88.76	0.415	-0.34
p-value	-	**0.0**	**0.0**	**0.0**	**0.0**
Medium	DES	**0.283**	**45.21**	**0.217**	**0.006**
	TES Trending Nested	1.152	76.27	0.939	-0.34
p-value	-	**0.0**	**0.0**	**0.0**	**0.0**
Long	DES	**0.652**	**34.18**	**0.519**	**-0.013**
	TES Trending Nested	3.504	69.83	2.865	-0.32
p-value	-	**0.0**	**0.0**	**0.0**	**0.0**

For this class of SKUs, we see that DES is significantly better than nested TES with a trend all across the board.

Class 11: Weekly seasonal SKUs with a trend. (67 items)

Table O.8: Effects of changing from DES to TES Trending Nested

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	DES	**0.183**	**83.5**	**0.135**	**0.073**
	TES Trending Nested	0.779	97.4	0.477	-0.212
p-value	-	**0.0**	0.1	**0.0**	**0.028**
Medium	DES	**0.316**	**64.13**	**0.248**	**0.089**
	TES Trending Nested	1.712	83.54	1.078	-0.221
p-value	-	**0.0**	**0.02**	**0.0**	0.081
Long	DES	**0.795**	**53.91**	**0.664**	**0.079**
	TES Trending Nested	5.111	77.58	3.357	-0.217
p-value	-	**0.0**	**0.0**	**0.0**	0.096

For this class of SKUs, DES is significantly better than nested TES with a trend for all cover periods.

Class 13: Daily-weekly seasonal SKUs with a trend. (294 items)

Table O.9: Effects of changing from DES to TES Trending Nested

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	DES	**0.219**	**61.78**	**0.17**	**0.046**
	TES Trending Nested	8.139E11	88.08	4.57E11	-0.382
p-value	-	0.318	**0.0**	0.318	**0.0**
Medium	DES	**0.317**	**42.49**	**0.245**	**0.045**
	TES Trending Nested	1.27E12	80.31	1.10E12	-0.379
p-value	-	0.318	**0.0**	0.318	**0.0**
Long	DES	**0.766**	**34.68**	**0.616**	**0.043**
	TES Trending Nested	3.828E12	77.29	3.267E12	-0.354
p-value	-	0.318	**0.0**	0.318	**0.0**

Again, DES outperforms nested TES by a significant margin based on both sMAPE and bias.

Class 14: Daily-monthly SKUs with a trend. (40 items)

Table O.10: Effects of changing from TES Trending to TES Trending Nested

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES Trending	**0.186**	**90.73**	**0.136**	**-0.076**
	TES Trending Nested	0.577	110.44	0.417	-0.233
p-value	-	**0.016**	**0.03**	**0.025**	**0.0**
Medium	TES Trending	**0.307**	**58.25**	**0.223**	**-0.106**
	TES Trending Nested	1.203	89.74	0.951	-0.228
p-value	-	**0.009**	**0.0**	**0.015**	**0.0**
Long	TES Trending	**0.764**	**40.81**	**0.543**	**-0.069**
	TES Trending Nested	3.602	78.66	2.864	-0.208
p-value	-	**0.005**	**0.0**	**0.009**	**0.0**

For this class of SKUs, we see that regular TES outperforms nested TES by a significant margin based on all of our performance metrics.

Class 15: Weekly-monthly SKUs with a trend. (33 items)

Table O.11: Effects of changing from TES Trending to TES Trending Nested

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES Trending	**0.225**	**91.07**	**0.154**	**-0.102**
	TES Trending Nested	2.867E1	106.75	1.417E1	-0.437
p-value	-	0.273	0.19	0.279	**0.003**
Medium	TES Trending	**0.427**	**67.37**	**0.293**	**-0.14**
	TES Trending Nested	6.530E1	91.69	3.363E1	-0.494
p-value	-	0.275	**0.05**	0.279	**0.006**
Long	TES Trending	**1.163**	**50.99**	**0.799**	**-0.119**
	TES Trending Nested	1.954E2	83.32	1.097E2	-0.498
p-value	-	0.276	**0.01**	0.28	0.068

For weekly-monthly SKUs with a trend, we see that the current method has better performance all across the board. Both medium and long cover times have significantly better sMAPE, and short and medium cover times have a significantly better bias. Therefore, we conclude the current method is the superior method.

Class 16: Daily-weekly-monthly SKUs with a trend. (279 items)

Table O.12: Effects of changing from TES Trending to TES Trending Nested

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES Trending	**0.22**	**76.22**	**0.157**	**-0.052**
	TES Trending Nested	2.497E12	98.77	1.426E12	-0.485
p-value	-	0.318	**0.0**	0.318	**0.0**
Medium	TES Trending	**0.388**	**53.42**	**0.274**	**-0.077**
	TES Trending Nested	3.853E12	87.91	3.359E12	-0.497
p-value	-	0.318	**0.0**	0.318	**0.0**
Long	TES Trending	**1.06**	**42.91**	**0.754**	**-0.086**
	TES Trending Nested	1.162E13	83.27	1.016E13	-0.507
p-value	-	0.318	**0.0**	0.318	**0.0**

For our last class of SKUs, being the SKUs with a trend and all levels of seasonality, we see that the current method is significantly better across all performance horizons based on both sMAPE and Bias. Despite the current method also being better by several trillions of orders of magnitude based on RMSE and MAD, the difference is not statistically significant. Again, this is likely because this difference is not a structural difference. Instead, this difference is caused by a handful of outliers.

P Performance of Adjusted Nested Exponential Smoothing Compared to the Current Method

Class 2: Daily seasonal SKUs without a trend. (1 283 items)

Table P.1: Effects of changing from SES to TES Nested Adjusted

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	SES	0.183	77.54	0.136	**-0.036**
	TES Nested Adjusted	**0.162**	**74.2**	**0.114**	-0.037
p-value	-	**0.0**	0.07	**0.0**	**0.0**
Medium	SES	0.26	49.33	0.195	-0.045
	TES Nested Adjusted	0.26	49.33	0.195	-0.045
p-value	-	1.0	1.0	1.0	1.0
Long	SES	0.542	33.2	0.423	-0.056
	TES Nested Adjusted	0.542	33.2	0.423	-0.056
p-value	-	1.0	1.0	1.0	1.0

The class of daily seasonal SKUs without a trend is an interesting class as this class only has its model adjusted for cover periods that have a number of days that is not a multiple of seven (the number of days in a week). In our research, this is the case for our short cover period only. For this period, we notice that the modifications we made to the nested exponential smoothing model actually result in the new model being better than the current model by a significant margin based on RMSE, MAD, and Bias.

We conclude nested triple exponential smoothing is better than simple exponential smoothing for this class of items, at least for our short cover time. For our medium and long cover times, performance is identical.

Class 3: Weekly seasonal SKUs without a trend. (24 items)

Table P.2: Effects of changing from SES to TES Nested Adjusted

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	SES	**0.182**	**98.41**	**0.135**	**-0.045**
	TES Nested Adjusted	0.21	102.1	0.148	0.178
p-value	-	0.241	0.79	0.504	**0.0**
Medium	SES	**0.3**	**70.82**	**0.234**	**-0.022**
	TES Nested Adjusted	0.375	75.58	0.277	0.248
p-value	-	0.126	0.7	0.27	**0.001**
Long	SES	**0.706**	**52.52**	**0.578**	**-0.031**
	TES Nested Adjusted	0.895	56.91	0.69	0.279
p-value	-	0.197	0.7	0.346	**0.003**

For weekly seasonal SKUs without a trend, we note that SES outperforms nested TES all across the board, although this difference is only statistically significant when looking at bias.

Class 4: Monthly seasonal SKUs without a trend. (5 items)

Table P.3: Effects of changing from TES to TES Nested Adjusted

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES	0.125	**123.39**	0.08	**0.075**
	TES Nested Adjusted	**0.117**	125.01	**0.069**	0.345
p-value	-	0.827	0.9	0.392	0.758
Medium	TES	0.234	71.99	0.142	**0.036**
	TES Nested Adjusted	**0.207**	**71.38**	**0.123**	0.426
p-value	-	0.75	0.97	0.361	0.663
Long	TES	0.603	48.35	0.355	**0.091**
	TES Nested Adjusted	**0.489**	**44.3**	**0.286**	0.628
p-value	-	0.644	0.67	0.368	0.158

Due to our adjustment to the way updating of our level works, we actually see the nested model perform differently from the current model, even though the nested model only uses monthly seasonality here. We see that, with the exception of short term sMAPE, this adjustment improves accuracy of the model at the cost of bias. This improvement is not statistically significant, however.

Class 5: Daily-weekly SKUs without a trend. (140 items)

Table P.4: Effects of changing from SES to TES Nested Adjusted

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	SES	0.204	72.11	0.157	**-0.065**
	TES Nested Adjusted	**0.19**	**69.63**	**0.136**	0.076
p-value	-	0.144	0.67	**0.006**	**0.0**
Medium	SES	**0.273**	**45.38**	**0.211**	-0.095
	TES Nested Adjusted	0.319	47.68	0.243	**0.007**
p-value	-	**0.009**	0.63	**0.022**	**0.0**
Long	SES	**0.606**	**31.42**	**0.486**	-0.111
	TES Nested Adjusted	0.723	33.15	0.566	**0.108**
p-value	-	**0.027**	0.65	0.056	**0.006**

For the daily-weekly seasonal SKUs that do not have a trend, we notice that the nested TES model is significantly better than the current method for short cover times based on MAD. Difference in RMSE and sMAPE is not statistically significant.

For medium cover periods, we see that SES is still the better method based on RMSE, MAD, and Bias. For the long term, our conclusion is the same, though it is only based on RMSE and Bias.

Class 6: Daily-monthly SKUs without a trend. (25 items)

Table P.5: Effects of changing from TES to TES Nested Adjusted

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES	0.175	94.4	0.13	**-0.031**
	TES Nested Adjusted	**0.169**	**93.43**	**0.116**	0.141
p-value	-	0.682	0.93	0.338	**0.023**
Medium	TES	0.261	**58.45**	0.195	**-0.051**
	TES Nested Adjusted	**0.256**	58.49	**0.19**	0.174
p-value	-	0.859	1.0	0.834	0.185
Long	TES	0.568	39.86	0.44	**0.016**
	TES Nested Adjusted	**0.528**	**39.0**	**0.41**	0.29
p-value	-	0.573	0.93	0.585	0.08

Here, we conclude the current method is superior to nested TES for the short term, based on Bias. Although the adjusted nested TES model outperforms the current method based on some measures, the difference is not statistically significant. This makes the current method better for the conservative approach, and the alternative method is better for the progressive approach.

Class 7: Weekly-monthly seasonal SKUs without a trend. (17 items)

Table P.6: Effects of changing from TES to TES Nested Adjusted

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES	**0.192**	**91.44**	**0.13**	-0.208
	TES Nested Adjusted	0.244	96.6	0.157	**0.129**
p-value	-	0.201	0.76	0.315	0.6
Medium	TES	**0.348**	**66.0**	**0.235**	-0.259
	TES Nested Adjusted	0.456	72.19	0.299	**0.173**
p-value	-	0.199	0.68	0.252	0.55
Long	TES	**0.861**	**51.56**	**0.607**	-0.265
	TES Nested Adjusted	1.072	56.36	0.754	**0.229**
p-value	-	0.374	0.73	0.363	0.798

For this class of SKUs, the current method is scores better than the alternative method on all of our accuracy measures, but the difference is not statistically significant.

Class 8: Daily-weekly-monthly seasonal SKUs without a trend. (131 items)

Table P.7: Effects of changing from TES to TES Nested Adjusted

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES	**0.214**	73.22	0.158	**-0.117**
	TES Nested Adjusted	0.234	**72.82**	**0.156**	0.125
p-value	-	0.168	0.94	0.911	**0.01**
Medium	TES	**0.34**	**50.45**	**0.248**	**-0.132**
	TES Nested Adjusted	0.437	54.37	0.31	0.16
p-value	-	**0.002**	0.45	**0.007**	0.077
Long	TES	**0.863**	**38.98**	**0.639**	**-0.175**
	TES Nested Adjusted	1.07	42.62	0.793	0.179
p-value	-	**0.026**	0.45	**0.029**	0.945

For this class of SKUs, we note that the current method and the alternative method perform similarly for the short period, as differences for our accuracy measures are not statistically significant. For medium and long cover times, the current method is more performant by a significant margin based on RMSE and MAD.

Class 10: Daily seasonal SKUs with a trend. (3 101 items)

Table P.8: Effects of changing from DES to TES Trending Nested Adjusted

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	DES	0.195	67.65	0.148	0.016
	TES Trending Nested Adjusted	**0.17**	**63.51**	**0.123**	**0.011**
p-value	-	**0.0**	**0.0**	**0.0**	**0.0**
Medium	DES	0.283	45.21	0.217	0.006
	TES Trending Nested Adjusted	**0.283**	**45.21**	**0.217**	**0.006**
p-value	-	1.0	1.0	1.0	1.0
Long	DES	0.652	34.18	0.519	**-0.013**
	TES Trending Nested Adjusted	**0.652**	**34.18**	**0.519**	-0.013
p-value	-	1.0	1.0	1.0	1.0

For our daily seasonal SKUs with a trend, we notice that adjusted nested triple exponential smoothing is better than double exponential smoothing by a significant margin for the short term. For medium and long cover times, performance is identical. Again, this is because, for cover times that have a length that is an integer multiple of seven days, daily seasonality is not relevant.

Class 11: Weekly seasonal SKUs with a trend. (67 items)

Table P.9: Effects of changing from DES to TES Trending Nested Adjusted

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	DES	**0.183**	**83.5**	**0.135**	**0.073**
	TES Trending Nested Adjusted	0.236	87.37	0.165	0.258
p-value	-	**0.003**	0.63	**0.029**	**0.001**
Medium	DES	**0.316**	**64.13**	**0.248**	**0.089**
	TES Trending Nested Adjusted	0.451	69.01	0.325	0.304
p-value	-	**0.0**	0.52	**0.009**	**0.001**
Long	DES	**0.795**	**53.91**	**0.664**	**0.079**
	TES Trending Nested Adjusted	1.161	57.77	0.873	0.331
p-value	-	**0.001**	0.6	**0.02**	**0.002**

For our weekly seasonal SKUs with a trend, DES remains the better model for all of our cover times.

Class 12: Monthly seasonal SKUs with a trend. (15 items)

Table P.10: Effects of changing from TES Trending to TES Trending Nested Adjusted

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES Trending	0.175	**114.7**	0.119	**-0.043**
	TES Trending Nested Adjusted	**0.157**	116.05	**0.104**	0.272
p-value	-	0.526	0.93	0.494	0.236
Medium	TES Trending	0.321	**82.45**	0.223	**-0.02**
	TES Trending Nested Adjusted	**0.269**	83.12	**0.192**	0.365
p-value	-	0.386	0.96	0.459	0.147
Long	TES Trending	0.82	61.27	0.582	**0.025**
	TES Trending Nested Adjusted	**0.614**	**59.71**	**0.459**	0.439
p-value	-	0.272	0.89	0.324	0.106

Similar to what we saw for monthly seasonal SKUs without a trend, adjustment of the level updating procedure has a positive though insignificant effect on RMSE and MAD at the cost of bias. Despite being marginally better based on RMSE and MAD, the difference is not significant. This leads us to believe that the current method is the better method for the conservative approach, and the alternative method is better for the progressive approach.

Class 13: Daily-weekly seasonal SKUs with a trend. (294 items)

Table P.11: Effects of changing from DES to TES Trending Nested Adjusted

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	DES	0.219	61.78	0.17	**0.046**
	TES Trending Nested Adjusted	**0.213**	**58.91**	**0.155**	0.183
p-value	-	0.443	0.44	**0.022**	**0.0**
Medium	DES	**0.317**	**42.49**	**0.245**	**0.045**
	TES Trending Nested Adjusted	0.381	44.88	0.29	0.207
p-value	-	**0.0**	0.5	**0.0**	**0.0**
Long	DES	**0.766**	**34.68**	**0.616**	**0.043**
	TES Trending Nested Adjusted	0.941	36.88	0.736	0.234
p-value	-	**0.0**	0.51	**0.002**	**0.0**

For our daily-weekly seasonal SKUs with a trend, we see that the adjusted version of nested triple exponential smoothing actually outperforms double exponential smoothing on the short term. This conclusion is based on MAD. For medium and long cover times, the current method is still the best approach.

We conclude that nested TES is better on the short term, and double exponential smoothing is better for our medium and long cover periods.

Class 14: Daily-monthly seasonal SKUs with a trend. (40 items)

Table P.12: Effects of changing from TES Trending to TES Trending Nested Adjusted

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES Trending	0.186	90.73	0.136	**-0.076**
	TES Trending Nested Adjusted	**0.159**	**88.12**	**0.113**	0.107
p-value	-	0.119	0.78	0.067	0.16
Medium	TES Trending	0.307	58.25	0.223	**-0.106**
	TES Trending Nested Adjusted	**0.26**	**56.47**	**0.197**	0.122
p-value	-	0.2	0.81	0.271	0.263
Long	TES Trending	0.764	40.81	0.543	**-0.069**
	TES Trending Nested Adjusted	**0.58**	**37.84**	**0.447**	0.191
p-value	-	0.111	0.56	0.161	0.107

For our daily-monthly seasonal SKUs with a trend, adjusted nested TES is better than the current method, though the difference is not statistically significant. The differences in performance are not statistically significant for any of our cover times. Looking at the table, we see that the current method is better under our conservative ideology, whilst the alternative method is better for our progressive ideology.

Class 15: Weekly-monthly seasonal SKUs with a trend. (33 items)

Table P.13: Effects of changing from TES Trending to TES Trending Nested Adjusted

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES Trending	**0.225**	**91.07**	**0.154**	**-0.102**
	TES Trending Nested Adjusted	0.254	95.38	0.167	0.212
p-value	-	0.429	0.68	0.623	0.108
Medium	TES Trending	**0.427**	**67.37**	**0.293**	**-0.14**
	TES Trending Nested Adjusted	0.483	73.32	0.334	0.244
p-value	-	0.497	0.53	0.483	0.181
Long	TES Trending	**1.163**	**50.99**	**0.799**	**-0.119**
	TES Trending Nested Adjusted	1.198	55.53	0.873	0.344
p-value	-	0.888	0.55	0.674	0.265

For this class of SKUs, we find the current method is the better method based on all of our performance metrics, though the difference is not statistically significant for any of them.

Class 16: Daily-weekly-monthly SKUs with a trend. (279 items)

Table P.14: Effects of changing from TES Trending to TES Trending Nested Adjusted

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES Trending	**0.22**	**76.22**	**0.157**	**-0.052**
	TES Trending Nested Adjusted	0.243	76.89	0.16	0.292
p-value	-	**0.03**	0.85	0.691	**0.0**
Medium	TES Trending	**0.388**	**53.42**	**0.274**	**-0.077**
	TES Trending Nested Adjusted	0.467	59.37	0.328	0.334
p-value	-	**0.001**	0.07	**0.001**	**0.0**
Long	TES Trending	**1.06**	**42.91**	**0.754**	**-0.086**
	TES Trending Nested Adjusted	1.203	48.6	0.877	0.366
p-value	-	**0.045**	0.06	**0.013**	**0.0**

For our final class of SKUs, we see that the current method is significantly better than the alternative method based on RMSE and Bias. For medium and long cover times, MAD is also significantly better.

Q Performance of Fourier Processed Exponential Smoothing Compared to the Current Method

Class 4: Monthly seasonal SKUs without trend. (5 items)

Table Q.1: Effects of changing from TES to TES Fourier

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES	**0.125**	123.39	**0.08**	0.075
	TES Fourier	0.183	**122.71**	0.087	**-0.03**
p-value	-	0.398	0.96	0.679	0.672
Medium	TES	**0.234**	71.99	**0.142**	**0.036**
	TES Fourier	0.379	**70.87**	0.162	-0.071
p-value	-	0.401	0.94	0.591	0.673
Long	TES	0.603	48.35	0.355	0.091
	TES Fourier	**0.554**	**46.57**	**0.327**	**0.057**
p-value	-	0.839	0.86	0.747	0.871

For monthly seasonal SKUs without a trend, we find no statistically significant differences between the two models. Being a class with just 5 items, we need more data to come to a conclusion.

Class 6: Daily-monthly SKUs without trend. (25 items)

Table Q.2: Effects of changing from TES to TES Fourier

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES	0.175	94.4	0.13	**-0.031**
	TES Fourier	**0.17**	**93.61**	**0.126**	-0.055
p-value	-	0.745	0.94	0.821	0.736
Medium	TES	0.261	58.45	0.195	**-0.051**
	TES Fourier	**0.244**	**56.82**	**0.183**	-0.08
p-value	-	0.476	0.88	0.542	0.587
Long	TES	0.568	39.86	0.44	**0.016**
	TES Fourier	**0.498**	**37.5**	**0.388**	-0.046
p-value	-	0.301	0.8	0.308	0.577

For daily-monthly seasonal SKUs without a trend, we do not find a statistically significant differences between the two models. Being a class with just 25 items, we need more data to find which of these models performs better.

Class 7: Weekly-monthly SKUs without trend. (17 items)

Table Q.3: Effects of changing from TES to TES Fourier

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES	0.192	**91.44**	0.13	-0.208
	TES Fourier	**0.182**	92.09	**0.129**	**-0.119**
p-value	-	0.69	0.97	0.969	0.476
Medium	TES	0.348	**66.0**	0.235	-0.259
	TES Fourier	**0.319**	67.21	**0.231**	**-0.14**
p-value	-	0.582	0.94	0.908	0.487
Long	TES	0.861	**51.56**	0.607	-0.265
	TES Fourier	**0.75**	53.14	**0.575**	**-0.202**
p-value	-	0.509	0.91	0.792	0.56

Similar to SKUs of class 4 and 6, this class of SKUs is very small with just 17 items. We do not find any significant differences between the two models.

Class 8: Daily-weekly-monthly SKUs without trend. (131 items)

Table Q.4: Effects of changing from TES to TES Fourier

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES	0.214	73.22	0.158	-0.117
	TES Fourier	**0.202**	**73.05**	**0.151**	**-0.045**
p-value	-	0.302	0.98	0.493	0.43
Medium	TES	0.34	50.45	0.248	-0.132
	TES Fourier	**0.304**	**50.44**	**0.227**	**-0.034**
p-value	-	0.072	1.0	0.153	0.348
Long	TES	0.863	**38.98**	0.639	-0.175
	TES Fourier	**0.73**	39.37	**0.56**	**-0.085**
p-value	-	**0.035**	0.94	0.094	**0.049**

We see the Fourier modified variant of the model outperform the current model, but this change is only significant for long-term RMSE and bias. We conclude the modified version is the better model for the long term, with us needing more data to validate whether this claim also holds for the short and medium cover periods.

Class 12: Monthly SKUs with trend. (15 items)

Table Q.5: Effects of changing from TES Trending to TES Trending Fourier

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES Trending	0.175	114.7	0.119	**-0.043**
	TES Trending Fourier	**0.159**	**113.6**	**0.108**	0.078
p-value	-	0.583	0.94	0.647	0.562
Medium	TES Trending	0.321	82.45	0.223	**-0.02**
	TES Trending Fourier	**0.276**	**81.09**	**0.199**	0.147
p-value	-	0.466	0.93	0.582	0.765
Long	TES Trending	0.82	61.27	0.582	**0.025**
	TES Trending Fourier	**0.664**	**60.49**	**0.489**	0.219
p-value	-	0.428	0.95	0.47	0.907

This class of items is very small at just 15 items, providing us with insufficient evidence to conclude which of the two models is better.

Class 14: Daily-monthly SKUs with a trend. (40 items)

Table Q.6: Effects of changing from TES Trending to TES Trending Fourier

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES Trending	0.186	90.73	0.136	-0.076
	TES Trending Fourier	**0.175**	**90.13**	**0.13**	**-0.009**
p-value	-	0.523	0.95	0.639	0.683
Medium	TES Trending	0.307	58.25	0.223	-0.106
	TES Trending Fourier	**0.275**	**56.8**	**0.206**	**-0.028**
p-value	-	0.412	0.85	0.496	0.806
Long	TES Trending	0.764	40.81	0.543	-0.069
	TES Trending Fourier	**0.65**	**38.73**	**0.481**	**-0.003**
p-value	-	0.365	0.71	0.394	0.841

We see the Fourier variant of the model outperform traditional TES all over the board, but changes are not significant.

Class 15: Weekly-monthly SKUs with a trend. (33 items)

Table Q.7: Effects of changing from TES Trending to TES Trending Fourier

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES Trending	0.225	**91.07**	0.154	**-0.102**
	TES Trending Fourier	**0.199**	91.41	**0.14**	0.161
p-value	-	0.296	0.97	0.444	0.82
Medium	TES Trending	0.427	**67.37**	0.293	**-0.14**
	TES Trending Fourier	**0.353**	68.0	**0.257**	0.179
p-value	-	0.175	0.95	0.336	0.723
Long	TES Trending	1.163	**50.99**	0.799	**-0.119**
	TES Trending Fourier	**0.916**	52.48	**0.686**	0.237
p-value	-	0.171	0.85	0.369	0.324

The Fourier variant of the model scores better on both RMSE and MAD for all forecasting horizons. The traditional variant scores better on sMAPE. None of these differences are statistically significant. Although bias of the traditional model is better, this difference is not statistically significant. This leads to the current model being better for the conservative approach, and the alternative model being better for the progressive approach.

Class 16: Daily-weekly-monthly SKUs with a trend. (279 items)

Table Q.8: Effects of changing from TES Trending to TES Trending Fourier

Horizon	**Model**	**RMSE**	**sMAPE**	**MAD**	**Bias**
Short	TES Trending	0.22	**76.22**	0.157	**-0.052**
	TES Trending Fourier	**0.202**	77.1	**0.148**	0.177
p-value	-	**0.04**	0.8	0.201	**0.013**
Medium	TES Trending	0.388	**53.42**	0.274	**-0.077**
	TES Trending Fourier	**0.338**	55.39	**0.253**	0.21
p-value	-	**0.008**	0.55	0.101	0.106
Long	TES Trending	1.06	**42.91**	0.754	**-0.086**
	TES Trending Fourier	**0.878**	45.59	**0.678**	0.211
p-value	-	**0.003**	0.38	0.069	0.531

For our last class of SKUs, we see that RMSE of the Fourier adaptation of the model is significantly lower, whilst sMAPE and MAD are not significantly different when compared to the traditional model. We conclude that the adapted model is an improvement over the traditional model.

R Performance of Fourier Processed Nested Exponential Smoothing Compared to the Current Method

Class 4: Monthly seasonal SKUs without a trend. (5 items)

Table R.1: Effects of changing from TES to TES Nested Fourier

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES	0.125	123.39	0.08	**0.075**
	TES Nested Fourier	**0.114**	**122.97**	**0.067**	0.277
p-value	-	0.751	0.97	0.286	1.0
Medium	TES	0.234	71.99	0.142	**0.036**
	TES Nested Fourier	**0.195**	**68.66**	**0.114**	0.335
p-value	-	0.647	0.83	0.187	0.983
Long	TES	0.603	48.35	0.355	**0.091**
	TES Nested Fourier	**0.444**	**38.95**	**0.248**	0.537
p-value	-	0.534	0.32	0.179	0.369

For this class of SKUs, we see the nested Fourier adaptation outperform the currently used model, the nested model, and the monthly Fourier model based on accuracy, though the difference is never statistically significant. We note that the bias of the nested Fourier version is worse than the currently used model, though this difference is also insignificant. We suspect the increase in bias mostly originates from the change in the updating method as applied in Section 6.5, as the increased bias was present there as well. If anything, the Fourier adaptation reduced the magnitude of this effect, but as none of the differences that are present are statistically significant, we are not able to jump to a sound conclusion.

Class 6: Daily-monthly SKUs without a trend. (25 items)

Table R.2: Effects of changing from TES to TES Nested Fourier

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES	0.175	94.4	0.13	**-0.031**
	TES Nested Fourier	**0.162**	**92.17**	**0.112**	0.058
p-value	-	0.388	0.85	0.216	0.42
Medium	TES	0.261	58.45	0.195	**-0.051**
	TES Nested Fourier	**0.237**	**56.21**	**0.177**	0.074
p-value	-	0.326	0.84	0.362	0.883
Long	TES	0.568	39.86	0.44	**0.016**
	TES Nested Fourier	**0.461**	**36.24**	**0.362**	0.156
p-value	-	0.106	0.7	0.126	0.442

For the daily-monthly SKUs without a trend, we see that performance based on accuracy also benefits from changes introduced by the new model, but the difference is not statistically significant anywhere. Like in our previous class, the nested Fourier version performs better than both nested TES and Fourier TES, but the difference is not statistically significant.

Class 7: Weekly-monthly seasonal SKUs without a trend. (17 items)

Table R.3: Effects of changing from TES to TES Nested Fourier

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES	**0.192**	**91.44**	**0.13**	-0.208
	TES Nested Fourier	0.214	95.9	0.146	**0.023**
p-value	-	0.494	0.79	0.502	0.484
Medium	TES	**0.348**	**66.0**	**0.235**	-0.259
	TES Nested Fourier	0.385	70.96	0.271	**0.047**
p-value	-	0.577	0.74	0.449	0.501
Long	TES	**0.861**	**51.56**	**0.607**	-0.265
	TES Nested Fourier	0.88	55.04	0.662	**0.041**
p-value	-	0.919	0.81	0.682	0.273

Interestingly, this class of SKUs shows higher accuracy for the current model, and better bias for the modified model. This is the complete opposite of what we saw before. As none of the modifications we made so far have resulted in a statistically significant difference, we cannot get to a conclusion on which model is better and why.

Class 8: Daily-weekly-monthly SKUs without a trend. (131 items)

Table R.4: Effects of changing from TES to TES Nested Fourier

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES	0.214	73.22	0.158	-0.117
	TES Nested Fourier	**0.199**	**70.6**	**0.136**	**0.052**
p-value	-	0.188	0.65	**0.019**	0.314
Medium	TES	**0.34**	**50.45**	**0.248**	-0.132
	TES Nested Fourier	0.353	51.62	0.257	**0.081**
p-value	-	0.566	0.82	0.623	0.819
Long	TES	0.863	**38.98**	0.639	-0.175
	TES Nested Fourier	**0.822**	39.49	**0.62**	**0.066**
p-value	-	0.553	0.92	0.708	0.109

This class of SKUs shows very interesting results. Starting off with our short cover periods, we previously saw neither nesting nor Fourier filtering was able to significantly perform the performance of the model. Now, when combining these two methods, we actually start to reap the benefits. Short term MAD was significantly improved, making the nested TES Fourier model the better model for short cover periods.

For medium cover periods, we see that nested TES Fourier is not better than the current model. We previously also saw that the Fourier adaptation was not significantly better than the current model. What is interesting is that, whereas the current model was previously significantly better than the nested model, this is no longer the case.

Although we previously saw the Fourier adaptation outperform the current method based on all metrics, with the difference in RMSE being statistically significant in case we have a significance level of $\alpha >= 0.072$, we could not conclude that the adaptation was actually beneficial for our dataset. Despite this, we see that this adaptation was able to undo some of the damage caused

by the inclusion of nested seasonal factors in the sense that this adaptation turned a statistically significant difference into a statistically insignificant difference, even though the sample population remained the same. This leads us to believe that, unless we later find an even better method, the Fourier adaptation to triple exponential smoothing is worth researching in more depth.

For our long cover periods, our conclusion is fairly similar, except the Fourier adaptation of the model was already proven to be better bases on our statistical analysis.

Class 12: Monthly seasonal SKUs with a trend. (15 items)

Table R.5: Effects of changing from TES Trending to TES Trending Nested Fourier

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES Trending	0.175	114.7	0.119	**-0.043**
	TES Trending Nested Fourier	**0.147**	**113.11**	**0.098**	0.184
p-value	-	0.327	0.92	0.342	0.932
Medium	TES Trending	0.321	82.45	0.223	**-0.02**
	TES Trending Nested Fourier	**0.242**	**79.26**	**0.174**	0.272
p-value	-	0.168	0.83	0.233	0.622
Long	TES Trending	0.82	61.27	0.582	**0.025**
	TES Trending Nested Fourier	**0.525**	**56.88**	**0.408**	0.335
p-value	-	0.102	0.7	0.145	0.671

For this class, we also find some interesting results. The original Fourier adapted version uses the updating of level as defined in literature, as neither Table P.3 nor Table P.10 proved the adjustment to be better based on our statistical analysis.

Looking purely at the numbers in this table, and comparing them to the Fourier-only adaptation (Table Q.5) as well as the adjusted nesting-only adaptation (Table P.10), we see that our accuracy-based performance is better based on numbers alone, even though the difference is not statistically significant. With p-values for the comparison of the combined model and the current model being much lower than the p-values found when combining the current model to either of the two adaptations alone, we suspect there is indeed a difference. Even though we are not able to prove this difference does indeed exist based on our data alone, we hypothesise evaluating the model on additional data may yield interesting improvements to our models.

Class 14: Daily-monthly SKUs with a trend. (40 items)

Table R.6: Effects of changing from TES Trending to TES Trending Nested Fourier

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES Trending	0.186	90.73	0.136	-0.076
	TES Trending Nested Fourier	**0.154**	**87.27**	**0.11**	**0.057**
p-value	-	0.055	0.71	**0.036**	0.871
Medium	TES Trending	0.307	58.25	0.223	-0.106
	TES Trending Nested Fourier	**0.245**	**55.09**	**0.187**	**0.065**
p-value	-	0.083	0.67	0.125	0.895
Long	TES Trending	0.764	40.81	0.543	**-0.069**
	TES Trending Nested Fourier	**0.531**	**35.55**	**0.409**	0.117
p-value	-	**0.038**	0.3	**0.042**	0.524

Like found when analysing the performance of this model on class 8 SKUs (daily-weekly-monthly seasonal SKUs without a trend), we find that our combined model produces significantly better results than the current model, even though neither adaptations of the model were significantly better by themselves. With our combined model being significantly better than the current model for both short and long cover periods, we conclude that, for the conservative approach, this model is better for short and long cover periods, whilst the current model is still better for medium cover periods. For the progressive approach, the adapted model is best for all of our cover periods.

We note that, as a class of just 40 SKUs, this class is fairly small. Despite this, differences for short and long cover periods are statistically significant. As the p-value for medium-term RMSE is just 0.083, we hypothesise this may also hold for our medium cover periods. Collection of a couple more samples may be able to back up this claim.

Class 15: Weekly-monthly seasonal SKUs with a trend. (33 items)

Table R.7: Effects of changing from TES Trending to TES Trending Nested Fourier

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES Trending	0.225	**91.07**	0.154	**-0.102**
	TES Trending Nested Fourier	**0.222**	94.19	**0.152**	0.149
p-value	-	0.899	0.76	0.896	0.715
Medium	TES Trending	0.427	**67.37**	0.293	**-0.14**
	TES Trending Nested Fourier	**0.401**	70.58	**0.287**	0.176
p-value	-	0.691	0.74	0.892	0.771
Long	TES Trending	1.163	**50.99**	0.799	**-0.119**
	TES Trending Nested Fourier	**0.976**	52.72	**0.735**	0.263
p-value	-	0.356	0.82	0.648	0.945

This class of SKUs does not see any significant benefits of adopting the combined model. In fact, performance appears to be worse than just the Fourier adaptation alone, even though gains made by this adaptation were also not statistically significant.

Class 16: Daily-weekly-monthly SKUs with a trend. (279 items)

Table R.8: Effects of changing from TES Trending to TES Trending Nested Fourier

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES Trending	0.22	76.22	0.157	**-0.052**
	TES Trending Nested Fourier	**0.208**	**75.77**	**0.144**	0.214
p-value	-	0.223	0.9	0.06	**0.0**
Medium	TES Trending	0.388	**53.42**	**0.274**	**-0.077**
	TES Trending Nested Fourier	**0.387**	57.95	0.287	0.249
p-value	-	0.958	0.16	0.368	**0.006**
Long	TES Trending	1.06	**42.91**	0.754	**-0.086**
	TES Trending Nested Fourier	**0.979**	47.51	**0.754**	0.253
p-value	-	0.195	0.12	0.99	0.823

With the Fourier adaptation outperforming the current model by a significant margin, we find that our numbers for RMSE are better across the board. Despite this, the difference is not statistically significant. We attribute this to the negative impact of the including of nesting, which we found

to be statistically inferior to the current method. For this class of SKUs, we conclude the Fourier adaptation without nesting is the best model.

S Performance of Prophet Compared to the Current Method

Class 1: Stationary SKUs. (894 items)

Table S.1: Effects of changing from SES to Prophet

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	SES	0.158	94.6	0.112	**-0.017**
	Prophet	**0.151**	**94.1**	**0.108**	-0.12
p-value	-	0.096	0.83	0.151	**0.0**
Medium	SES	**0.241**	**66.03**	**0.18**	**-0.011**
	Prophet	0.25	68.13	0.19	-0.151
p-value	-	0.166	0.32	**0.033**	**0.0**
Long	SES	**0.519**	**44.9**	**0.411**	**-0.011**
	Prophet	0.564	47.7	0.449	-0.209
p-value	-	**0.008**	0.1	**0.006**	**0.0**

For this class of SKUs, we notice that SES performs better on all of our performance metrics for both medium and long horizons. For the short horizon, Prophet outperforms SES on the metrics RMSE, sMAPE, and MAD, but the difference is not significant. SES scores a better bias on the short term, with the difference between SES and Prophet being statistically significant. We conclude that this class of SKUs is best forecasted using SES.

Class 2: Daily seasonal SKUs without a trend. (1 283 items)

Table S.2: Effects of changing from SES to Prophet

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	SES	0.183	77.54	0.136	**-0.036**
	Prophet	**0.166**	**74.93**	**0.12**	-0.111
p-value	-	**0.0**	0.16	**0.0**	**0.0**
Medium	SES	**0.26**	**49.33**	**0.195**	**-0.045**
	Prophet	0.27	50.55	0.206	-0.147
p-value	-	0.082	0.44	**0.005**	**0.0**
Long	SES	**0.542**	**33.2**	**0.423**	**-0.056**
	Prophet	0.586	34.89	0.461	-0.194
p-value	-	**0.002**	0.2	**0.0**	**0.0**

For daily seasonal SKUs without a trend, we see that Prophet performs significantly better on both RMSE and MAD for the short horizon, although we note that its bias is significantly worse than that of SES. Despite having a worse bias, we identify Prophet as the better model for short horizons as by the selection rules proposed in Section 4.2.1.

For both medium and long horizons, SES remains the best model as it performs better on all metrics.

Class 3: Weekly seasonal SKUs without a trend. (24 items)

Table S.3: Effects of changing from SES to Prophet

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	SES	0.182	98.41	0.135	**-0.045**
	Prophet	**0.169**	**97.42**	**0.124**	-0.058
p-value	-	0.497	0.94	0.499	0.57
Medium	SES	0.3	70.82	0.234	**-0.022**
	Prophet	**0.285**	**70.75**	**0.22**	-0.031
p-value	-	0.681	1.0	0.643	0.491
Long	SES	0.706	**52.52**	0.578	-0.031
	Prophet	**0.672**	54.16	**0.539**	**-0.025**
p-value	-	0.766	0.89	0.687	0.561

For this class of SKUs, we see Prophet outperform the current method by most measures, but none of the differences are significant. Either method will work fine here. This means the conservative approach will stick to SES, whilst the progressive approach changes to Prophet.

Class 4: Monthly seasonal SKUs without a trend. (5 items)

Table S.4: Effects of changing from TES to Prophet

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES	0.125	123.39	0.08	**0.075**
	Prophet	**0.11**	**121.04**	**0.065**	0.186
p-value	-	0.648	0.86	0.253	0.944
Medium	TES	0.234	71.99	0.142	**0.036**
	Prophet	**0.193**	**69.19**	**0.116**	0.178
p-value	-	0.615	0.85	0.222	0.933
Long	TES	0.603	48.35	0.355	**0.091**
	Prophet	**0.466**	**42.49**	**0.265**	0.274
p-value	-	0.586	0.56	0.259	0.557

We notice Prophet outperform the current method on all error metrics. We notice the current method scores better on bias. Despite Prophet having better scores by a wide margin, we cannot conclude the model is significantly better. This is likely because this class of SKUs consists of just five products. Again, either method will work fine here. Like before, the conservative approach will stick to the current method, whilst the progressive method will switch over.

Class 5: Daily-Weekly seasonal SKUs without a trend. (140 items)

Table S.5: Effects of changing from SES to Prophet

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	SES	0.204	72.11	0.157	**-0.065**
	Prophet	**0.168**	**67.24**	**0.124**	-0.134
p-value	-	**0.0**	0.39	**0.0**	**0.006**
Medium	SES	**0.273**	**45.38**	**0.211**	**-0.095**
	Prophet	0.274	45.52	0.213	-0.174
p-value	-	0.962	0.98	0.85	**0.026**
Long	SES	**0.606**	31.42	0.486	**-0.111**
	Prophet	0.609	**30.97**	**0.484**	-0.228
p-value	-	0.948	0.9	0.956	**0.002**

For our short horizon, we notice Prophet outperform SES by a significant margin based on RMSE and MAD. For medium horizons, the difference between the two models is not statistically significant. For long horizons, the difference in accuracy is not statistically significant, but SES has a significantly better bias. This makes Prophet the best method for short term, and SES remains the better method for medium and long term horizons.

Class 6: Daily-Monthly SKUs without a trend. (25 items)

Table S.6: Effects of changing from TES to Prophet

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES	0.175	94.4	0.13	**-0.031**
	Prophet	**0.159**	**91.37**	**0.113**	-0.06
p-value	-	0.283	0.79	0.214	0.068
Medium	TES	0.261	58.45	0.195	**-0.051**
	Prophet	**0.235**	**55.86**	**0.176**	-0.096
p-value	-	0.268	0.81	0.333	0.151
Long	TES	0.568	39.86	0.44	**0.016**
	Prophet	**0.458**	**36.27**	**0.359**	-0.091
p-value	-	0.089	0.7	0.104	0.085

We again see a very small class of SKUs, having just 25 items. This is not sufficient to conclude either model is better than the other. Despite being not significant, we notice that Prophet has better scores on its error metrics by a large margin.

Class 7: Weekly-Monthly SKUs without a trend. (17 items)

Table S.7: Effects of changing from TES to Prophet

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES	0.192	**91.44**	0.13	**-0.208**
	Prophet	**0.167**	91.57	**0.121**	-0.294
p-value	-	0.29	0.99	0.652	0.822
Medium	TES	0.348	**66.0**	0.235	**-0.259**
	Prophet	**0.293**	67.61	**0.224**	-0.36
p-value	-	0.285	0.92	0.775	0.891
Long	TES	0.861	**51.56**	0.607	**-0.265**
	Prophet	**0.685**	52.51	**0.558**	-0.511
p-value	-	0.289	0.95	0.696	0.384

This class has just 17 items, which again is very little to conclude whether either of our two models is better than the other. We notice TES is better based on sMAPE and bias, and Prophet scores better on RMSE and MAD. None of the differences are significant so we conclude either method is fine.

Class 8: Daily-Weekly-Monthly seasonal SKUs without a trend. (131 items)

Table S.8: Effects of changing from TES to Prophet

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES	0.214	73.22	0.158	**-0.117**
	Prophet	**0.165**	**67.82**	**0.119**	-0.131
p-value	-	**0.0**	0.34	**0.0**	0.618
Medium	TES	0.34	50.45	0.248	**-0.132**
	Prophet	**0.275**	**48.42**	**0.207**	-0.154
p-value	-	**0.001**	0.69	**0.003**	0.971
Long	TES	0.863	38.98	0.639	**-0.175**
	Prophet	**0.636**	**36.54**	**0.488**	-0.26
p-value	-	**0.0**	0.61	**0.001**	0.138

We see Prophet outperform TES on all error metrics, with the difference in RMSE and MAD being significant. This makes Prophet the best model for this class.

Class 9: Trending SKUs. (3 071 items).

Table S.9: Effects of changing from DES to Prophet

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	DES	0.183	75.14	0.137	**0.006**
	Prophet	**0.167**	**72.26**	**0.123**	-0.169
p-value	-	**0.0**	**0.02**	**0.0**	**0.0**
Medium	DES	0.283	52.58	0.217	**-0.008**
	Prophet	**0.281**	**52.44**	**0.217**	-0.213
p-value	-	0.703	0.9	0.868	**0.0**
Long	DES	**0.671**	39.92	**0.538**	**-0.031**
	Prophet	0.675	**39.81**	0.538	-0.273
p-value	-	0.716	0.91	0.992	**0.0**

We notice Prophet outperform DES on all of our error metrics on the short term. This makes Prophet the better model, despite DES having a lower bias. For medium and long horizons, either method works fine as none of the differences on our performance metrics are significant.

Class 10: Daily seasonal SKUs with a trend. (3 101 items)

Table S.10: Effects of changing from DES to Prophet

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	DES	0.195	67.65	0.148	**0.016**
	Prophet	**0.172**	**63.69**	**0.126**	-0.141
p-value	-	**0.0**	**0.0**	**0.0**	**0.0**
Medium	DES	**0.283**	**45.21**	**0.217**	**0.006**
	Prophet	0.285	45.25	0.219	-0.177
p-value	-	0.514	0.97	0.354	**0.0**
Long	DES	**0.652**	34.18	**0.519**	**-0.013**
	Prophet	0.665	**34.11**	0.524	-0.22
p-value	-	0.185	0.94	0.507	**0.0**

We see Prophet perform significantly better on the short term, despite bias being worse when compared to the current method. For medium and long horizons, DES remains the better method due to having a significantly better bias whilst not being statistically different based on our other metrics.

Class 11: Weekly seasonal SKUs with a trend. (67 items)

Table S.11: Effects of changing from DES to Prophet

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	DES	0.183	83.5	0.135	**0.073**
	Prophet	**0.166**	**79.67**	**0.119**	-0.168
p-value	-	0.135	0.64	0.113	0.087
Medium	DES	0.316	64.13	0.248	**0.089**
	Prophet	**0.286**	**60.89**	**0.219**	-0.208
p-value	-	0.177	0.68	0.13	0.053
Long	DES	0.795	53.91	0.664	**0.079**
	Prophet	**0.677**	**48.89**	**0.541**	-0.264
p-value	-	0.093	0.5	**0.035**	**0.03**

For these SKUs, DES has a better bias across the board. Prophet scores better on all other metrics, with only long-term MAD being statistically significant. This makes Prophet the better model for long term. For medium and long term, the best model for the conservative approach is DES, and for the progressive approach Prophet is the better model.

Class 12: Monthly Seasonal SKUs with a trend. (15 items)

Table S.12: Effects of changing from TES Trending to Prophet

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES Trending	0.175	114.7	0.119	-0.043
	Prophet	**0.136**	**111.12**	**0.089**	**0.033**
p-value	-	0.157	0.82	0.156	0.773
Medium	TES Trending	0.321	82.45	0.223	**-0.02**
	Prophet	**0.22**	**76.15**	**0.158**	0.068
p-value	-	0.077	0.67	0.107	0.802
Long	TES Trending	0.82	61.27	0.582	**0.025**
	Prophet	**0.431**	**50.91**	**0.328**	0.052
p-value	-	**0.03**	0.37	**0.03**	0.908

We see Prophet being the better model on for our long cover period, having a significantly better RMSE and MAD. For the short and medium term, either method will work fine as none of the differences are statistically significant. Again, this class contains very little SKUs and thus we have very little evidence Prophet is the better model despite its accuracy being better all across the board.

Class 13: Daily-Weekly seasonal SKUs with a trend. (294 items)

Table S.13: Effects of changing from DES to Prophet

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	DES	0.219	61.78	0.17	**0.046**
	Prophet	**0.182**	**56.0**	**0.134**	-0.12
p-value	-	**0.0**	0.11	**0.0**	**0.0**
Medium	DES	0.317	42.49	0.245	**0.045**
	Prophet	**0.306**	**41.37**	**0.234**	-0.149
p-value	-	0.387	0.75	0.261	**0.0**
Long	DES	0.766	34.68	0.616	**0.043**
	Prophet	**0.73**	**32.96**	**0.565**	-0.171
p-value	-	0.335	0.61	0.085	**0.0**

We see Prophet outperform DES on the short term, having a significantly better RMSE and MAD. sMAPE is also better, hut the difference is not significant. For the medium long term, accuracy is not statistically different on any of our accuracy measures, whilst DES has a significantly better bias. Like before, selecting the best model for medium and long term will depend on your ideology.

Class 14: Daily-Monthly SKUs with a trend. (40 items)

Table S.14: Effects of changing from TES Trending to Prophet

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES Trending	0.186	90.73	0.136	-0.076
	Prophet	**0.149**	**86.26**	**0.109**	**-0.06**
p-value	-	**0.027**	0.63	**0.027**	0.985
Medium	TES Trending	0.307	58.25	0.223	-0.106
	Prophet	**0.239**	**54.71**	**0.186**	**-0.077**
p-value	-	0.053	0.63	0.104	0.715
Long	TES Trending	0.764	40.81	0.543	-0.069
	Prophet	**0.515**	**34.7**	**0.401**	**-0.051**
p-value	-	**0.026**	0.23	**0.03**	0.524

Here, we see Prophet perform better all across the board, though the difference is only significant for short and long forecasting horizons. This makes Prophet the better model for short and long forecasting horizons, and either model being fine for medium horizons. As Prophet has better numbers for the medium horizon as well, combined with the fact that p-value for medium-term RMSE is just 0.053, we expect it is the better model here as well. With this class having just 40 SKUs, we need more data to back up this claim.

Class 15: Weekly-Monthly SKUs with a trend. (33 items)

Table S.15: Effects of changing from TES Trending to Prophet

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES Trending	0.225	91.07	0.154	-0.102
	Prophet	**0.166**	**86.49**	**0.115**	**-0.062**
p-value	-	**0.01**	0.66	**0.023**	0.156
Medium	TES Trending	0.427	67.37	0.293	-0.14
	Prophet	**0.281**	**64.17**	**0.209**	**-0.092**
p-value	-	**0.004**	0.74	**0.015**	0.278
Long	TES Trending	1.163	50.99	0.799	-0.119
	Prophet	**0.661**	**46.43**	**0.507**	**-0.112**
p-value	-	**0.003**	0.56	**0.01**	0.736

We see Prophet being dominant all across the board, making it the better model. Note that only differences in RMSE and MAD are statistically significant.

Class 16: Daily-Weekly-Monthly SKUs with a trend. (279 items)

Table S.16: Effects of changing from TES Trending to Prophet

Horizon	Model	RMSE	sMAPE	MAD	Bias
Short	TES Trending	0.22	76.22	0.157	**-0.052**
	Prophet	**0.159**	**69.57**	**0.114**	-0.199
p-value	-	**0.0**	0.06	**0.0**	**0.002**
Medium	TES Trending	0.388	53.42	0.274	**-0.077**
	Prophet	**0.271**	**50.0**	**0.204**	-0.259
p-value	-	**0.0**	0.29	**0.0**	**0.003**
Long	TES Trending	1.06	42.91	0.754	**-0.086**
	Prophet	**0.635**	**38.45**	**0.49**	-0.377
p-value	-	**0.0**	0.13	**0.0**	**0.0**

Prophet has a significantly better RMSE and MAD score for all forecasting horizons and thus we conclude it is our best model.

T Performance of Prophet Compared to the Current Method

Short Cover Times

Table T.1: Performance of the Prophet Model with and without Holidays for Short Cover Times

Model	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias
Prophet Holidays	0.167	**71.55**	0.122	-0.144
Prophet No Holidays	**0.167**	71.56	**0.122**	**-0.122**
p-value	0.92	0.985	0.93	0.611

We find that the models are tied on RMSE and MAD. Inclusion of holidays yields a better performance on sMAPE, and exclusion yields a better bias. Performing a t-test with $\alpha = 0.05$ for all four of our metrics, we find the difference between the two models is not significant for any of our metrics.

Medium Cover Times

Table T.2: Performance of the Prophet Model with and without Holidays for Medium Cover Times

Model	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias
Prophet Holidays	0.278	50.97	0.213	-0.182
Prophet No Holidays	**0.278**	**50.94**	**0.213**	**-0.176**
p-value	0.916	0.959	0.881	0.63

We again find that the model that does not include holidays performs slightly better. Using a two-sided t-test with $\alpha = 0.05$, we conclude the difference is not significant for any of our four performance metrics.

Long Cover Times

Table T.3: Performance of the Prophet Model with and without Holidays for Long Cover Times

Model	Normalised RMSE	Average sMAPE	Average Normalised MAD	Average Bias
Prophet Holidays	0.646	37.72	0.511	-0.234
Prophet No Holidays	**0.646**	**37.67**	**0.51**	**-0.227**
p-value	0.898	0.936	0.852	0.819

Again, we find that the model that does not include holidays performs slightly better. Using a two-sided t-test with $\alpha = 0.05$, we conclude the difference is not significant for any of our four performance metrics.

U Performance Improvements as a Result of Model Changes

In this appendix, we analyse the performance gained from switching forecasting models based on RMSE. We do this for each cover time, for each ideology, for each class of SKU. We start off by looking at short cover times.

U.1 Short Cover Times

We start off by analysing Table U.1, where the improvements that can be made to short-term performance under a progressive ideology are shown. Changing our forecasting method to Prophet for all of our classes yields the greatest improvement in RMSE, for an overall improvement in RMSE of 11.70%. We conclude that completely switching to Prophet is the best decision.

The exception to this is class 10: Daily seasonal SKUs with a trend. For this class, the level-updated adjusted variant of nested triple exponential smoothing is the best model, although differences with Prophet are very small. Using the level-updated adjusted variant of nested triple exponential smoothing instead of yields an improvement over the current method of 12.82% rather than 11.79% for this class. Overall performance increase after rounding is still 11.70%.

Analysing our results, we see that the improved performance of our models is statistically significant for nine out of our sixteen classes. This means that, under a conservative approach, seven out of our sixteen classes do not see a model change. The classes that do not see a model change are all very small, with the exception of the class of stationaries. Under a conservative approach, we see an overall improvement to performance of 11.17% based on RMSE. Full results are provided in Table U.2.

U.2 Medium Cover Times

We now look at our medium cover times under a progressive ideology. Results are provided in Table U.3. Analysing this table, we see that model changes are not as prevalent as they are for short cover times. Twelve of our sixteen classes see a model change under a progressive ideology for an overall performance improvement of 2.48% based on RMSE.

Under a conservative ideology, we see only three of our sixteen classes change models for a total improvement of 1.77% based on RMSE. Full results are provided in Table U.4.

U.3 Long Cover Times

For long cover times, we see that, under a progressive ideology, we switch to Prophet for eleven of our sixteen classes for a total performance improvement of 3.37% based on RMSE. Under a conservative ideology, we see three classes see a model change for a total improvement of 3.07% based on RMSE. Full results are provided in Tables U.5 and U.6.

Table U.1: Short Cover Periods: Progressive Approach

SKU types	Current Model	New Model	Number of SKUs	RMSE Improvement (%)
1: (T: n, D: n, W: n, M: n)	SES	Prophet	894	4.43
2: (T: n, D: y, W: n, M: n)	SES	Prophet	1 283	9.29
3: (T: n, D: n, W: y, M: n)	SES	Prophet	24	7.14
4: (T: n, D: n, W: n, M: y)	TES	Prophet	5	12.00
5: (T: n, D: y, W: y, M: n)	SES	Prophet	140	17.65
6: (T: n, D: y, W: n, M: y)	TES	Prophet	25	9.14
7: (T: n, D: n, W: y, M: y)	TES	Prophet	17	13.02
8: (T: n, D: y, W: y, M: y)	TES	Prophet	131	22.90
9: (T: y, D: n, W: n, M: n)	DES	Prophet	3 071	8.74
10: (T: y, D: y, W: n, M: n)	DES	Trending TES Nested Adjusted	3 101	12.82
11: (T: y, D: n, W: y, M: n)	DES	Prophet	67	9.29
12: (T: y, D: n, W: n, M: y)	TES Trending	Prophet	15	22.29
13: (T: y, D: y, W: y, M: n)	DES	Prophet	294	16.89
14: (T: y, D: y, W: n, M: y)	TES Trending	Prophet	40	19.89
15: (T: y, D: n, W: y, M: y)	TES Trending	Prophet	33	26.22
16: (T: y, D: y, W: y, M: y)	TES Trending	Prophet	279	27.73
Total			9 419	11.70

Table U.2: Short Cover Periods: Conservative Approach

SKU types	Current Model	New Model	Number of SKUs	RMSE Improvement (%)
1: (T: n, D: n, W: n, M: n)	SES	SES	894	0.00
2: (T: n, D: y, W: n, M: n)	SES	Prophet	1 283	9.29
3: (T: n, D: n, W: y, M: n)	SES	SES	24	0.00
4: (T: n, D: n, W: n, M: y)	TES	TES	5	0.00
5: (T: n, D: y, W: y, M: n)	SES	Prophet	140	17.65
6: (T: n, D: y, W: n, M: y)	TES	TES	25	0.00
7: (T: n, D: n, W: y, M: y)	TES	TES	17	0.00
8: (T: n, D: y, W: y, M: y)	TES	Prophet	131	22.90
9: (T: y, D: n, W: n, M: n)	DES	Prophet	3 071	8.74
10: (T: y, D: y, W: n, M: n)	DES	Trending TES Nested Adjusted	3 101	12.82
11: (T: y, D: n, W: y, M: n)	DES	DES	67	0.00
12: (T: y, D: n, W: n, M: y)	TES Trending	TES Trending	15	0.00
13: (T: y, D: y, W: y, M: n)	DES	Prophet	294	16.89
14: (T: y, D: y, W: n, M: y)	TES Trending	Prophet	40	19.89
15: (T: y, D: n, W: y, M: y)	TES Trending	Prophet	33	26.22
16: (T: y, D: y, W: y, M: y)	TES Trending	Prophet	279	27.73
Total			9 419	11.17

Table U.3: Medium Cover Periods: Progressive Approach

SKU types	Current Model	New Model	Number of SKUs	RMSE Improvement (%)
1: (T: n, D: n, W: n, M: n)	SES	SES	894	0.00
2: (T: n, D: y, W: n, M: n)	SES	SES	1 283	0.00
3: (T: n, D: n, W: y, M: n)	SES	Prophet	24	5.00
4: (T: n, D: n, W: n, M: y)	TES	Prophet	5	17.52
5: (T: n, D: y, W: y, M: n)	SES	SES	140	0.00
6: (T: n, D: y, W: n, M: y)	TES	Prophet	25	9.96
7: (T: n, D: n, W: y, M: y)	TES	Prophet	17	15.80
8: (T: n, D: y, W: y, M: y)	TES	Prophet	131	19.12
9: (T: y, D: n, W: n, M: n)	DES	Prophet	3 071	0.71
10: (T: y, D: y, W: n, M: n)	DES	DES	3 101	0.00
11: (T: y, D: n, W: y, M: n)	DES	Prophet	67	9.49
12: (T: y, D: n, W: n, M: y)	TES Trending	Prophet	15	31.46
13: (T: y, D: y, W: y, M: n)	DES	Prophet	294	3.47
14: (T: y, D: y, W: n, M: y)	TES Trending	Prophet	40	22.15
15: (T: y, D: n, W: y, M: y)	TES Trending	Prophet	33	34.19
16: (T: y, D: y, W: y, M: y)	TES Trending	Prophet	279	30.15
Total			9 419	2.48

Table U.4: Medium Cover Periods: Conservative Approach

SKU types	Current Model	New Model	Number of SKUs	RMSE Improvement (%)
1: (T: n, D: n, W: n, M: n)	SES	SES	894	0.00
2: (T: n, D: y, W: n, M: n)	SES	SES	1 283	0.00
3: (T: n, D: n, W: y, M: n)	SES	SES	24	0.00
4: (T: n, D: n, W: n, M: y)	TES	TES	5	0.00
5: (T: n, D: y, W: y, M: n)	SES	SES	140	0.00
6: (T: n, D: y, W: n, M: y)	TES	TES	25	0.00
7: (T: n, D: n, W: y, M: y)	TES	TES	17	0.00
8: (T: n, D: y, W: y, M: y)	TES	Prophet	131	19.12
9: (T: y, D: n, W: n, M: n)	DES	DES	3 071	0.00
10: (T: y, D: y, W: n, M: n)	DES	DES	3 101	0.00
11: (T: y, D: n, W: y, M: n)	DES	DES	67	0.00
12: (T: y, D: n, W: n, M: y)	TES Trending	TES Trending	15	0.00
13: (T: y, D: y, W: y, M: n)	DES	DES	294	0.00
14: (T: y, D: y, W: n, M: y)	TES Trending	TES Trending	40	0.00
15: (T: y, D: n, W: y, M: y)	TES Trending	Prophet	33	34.19
16: (T: y, D: y, W: y, M: y)	TES Trending	Prophet	279	30.15
Total			9 419	1.77

Table U.5: Long Cover Periods: Progressive Approach

SKU types	Current Model	New Model	Number of SKUs	RMSE Improvement (%)
1: (T: n, D: n, W: n, M: n)	SES	SES	894	0.00
2: (T: n, D: y, W: n, M: n)	SES	SES	1 283	0.00
3: (T: n, D: n, W: y, M: n)	SES	Prophet	24	4.82
4: (T: n, D: n, W: n, M: y)	TES	Prophet	5	22.72
5: (T: n, D: y, W: y, M: n)	SES	SES	140	0.00
6: (T: n, D: y, W: n, M: y)	TES	Prophet	25	19.37
7: (T: n, D: n, W: y, M: y)	TES	Prophet	17	20.44
8: (T: n, D: y, W: y, M: y)	TES	Prophet	131	26.30
9: (T: y, D: n, W: n, M: n)	DES	DES	3 071	0.00
10: (T: y, D: y, W: n, M: n)	DES	DES	3 101	0.00
11: (T: y, D: n, W: y, M: n)	DES	Prophet	67	14.84
12: (T: y, D: n, W: n, M: y)	TES Trending	Prophet	15	47.44
13: (T: y, D: y, W: y, M: n)	DES	Prophet	294	4.70
14: (T: y, D: y, W: n, M: y)	TES Trending	Prophet	40	32.59
15: (T: y, D: n, W: y, M: y)	TES Trending	Prophet	33	43.16
16: (T: y, D: y, W: y, M: y)	TES Trending	Prophet	279	40.09
Total			9 419	3.37

Table U.6: Long Cover Periods: Conservative Approach

SKU types	Current Model	New Model	Number of SKUs	RMSE Improvement (%)
1: (T: n, D: n, W: n, M: n)	SES	SES	894	0.00
2: (T: n, D: y, W: n, M: n)	SES	SES	1 283	0.00
3: (T: n, D: n, W: y, M: n)	SES	SES	24	0.00
4: (T: n, D: n, W: n, M: y)	TES	TES	5	0.00
5: (T: n, D: y, W: y, M: n)	SES	SES	140	0.00
6: (T: n, D: y, W: n, M: y)	TES	TES	25	0.00
7: (T: n, D: n, W: y, M: y)	TES	TES	17	0.00
8: (T: n, D: y, W: y, M: y)	TES	Prophet	131	19.12
9: (T: y, D: n, W: n, M: n)	DES	DES	3 071	0.00
10: (T: y, D: y, W: n, M: n)	DES	DES	3 101	0.00
11: (T: y, D: n, W: y, M: n)	DES	DES	67	0.00
12: (T: y, D: n, W: n, M: y)	TES Trending	TES Trending	15	0.00
13: (T: y, D: y, W: y, M: n)	DES	DES	294	0.00
14: (T: y, D: y, W: n, M: y)	TES Trending	TES Trending	40	0.00
15: (T: y, D: n, W: y, M: y)	TES Trending	Prophet	33	34.19
16: (T: y, D: y, W: y, M: y)	TES Trending	Prophet	279	30.15
Total			9 419	3.07