Erasmus University Rotterdam
Erasmus School of Economics

Master Thesis Business Analytics and Quantitative Marketing

Practical demand forecasting:
leveraging unrecorded product transitions to improve forecasting performance for new products

L.W. van Someren (455031)
November 9, 2022

Supervisor:	prof.dr.ir. R. Dekker
Second Assessor:	dr. O Kuryatnikova
Supervisor Company:	S. Chowdhury
Supervisor Company:	dr. E. Boattini

The content of this thesis is the sole responsibility of the author and does not reflect the view of the supervisor, second assessor, Erasmus School of Economics or Erasmus University.

Abstract: In demand forecasting, historical data is often limited. In this thesis, we propose a method that leverages unrecorded predecessor products to improve demand forecasts for current products. This increases the sales history available to our forecasting model, thereby increasing its accuracy. Furthermore we apply our method to model ongoing transitions to improve forecasts for newly introduced items. Based on our experiments, we find historical transitions can be used in most cases, but their effect on forecasting performance is limited. In contrast, we found that ongoing transitions can have a substantial effect on forecasting performance. To accurately model them, however, one requires a dataset of sufficient size and quality. The forecasting methods we consider are ETS and SARIMA. In addition to these forecasting methods, for ongoing transitions Gradient Boosting and Random Forest are employed.

Contents

1 Introduction

Accurate demand forecasts are essential in business. Many processes within companies can be optimised when these forecasts are both available and accurate. If they are unavailable, however, the results can be detrimental. Consider the case of a retailer. If he has no accurate demand forecast available, sensible procurement is near impossible. In case demand is underestimated, some demand will remain unfulfilled, which leads to out-of-stock situations and lost sales. Additionally, it also negatively impacts customer satisfaction. This can harm reputation and sales in the long-term. On the other hand, if demand is overestimated the consequences can be even worse. The excess product needs to be stored and financed, which brings additional costs. Sometimes, the excess needs to be entirely discarded or sold at a discount, which is costly. This is often the case if the product is perishable, such as with food and pharmaceuticals, or subject to fast changing trends such as in the fashion industry. Which brings significant financial risk. Other examples of processes that can be improved with better demand forecasts can be found in all aspects of a company. These can be related, among others, to supply chains, logistics, human resources and inventory (Aburto & Weber, 2007; Hart et al., 2013; Kot et al., 2011). Better management of these factors ultimately leads to lower costs, better margins and improved service to customers.

Historically, demand forecasts were made by a company's sales department. The sales representatives request information from the different clients regarding their purchasing intentions and combine it with their *expert* knowledge of the business to create a forecast. This approach is called *judgemental forecasting*. Examples of the *expert* knowledge that may be used include the interactions between items, major events affecting one or more clients or commercial actions such as a promotion. In recent years, many companies have transitioned towards statistical forecasting methods. Almost universally, statistical forecasting methods have lead to increased forecasting performance, when compared to judgemental approaches (Webby & O'Connor, 1996). While on average performance increases, this might not be the case for individual products. This is because, often, some *expert* knowledge is lost during the transition to a statistical method. While an experienced sales representative will *implicitly* consider all information significant to the forecast, a statistical model can only leverage the information it is explicitly provided with. When forecasting at scale, some product specific information might be lost.

One such case are so-called product transitions. A transition occurs when an *established* product is replaced by a *new* product. For the purpose of this thesis, we will focus on one-to-one transitions since they are easiest to detect and model. Note, however, that one-to-many or many-to-one transitions can also occur. A transition can be due to a new product iteration being introduced, as is often the case with electronics. Another example, is that a minor change is made to a product's packaging. However small a change might be, often this results in a new SKU code. In case we are forecasting on a SKU level, it is essential this change is recorded. Otherwise, a model might mistakenly overestimate the demand for the *old* product. Forecasts for the *new* product will be even worse, as there is no available demand history. Despite the obvious relationship between the two products, any meaningful forecasts for the *new* product are therefore impossible in the first month after introduction. While with each month, the forecast will gradually improve, the sample size remains very small. This limits the complexity of suitable forecasting models. For example, only a few months of data do not allow for seasonal effects to be correctly modelled. This has a negative impact on forecasting performance. Furthermore, during the transition, as the *new* product replaces the *old* product, sales will increase almost by definition. A model trained only on these months will

therefore overestimate demand in the months following the transition.

To illustrate this effect, we will now consider an example. In Figure 1 we see a product transition. From 2018 until August 2020, only the *old* product is sold. During that month, a *new* product is introduced, which by January 2021 completely replaces the the *old* product. In the first plot we see a one-month-ahead forecast based only on the new product. At the first forecasting month, it is based on 12 months of demand history. The resulting forecast for the *new* item is very poor and has a large confidence interval. In the second plot, we aggregate the demand for the two products. The resulting model dramatically improves the forecast. Furthermore we see that the confidence interval is much smaller, indicating a higher degree of certainty.

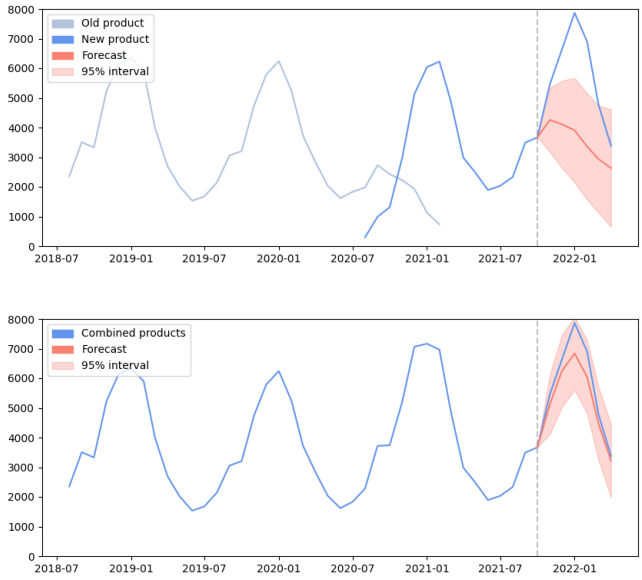

Figure 1: Improved predictions by leveraging discontinued products

Given the profound impact of these product transitions, it is essential they are recorded. The necessity of this data management, however, is often only realised when a company starts applying statistical forecasting techniques. In this case, historical transitions might not have been recorded. The obvious solution in this case is to record the transitions retroactively. While this can be done manually, by examining historical records or correspondence, this may not be possible or feasible. The relevant information might be lost, or it can be prohibitively expensive. In these cases, an statistical method is needed to record transitions.

To develop this method, we propose the following central research question. *Can unrecorded product transitions be identified and used to improve forecasting performance*? To partition the problem we will develop a general framework. Each part of the framework corresponds to sub question. The first question is *how can we optimally identify historical product transitions*? The detection of unrecorded transitions is essential as they contribute to the forecast and form the basis for the next stage. The second question we will answer is *how can we optimally identify ongoing*

product transitions? Ongoing transitions have a very large impact when forecasting in real time, hence this phase will contribute most to the practical value of our method. It is also the most challenging. Since the number of available observations often is small, some forecasting methods are not appropriate or perform badly. Therefore, we will answer *which forecasting method is best suited to deal with the various challenges of this prediction task*?

The remainder of this thesis is structured as follows: In the next section we will provide a detailed description of the problem. Next, in Section 3 we will provide an overview of the relevant literature. In Sections 4 and 5 we will discuss the datasets at our disposal, as well as a detailed description of our proposed framework, respectively. Finally, in Sections 6 and 7 we will present out experimental results and conclusion.

2 Problem Description

As described in the previous section, accurate demand forecasts can yield significant rewards for companies. The potential reward has driven many companies towards a more *data driven* approach (Kühne & Böhmann, 2019). For this approach to be successful, several key ingredients are needed. Perhaps the most important factor is the presence of appropriate infrastructure to store, process and manipulate data. Over the last few years platforms such as Amazon Web Services and Microsoft Azure have made cloud storage and -computing more accessible than ever (Saini et al., 2019). The second ingredient is appropriate data management. This encompasses the implementation of procedures and standards to gather and store data across the company. It ensures that relevant information is kept and accessible for analysis. The third and final ingredient is a statistical or machine learning method. While continued research has certainly improved methods over the past few years, most have been around for several decades.

When these three ingredients are combined, it has yielded significant results. Consider, automated quality control based on Convolutional Neural Networks (CNN) (Imoto et al., 2018). Another example can be found in e-commerce, where accurate churn prediction is being combined with personalised marketing materials to increase customer lifetime value (Srigopal, 2018). Contrary to popular belief, many machine learning methods have been around for decades (Fradkov, 2020). Despite this, their practical application remained limited until recently. In addition to the rise of cloud storage and computing, data availability has been the single biggest driver of innovation. An E-commerce company has all kinds of data on *thousands* or *millions* of customers. This includes purchase history, basic demographics, address information and even browsing behaviour through the use of cookies (Akter & Wamba, 2016). This allows them to accurately predict customer churn, and identify factors that might reduce it. Similarly, manufacturers these days have data on thousands of products they produce, which can be used to train the anomaly detection network. This allows for a complex and accurate technique such as a CNN.

One area that has not experienced the same improvement, despite continued research, is demand forecasting. Popular methods include classical time-series models such as Auto Regressive Moving Average (ARMA) models and state of the art Recurrent Neural Network (RNN) architectures such as Gated Recurrent Unit (GRU) and Long Short Tem Memory (LSTM). In the context of demand forecasting, however, these advanced methods often fail. This can mostly be attributed to poor data management. The main culprit is data availability. Since demand is measured over time, data quantity is closely related to the duration and frequency of the measuring process. In business time-series the frequency of orders is often insufficient to support hourly or even daily observations. These time-series will be too sparse to fit models and make meaningful predictions. More common are weekly, monthly or even quarterly frequencies (Daly & Ortuzar, 1990). While this reduces the noise in the data, it also severely reduces the number of available data points. Which in turn limits options when forecasting in terms of model complexity and performance.

In addition to these quantitative limitations there may also be qualitative limitations. Poor data quality can have a multitude of causes. In case it is captured during, for example, a production process, the accuracy of the capturing device has a direct influence on the data quality. More frequently, the transformation from analogue to digital causes a loss of information. Consider the *judgemental* forecast discussed in the previous section. In this analogue setting, the forecast is produced by a company's sales department. The sales people request information from the different

clients regarding their purchasing intentions and then use their experience and expert knowledge of the business to create a forecast. While some aspects of the forecast can certainly be improved using a statistical method, it is the expert knowledge that is vital in this process. This knowledge could be anything related to the forecast. Examples may include global events such as geopolitical tensions or a pandemic. Or it could be inherent to the client or the business itself. Such as legal trouble for a client, a change in the production process or a product transition.

When applying a statistical model in the case of demand forecasting, it is difficult to incorporate these factors in the model. Furthermore, due to the limited sample size that is often available in a demand forecasting setting, model performance can be significantly influenced by the effect of a relevant event if it is not modelled. Suppose we would like to forecast sales of some fictional smartphone, the XPhone. As can be seen in Figure 2, monthly sales follow a relatively stable (seasonal) pattern. Based on this data, forecasting Xphone sales next month should be relatively easy. Now suppose the task is to forecast the demand for the *latest* Xphone model. In addition, suppose it is not known which Xphone model is the current model's predecessor. This scenario is represented in the bottom of Figure 2, where we try to model the pink spike on the right, without knowing all the spikes that happened before. In this case, the forecasting task is much harder, since there is no demand history available. Furthermore, the similarity or temporal ordering of other products is not known.

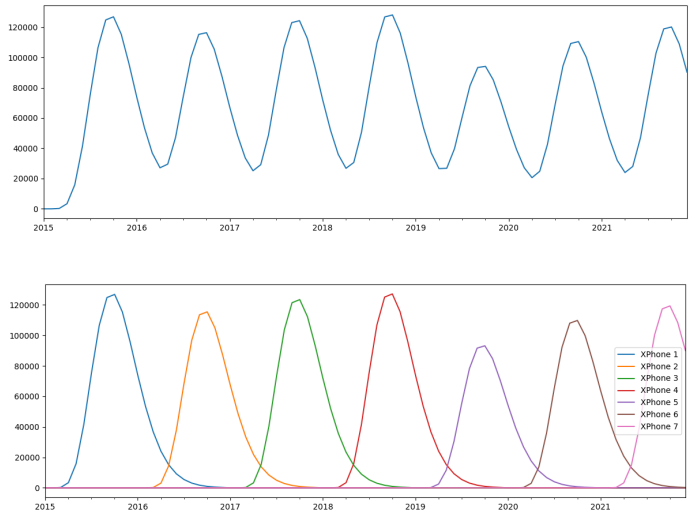

Figure 2: Sales for Xphone smartphones

While the importance of previous product iterations in the Xphone example is clear, in other settings it might be less pronounced. Consider an e-commerce giant, that sells millions of products. If a single supplier makes a minor change to the product packaging, for example, the product may receive a new SKU code. A sales representative will *implicitly* take this change into account when making a forecast. A statistical model, however, can only use the information it is *explicitly* given. Often, these product transitions are not recorded. In this case, when a statistical method

is applied, the model cannot leverage this information, which will have a negative impact on the forecast quality.

Consider the example of the Xphone previously discussed. In case no *succession information* is recorded, any statistical method will only base its forecasts on the very small demand history sample that is available for the *latest* Xphone. While, the overall demand for Xphones is a well documented process that is easily modelled. The difference in forecasting performance between the same method with only a few observations, or the combined demand history of the current product with the previous product cycle can be significant. There are two main reasons for this.

First of all, having a longer demand history allows for more complex models. These models could for example incorporate seasonal effects, which often explain a portion of the observed variance. Secondly, incorporating the demand of the previous product cycle limits the upward bias that occurs after the transition. This bias is caused by the transition. When a product is introduced, by definition its sales start at zero. Hence, the demand can only grow initially. During the same time, the old product shows a significant, and otherwise unexplained, decrease in demand. If we fit a statistical model, that only makes use of these first few months of the new product, we might predict that the increase will keep going forever, as can be seen in Figure 3. This does not represent reality. When we include the previous cycle, however, the demand process is more stationary. In this case we are predicting the *total demand* for Xphones, and attributing a share to the *latest* Xphone.

For the purpose of this thesis we will limit the research to one-to-one transitions. Situations where multiple old products are merged, or a product line is split will not be considered, as this will increase the search-space exponentially. In the context of our Xphone example, we will consider only a single *new* Xphone. Transitions where a the Xphone product line is split, for example with the introduction of the Xphone XL and the Xphone XS, are beyond the scope of this research.

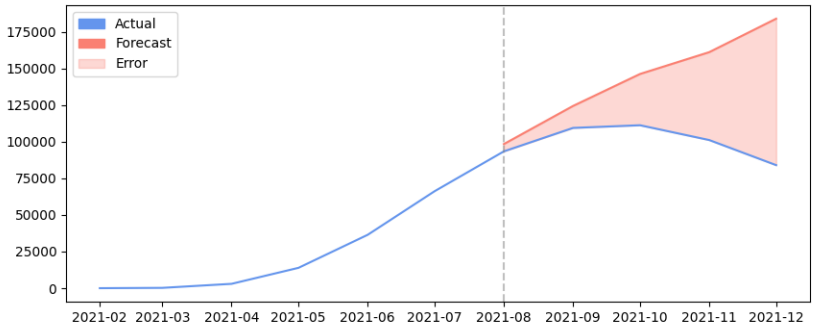

Figure 3: Xphone7 forecasting performance

3 Literature

As described in the previous section, the problem is one of demand forecasting. In subsection 3.1 we will briefly introduce the concepts and methods that have previously been proposed in this field. Next, in subsection 3.2, we will present an overview of the literature regarding the challenge of forecasting new products. Finally, in subsection 3.3 we will discuss the literature on simulated data.

3.1 Time series forecasting

A time series is a series of observations that have a temporal ordering (Wei, 2006). Due to this ordering, the patterns and dependencies in time series are different from those found in cross-sectional datasets. In particular, phenomena such as auto correlation, seasonality and trend should be considered. Autocorrelation refers to the short-term dependence within the data, that is the relation that exists between the current observation and recent lags. Seasonality refers to a stable, fixed frequency repeating pattern in the data. This could be hours in a day, days in a week or months of the year. A trend refers to the long term movement of a time series over multiple seasonal periods. Trends can be constant (i.e. non-zero mean), linear, or follow a more complex pattern.

In classical time series modelling we extend the concept of linear regression to include these dependencies, which results in the Auto Regressive Moving Average (ARMA) model family (Gershenfeld & Gershenfeld, 1999). These models consider lagged dependent variables (AR), as well as lagged error terms (MA) as regressors. In case a time series is *non-stationary*, one can apply one or more orders of differencing, resulting in the Auto Regressive Integrated Moving Average (ARIMA) model. When a time series contains a seasonal pattern, one can include seasonal ARMA parameters, resulting in a *seasonal* ARIMA, or SARIMA model. Including static exogenous regressors within the model specification results in the SARIMAX model.

A limitation of the ARIMA approach is the assumption that a time series is stationary (Koreisha & Pukkila, 1993). In many cases this stationarity can be obtained by taking one or more first differences of the data. In case the normality assumption does not hold, applying a power law transformation such as a Box-Cox transformation may improve the model fit.

Selecting an appropriate model specification is often done by minimising the Akaike Information Criterion (AIC) or the Bayesian Information Criterion (BIC) (Hurvich & Tsai, 1989). These information criteria are based on the likelihood function, together with a penalty on the number of parameters. This is essential to avoid overfitting. While the likelihood of a model will improve with each additional parameter, given a fixed set of observations, this means each parameter can be optimised for fewer training samples. The main idea here is to identify the optimal trade-off between goodness-of-fit and model complexity.

Another approach are so-called Structural Time Series Models (STS) (C. Kim & Nelson, 1998). This approach is based on the assumption that a time series can be decomposed into different latent components. The series is then interpreted as the sum or product of these components. Since each component can be specified separately, based on prior knowledge, this approach has a clear link with Bayesian inference. A simple example is the Local Level Model, which specifies a model with a time-varying mean. Similarly, trend and seasonal components can also be specified to vary over

time, as is the case in the Local Linear Trend model (Shephard & Harvey, 1990). To estimate the parameters for the different components in STS models maximum likelihood is used. The state distribution can then be estimated using the Kalman Filter (Li & Zhu, 2021). A state of the art model in this class is Trigonometric Box-Cox ARMA Trend and Seasonal components (TBATS) (De Livera et al., n.d.). Other succesful approaches based on STS models include Error Trend Seasonality (ETS) models, which apply exponential smoothing. Exponential smoothing refers to taking moving averages of past observations, with exponentially decaying weights (Holt, 2004). Each of the three components in the ETS model can either be specified as additive or multiplicative.

Another noteable approach within this class is Facebook's Prophet (Taylor & Letham, 2018). As stated by the authors, the method seeks to solve the forecasting problem by optimally fitting a curve to the time series without specifically modelling the temporal relationships within the data. The method is an additive model with three main components: seasonality, trend and holidays. The authors argue that these components are the most important when modelling business time series, and justify this restriction by citing increased interpretability and ease of use. Especially for businesses without access to highly specialised knowledge of time series modelling.

Recurrent Neural Networks (RNN) have become a popular time series forecasting method. They have been successfully applied in a wide range of settings including, among others, modelling electricity demand (Muzaffar & Afshari, 2019), air pollution (Chang et al., 2020), traffic (Zhao et al., 2017) and financial time series (Cao et al., 2019). Two successful kinds of RNN's are the Gated Recurrent Unit (GRU) and the Long Short Term Memory (LSTM). The LSTM model architecture was proposed to solve the vanishing and exploding gradient problem that typically occures in RNN's (Hochreiter & Schmidhuber, 1997). Each unit in the network is comprised of a cell with an input, output and forget gate. One limitation of LSTM's are the large number of parameters, subsequently they require substantial amount of data to train. Since data quantity is often an issue in the context of time series, GRU was proposed (Cho et al., 2014). The GRU is very similar to a LSTM, but it does not have an output gate. The simpler architecture performs equally well on many problems and it is easier to train. On smaller datasets with low frequency, GRU has been shown to outperform LSTM (Chung et al., 2014).

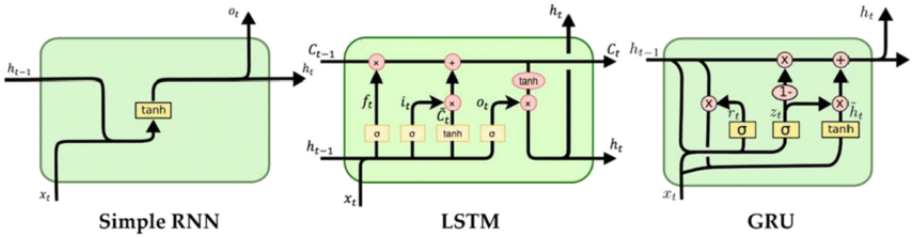

Figure 4: Comparision of RNN, LSTM and GRU (Tembhurne & Diwan, 2021)

In addition to methods specifically developed for time series analysis, more generic supervised learning approaches have recently been adopted to perform time series tasks (Bontempi et al., 2012). This is done by converting the forecasting problem into a supervised learning task. The conversion is a simple manipulation of the time series such that you are left with an input that contains the lags and exogenous variables that you would like the model to consider, as well as the period you would like to predict as the dependent variable. The popularity of this approach can be derived

from the results from the famous M5 competition on time series forecasting. The M5 competition is the latest iteration of the M competition first organised in 1982. During this edition, 5507 teams were tasked to create forecasts for 100,000 individual time series. It was found that methods based on the Light Gradient Boosting Machine (LGBM) was substantially superior to other methods in predicting hierarchical retail sales (Makridakis et al., 2022). In our instance, there is a notable issue with this approach. Like an RNN, LGBM needs sufficient samples to converge to a stable solution. Therefore, they might not be appropriate in our case, as our time series are of limited length.

3.2 Forecasting with limited historical data

While the methods in the previous section have proven to be successful in many instances, they often perform poorly or cannot be implemented in business settings. The main culprit here is data availability. In case there is insufficient historical sales data available, one should look to alternative sources of information.

One of the earliest attempts at modelling this problem was the Bass diffusion model (Bass, 1969). It was developed in the sixties, when new household appliances such as colour televisions, refrigerators and other consumer durables were introduced in rapid succession. The model is a theoretical framework that considers two groups of consumers, innovators and imitators. Based on the estimated rate of adoption of these two groups, the model predicts a profile for the ratio of adoption within the population. The Bass model has long been obsolete, since the assumptions made are not reflecting the more complex dynamics we find in practice.

More recently, a number of methods have been proposed to deal with specific kinds of new products and substitutions. These methods are developed to forecast products with short life cycles, like a single season (fashion), single occasion (Mother's Day) or planned obsolescence (technology) (Berbain et al., 2011). Almost all approaches that deal with this problem propose to analyse historical introductions of similar products. This is also known as *analogous* forecasting (Szozda, 2010). These methods create a forecast based on the introduction profiles of similar observations, and have been successfully applied in predicting box office revenue for movies (Neelamegham & Chintagunta, 1999), weather forecasting (Van den Dool, 1989), as well as in predicting the spread of influenza epidemics (Viboud et al., 2003). Another approach to forecast new products combines K-means clustering with a Quantile Regression Forest van Steenbergen and Mes (2020). Within the fashion industry, Thomassey and Happiette (2007) applied K-means clustering combined with a probabilistic neural network to predict the sales profiles for new products. Similar research was done applying Extreme Learning Machines to the problem (Sun et al., 2008).

3.3 Simulated Data

We will now briefly discuss the use of simulated data. Simulation is an established practice in many fields of study. It is widely used in meteorology (Noh et al., 2003), engineering (Bird, 1981) and physics (Georgescu et al., 2014). In statistics and econometrics, simulation essentially refers to sampling from some distribution (Kleijnen & van Groenendaal, 1992). It is very common in Bayesian Inference, where Monte Carlo methods are used to estimate the distribution of parameters (Bird, 1981; Chen et al., 2012).

Data can be simulated in a similar way by sampling from a number of distributions. This simu-

lated data gives researchers much more control over the data generating process (DGP) (Shannon, 1998). This allows for clean comparison between and comprehensive analysis of different methods. Depending on the research requirements, data can be simulated from a wide range of distributions. When researching clustering methods, for example, one might sample from several Gaussian distributions. By specifying these distributions with different means and variances, different clusters of observations can be simulated (Milligan & Cooper, 1987). Other distributions commonly used for simulation include Exponential-, Gamma-, Uniform- and Poisson distributions.

4 Data

To evaluate our proposed method we will make use of two types of data. The practical evaluation of the method is done using a real dataset provided by a global aqua feed producer. For this dataset, the *true* product mappings are unrecorded. Therefore, we will use simulated datasets to develop and test the method. This will allow us to evaluate and optimise each component individually.

4.1 Fish feed data

The data for these experiments was provided by a market leader in aqua feed production. They produce feed for a wide range of fish species, including salmon, trout, shrimp etc. They operate in markets all around the world. Most of their products are specialised products, such as medicated feeds, which are produced in small quantities and made to order (MTO). This production style, however, is not suited to all of their products, as for some the quantity required is simply too large.

Since many products are perishable and warehouse space is limited, made to stock (MTS) production requires accurate demand forecasts. For the purpose of this paper we will focus on 300 products that were identified as candidates for MTS production. This subset of products represents the bulk of total production output. Hence, improving these forecasts will have the greatest impact on the business. The data is organised in three tables.

The first is historical sales data. This table is comprised of 15606 transactions. It contains information on the product, quantity, customer and delivery date; as well as information about the production process such as at which factory, production line and warehouse the product was produced. Example entries of the relevant columns can be found in Table 1. For the purpose of our analysis we will aggregate the transactions by product and by month. Since we have a little over four years of data, this leaves us with 300 time series comprised of 52 months.

We also have a brief description of each of the products. This table contains information about product characteristics, such as the intended fish family and life stage; as well as information about the products profit margin, production speed and maximum storage time. Example entries can be found in Table 2. This table will be used in the clustering and classification stage of our method, to determine similarity between different products. Therefore, we will apply some feature engineering to this table. More specifically, we extract relevant information from the product names. This includes information on the product composition, intended fish weight and production methods.

Finally we were provided the *expert* forecast made by the company's sales department. This monthly forecast is made for most products and is created by aggregating the expectation of all account managers. It therefore leverages their knowledge of long term contracts, as well as the intention of clients. For most products this forecast has not proven to be reliable, nevertheless it contains information that could be beneficial to our analysis. Specifically, we use this forecast to derive the expected introduction and discontinuation date of each product

Within our dataset we observe a significant change in late 2021. A large number of products was discontinued around this time, while a similar number of new products was introduced. After inspecting the product descriptions, it was found that this group of products had been re-branded

and are now sold under a new name. The goal of our method is to correctly detect and model this change.

Product ID	*Customer ID*	*Transaction ID*	*Order Date*	*Requested Delivery Date*	*Invoiced Quantity*
1234	456	123342451	2022-04-23	2022-05-13	35000
1253	342	121234432	2021-09-12	2021-09-28	61200

Table 1: Example rows for sales data

Product ID	*product Name*	*Brand*	Fish family	Life stage
1234	Grow 60A25C RC 9.0MM 1T	Growbest	Atlantic	Grower Diets
1253	Super 30A40C 12.0MM 1T	Super	Trout (freshwater)	Smolt Diets

Table 2: Example rows for product data

Product ID	*Date*	*Forecast Made*	Quantity
1234	2019-03-01	2019-01-01	75000
1253	2021-10-01	2020-12-01	8000

Table 3: Example rows for *expert* forecast

4.2 Simulated Data

The goal of our simulated experiments is to evaluate each of our method's component. This allows us to better develop the method, and will ultimately lead to improved performance on the real data. The simulated dataset will closely resemble the fish feed dataset, with the exception that the true product mappings are known.

Simulating the dataset is comprised of three main stages. Each dataset consists of 200 product families. For each of these, we generate a time series with a length of 60 months and assign them into groups of size d. This *difficulty* parameter represents the number of distinct product families within a candidate group. In our experiments we will consider datasets for d between 2 and 5.. The time series are generated based on the structural time series (STS) model family (Harvey & Shephard, 1993). Since sales data is non-negative, we specify the data generating process (DGP) in terms of its components, and then take the exponent. This results in a multiplicative DGP. The components that we consider are level, trend and seasonality. Each of these are specified randomly using Gaussian, Uniform and Exponential distributions (Wackerly et al., 2014). Finally we introduce some Gaussian noise and outliers to each of the series. The level is drawn from a Gaussian distribution. The trend component is specified as the cumulative sum of Gaussian draws and can hence be seen as a random walk. The seasonal component is created based on a sine function. We offset this function a random number of periods by drawing from a Uniform distribution, and add some Gaussian noise.

Next, we simulate the product features. We specify two types of features, numerical and categorical. This choice was made to resemble the kind of features present in the fish feed data. The

features can be interpreted as the product characteristics, such as ingredient content or brand. A change in one of the numerical features can be seen as a proxy for changes to the product itself such as an *improved formula* or a new iteration. The categorical features represent features such as the product packaging, branding or intended fish specie. These kind of variables can be further subdivided into those that are and are not allowed to change. An example of the latter is the intended fish species. products that are not in the same category here cannot be candidates for succession. In our simulated dataset we capture these features in a single *group* variable. Other categorical variables, such as packaging quantity and brand, can change during a transition. Hence these will be represented by a number of variables in our dataset.

To simulate the numerical features, we sample each from a Uniform distribution. Next we divide by their sum, such that they can be interpreted as percentages. This represents the aforementioned product composition. Next we consider categorical variables. Here we specify the number of levels and assign each product according to a Uniform distribution.

Finally we introduce an product mapping into our data. For each of the products we draw the number of transitions per product family from a Geometric distribution. Then, for each product and each transition, we generate a random transition profile, and distribute the demand across our new products. We then generate new features for these products by adding some Gaussian noise to the numerical features, and changing a subset of categorical variables.

	Content					Category					
Product ID	1	2	3	4	5	1	2	3	4	5	Group
54607	0.136	0.121	0.045	0.352	0.346	0	0	7	1	1	11
74360	0.182	0.121	0.301	0.266	0.130	0	1	2	2	0	11
87291	0.245	0.253	0.277	0.037	0.188	1	2	3	2	1	11
40060	0.184	0.163	0.000	0.294	0.407	0	0	7	1	1	11

Table 4: Example of simulated product characteristics

5 Methodology

5.1 General framework

The general framework of our method consists of three stages. Each stage is related to one of our research questions. First we leverage historical data to detect historical transitions. Secondly, we use these historical transitions to train our a classification model to predict ongoing transitions. Finally, we use these transitions to create a forecast.

Within the scope of our problem, products can be classified into four types: new, recent, established and discontinued. With our method, we seek to improve forecasts for new and recent products. To do so, we will use established and discontinued products. Established products are products that have been around for a while, such that there is enough data available to incorporate seasonal effects into the model. Hence, they can already be accurately modelled. Discontinued products are products that are no longer actively being sold. However, they are an essential part of our approach as their demand history contains valuable information about new and recent products.

Recent products are defined as products that have been introduced between approximately 6 and 20 months ago. This means that we can apply a simple statistical model, but this model will not be able to account for any seasonal effects. If these effects are present, the forecast will not be accurate. Finding the correct discontinued product that preceded it, essentially gives more historical data, allowing a more complex model and better forecasts. A schematic overview of how we forecast recent products can be found in Figure 5 Finally, we have new products. These are the most challenging to model, since often very little information is available. Most importantly there is no demand history available, since the product has not yet been sold. Additionally, there could still be an ongoing transition, that could influence the demand. To identify predecessors in this case we will rely on the historical transitions previously identified. Figure 6 shows how forecasts for new products are created.

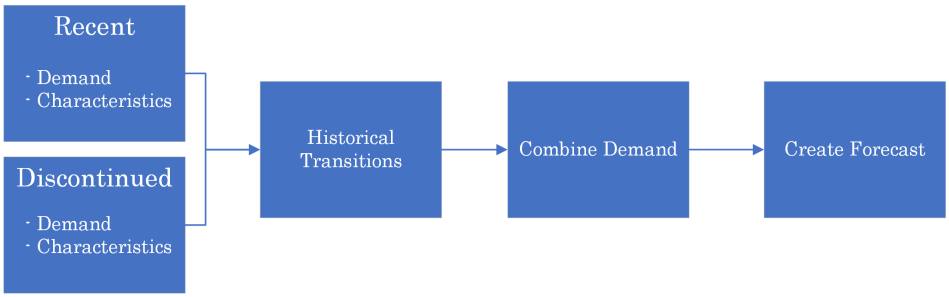

Figure 5: Method overview when forecasting recent products

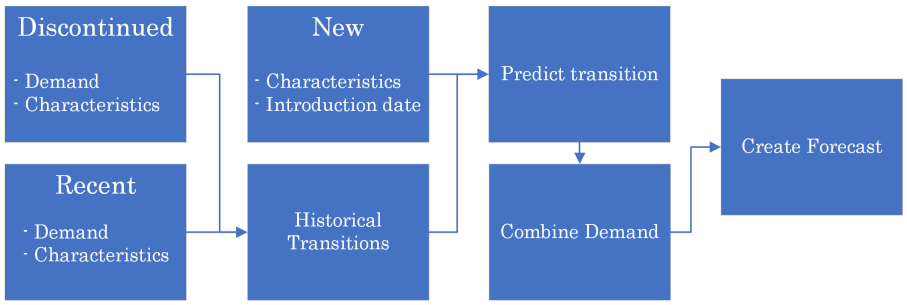

Figure 6: Method overview when forecasting new products

5.2 Detecting historical transitions

The detection of historical transitions is the backbone of our method. Not only do they contribute to the forecast for recent products, they are also used to predict ongoing transitions. To detect historical transitions we use sales and product data. Whenever a new product is introduced we apply a three staged process, which can be seen in Figure 7.

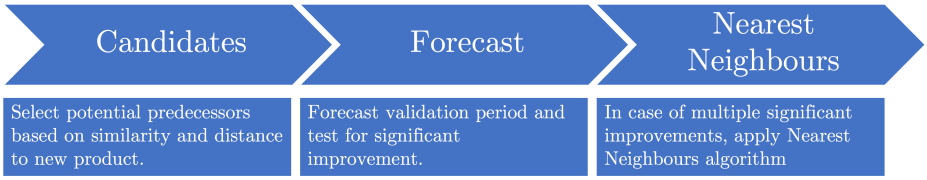

Figure 7: Three staged detection approach

We begin our search by identifying a set of candidate predecessor products. Candidates are defined according to some predetermined parameters. These parameters and their values depend on the dataset. In general they should include, among others, product hierarchy and maximum distance or overlap. Including product hierarchy information, ensures only products from the same group are considered as candidates. Distance or overlap refers to the time between the introduction of the new product and discontinuation of the old product. If the distance or overlap is large, it is unlikely a transition is occurring. In case no products meet these criteria, no candidates are available and no predecessor is detected. Therefore, the candidate selection stage is a trade-off between precision an efficiency. In case strict restrictions are imposed, the method will work will be very efficient. At the same time, however, some transitions will be missed. Without any restrictions, on the other hand, the number of comparisons will increase exponentially with regards to the number of new products that are introduced. This is very inefficient. Furthermore, a large number of very distant predecessors may lead to incorrectly detected transitions by increasing false positives. Hence it is essential the restrictions are chosen in a way that is appropriate for the dataset.

In the second stage we create a forecast on a predetermined validation interval. This interval should be chosen such that it includes several months of post-transition data, but remains close to

the transition. In our case, as will be explained in section 6.2, we chose a validation period from the fourth to the eight month of the new product. With the validation period chosen, we create a forecast for this period based on the aggregated demand of each of the candidates with the new product. We then compare the forecasting performance for each to a forecast without a predecessor and test if any yield a significant improvement. For this, we use the Diebold Mariano test, which will be described in Section 5.2.1. In case no significant improvement is found, we conclude there is no predecessor. In case a single candidate yields a significant improvement, this candidate is classified as a predecessor and a transition is detected.

In case there are multiple products that yield a significant improvement the forecast, we proceed to the third stage. Here we will consider two approaches. In the first approach, we will select the product that yields the biggest improvement in the forecast. Alternatively, we apply the Nearest Neighbour algorithm on the product characteristics. An overview of the latter approach can be found in Algorithm 1.

Algorithm 1 Historical Transition Detection

Data: Historical demand D, product characteristics I
Result: Transitions $(k, l) \in T$ where $k, l \in I$
$\Phi = getNewproducts(D)$
for $i \in \Phi$ **do**
  $\hat{y}_{it} = createForecast(D_i)$
  C = *getCandidates(i, D, I)*
  for $c \in$ candidates **do**
    $D_{i,c} = D_i + D_c$
    $\hat{y}_{ct} = createForecast(D_{i,c})$
  end for
  Let $B \subset I$ such that MSE$(\hat{y}_c, y_i)$ <MSE$(\hat{y}_i, y_i)$ for $c \in C$
  Let $F \subset B$ such that *dieboldMariano*$(\hat{y}_c, \hat{y}_i)$ is significant
  if $|B| \geq 2$ **then**
    p = *selectNearest*(B, I_i)
    $T \cup \{(i, p)\}$
  else
    if $B = \{b\}$ **then**
      $t = b$
      $T \cup \{(i, p)\}$
    end if
  end if
end for

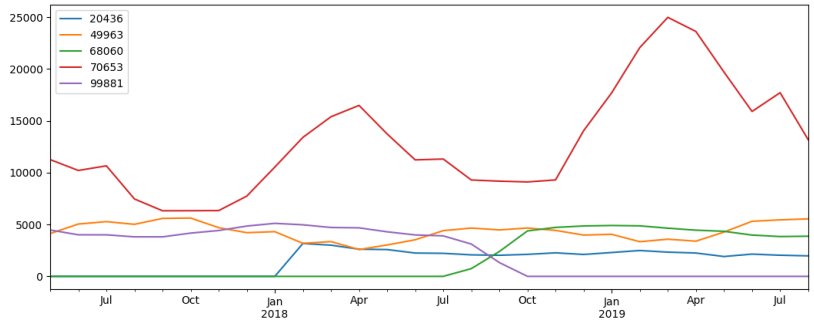

Figure 8: Example of a new product (green), with its candidate predecessors

To further illustrate the mechanics of the method, we will consider an example. In Figure 8 we see a newly introduced product (green) with several candidate predecessors. These candidates were selected using product and demand data. In stage two, we consider the validation forecast error with only the demand history for the new product. This can be seen in the top graph of Figure 9. We see that the forecast performs well during the first thee months. It then spikes, resulting in a very large error. This is likely due to the large spike during the very small training period. Hence the model is very sensitive to upward spikes, resulting in some large errors. When we consider an incorrect predecessor, the prediction error is worse than before, as can be seen in the middle graphs of Figure 9. At the bottom of the Figure, we see the prediction error for a correct predecessor. Here the error is much smaller than in all cases before. Since in this example, there are no other candidates who outperformed the benchmark of *no predecessor*, we will select this candidate as the predecessor.

Figure 9: Prediction error with no (top), wrong (middle) and correct (bottom) predecessors

5.2.1 Diebold Mariano Test

The Diebold Mariano (DB) statistical test is used to test if two forecasts perform significantly different (Diebold, 2015). Suppose we have a *baseline* forecasting model, in our case this would be the forecast without a predecessor. We now compare the actual values with the predictions made by the *baseline* and *alternative* models. Let f_{it} denote the forecast for model $i \in \{b, a\}$ at time t. We then have the forecast error e_{it} as:

$$e_{bt} = y_t - f_{bt} \qquad\qquad e_{at} = y_t - f_{at}$$

We can now calculate the average *loss differential* $\bar{d}$ by taking the mean of the difference in squared errors for both models.

$$\bar{d} = \frac{1}{T}\sum_{t=1}^{T} d_t \qquad \text{where} \qquad d_t = e_{bt}^2 - e_{at}^2$$

In this instance, $\bar{d}$ is based on the mean squared error (MSE). Note that we could equivalently base $\bar{d}$ on another metric such as the mean absolute error (MAE). In that case the d_t would be defined as the difference in absolute errors. We now calculate the auto-covariance γ_k of the time series d_t, where k denotes the number of lags as:

$$\gamma_k = \frac{1}{T}\sum_{t=k+1}^{T} (d_t - \bar{d})(d_{t-k} - \bar{d})$$

We can now calculate the DB test for $h \geq 1$ statistic as:

$$DM = \frac{\bar{d}\sqrt{T}}{\sqrt{\gamma_0 + 2\sum_{k=1}^{h-1}\gamma_k}}$$

Where h denotes the forecast sample size. Under the H_0 of equal predictive performance, the DB statistic (asymptotically) has a $\mathcal{N}(0, 1)$ distribution. Hence we reject the null hypothesis if it is far from zero. For the purpose of this thesis, we will use a p-value of 0.05 as a significance threshold.

5.2.2 Nearest Neighbour Search

The Nearest Neighbour Search (NNS) problem is concerned with finding observations that are *close* to a query point (Knuth et al., 1973). Suppose we have $S \subset M$, where S represents a set of points in M dimensional space. The nearest neighbour of a query point $q \in M$, is the point $n \in S$ that minimises some distance metric. Often this metric is the euclidean distance. Solving this problem can be done in several ways. A simple approach would be to compute the pairwise distance of the query point to each of the points in S. This approach is known as a a linear search since it runs in linear time in terms of number of points in S. When searching large numbers of points, this approach becomes slow. To improve the speed of the search, several space partitioning methods have been proposed. We will apply the k-dimensional tree based method (Mark et al., 2008) implemented in the Scikit-learn package (Pedregosa et al., 2011),

5.3 Predicting ongoing transitions

Correctly predicting ongoing transitions presents some additional challenges. Since there is no or very little sales history, we have to mainly rely on product characteristics to predict transitions. Furthermore, since the transition is ongoing, we must split the demand between the old and new product. To answer our research question we propose a three staged approach, which can be found in Figure 10.

Figure 10: Three staged prediction approach

The candidate stage of our approach is exactly the same as the candidate stage for the detection method described in section 5.2. This is by design as our classification method will be trained on historical candidates and transitions. Since we are training the classification models to predict the relationship between two products, we cannot directly use the product characteristics. Instead, we transform the original product characteristics based on the type of data, such that they capture (dis-)similarity between the two products. This is done in a generic way, such that this approach is applicable to multiple datasets. In addition, when available, we include the prediction error for the previous months as well as the new product's share of joint demand in the previous month. A detailed overview of the features can be found in the table below. As the set and distribution of the features is different for each month, we will create four different classification models. The methods that we will consider for pair classification are the Random Forest and Gradient Boosting algorithms, which will be discussed in Section 5.5.6.

Feature	*Description*
Numerical Difference	For numerical product characteristics, we calculate the difference between the old and new products. Examples of these characteristics are ingredient content or price per tonne. In this case the transition features become difference in in ingredient content and price.
Ordinal Distance	For ordinal product features, we compare the distance between the groups. An example for fish feed would be the *target weight*. Common values are 500, 1000, 2000 and 3500. Since this relationship is nonlinear, the difference can give skewed results. Treating it as a categorical variable results in a loss of information. Hence we count the number of steps between the old and new product.
Categorical Equal	For boolean or categorical features we compare the value for both products. In case these are equal we return true, and false otherwise.
Transition Duration	The transition duration is the number of months between first sale of the new product and the (expected) last sale of the old product. In case the old product has not yet been discontinued we use the *expert forecast* that was provided by the company as an indicator of the expected date of last sale. Note that in case there is an overlap between the two products, this number is positive, and in case there is a gap it is negative.
Prediction Error	This feature captures the explanatory power of the new product over the prediction error of the old product. It is calculated of as the a WMAPE, where we predict the old product and have the sum of old and new products as actuals. Note that as this feature requires demand history it is not calculated when forecasting the first month.
Percentage New	This feature calculates the share of new product in the previous month. In case of a true transition, one would expect this ratio to approach one as the new product replaces the old. Otherwise, the two would be uncorrelated. As with the previous feature, since this feature requires sales, it is included from the second month onward.

Once a likely transition has been identified, we apply one of our forecast methods to predict the joint demand of both products. To split the demand between them, we use techniques inspired by van Steenbergen and Mes (2020). The transition profile of a product pair is the ratio of their demand the in the first four months after the introduction of the new product. An example can be seen in the left plot of Figure 11. We calculate the transition profiles for all historical transitions identified in the previous section. We then, for each month train a regression model that predicts the demand split for future transitions based on the transition characteristics described above. The forecast for the new product is calculated by multiplying the joint demand with the predicted transition profile.

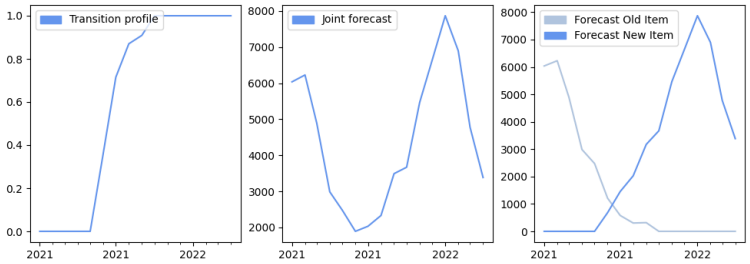

Figure 11: Splitting the forecasted joint demand between the old and new product

5.4 Forecasting methods

We will now describe the forecasting models that we will use in our method. These methods will be used while identifying historical transitions, as well as making the final predictions on which our approach will be evaluated.

5.4.1 SARIMAX

Seasonal Auto-Regressive Integrated Moving Average with Exogenous variables, (SARIMAX) models are a generalisation of extension of *Auto Regressive Moving Average*(ARMA) models (Box et al., 2015). ARMA models are a class of linear models that generalise linear regression regression to time series settings (Franses, Franses, et al., 1998). The AR component of the model can be expressed as a linear regression in terms of lagged dependent variables. As can be seen in the AR(2) example, the model can be estimated using *ordinary least squares* (OLS). Similarly we have the MA component. This is specified as a moving average of the error term ϵ. Since the regressors are unknown, however, OLS cannot be applied. Estimation of MA models can be done using *non-linear least squares* (NLS) or *maximum likelihood estimation* (MLE).

$$\begin{aligned} AR(2): &\quad y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \epsilon_t \\ MA(2): &\quad y_t = \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \epsilon_t \end{aligned}$$

In addition to an auto-regressive and a moving average component, SARIMAX models also implement a differencing order (I) and a seasonal component (S). Additionally, exogenous variables (X) may be specified. In general a SARIMA model is specified in terms of its order $(p, d, q) \times (P, D, Q)_s$.

$$\phi_p(L)\Phi_P(L^s)\nabla^d\nabla_s^D y_t = \theta_q(L)\Theta_Q(L^s)\varepsilon_t$$

Here y_t is the dependent variable. $\phi_p(L)$ and $\theta_q(L)$ represent AR and MA polynomials of order p and q, with L as the lag operator. That is $L^k(y_t)$ is equal to y_{t-k}. Similarly, $\Phi_p(L^s)$ and $\Theta_Q(L^s)$ correspond to seasonal AR and MA polynomials of orders P and Q. Finally we have the differencing operators ∇^d and ∇_s^D, which enforce stationary in both components by differencing with orders d and D respectively. For example, a model with specification $(1, 0, 1) \times (1, 0, 1)_{52}$ can be written out like:

$$y_t = \phi_1 y_{t-1} + \theta_1 \epsilon_{t-1} + \Phi_1 y_{t-53} + \Theta_1 \epsilon_{t-53} + \epsilon_t$$

And in the case we have $(1,1,1) \times (1,0,1)_{52}$ with exogenous regressor x we difference the dependent variable once and add a term for x. This yields the following model:

$$\Delta y_t = \phi_1 \Delta y_{t-1} + \theta_1 \epsilon_{t-1} + \Phi_1 \Delta y_{t-53} + \Theta_1 \epsilon_{t-53} + \beta x_t + \epsilon_t$$

Estimation of SARIMAX models can be done using MLE.

5.4.2 ETS

Our second forecasting method are *Error Trend Seasonality* models. ETS models, are a kind of state space model that use exponential smoothing. State space models, sometimes referred to as structural time series or dynamic linear models, are time series models based on the assumption that a time series can be decomposed into different latent components (C. Kim & Nelson, 1998). Here each of the components is considered a stochastic factor, with its own stochastic process. The simplest example is a Local Level model, which can be seen below.

$$\begin{aligned} y_t &= \mu_t + \epsilon_t & \epsilon_t &\sim \mathcal{N}(0, \sigma_\epsilon^2) \\ \mu_t &= \mu_{t-1} + \eta & \eta &\sim \mathcal{N}(0, \sigma_\epsilon^2) \end{aligned}$$

Here we see that the dependent variable y_t depends on a time varying mean μ_t plus an error ϵ_t. In this case, the mean component μ_t is defined as a random walk. Suppose now we would like to include a linear trend β_t in the model. In this case we re-specify the mean component as follows, which gives us the Local Linear Trend model.

$$\begin{aligned} y_t &= \mu_t + \epsilon_t & \epsilon_t &\sim \mathcal{N}(0, \sigma_\epsilon^2) \\ \mu_t &= \mu_{t-1} + \beta_t + \eta & \eta &\sim \mathcal{N}(0, \sigma_\eta^2) \\ \beta_t &= \beta_{t-1} + \zeta_t & \zeta &\sim \mathcal{N}(0, \sigma_\zeta^2) \end{aligned}$$

Exponential smoothing is a technique that is used in signal processing and time series analysis to remove noise from the data (Rabiner & Gold, 1975). It is defined in terms of the recurrent relation found in Equation 1. Here α is referred to as the *smoothing parameter*, and represents the exponential rate at which the weights of the function decay. The terms y_t and x_t represent the smoothed and original signal. If we rewrite Equation 1 to its moving average, we are left with Equation 2 for the 1 step ahead forecast.

$$\begin{aligned} y_0 &= 0 \\ y_t &= \alpha x_t + (1-\alpha) y_{t-1} \end{aligned} \tag{1}$$

$$\begin{aligned} \hat{y}_{T+1|T} &= \alpha y_T + \alpha(1-\alpha) y_{T-1} + \dots \\ &= \sum_{j=0}^{T-1} \alpha (1-\alpha)^j y_{T-j} + (1-\alpha)^T x_0 \end{aligned} \tag{2}$$

When we combine exponential smoothing with state space models we are left with ETS models. They are defined in terms of their three components. These can have an additive (A) or multiplicative (M) specification or, in case of trend and seasonality, remain unspecified (N) (Hyndman & Athanasopoulos, 2018). The example in the previous section, is referred to as an ETS(A,N,N) model and has the following model specification, where $\epsilon_t \sim \mathcal{N}(0, \sigma_\epsilon^2)$.

$$\begin{aligned} y_t &= l_{t-1} + \epsilon_t \\ l_t &= l_{t-1} + \alpha\epsilon_t \end{aligned}$$

A more complex specification is the ETS(A,A,A), or the ETS model with additive errors, trend and seasonality. Its specification can be found in Equation 3. In order, the formulas in this equation are the measurement equation y_t, followed by the state equations l_t, b_t, s_t for the error, trend and seasonality processes. An overview of all ETS model specifications can be found in Section 7.5 of Hyndman and Athanasopoulos (2018). Parameter estimation for these models is done through maximum likelihood.

$$\begin{aligned} y_t &= l_{t-1} + b_{t-1} + s_{t-m}\epsilon_t \\ l_t &= l_{t-1} + b_{t-1} + \alpha\epsilon_t \\ b_t &= b_{t-1} + \beta\epsilon_t \\ s_t &= s_{t-m} + \gamma\epsilon_t \end{aligned} \tag{3}$$

5.4.3 Model selection

For both of the aforementioned approaches, the optimal model specification is typically found by minimising an information criterion. In this case we will consider the *Akaike Information Criterion* (AIC) (Stoica & Selen, 2004). The AIC estimates the trade-off between the number of estimated parameters K and the negative log likelihood $\hat{L}$, and is defined in Equation 4. We see that the AIC applies a penalty to the log likelihood based o the number of parameters in the model. For each additional parameter, it judges the marginal improvement in log likelihood. Therefore its minimum can be interpreted as the optimal balance between model complexity and fit.

$$AIC = 2k - 2\ln(\hat{L}) \tag{4}$$

It has been shown that, for a small sample size, the penalty applied by the AIC is too small. The AIC thus underestimates model complexity, which may lead to overfitting (McQuarrie & Tsai, 1998). Therefore we will use the *corrected Akaike Information Criterion* (AICc). The AICc adds an extra penalty, which decreases as sample size n increases.

$$AICc = AIC + \frac{2k^2 + 2k}{n - k - 1}$$

5.5 Supervised Learning

5.5.1 Supervised versus unsupervised learning

Within the framework defined in the previous sections we we apply several machine learning techniques. An important distinction between different types of techniques, is whether they are *supervised* or *unsupervised*. In supervised learning, the targets are known (Love, 2002), while in the

unsupervised case they are not. This difference is illustrated in Figure 12. In supervised learning we are trying to optimise the model to predict these labels. Within supervised learning we consider two cases, categorical or continuous labels. If our task is the former we refer to it as classification, and to the latter as regression.

When the labels are unknown, learning tasks are unsupervised. Two important kinds of unsupervised learning are clustering and dimensionality reduction. Clustering algorithms seek to partition the data into homogeneous groups (Estivill-Castro, 2002). By clustering observations, the algorithm uncovers hidden patterns. This is very useful in, for example, exploratory data analysis or nearest neighbour search. Popular clustering algorithms include K-means clustering (Pelleg & Moore, 1999), Hierarchical clustering (Landau et al., 2011) and DBSCAN (Ester et al., 1996). Dimensionality reduction (DR) seeks to reduce the dimensionality of the feature space (Van Der Maaten et al., 2009). This can be desirable as to avoid the curse of dimensionality, as well as to reduce overfitting. The general idea here is to project the high-dimensional features into low-dimensional space, while preserving their variation. Popular approaches include Principal Component Analysis (Daffertshofer et al., 2004) and Linear Discriminant Analysis (Yu & Yang, 2001). Often, DR techniques are applied sequentially with a clustering algorithm. This can lead to sub optimal results, since the clusters in low-dimensional space may not correctly represent the original variation (Milligan, 1996). To combat this, several techniques, such as Factorial K-means and Reduced K-means have been proposed (Timmerman et al., 2010). These methods improve the outcome by combining the optimisation of the two stages into a single objective function.

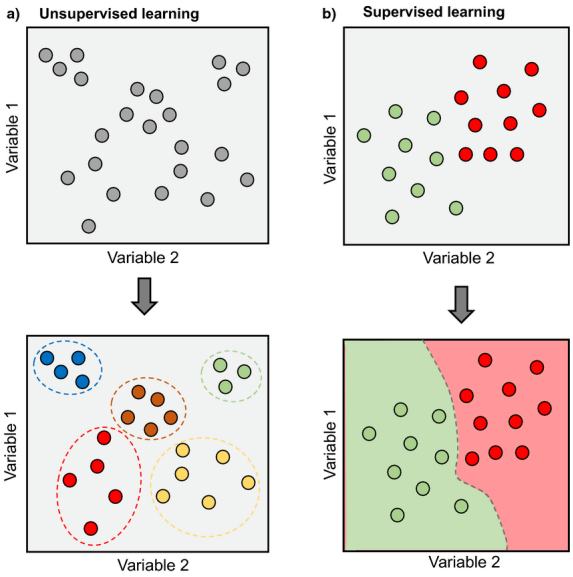

Figure 12: Unsupervised versus supervised learning (Morimoto & Ponton, 2021)

5.5.2 History of supervised learning

In general terms, supervised learning problems optimise a *loss function*, and can be subdivided into two classes: regression and classification. The main difference in these cases is the loss function

that is being optimised. In case of regression this often is the *mean squared loss* or *mean absolute loss*. While in classification *logistic* or *tangent* loss functions can be used. In general the problem can be represented as a minimisation of the loss function L with respect to the dependent variable y and a model output $\hat{y} = f(x)$.

$$\hat{f} = \underset{f}{\operatorname{argmax}} \quad \mathbb{E}\left[L(y, f(x))\right] \tag{5}$$

Regression finds its origins in the 17th century. It was during this time it was discovered, that the accuracy of a prediction could be increased by averaging repeated measurements (Stigler, 1986). Several decades later, Legendre (1806) and Gauss (1809) independently proposed the idea of *least squares* to draw linear lines in noisy astronomy measurements.

5.5.3 Linear Regression

Their method is now known as *ordinary least squares* (OLS), and can be used to estimate parameters in a linear model. Other estimation techniques include *maximum likelihood* (ML) and *(general) method of moments* (GMM). The most basic form of a linear model is known as *simple linear regression* (Zou et al., 2003). The model can be represented as $y = \alpha + x\beta$, where y is known as the *target* or *dependent* variable and x represents the *regressor* or *independent* variable. The coefficient α can be interpreted as the mean value of y when $x = 0$. The coefficient β, can be interpreted as the *total effect* of x on y. That is, all else being equal, in case x increases with 1 unit, y will increase by β units. Note that this *total effect* is not necessarily equal to the *true effect* of x on y, as omitted variable bias may be present. We can extend the model to *multiple linear regression* to account for multiple regressors x_i. In general, for $k \in \mathbb{N}$ regressors this model is defined as per Equation 6. Here we see that y essentially is a weighted sum of x_i, where the weights are represented by β_i.

$$y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i \tag{6}$$

As the name implies, the linear regression model cannot be used to model non-linear relationships. Hence it cannot be used in a classification setting. Furthermore, it can only be used to estimate linear relationships. To overcome these restrictions, many extensions were proposed.

5.5.4 Generalised Linear Model

One class of these extensions is the so-called *generalised linear models* (GLM) (Nelder & Wedderburn, 1972). GLM's can handle different kinds of data such as numerical, categorical and count data. One example is the *logistic regression model* for binary data. This model applies the logistic function as a link function to map continuous input variables to an output between 0 and 1, and is defined in Equation 7. These outputs can be interpreted as class membership probabilities, and hence *logistic regression* is a classification method.

$$\mathbb{P}(Y = 1|X, \mathbf{b}) = \frac{1}{1 + \exp\{\beta_0 + \sum_{i=1}^{k} \beta_i x_i\}} \tag{7}$$

In case the dependent variable represents count data, we can use a Poisson link function. In this case we define the rate λ parameter of the Poisson process as the expectation of Y given X,

which gives us $\lambda = \exp\{\beta_0 + \sum_{i=1}^{k} \beta_i x_i\}$. We then have the probability mass function of the Poisson distribution as:

$$\mathbb{P}(Y = l|X, \mathbf{b}) = \frac{\exp\{y(\beta_0 + \sum_{i=1}^{k} \beta_i x_i) - \exp\{\beta_0 + \sum_{i=1}^{k} \beta_i x_i\}\}}{l!} \tag{8}$$

This model is known as the *Poisson Regression Model*, and has wide applications to model count data Hayat and Higgins, 2014. It has, for instance, been used to model accident- (Joshua & Garber, 1990) and call frequencies (Shen & Huang, 2008). The parameters in GLM's can be estimated using maximum likelihood estimation (MLE). It is based on the concept of likelihood of the data given the assumed model. The likelihood function is defined as the joint probability of the data given the model parameters. In many cases, one can maximise this function in an exact way by differentiating the log-likelihood function with respect to the model parameters, and taking first order conditions (Le Cam, 1990). In the digital age, MLE has become very popular since it can be applied as an iterative procedure. In this case the numerical optimisation methods such as Gradient Descent and Newton-Raphson are applied to find the maximum of the likelihood function (Fletcher, 2013).

5.5.5 (non-) Parametric methods

The GLM's introduced in the previous section are very capable models. Since they depend on distributional assumptions, results may vary if these assumptions are not met. While not necessarily a limitation, this does require extensive verification of these assumptions. Over the last few decades a several *non-parametric* methods have been proposed. Different from parametric methods, these methods do not have any distributional assumptions. This makes them radically different, and very flexible. They can often be applied to both regression and classification tasks (Zhou, 2021). Examples include *decision* and *regression* Trees (Section 5.5.6, Random Forest (Section 5.5.7), Gradient Boosting (Section 5.5.8).

5.5.6 Classification and Regression Trees

As can be seen in Figure 13, a decision tree consists of a series of *choices*. In every node it considers one or more input features and makes a decision based on their values. When a leaf is reached, an output value is generated. More formally, at each node, the tree partitions the outcome space, such that at the leaf an optimal value is predicted. This process is know as *recursive partitioning*, and is complete when additional splits no longer yield an improved loss function. Several algorithms to learn decision trees, such as ID3, C4.5 and MARS have been proposed over the years. Of these, C4.5 is most commonly used (Quinlan, 2014). An essential component of the algorithm is its splitting criterion, *normalised information gain* (Kotsiantis et al., 2007). Its expected value is known as the *mutual information*, and can be interpreted as the reduction of entropy in case the value of a random variable is known. More formally, the information gain (Equation 9) of node η is defined as the Kullback-Leibner divergence between the flat prior distribution $\mathbb{P}(x|I)$ of X and posterior distribution $\mathbb{P}(x|\eta)$ of x (Larose & Larose, 2014).

$$IG(X, \eta) = \mathcal{D}_{KL}[\mathbb{P}(x|\eta), \mathbb{P}(x|I)] \tag{9}$$

The C4.5 algorithm is a recursive algorithm. Upon initialisation, several base conditions are checked. In case all products belong to a single class, we create a leaf node that classifies products according to that class. In case no information gain is found, we move up the tree one level and create a

decision node. If a new class is encountered, we also move up the tree and create a decision node there. In case none of the base cases occur, we proceed to calculate the information gain based on all of the available features. We select the best feature, and split the observations accordingly. We then recurrently initialise the algorithm at the children of the node.

Algorithm 2 C4.5 (Lungu & Pîrjan, 2010)

Base cases:

- All products in single class. Create leaf node.
- No features yield any information gain. Create decision node with class expectation. Move up one level on the tree.
- New class encountered. Create decision node with expectation. Move one level up the tree.

let $\mathcal{F}$ be the set of features
let $\mathcal{I}(f)$ be information gain of f in the current node
Algorithm:
Check base cases
for $f \in \mathcal{F}$ **do**
$I_f = \mathcal{I}(f)$
end for
$f_{best} = \underset{f \in \mathcal{F}}{\operatorname{argmax}} I_f$
Create node at f_{best}
Move down the tree and recur at f_{best}

Decision trees on their own are notoriously unstable and prone to overfitting (James et al., 2013). This is especially the case if the tree is deep. Smaller trees have proven very useful however, as the basis for Random Forest and Gradient Boosting approaches, where a large number of them is used to form an *ensemble*. Note that a tree can be used for both classification and regression problems. When used in classification, each leaf corresponds to a binary or categorical output. In this case the tree is known as a *decision tree*. Similarly, in a *regression tree*, each of the leaves corresponds to a continuous value. This allows (deep) regression trees to approximate very complex functions, which make them very flexible but also prone to overfitting.

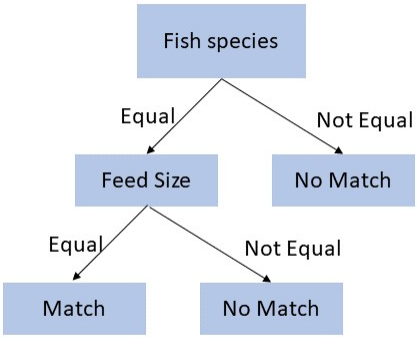

Figure 13: Example of a Decision Tree

5.5.7 Random Forest

To overcome the limitations of single trees, Random Forest was proposed (Ho, 1995). A Random Forest is an example of a *bagging* method. Bagging stands for *bootstrap aggregating*. It refers to combining a large number *base learners* that are combined to create a prediction. In case of regression this aggregation corresponds to taking the mean of all predictions generated, which can be seen in Equation 10. For classification, we tabulate the class predictions made by each of the base learners and select the class with the highest count. This is known as *majority voting*, and is shown in Equation 11. Each of the trees in the Random Forest is trained on a *bootstrapped* sample of the observations. Bootstrapping is a procedure where a new sample is created by sampling with replacement from the old sample. It can be used to asses the stability of a model by evaluating the variation in the predictions that result from each of the samples. This reduces the model's sensitivity to outliers, as they will not always be included in the tree. In addition to bootstrapping, Random Forest uses a second level of randomness. Instead of optimising feature selection for each of the trees, it randomly assigns each node of each tree a subset of features. This ensures trees are not dominated by a few strong features, and that they are uncorrelated.

$$\hat{f}(\mathbf{x}) = \frac{1}{M} \sum_{k=1}^{M} f_k(\mathbf{x}) \tag{10}$$

$$\hat{f}(\mathbf{x}) = \underset{c \in C}{\operatorname{argmax}} \left\{ \sum_{k=1}^{K} \mathbb{1}(f_k(\mathbf{x}) = c) | c \in C \right\} \tag{11}$$

5.5.8 Gradient Boosting

As the name suggests, Gradient Boosting (GB) is a technique that relies on *boosting*. Instead of blindly creating a large ensemble of estimators, as is done with *bagging*, it starts with a single estimator. The algorithm then iteratively adds estimators. First, it assesses the performance of the current ensemble and identifies which observations are classified. It then trains a new estimator focused on correcting these mistakes and adds it to the ensemble. Since the algorithm is an iterative

procedure, it solves the optimisation problem by means of a greedy heuristic (Friedman, 2001). While this approach often outperforms the Random Forest, it also poses a much higher risk of overfitting. In the context of the definition of supervised learning problems in Equation 5, we can define the GB algorithm as an expanding, weighted sum of M base learners h_m. The concept for the GB algorithm is defined in Equations 12 and 13.

$$f_0 = \underset{c}{\text{argmax}} \sum_{i=1}^{N} L(y_i, c) \tag{12}$$

$$f_m = f_{m-1}(x) + \gamma \underset{h_m}{\text{argmax}} \sum_{i=1}^{N} L(y, f_{m-1}(x_i)) \tag{13}$$

The model starts out with by finding an optimal constant $c \in \mathcal{R}$. With each subsequent iteration, it adds another base learner to the ensemble, by fitting this tree h_m to the residuals of the previous ensemble. Given an arbitrary loss function L, the solution to this problem is infeasible. Therefore, *functional gradient descent* is used (Mason et al., 1999). Given the weight γ, we determine the optimal descent direction of L. Since the gradient $\nabla_{f_{m-1}}$ is assumed to be non positive, Equation 14 shows that for sufficiently small γ we have $L(y_i, f_m(x_i)) \leq L(y_i, f_{m-1}(x_i))$. And hence the GB algorithm gets its name from iterative improvements found using the gradient.

$$f_m = f_{m-1}(x) + \sum_{i=1}^{N} \nabla_{f_{m-1}} L(y, f_{m-1}(x_i) + h_m(x)) \tag{14}$$

Popular implementations of the GB algorithm include *extreme gradient boosting*, or XGBoost, and *light gradient boosting machine*, or LGBM. For the purpose of this thesis we will use the LightGBM implementation of the algorithm (Ke et al., 2017). Originally developed by Microsoft, LGBM has been developed with scalability and performance in mind. These features make it especially suitable for large datasets.

5.5.9 Support Vector Machines

Support Vector Machines (SVM) are similar to Tree based methods in the sense that they can be applied both in regression and classification settings. They work in a very different way, however. Where tree based methods rely on *recursive partitioning* to split observations into ever-smaller groups until these groups are homogeneous, SVMs instead transform the *data* such that it can be partitioned using a single *hyperplane*. The SVM searches for a hyperplane such that the margin between the hyperplane and the nearest observation is maximised. Often a good linear boundary such as in Figure 14 is impossible. In this case a kernel is used to transform the data into a higher dimension, where a linear boundary is possible.

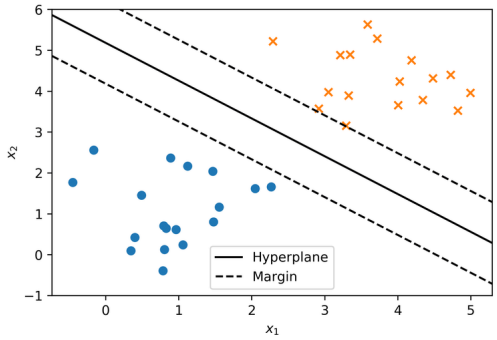

Figure 14: Example of a (linear) SVM (Mir & Nasiri, 2019)

5.6 Evaluation

5.6.1 Forecasting

To evaluate our predictions we will use two different metrics. Firstly, the Root Mean Squared Error (RMSE) will be used to select potential predecessors, as well as compare the different models. To provide a more interpretable metric, we will include the weighted Mean Absolute Percentage Error (wMAPE). wMAPE is a variation of the MAPE. It was proposed for cases where there are (close-to) zero values present in the data. In this case MAPE suffers from division by zero and therefore tends to infinity (S. Kim & Kim, 2016). Finally when we compare our methods against a benchmark, we will compare the fraction of products that improve in terms od RMSE relative to the benchmark.

$$RMSE = \sqrt{\sum_{t=0}^{T}(y_t - \hat{y}_t)^2} \qquad wMAPE = \sum_{t=0}^{T} \frac{|y_t - \hat{y}_t|}{|y_t|}$$

5.6.2 Detection and Classification

For our simulated dataset we have the true transitions. Hence we can evaluate the detection and classification stages individually. Both these problems are binary, hence our evaluation metrics can be derived from the *confusion matrix*, an example of which can be seen in Table 6. It essentially captures the classification performance by counting the outcomes. In case of binary classification, we have four potential outcomes. In case the actual and predicted value are True, the observation is a True Positive (TP). In case both are negative the observation is a True Negative (TN). In case we predict False, while the observation is actually True, it is a False Negative (FN). Similarly, actual False observations that we predict as True are False Positives (FP).

	Predicted	
Actual	True	False
True	TP	FN
False	FP	TN

Table 6: Example confusion matrix

The first of our metrics is the classification Accuracy. In terms of the confusion matrix it is defined as the sum of true positives and true negatives, divided by the total number of observations. Hence, it captures the ratio of correctly predicted observations. While this measure is very intuitive, it has some limitations. The accuracy is highly dependent on the class distribution. In case the dataset is unbalanced, such that one of the classes is larger than the other, bad models can have high accuracy. Suppose we have a dataset that has 99% False observations, as is often the case in applications such as fraud detection. In this case, a model that always predicts the majority class still achieves 99% accuracy.

Since the data in our case is also imbalanced, we will also consider precision and recall. Recall is defined as the ratio of true positives and the total number of actual positives. Similarly, precision is defined as the ratio of true positives and the total number of predicted positives. Precision and recall are often a trade off. A model with very high precision, meaning it will only predict True in case it is absolutely certain, will likely miss a lot of True observations. Hence it will have low recall. Similarly, a model with high recall will likely have low precision. To balance this trade-off we consider the F1 score. This is defined as the harmonic mean between recall and precision.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \qquad Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN} \qquad F1 = 2 \times \frac{precision \times recall}{precision + recall}$$

6 Experimental Setup and Results

In this section we will discuss the setup and results of our experiments. First we will introduce the different experiments, and describe how they were implemented. We will then present and discuss our results.

6.1 Experimental setup

To evaluate our methods and answer our research questions we will perform several experiments. These experiments are split into three parts, each corresponding to a different question. In the first two parts, we will evaluate the transition detection and prediction performance of the method. As we have no ground *truth* for the fish feed dataset, we will only perform these experiments on the simulated data. Finally, we will evaluate the forecasting performance of our method. In this case both datasets are considered.

First, we will evaluate the historical transition detection performance for products introduced during the detection phase. This will be done using the classification metrics introduced in section 5.6.2. We will then train our predictive models using these transitions to predict transitions during the prediction phase. After we have evaluated our transition predictions we will consider the forecasting performance of our methods.

After evaluating the detection and prediction performance using the simulated data, we will answer our third research question and evaluate the forecasting performance. For the simulated data, we will consider three different experiments. First, we will consider the forecasting performance for recent products. That is, we evaluate the effect of historical transition detection on forecasting performance. In this analysis, we compare the RMSE, wMAPE and the percentage of improved forecasts of our different method configurations.

Our second and third experiment evaluate the effect of predicting ongoing transitions. Hence we only consider newly introduced products. In the second experiment we compare the forecasted joint demand to the actual demand of the new product. In this case, intuitively, the error should start out high, and progressively decrease as the transition completes. While it may seem an odd choice to compare the forecasted *joint* demand to the actual demand for only the new product, there is good reason to do so. Suppose the wrong predecessor is predicted and this predecessor dominates the *new* product in terms of quantity. In this case the forecasting performance might still be very good, since the error introduced by the new product does not effect the forecast. A comparison of the predicted *joint* demand to the actual *new* demand, meanwhile, will show a very large error in this case. Hence this experiment is robust with respect to a dominating incorrect predecessor. It turn, however, this choice does sacrifice some information in the first (few) month(s), as the error will never be small in this period. Finally, in our third experiment we will apply our predicted transition profiles to the *joint* forecast and compare the demand attributed to the new product with the actual demand for the new product. This will allow us to evaluate the forecasting performance of our entire method.

We then turn to the fish feed data. In this case we cannot directly evaluate the detection and prediction performance, as we do not have the true transitions. Instead, we will evaluate our method based on its effect on forecasting performance. To answer our first research question, we will estimate the effect of including a transition detection method in the forecasting pipeline. Next,

we will evaluate the contribution of predicting ongoing transitions. Finally, we estimate the total effect of our method on forecasting performance.

Our experiments are implemented using the Python3 programming language (Van Rossum & Drake, 2009). The the general framework of the method is coded by hand, and the Pandas *DataFrame* (pandas development team, 2020) to store and manipulate our data. On occasion Numpy operations (Harris et al., 2020) are also used. To create the plots and figures in this thesis we use the Matplotlib (Hunter, 2007) library. For the machine learning and forecasting methods, some common packages were used. Forecasting was done, using the ETS and SARIMA implementations in the Statsmodels library (Seabold & Perktold, 2010). The Nearest Neighbour algorithm, as well as the Random Forest, are implemented in the Sci-kit learn library (Pedregosa et al., 2011). For Gradient Boosting, we used LigthGBM (Ke et al., 2017).

First we will consider the results for the individual components of our method, using the simulated dataset as described in Section 4.2. We will then apply the method as a whole to our simulated data and fishfeed dataset. In our results tables we will abbreviate the model specification. These abbreviations are described below.

Detection Methods

Model	Description
Random	Random transition classification. This model serves as a benchmark and to illustrate the data difficulty parameter D.
True	No detection method, instead True transitions passed to classification stage. Used to evaluate the individual performance of our classification methods.
S	First stage detection using SARIMA models, second stage select best forecast.
E	First stage detection using ETS models, second stage select best forecast.
S+NN	First stage forecast detection using SARIMA models, second stage select Nearest Neighbour.
E+NN	First stage forecast detection using ETS models, second stage select Nearest Neighbour.

Classification Methods

Model	Description
True	Serves as a benchmark to evaluate forecasting performance with perfect detection and classification.
RF	Classification using Random Forest
GB	Classification using Gradient Boosting.

Forecasting Methods

Model	Description
S	SARIMA forecast
E	ETS forecast

6.2 Parameter Tuning and Restrictions

One of the main advantages of our method, is the limited number of parameters that need to be tuned. Nevertheless, each of the stages requires some tuning. The most parameters in the detection stage are the choice of validation period. The validation period is defined in terms of its length

and starting point, relative to an product's introduction date. In case the starting point is far away from the introduction, the effect of a predecessor might be limited. Hence the transition would be more difficult to detect. In case the starting point is very close to the introduction date, the transition might not yet be complete. This will result in a large prediction error, and hence the transition might go undetected. Similarly, while a larger validation period might make detection more robust, it also requires more data before a detection can be made. After experimentation, we used a validation period of 4 months, starting 4 months after a product was introduced. This implies that a total of 8 months of available data is needed to detect a transition using our forecast based approach.

With pair classification and profile regression we mostly used the LGBM and Random Forest default settings. Further optimisation was deemded unnecessary, since the performance was near perfect with default parameters with the simulated data. In case of the fish feed data, the very small sample size made improvement with different parameters unlikely. Due to this small sample size we limited the number of estimators in the ensembles to 100 for the simulated data and 25 for the fish feed data.

For both forecast methods and datasets, no significant tuning was done. In case of the ETS model, all *statsmodels* default parameters were used. For SARIMA we found that in all cases, based on the AICc, a very simple model was selected. The maximum order for p, q, P and q was rarely above 2. To improve the run time of our approach, we therefore set their maximum order to 2. Similarly, in no case was the differencing order d and D higher than 1. Hence we restricted the maximum differencing order to 1.

The criteria used to select candidate turned out to have substantial impact on the results. As described in section 5.2, these criteria represent rules to determine a candidates eligibility. In addition to disqualifying products based their characteristics, we tuned three more restrictions: distance, overlap and contribution. These are used find a balance between precision and computational feasibility, and are chosen in a generic way, such that they can be applied to multiple datasets. To find the optimal values we applied a grid search.

- Distance - The time between the discontinuation of a candidate and the introduction of the new product may not be greater than 12 months.
- Overlap - The time between the introduction of the new product and discontinuation of a candidate and the new product may not be greater than 6 months.
- Contribution - A candidate must have a minimum of 6 months sales history before introduction of the new product.

6.3 Detecting historical transitions

The results for our historical transitions can be found in Table 8. To evaluate the performance of our classification methods, we included a *Random* assignment benchmark. In this case, if there more than one candidate according to the criteria outlined in the previous section, the predecessor is assigned randomly. The data difficulty parameter D represents the number of *sub-groups* in each of our disjoint groups. A higher value of this parameter, therefore, implies that a match must be found among more candidates. This increases the difficulty of our detection task, which is reflected in the

decreasing F1 score for the random benchmark with increasing difficulty. Despite this increasing difficulty, the detection performance remains high. We see that with detection based on SARIMA, the predecessor is almost always detected correctly. When using the ETS-model we find a slight improvement. A further improvement is found when we combine the forecast based methods with Nearest Neighbours. Here, for each of the datasets, the SARIMA and Nearest neighbour method has a slight advantage over the ETS based method. As the detection performance is better for methods based on both forecast and nearest neighbour search, we will only consider the *S+NN* and *E+NN* detection methods in the further stages.

D	Model	Accuracy	Precision	Recall	F1
2	Random	0.57	0.63	0.386	0.479
	S	0.913	0.882	0.877	0.879
	E	0.899	0.888	0.825	0.855
	S+NN	0.979	0.974	0.968	0.971
	E+NN	0.955	0.972	0.903	0.936
3	Random	0.619	0.526	0.345	0.416
	S	0.907	0.840	0.840	0.840
	E	0.885	0.826	0.763	0.793
	S+NN	0.974	0.955	0.955	0.955
	E+NN	0.959	0.965	0.891	0.927
4	Random	0.663	0.431	0.301	0.355
	S	0.922	0.828	0.817	0.822
	E	0.923	0.862	0.778	0.818
	S+NN	0.986	0.974	0.961	0.967
	E+NN	0.967	0.971	0.876	0.921
5	Random	0.687	0.338	0.253	0.289
	S	0.937	0.824	0.824	0.824
	E	0.904	0.763	0.676	0.717
	S+NN	0.973	0.926	0.926	0.926
	E+NN	0.970	0.969	0.858	0.910

Table 8: Historical transitions detection performance

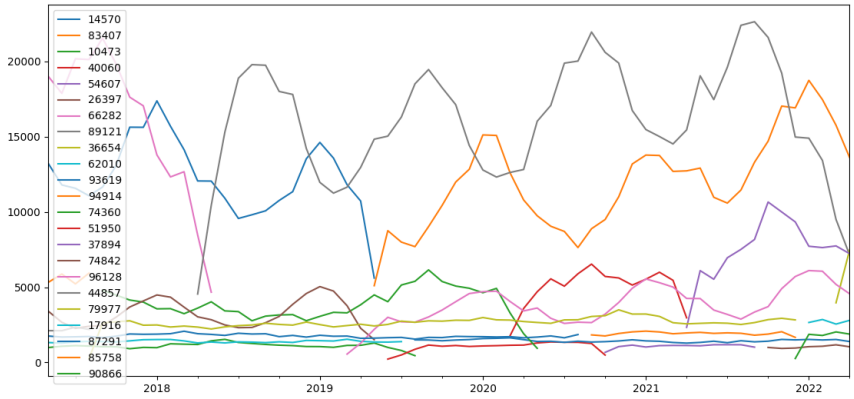

Figure 15: Products in group 11

To further illustrate these results we will now consider an example from group 11 in the simulated dataset with *difficulty* 4 . For the purpose of this example we will use ETS forecasting. An overview of this product group can be see in Figure 15. After applying the candidate selection criteria, we are left with potential 3 candidates for new product 54607. For each of these we apply the ETS+NN method. The results can be found in Figure 16 and Table 9. In Table 9 we see the MSE and wMAPE scores for both candidates, as well as the P-value for the Diebold Mariano test described in Section 5.2.1. Note that two out of three candidates significantly (5% level) outperform the forecast without a predecessor. Hence we will apply the Nearest Neighbour algorithm, based on the product characteristics in Table 10. A visual inspection of the table clearly shows that for product 40060 is closer to our new product than product 74360. This is consistent with the prediction made by the NN algorithm, which finds product 40060 is the predecessor of product 54607.

Predecessor	MSE	wMAPE	P-value DB
None	111	0.090	-
74360	87	0.057	0.02
40060	67	0.035	0.00
87291	334	0.288	0.00

Table 9: Predecessor detection for product 54607

	Content					Category					
Product ID	1	2	3	4	5	1	2	3	4	5	Group
54607	0.136	0.121	0.045	0.352	0.346	0	0	7	1	1	11
40060	0.184	0.163	0.000	0.294	0.407	0	0	7	1	1	11
74360	0.182	0.121	0.301	0.266	0.130	0	1	2	2	0	11

Table 10: Candidates characteristics for new product 54607

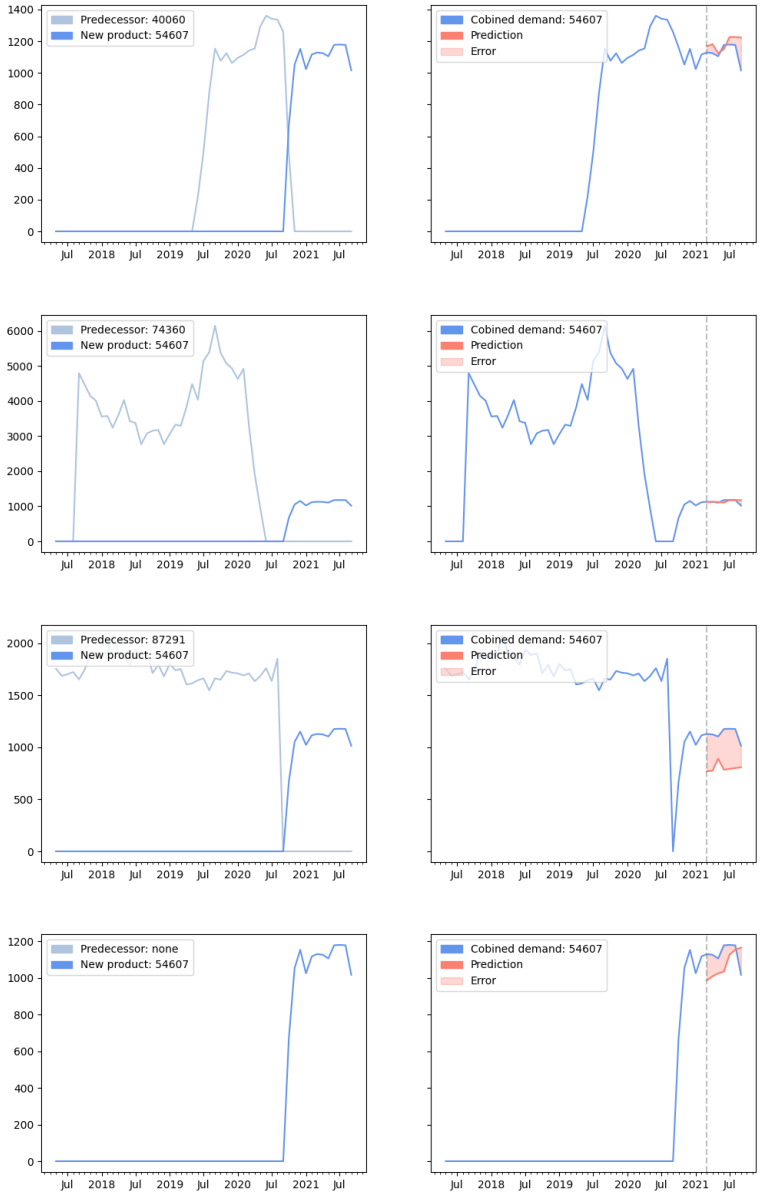

Figure 16: Validation forecast with different candidates (note: scales differ between plots)

6.4 Predicting Ongoing Transitions

We will now evaluate our classification methods for new products. For the first four months of a new product's lifespan, we will evaluate the classification performance using gradient boosting and

random forest techniques. In Table 11 we see the classification performance for the first month. It can immediately be seen that the classification performance is close to perfect. The difference between the two classification methods seems insignificant. The results for the following months can be found in Tables 20, 21 and 22 respectively, which are included in Appedix B. The performance in this case is very similar to the performance for the first month.

	Detect	Random Forest				Gradient Boosting			
D	Model	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
2	True	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	S+NN	1.0	1.0	1.0	1.0	0.989	0.976	0.976	0.976
	E+NN	1.0	1.0	1.0	1.0	0.989	0.976	0.976	0.976
3	True	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	S+NN	1.0	1.0	1.0	1.0	0.987	0.956	0.956	0.956
	E+NN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4	True	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	S+NN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	E+NN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
5	True	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	S+NN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	E+NN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Table 11: Ongoing Transitions classification first month

Coming back to the group 11 example described in the previous section, we now have a *new* product that needs to be predicted. To predict the correct predecessor for product 26397, we will use the RF classifier. After applying our candidate selection rules we were left with three candidates, whose characteristics can be seen in Table 12. As described in Section 5.3, we transform these product characteristics to transition characteristics. We then apply the RF classifier trained on historical transitions. The transition characteristics, as well as their predicted transition probabilities can be found in Table 13. Based on the probabilities, we clearly predict product 54607 to be the predecessor of product 26397. In addition, after applying their transition characteristics to our RF regressor, we predict an immediate transition upon introduction of the new product. Both these findings are consistent with the transition seen in Figure 15.

	Content					Category					
Product ID	1	2	3	4	5	1	2	3	4	5	Group
26397	0.179	0.191	0.005	0.284	0.340	0	0	7	1	1	11
79977	0.345	0.072	0.152	0.179	0.252	0	1	1	0	0	11
54607	0.136	0.121	0.045	0.352	0.346	0	0	7	1	1	11
85758	0.345	0.219	0.240	0.031	0.163	1	2	3	2	1	11

Table 12: Candidate characteristics for new product 26397

Predecessor	Numerical difference					Categorical equal						Transition
ID	1	2	3	4	5	1	2	3	4	5	Group	Probability
79977	0.166	0.119	0.147	0.105	0.088	1	0	0	0	0	1	0.01
54607	0.043	0.070	0.040	0.068	0.006	1	1	1	1	1	1	0.95
85758	0.166	0.028	0.235	0.031	0.177	1	1	1	1	1	1	0.04

Table 13: Transition characteristics for new product 26397

6.5 Forecasting performance

We will now discuss the forecasting performance of our methods on our datasets. First we will consider the performance on simulated data. Next we will discuss our fish feed data.

6.5.1 Simulated Data

First we will look at the forecasting performance for each of our simulated datasets. Recall from Section 4.2 that each dataset is defined by its *difficulty* parameter. This parameter represents the number of different product families in each product group. Hence a higher *difficulty* value results in more candidates and therefore more difficult detection. As described in section To evaluate the contribution of historical transitions identified in section 6.3, we look at the forecasting performance for recent products. That is, we consider the group of products for which a predecessor has been identified. Depending on the dataset, the number of recent products is around 130. In Table 14 we compare the performance of our detection methods against the performance of true and no predecessor detection, across three metrics: RMSE, wMAPE and % improved. Recall that the third metric is defined as the fraction of products that are improved relative to to the baseline, in this case *no detection*. For each of the datasets, we underline the forecast specification that performed best with our given metrics.

For both forecasting methods, and all four datasets, we see that our methods perform better than the benchmark without predecessor detection. Furthermore, in most cases the performance is very close to the *True classification* benchmark. Between the two forecasting methods, ETS models seem to perform slightly better than SARIMA models, despite the higher detection accuracy we found in the previous section. Note that we see the same difference for the models with *true* detection. One possible explanation is that our simulated DGP is multiplicative. The relevance of this bias is limited, however, since the difference between the methods is very small compared to the improvement found relative to the no predecessor detection benchmark.

		difficulty = 2			*difficulty = 3*		
Model specification		Metrics			Metrics		
D	F	RMSE	wMAPE	% improved	RMSE	wMAPE	% improved
True	S	1597	0.137	1	3420	0.167	1
S+NN	S	3135	0.198	1	3557	0.196	1
-	S	7922	0.992	-	10387	0.993	-
True	E	1013	0.095	0.916	1480	0.108	0.886
E+NN	E	1349	0.103	0.884	2217	0.114	0.938
-	E	3285	0.297	-	4300	0.318	-
		difficulty = 4			*difficulty = 5*		
Model specification		Metrics			Metrics		
D	F	RMSE	wMAPE	% improved	RMSE	wMAPE	% improved
True	S	1653	0.157	1	5249	0.213	1.0
S+NN	S	2119	0.19	0.993	6314	0.297	1
-	S	7546	0.992	-	10241	0.993	-
True	E	1539	0.081	0.971	5819	0.165	0.902
E+NN	E	2045	0.087	0.942	6191	0.187	0.962
-	E	6544	0.338	-	13450	0.319	-

Table 14: Forecasting performance for recent products

Recall the example introduced in Section 6.3, where we identified product 40060 as a predecessor for product 54607. Figure 17 shows the subsequent forecast improvement for the period after the validation period. We see that, especially in earlier months, the predecessor has a substantial effect on the forecast error. In this instance, the wMAPE decreases from 0.15 to only 0.04.

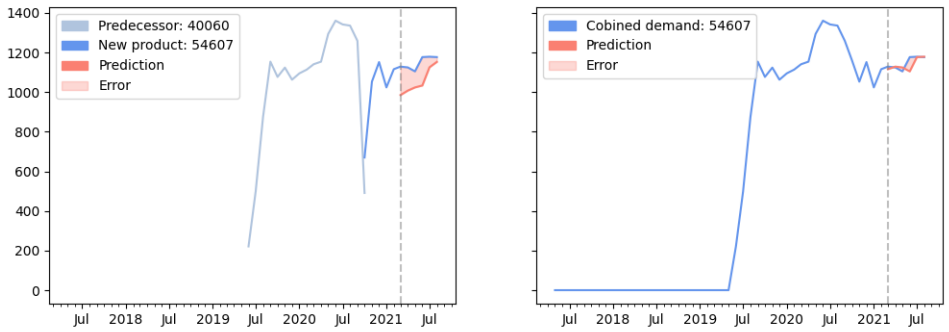

Figure 17: Forecast improvement for product 54607

We will now consider the results for our newly introduced products. Table 15 shows the forecasting performance of our joint method relative to the actual demand. As expected we see a relatively large error in the first month, due to the ongoing transition. In subsequent months the error decreases, as more transitions are finished. The performance of the method does not seem to depend on the classification method used. As can be seen in Table 15, for all detection and

forecasting specifications, the performance is equal or very similar for both Random Forest and Gradient Boosting. This implies that the two classification methods are equivalent for this task. With regards to the forecasting method, again the ETS based method performs slightly better than SARIMA in most cases. In addition, across the datasets, the performance for both ETS and SARIMA based methods does not seem to decrease for increasing data difficulty. This implies that the method does not have trouble selecting correct predecessors from a larger group of candidates, which is supported by the consistent classification results found in the previous section.

Table 15: Forecasting new products without transition profile

difficulty = 2

Specification			Month 1		Month 2		Month 3		Month 4	
D	C	F	RMSE	wMAPE	RMSE	wMAPE	RMSE	wMAPE	RMSE	wMAPE
S+NN	RF	S	3204	0.836	2167	0.666	2281	0.475	1672	0.486
S+NN	GB	S	3204	0.837	2153	0.666	2198	0.475	1672	0.486
E+NN	RF	E	3179	0.837	2111	0.640	2183	0.459	1607	0.477
E+NN	GB	E	3179	0.838	2103	0.639	2105	0.461	1608	0.479
True	True	S	3089	0.814	1384	0.433	1072	0.359	1222	0.343
True	True	E	3307	0.831	1296	0.405	1173	0.372	1236	0.354

difficulty = 3

Specification			Month 1		Month 2		Month 3		Month 4	
D	C	F	RMSE	wMAPE	RMSE	wMAPE	RMSE	wMAPE	RMSE	wMAPE
S+NN	RF	S	5944	0.812	5900	0.589	5688	0.587	3649	0.515
S+NN	GB	S	6674	0.829	5732	0.569	5296	0.587	3521	0.498
E+NN	RF	E	5766	0.807	5279	0.559	5485	0.571	3430	0.551
E+NN	GB	E	5766	0.807	4452	0.543	5117	0.577	3689	0.551
True	True	S	5532	0.797	2531	0.472	1586	0.343	1564	0.318
True	True	E	5433	0.785	2751	0.476	1573	0.347	1704	0.324

difficulty = 4

Specification			Month 1		Month 2		Month 3		Month 4	
D	C	F	RMSE	wMAPE	RMSE	wMAPE	RMSE	wMAPE	RMSE	wMAPE
S+NN	RF	S	9873	1.137	4095	0.641	4021	0.611	3182	0.507
S+NN	GB	S	9860	1.137	4109	0.641	4027	0.611	3191	0.508
E+NN	RF	E	9322	1.123	4060	0.634	3821	0.603	3274	0.513
E+NN	GB	E	9322	1.123	4016	0.634	3829	0.603	3389	0.513
True	True	S	7471	1.032	4330	0.616	2729	0.42	1706	0.338
True	True	E	7324	1.030	3794	0.588	2469	0.429	1665	0.345

difficulty = 5

Specification			Month 1		Month 2		Month 3		Month 4	
D	C	F	RMSE	wMAPE	RMSE	wMAPE	RMSE	wMAPE	RMSE	wMAPE
S+NN	RF	S	6403	1.0	4037	0.649	4330	0.626	4594	0.655
S+NN	GB	S	6403	1.0	3950	0.649	3994	0.626	3850	0.655
E+NN	RF	E	6520	1.002	4173	0.661	4223	0.614	4568	0.649
E+NN	GB	E	6520	1.002	4083	0.661	3896	0.614	3828	0.649
True	True	S	4933	0.865	2927	0.546	1958	0.312	1565	0.249
True	True	E	4872	0.868	3001	0.562	1853	0.331	511	0.242

After we apply the predicted transition profiles, we are left with the results in Table 16. We now see that the error in the first month is much lower, since part of the demand is now allocated to the preceding product. Despite this decrease for most of the first months, we do not observe a further decrease in the subsequent months. With a wMAPE between 0.5 and 0.6 in most cases, the error remains high relative to the true benchmark, which is on average just below 0.3. The influence of the chosen classification (and profile regression) methods remain of limited impact, similar to the previous table. We also see that the difference in forecast error remains small between the forecasting methods, with ETS performing slightly better again.

Table 16: Forecasting performance for new products with transition profile

difficulty = 2

Specification			Month 1		Month 2		Month 3		Month 4	
D	C	F	RMSE	wMAPE	RMSE	wMAPE	RMSE	wMAPE	RMSE	wMAPE
S+NN	RF	S	1126	0.501	987	0.377	2147	0.428	3872	0.599
S+NN	GB	S	1034	0.489	925	0.394	2083	0.438	3444	0.612
E+NN	RF	E	1058	0.485	900	0.363	2043	0.411	3824	0.598
E+NN	GB	E	985	0.468	893	0.369	2000	0.431	3355	0.610
True	True	S	372	0.264	741	0.279	572	0.273	1146	0.319
True	True	E	733	0.326	644	0.248	670	0.291	1154	0.331

difficulty = 3

Specification			Month 1		Month 2		Month 3		Month 4	
D	C	F	RMSE	wMAPE	RMSE	wMAPE	RMSE	wMAPE	RMSE	wMAPE
S+NN	RF	S	3360	0.536	2507	0.435	5521	0.566	4822	0.555
S+NN	GB	S	4797	0.587	2351	0.456	5174	0.579	4112	0.567
E+NN	RF	E	3058	0.562	2171	0.389	5335	0.554	5207	0.572
E+NN	GB	E	3084	0.572	2167	0.383	4986	0.564	4452	0.591
True	True	S	1168	0.298	1201	0.288	1084	0.261	1158	0.267
True	True	E	740	0.265	1476	0.286	1021	0.268	1337	0.274

difficulty = 4

Specification			Month 1		Month 2		Month 3		Month 4	
D	C	F	RMSE	wMAPE	RMSE	wMAPE	RMSE	wMAPE	RMSE	wMAPE
S+NN	RF	S	1872	0.497	3513	0.479	3948	0.561	3667	0.613
S+NN	GB	S	1896	0.493	3440	0.477	4041	0.532	3501	0.613
E+NN	RF	E	1113	0.462	2118	0.459	3145	0.515	2945	0.594
E+NN	GB	E	1160	0.471	2576	0.482	2962	0.512	2892	0.599
True	True	S	368	0.26	683	0.249	593	0.245	703	0.259
True	True	E	319	0.256	812	0.273	848	0.280	722	0.272

difficulty = 5

Specification			Month 1		Month 2		Month 3		Month 4	
D	C	F	RMSE	wMAPE	RMSE	wMAPE	RMSE	wMAPE	RMSE	wMAPE
S+NN	RF	S	2686	0.593	913	0.371	3876	0.57	4591	0.651
S+NN	GB	S	2666	0.595	960	0.381	3554	0.559	3834	0.644
E+NN	RF	E	2816	0.595	945	0.369	3824	0.550	4565	0.645
E+NN	GB	E	2794	0.596	995	0.379	3559	0.549	3806	0.636
True	True	S	237	0.206	639	0.270	731	0.269	560	0.244
True	True	E	319	0.238	688	0.275	765	0.286	479	0.226

We now return to the Group 11 example. Recall from 6.4 that we predicted product 54607 to be a predecessor of product 25397. In Figure 18 we see that the ETS forecast without a predecessor behaves as the mean of the available observations. For the first month, the forecast is zero as no information is available. From the second month onwards, the forecast performs better but is slow to adjust. On the right we see the forecast with a predecessor included. Here we see a dramatic improvement in the forecast. The wMAPE in the first case is 0.29, mainly due to the very large error in the first month. With the predecessor included, the wMAPE drops to 0.07.

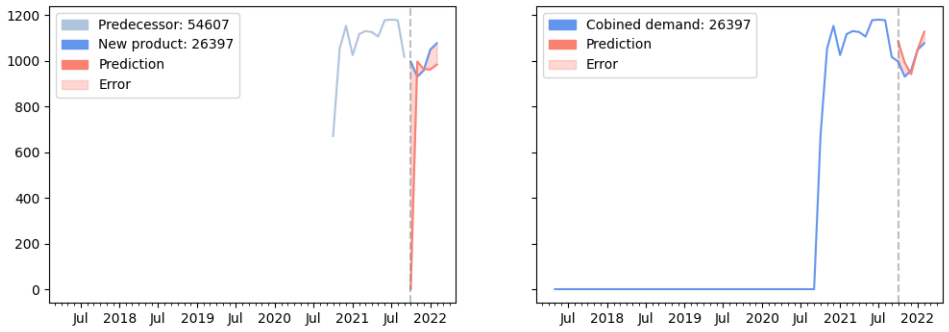

Figure 18: Forecast performance after introduction for product 26397

6.5.2 Fish Feed Data

We now look at the results for the fish feed data. First we consider the transitions that were detected by our methods. Since we cannot compare these to a *ground truth*, we will consider the mean forecasting performance for these products. In Table 17 we see that the SARIMA based method performs significantly better than the ETS based method. Not only is the number of transitions detected higher, the mean RMSE and wMAPE scores are also lower. Hence we obtain a better forecast for more products. Note that 5 detected transitions is limited, especially given that there were 24 products introduced during this period. This very small sample size will make the classification of ongoing transitions very difficult. To further illustrate the improvement at an product level, we see three of the transitions detected by SARIMA in Figure 19.

Specification					
Detect	Forecast	#found	RMSE	wMAPE	% improved
S+NN	S	5	492	0.642	21.0
S	S	1	551	0.891	4.2
-	S	-	566	0.923	-
E+NN	E	3	507	0.691	12.6
E	E	0	675	0.947	0
-	E	-	675	0.947	-

Table 17: Forecasting performance for products with detected predecessor

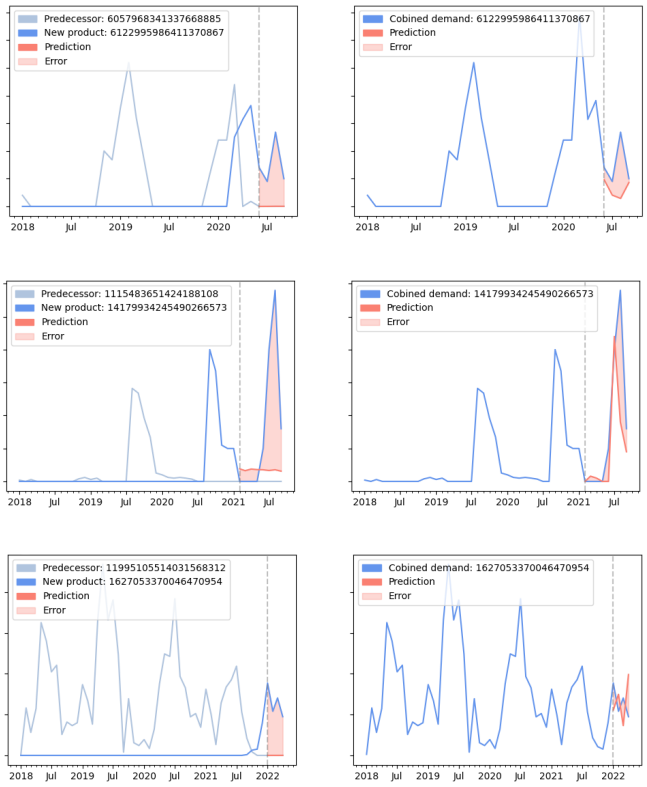

Figure 19: Validation period forecast improvement (SARIMA)

The results for new products can be found in Table 18. Here we evaluate the forecasting performance of our models with predecessor detection, regular SARIMA and ETS models as well as a zero and naive benchmark. The zero benchmarks considers a forecast that is always 0, whereas the naive benchmark uses the actual value of last period as the prediction. For the first month we see that the SARIMA, ETS and Naive predictions are equal to the zero predictions. This makes sense as no data is available at this point. The detection based methods perform very badly, with a RMSE more than twice as large as the zero benchmark. This implies that the sample size discussed in the previous section is indeed too small and the model has trouble predicting true transitions. For subsequent months we see that the detection based models improve significantly as there is more demand history to consider. They are, however, outperformed by SARIMA and ETS as well as the two benchmarks. In the 4th month the Naive benchmark performs the best. This implies that none of the models is able to make an accurate forecast.

Specification			Month 1		Month 2	
Detect	Forecast	Split	RMSE	wMAPE	RMSE	wMAPE
S+NN	S	GB	1895	1.912	1334	1.272
S+NN	S	RF	1905	1.867	1345	1.255
-	S	-	961	1.000	983	0.892
E+NN	E	GB	2346	2.160	1547	1.634
E+NN	E	RF	2298	2.051	1523	1.651
-	E	-	961	1.000	983	0.892
Predict zero			961	1.000	1345	1.000
Predict naive			961	1.000	983	0.892

Specification			Month 3		Month 4	
Detect	Forecast	Split	RMSE	wMAPE	RMSE	wMAPE
S+NN	S	GB	1353	1.314	1176	1.152
S+NN	S	RF	1289	1.205	1246	1.241
-	S	-	961	1.000	983	0.892
E+NN	E	GB	1691	1.024	1101	0.948
E+NN	E	RF	1694	1.024	1086	0.946
-	E	-	918	0.871	962	0.831
Predict zero			1872	1.000	1651	1.000
Predict naive			1057	0.862	947	0.764

Table 18: Forecasting performance for new products

Finally we will consider a 6-month simulation of different forecast methods, the results of which can be found in Table 19. We will compare our methods relative to the *expert* forecast benchmark. As described in Section 4, this forecast was provided to us by the company. It is not based on a statistical model. Instead it is created by the sales department, and is the aggregated expected demand based experience of the company sales department. The models we will consider are SARIMA and ETS based forecasting methods, with and without predecessor detection. In the case of predecessor detection, we will also consider GB or RF based transition prediction.

We see that almost all methods, except for the regular ETS model, manage to outperform the *expert* benchmark. This indicates that the benchmark can be improved upon, and that the company can benefit from a statistical forecasting methods. Between SARIMA and ETS based methods, we see a substantial difference. Both in terms of RMSE and wMAPE, the SARIMA based methods perform much better. The model that yields the greatest improvement on the benchmark is SARIMA with predecessor detection, without transition prediction. This result indicates that our transition prediction methods do not perform well on this dataset. This result is consistent with the results found in Tables 17 and 18. A likely cause for this result is the small number of historical transitions that were detected. This leads to poorly trained classifiers and incorrectly predicted transitions.

Detect	Forecast	Split	RMSE	wMAPE	% improved
S+NN	S	GB	387	0.743	78.4
S+NN	S	RF	393	0.758	73.6
S+NN	S	-	393	0.759	73.0
-	S	-	401	0.834	64.9
E+NN	E	GB	430	0.896	33.7
E+NN	E	RF	431	0.896	33.9
E+NN	E	-	423	0.887	36.2
-	E	-	450	0.924	11.0
-	Expert	-	436	0.903	-

Table 19: 4 months average forecasting performance

Finally we will consider a transition that was detected by both methods. In Figure 20 we see that both methods result in a significant improvement of the forecast. Note that without considering the forecast SARIMA performs much better than ETS in this case. After we incorporate the transition, the results are closer. In the case of SARIMA, we initially have a wMAPE of 0.47. After incorporating the transition into the forecast, this drops to 0.19. For ETS the baseline wMAPE is even larger, with a value of 0.55. With the predecessor we find a wMAPE of 0.32. While both methods improve, SARIMA clearly performs better for this example.

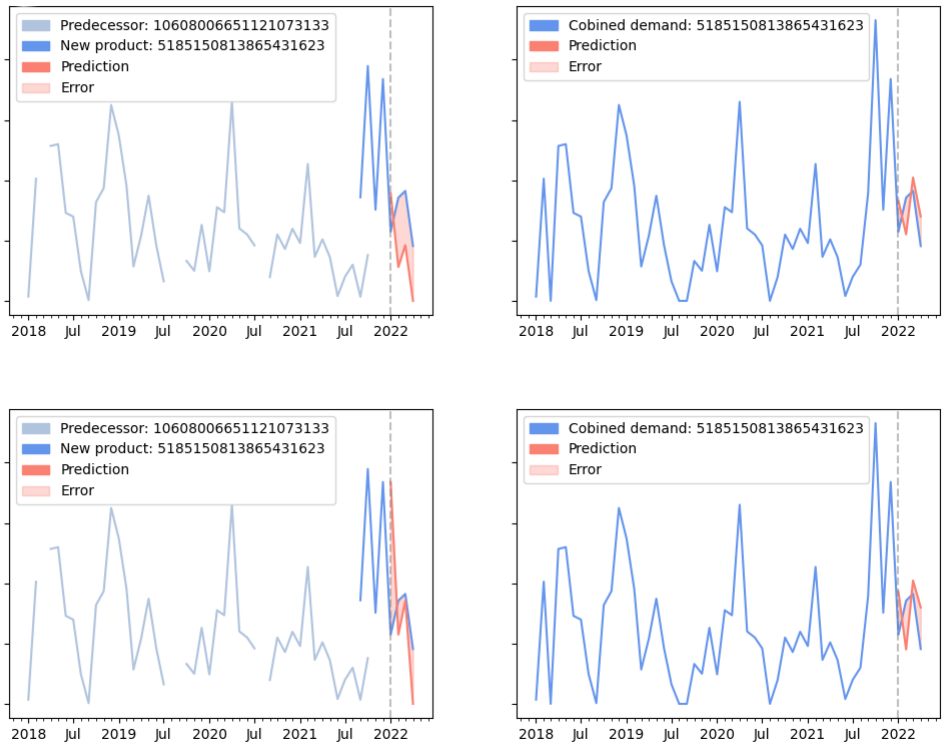

Figure 20: Forecast with transition for SARIMA (top) and ETS (bottom)

7 Conclusion

In this thesis we set out to answer the following research question: *Can unrecorded product transitions be identified and used to improve forecasting performance.* To guide this process we considered several sub questions. The first of which is *how can we optimally identify historical product transitions*? To answer this question we developed a framework to detect historical transitions and performed several experiments which considered several methods within our broader framework. We considered four model specifications, SARIMA and ETS on their own, as well as a combination with the Nearest Neighbour algorithm. The detection method performed very well in our simulation experiments. Our SARIMA+NN model was able to achieve an F1 score of over 0.90 in all scenarios. For the experiments with the fish feed data, given the quality of the data, we also found promising results. Again the SARIMA+NN model performed the best. In this real-world dataset, however, the number of transitions was limited. Hence the substantial improvement for a small number of items had a limited effect on forecasting performance across the complete dataset. Despite this limited effect overall, we were able to improve the forecast of each of the items with a detected predecessor.

The second question we considered was *how can we optimally identify ongoing product transitions?* For this task we considered two methods: Random Forest and Gradient Boosting. Both algorithms showed very similar results. In our simulated experiments we found no significant difference in the performance between these methods. There was, however, a substantial gap between the forecasting performance of our methods and the *True classification* benchmark. This is despite achieving near perfect classification results. With the fish feed dataset results are not good. The predictions for newly introduced products failed to match the performance of simple benchmarks. This is most likely due to the very small sample sizes the models were trained on. These small samples combined with a large variation in the item characteristics meant that an accurate prediction of ongoing product transitions was next to impossible. Hence we must conclude that further experiments are needed, and the performance of our method with newly introduced items is highly dependent on the size and quality of our data.

The final question considered was *which forecasting method is best suited to deal with the various challenges of this prediction task?* In most of our experiments, the results for our forecasting methods were very similar. The simulated data favoured ETS models slightly, but may be due to the multiplicative DGP. Nevertheless, the difference between the methods was very small compared to the overall improvement. With the fish feed data we found that SARIMA performed slightly better.

Based on these results, we conclude that unrecorded product transitions can be identified in some cases. Historical transitions can easily be used. However, there are some limitations. The more recent a transition occurs, the larger its impact on forecasts will be. Too recent, however, and it becomes very hard to accurately detect or predict. Ongoing transitions can be used to forecast demand in case two conditions are met. Firstly, the dataset should be of sufficient size. This ensures that the classification methods have sufficient samples to train on. Furthermore, the data on item characteristics should be sufficiently detailed, such that classification methods can differentiate between true and false transitions. If these conditions are met, then ongoing transitions can be used to improve forecasting performance. Gradient boosting and random forest will give similar results. To further refine the method we suggest applying the method to different

datasets with the aforementioned requirements. We suspect that short sale cycle products that go through iterative improvements are good candidates. Examples include the sale of new vehicles and consumer electronics, which are typically updated yearly. Another option is to expand the framework to account for more complex transitions, such as one-to-multiple or multiple-to-one. This does, however, increase the search space exponentially, and could make the problem intractable.

ERASMUS UNIVERSITEIT ROTTERDAM

A Detection Confusion Matrices

The detection methods considered in the table are: Random detection (R), SARIMA or ETS based detection (S, E), and combined forecasting nearest neighbours based detection (S+NN, E+NN).

Predicted

difficulty = 2

Actual	R		S		E		S+NN		E+NN	
	T	F	T	F	T	F	T	F	T	F
T	51	81	120	12	113	19	129	3	121	11
F	17	53	12	58	11	59	3	67	3	67

difficulty = 3

Actual	R		S		E		S+NN		E+NN	
	T	F	T	F	T	F	T	F	T	F
T	41	92	118	15	111	22	130	3	122	11
F	25	57	14	68	14	68	2	80	3	79

difficulty = 4

Actual	R		S		E		S+NN		E+NN	
	T	F	T	F	T	F	T	F	T	F
T	40	91	112	19	102	29	126	5	115	16
F	18	93	16	95	15	96	2	109	2	109

difficulty = 5

Actual	R		S		E		S+NN		E+NN	
	T	F	T	F	T	F	T	F	T	F
T	31	96	111	16	97	30	119	8	108	19
F	28	136	14	150	15	149	6	158	4	160

ERASMUS UNIVERSITEIT ROTTERDAM

B Simulated data classification performance

	Detect	Random Forest				Gradient Boosting			
D	Model	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
2	True	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	S+NN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	E+NN	1.0	1.0	1.0	1.0	0.989	0.976	0.976	0.976
3	True	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	S+NN	0.997	0.977	1.0	0.989	0.99	0.955	0.977	0.966
	E+NN	0.997	0.977	1.0	0.989	0.997	0.977	1.0	0.989
4	True	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	S+NN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	E+NN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
5	True	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	S+NN	0.998	0.979	1.0	0.989	0.998	0.979	1.0	0.989
	E+NN	0.998	0.979	1.0	0.989	0.998	0.979	1.0	0.989

Table 20: Ongoing Transitions classification second month

	Detect	Random Forest				Gradient Boosting			
D	Model	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
2	True	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	S+NN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	E+NN	1.0	1.0	1.0	1.0	0.988	0.974	0.974	0.974
3	True	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	S+NN	0.996	0.973	1.0	0.986	0.988	0.946	0.972	0.959
	E+NN	0.996	0.973	1.0	0.986	0.996	0.973	1.0	0.986
4	True	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	S+NN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	E+NN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
5	True	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	S+NN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	E+NN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Table 21: Ongoing Transitions classification third month

D	Detect Model	Random Forest Accuracy	Precision	Recall	F1	Gradient Boosting Accuracy	Precision	Recall	F1
2	True	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	S+NN	0.992	0.966	1.0	0.982	0.992	0.966	1.0	0.982
	E+NN	0.992	0.966	1.0	0.982	0.976	0.931	0.964	0.947
3	True	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	S+NN	1.0	1.0	1.0	1.0	0.989	0.964	0.964	0.964
	E+NN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4	True	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	S+NN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	E+NN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
5	True	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	S+NN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	E+NN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Table 22: Ongoing Transitions classification fourth month

References

Aburto, L., & Weber, R. (2007). Improved supply chain management based on hybrid demand forecasts. *Applied Soft Computing*, *7*(1), 136–144.

Akter, S., & Wamba, S. F. (2016). Big data analytics in e-commerce: A systematic review and agenda for future research. *Electronic Markets*, *26*(2), 173–194.

Bass, F. M. (1969). A new product growth for model consumer durables. *Management science*, *15*(5), 215–227.

Berbain, S., Bourbonnais, R., & Vallin, P. (2011). Forecasting, production and inventory management of short life-cycle products: A review of the literature and case studies. *Supply Chain Forum: An International Journal*, *12*(4), 36–48.

Bird, G. (1981). Monte-carlo simulation in an engineering context. *Progress in Astronautics and Aeronautics*, *74*, 239–255.

Bontempi, G., Ben Taieb, S., & Borgne, Y.-A. L. (2012). Machine learning strategies for time series forecasting. *European business intelligence summer school*, 62–77.

Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: Forecasting and control*. John Wiley & Sons.

Cao, J., Li, Z., & Li, J. (2019). Financial time series forecasting model based on ceemdan and lstm. *Physica A: Statistical mechanics and its applications*, *519*, 127–139.

Chang, Y.-S., Chiao, H.-T., Abimannan, S., Huang, Y.-P., Tsai, Y.-T., & Lin, K.-M. (2020). An lstm-based aggregated model for air pollution forecasting. *Atmospheric Pollution Research*, *11*(8), 1451–1463.

Chen, M.-H., Shao, Q.-M., & Ibrahim, J. G. (2012). *Monte carlo methods in bayesian computation*. Springer Science & Business Media.

Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Daffertshofer, A., Lamoth, C. J., Meijer, O. G., & Beek, P. J. (2004). Pca in studying coordination and variability: A tutorial. *Clinical biomechanics*, *19*(4), 415–428.

Daly, A., & Ortuzar, J. D. (1990). Forecasting and data aggregation: Theory and practice. *Traffic Engineering and Control*, *31*(12), 632–643.

De Livera, A. M., Hyndman, R. J., & Snyder, R. D. (n.d.). Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American statistical association*, *106*(496), 1513–1527.

Diebold, F. X. (2015). Comparing predictive accuracy, twenty years later: A personal perspective on the use and abuse of diebold–mariano tests. *Journal of Business & Economic Statistics*, *33*(1), 1–1.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *kdd*, *96*(34), 226–231.

Estivill-Castro, V. (2002). Why so many clustering algorithms: A position paper. *ACM SIGKDD explorations newsletter*, *4*(1), 65–75.

Fletcher, R. (2013). *Practical methods of optimization*. John Wiley & Sons.

Fradkov, A. L. (2020). Early history of machine learning. *IFAC-PapersOnLine*, *53*(2), 1385–1390.

Franses, P. H., Franses, P. H. B. et al. (1998). *Time series models for business and economic forecasting*. Cambridge university press.

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

Gauss, C. F. (1809). Theoria motus corporum coelestum. *Werke*.

Georgescu, I. M., Ashhab, S., & Nori, F. (2014). Quantum simulation. *Reviews of Modern Physics*, *86*(1), 153.

Gershenfeld, N. A., & Gershenfeld, N. (1999). *The nature of mathematical modeling*. Cambridge university press.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., ... Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hart, M., Lukoszová, X., & Kubıéková, J. (2013). Logistics management based on demand forecasting. *Research in logistics & production*, *3*.

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

Hayat, M. J., & Higgins, M. (2014). Understanding poisson regression. *Journal of Nursing Education*, *53*(4), 207–215.

Ho, T. K. (1995). Random decision forests. *Proceedings of 3rd international conference on document analysis and recognition*, *1*, 278–282.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

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

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Hurvich, C. M., & Tsai, C.-L. (1989). Regression and time series model selection in small samples. *Biometrika*, *76*(2), 297–307.

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice*. OTexts.

Imoto, K., Nakai, T., Ike, T., Haruki, K., & Sato, Y. (2018). A cnn-based transfer learning method for defect classification in semiconductor manufacturing. *2018 international symposium on semiconductor manufacturing (ISSM)*, 1–3.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112). Springer.

Joshua, S. C., & Garber, N. J. (1990). Estimating truck accident rate and involvements using linear and poisson regression models. *Transportation planning and Technology*, *15*(1), 41–58.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, *30*, 3146–3154.

Kim, C., & Nelson, C. (1998). State-space models with markov-switching: Classical and gibbs-sampling approproaches with applications.

Kim, S., & Kim, H. (2016). A new metric of absolute percentage error for intermittent demand forecasts. *International Journal of Forecasting*, *32*(3), 669–679.

Kleijnen, J. P., & van Groenendaal, W. (1992). *Simulation: A statistical perspective*. John Wiley & Sons, Inc.

Knuth, D. E. et al. (1973). *The art of computer programming* (Vol. 3). Addison-Wesley Reading, MA.

Koreisha, S. G., & Pukkila, T. M. (1993). New approaches for determining the degree of differencing necessary to induce stationarity in arima models. *Journal of statistical planning and inference*, *36*(2-3), 399–412.

Kot, S., Grondys, K., & Szopa, R. (2011). Theory of inventory management based on demand forecasting. *Polish journal of management studies*, *3*, 147–155.

Kotsiantis, S. B., Zaharakis, I., Pintelas, P., et al. (2007). Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, *160*(1), 3–24.

Kühne, B., & Böhmann, T. (2019). Data-driven business models-building the bridge between data and value. *ECIS*.

Landau, S., Leese, M., Stahl, D., & Everitt, B. S. (2011). *Cluster analysis*. John Wiley & Sons.

Larose, D. T., & Larose, C. D. (2014). *Discovering knowledge in data: An introduction to data mining* (Vol. 4). John Wiley & Sons.

Le Cam, L. (1990). Maximum likelihood: An introduction. *International Statistical Review/Revue Internationale de Statistique*, 153–171.

Legendre, A. M. (1806). *Nouvelles méthodes pour la détermination des orbites des comètes; par am legendre...* chez Firmin Didot, libraire pour lew mathematiques, la marine, l ...

Li, G., & Zhu, Z. H. (2021). Estimation of flexible space tether state based on end measurement by finite element kalman filter state estimator. *Advances in Space Research*, *67*(10), 3282–3293.

Love, B. C. (2002). Comparing supervised and unsupervised category learning. *Psychonomic bulletin & review*, *9*(4), 829–835.

Lungu, I., & Pîrjan, A. (2010). Research issues concerning algorithms used for optimizing the data mining process. *Romanian Economic Business Review*, *4*, 108–125.

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2022). M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*.

Mark, d. B., Otfried, C., Marc, v. K., & Mark, O. (2008). *Computational geometry algorithms and applications*. Spinger.

Mason, L., Baxter, J., Bartlett, P., & Frean, M. (1999). Boosting algorithms as gradient descent. *Advances in neural information processing systems*, *12*.

McQuarrie, A. D., & Tsai, C.-L. (1998). *Regression and time series model selection*. World Scientific.

Milligan, G. W. (1996). Clustering validation: Results and implications for applied analyses. *Clustering and classification* (pp. 341–375). World Scientific.

Milligan, G. W., & Cooper, M. C. (1987). Methodology review: Clustering methods. *Applied psychological measurement*, *11*(4), 329–354.

Mir, A., & Nasiri, J. A. (2019). Lighttwinsvm: A simple and fast implementation of standard twin support vector machine classifier. *Journal of Open Source Software*, *4*, 1252. https://doi.org/10.21105/joss.01252

Morimoto, J., & Ponton, F. (2021). Virtual reality in biology: Could we become virtual naturalists? *Evolution: Education and Outreach*, *14*. https://doi.org/10.1186/s12052-021-00147-x

Muzaffar, S., & Afshari, A. (2019). Short-term load forecasts using lstm networks. *Energy Procedia*, *158*, 2922–2927.

Neelamegham, R., & Chintagunta, P. (1999). A bayesian model to forecast new product performance in domestic and international markets. *Marketing Science*, *18*(2), 115–136.

Nelder, J. A., & Wedderburn, R. W. (1972). Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, *135*(3), 370–384.

Noh, Y., Cheon, W., Hong, S., & Raasch, S. (2003). Improvement of the k-profile model for the planetary boundary layer based on large eddy simulation data. *Boundary-layer meteorology*, *107*(2), 401–427.

pandas development team, T. (2020). *Pandas-dev/pandas: Pandas* (Version latest). Zenodo. https://doi.org/10.5281/zenodo.3509134

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pelleg, D., & Moore, A. (1999). Accelerating exact k-means algorithms with geometric reasoning. *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 277–281. https://doi.org/10.1145/312129.312248

Quinlan, J. R. (2014). *C4. 5: Programs for machine learning*. Elsevier.

Rabiner, L. R., & Gold, B. (1975). Theory and application of digital signal processing. *Englewood Cliffs: Prentice-Hall*.

Saini, H., Upadhyaya, A., & Khandelwal, M. K. (2019). Benefits of cloud computing for business enterprises: A review. *Proceedings of International Conference on Advancements in Computing & Management (ICACM)*.

Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with python. *9th Python in Science Conference*.

Shannon, R. E. (1998). Introduction to the art and science of simulation. *1998 winter simulation conference. proceedings (cat. no. 98ch36274)*, *1*, 7–14.

Shen, H., & Huang, J. Z. (2008). Forecasting time series of inhomogeneous poisson processes with application to call center workforce management. *The Annals of Applied Statistics*, *2*(2), 601–623.

Shephard, N. G., & Harvey, A. C. (1990). On the probability of estimating a deterministic component in the local level model. *Journal of time series analysis*, *11*(4), 339–347.

Srigopal, V. (2018). Predicting customer churn risks and optimizing retention investments using reliability and maintenance engineering.

Stigler, S. M. (1986). *The history of statistics: The measurement of uncertainty before 1900*. Harvard University Press.

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

Sun, Z.-L., Choi, T.-M., Au, K.-F., & Yu, Y. (2008). Sales forecasting using extreme learning machine with applications in fashion retailing. *Decision Support Systems*, *46*(1), 411–419.

Szozda, N. (2010). Analogous forecasting of products with a short life cycle. *Decision Making in Manufacturing and Services*, *4*(1-2), 71–85.

Taylor, S. J., & Letham, B. (2018). Forecasting at scale. *The American Statistician*, *72*(1), 37–45.

Tembhurne, J. V., & Diwan, T. (2021). Sentiment analysis in textual, visual and multimodal inputs using recurrent neural networks. *Multimedia Tools and Applications*, *80*(5), 6871–6910.

Thomassey, S., & Happiette, M. (2007). A neural clustering and classification system for sales forecasting of new apparel items. *Applied Soft Computing*, *7*(4), 1177–1187.

Timmerman, M. E., Ceulemans, E., Kiers, H. A., & Vichi, M. (2010). Factorial and reduced k-means reconsidered. *Computational Statistics & Data Analysis*, *54*(7), 1858–1871.

Van den Dool, H. (1989). A new look at weather forecasting through analogues. *Monthly weather review*, *117*(10), 2230–2247.

Van Der Maaten, L., Postma, E., Van den Herik, J., et al. (2009). Dimensionality reduction: A comparative. *J Mach Learn Res*, *10*(66-71), 13.

Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual*. CreateSpace.

van Steenbergen, R., & Mes, M. (2020). Forecasting demand profiles of new products. *Decision Support Systems*, *139*, 113401. https://doi.org/https://doi.org/10.1016/j.dss.2020.113401

Viboud, C., Boëlle, P.-Y., Carrat, F., Valleron, A.-J., & Flahault, A. (2003). Prediction of the spread of influenza epidemics by the method of analogues. *American Journal of Epidemiology*, *158*(10), 996–1006.

Wackerly, D., Mendenhall, W., & Scheaffer, R. L. (2014). *Mathematical statistics with applications*. Cengage Learning.

Webby, R., & O'Connor, M. (1996). Judgemental and statistical time series forecasting: A review of the literature. *International Journal of forecasting*, *12*(1), 91–118.

Wei, W. W. (2006). Time series analysis. *The oxford handbook of quantitative methods in psychology: Vol. 2*.

Yu, H., & Yang, J. (2001). A direct lda algorithm for high-dimensional data—with application to face recognition. *Pattern recognition*, *34*(10), 2067–2070.

Zhao, Z., Chen, W., Wu, X., Chen, P. C., & Liu, J. (2017). Lstm network: A deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems*, *11*(2), 68–75.

Zhou, Z.-H. (2021). *Machine learning*. Springer Nature.

Zou, K. H., Tuncali, K., & Silverman, S. G. (2003). Correlation and simple linear regression. *Radiology*, *227*(3), 617–628.