LiU-ITN-TEK-A--21/055-SE

Machine learning for planning in warehouse management

Anton Tynong

2021-06-23

Department of Science and Technology
Linköping University
SE-601 74 Norrköping, Sweden

Institutionen för teknik och naturvetenskap
Linköpings universitet
601 74 Norrköping

LiU-ITN-TEK-A--21/055-SE

Machine learning for planning in warehouse management

The thesis work carried out in Transportsystem
at Tekniska högskolan at
Linköpings universitet

Anton Tynong

Norrköping 2021-06-23

Department of Science and Technology
Linköping University
SE-601 74 Norrköping, Sweden

Institutionen för teknik och naturvetenskap
Linköpings universitet
601 74 Norrköping

Abstract

In today's era of e-commerce with high expectations on high service levels and demanding competition, well developed warehouse management and planning is required. Recent trends in digitalization give access to a large amount of data and big efforts and improvements are made to use that data in warehouse management. This study considers the software company *Millbyte*'s warehouse management system *Alyson WMS* and how customer data can be used to improve the planning within warehouses. The approach taken is to evaluate if Machine Learning has potential to give benefits to the *Alyson WMS* users. The objective was to identify and evaluate possible applications within the *Alyson WMS* as well as comparing different Machine learning techniques with each other and naive methods.

The study was done in the Azure ML-studio platform, with two different data sets tested on two different identified applications; *Demand prediction* and *Return prediction*. The Machine learning algorithms used was *Boosted Decision Tree, Linear regression, Decision Forests* and *Neural network*. The best found algorithm for the *Demand prediction* application was the *Decision forest*-algorithm which performed a MAPE of 30,2% and 45% for the two data sets respectively, compared to 34,8% and 54,3% for the best naive method. The best found algorithm for the *Return prediction* was the *Boosted decision tree*-algorithm, which also performed better than the best naive method for the application with a MAPE of 27,5% and 50,2% for the two data sets respectively, compared to 35,1% and 54,2% for the best naive method.

The results in this study suggests that valuable ML-applications within the *Alyson WMS* can be identified and that the most suitable algorithms for the case study can give better predictions compared to naive methods. Existing customers using *Alyson WMS* could have benefits using these predictions as a supplement in their warehouse planning. The most promising algorithms are Decision tree-types of algorithms which have the benefits of being flexible and able to handle large amounts of data. The results and algorithm performance can be concluded to be highly dependent on the characteristics of the data and the included variables and features. Future research with Machine learning algorithms within the *Alyson WMS* should consider inclusion of external data from customers, such as sales and promotions, as input variables if using the same applications as in this study. Furthermore, efforts can be done to study other applications where consistent and timely data exists.

Acknowledgements

A long struggle is finally coming to an end with the completion of my master thesis and this report. With new insights about Machine Learning I am eager to expanding on the research done in this study together with *Millbyte Solutions*, and hopefully in the near future deploying new features within *Alyson WMS* based on this report. So first and foremost, I am thankful to *Millbyte Solutions* for the opportunity, and I am hopeful and optimistic that it will be an investment and experience well spent, both for the company and me. I am also very thankful to my supervisor David for the discussions, guidance and help to keep me on track towards the right goals. And finally, I would not be here without my family and friends, whom with their encouragement and support, have been key to the completion of my studies that is now coming to an end with this master thesis. I speak for everyone when I am saying that it is about time to close this chapter in life and start writing a new one.

Table of Contents

1. Introduction

This chapter explains the background followed by the objectives and research questions for the study.

1.1. Background

Keeping inventory, often also referred to as *Warehousing*, has been a concept existing since the Age of Agriculture. Although the way it has been used through history has been similar, there are two major shifts since the age of agriculture, both for the whole civilization and the concept of warehousing. The first shift was when the *Industrial Era* came along two or three centuries ago, and the latest shift was just about five decades ago, to the *Age of Information*. During the latter, much of the basic concepts of warehousing are the same as before, but tools, operations and tasks have been drastically changed. The recent technological development, mass-globalization and increased network complexity have put a demand of efficient supply chains where warehousing is a key component (Ackerman, 1997). Especially with the explosion of e-commerce and online purchases which have resulted in a requirement for faster movements and increasing level of customer service. Therefore, well managed inventories are required to be able to compete on the market.

Almost all organizations and companies keep inventory of physical, tangible items and their counterpart, intangible items, such as records or data. Efficient inventory management is of great importance, especially for supplying companies, in order to have predictability, handle variations in demand and have a hedge against unreliability of supply (Muller, 2011). Bartholdi and Hackman (2019) also include consolidation of products as a motivation and a relatively new key function of warehousing, mainly due to the benefits of reduction of transportation costs and increase in level of customer service.

The movement of Just-in-Time about 20 years ago, and the Lean-movement during the last decade, aimed at reducing or removing warehousing. Warehouses still have an important role and are adding value to supply chains, mainly to customer service through availability and short response time. Also included in some warehouses are value-adding services like handling of returns, customization and consolidation. Warehousing can also have cost-reducing functionalities when for example production need long runs because of expensive and time-consuming setups, or when companies face large variance of demand (Frazelle, 2016).

Warehouse management and Inventory management are often coupled together or sometimes even thought of as the same concept, but there are however some differences. Inventory management is more focused on the actual inventory and warehouse management is commonly extended to include all operations and processes connected to the warehouse. On the other hand, inventory management will instead in a higher degree relate and be considered in the big-picture, typically in business management. This study concerns warehouse management and its' broader description, focusing on the operations within the warehouse. For simplicity, further on in this report the term warehouse management will be used even though it could relate to both or either of the concepts in other literature.

Warehouse management is the concept of managing and planning within the warehouse. Except for planning the amounts of inventory, planning in a warehouse consists of three main elements: people, space and equipment (Ackerman, 1997). Each element will in turn be affected by the operations performed in the warehouse. These operations are most often categorized as inbound operations, e.g. receiving and put-away, and outbound operations, e.g. order-picking, checking, packing and shipping (Bartholdi & Hackman 2019).

Plans and decisions for the three elements and for the operations need to consider for example demand forecasts, order quantities, storage location systems, storage placement, order-picking system, lot sizing, labour scheduling, equipment capacity, safety stock rules, replenishment policies, returns handling systems and returns predictions (Ackerman 1997). These are sub-processes to support decisions and planning of the elements and operations. The sub-process will further on be called applications, since it is processes where different techniques can be applied for calculations or decisions. In other words, applications are processes that can be optimized within a warehouse, and techniques are the methods to use for the optimization.

Although much have changed in the last decades, a lot of theory and concepts about techniques for the applications have been practiced for even longer and is still present within warehouses today. According to Bartholdi and Hackman (2019) too much of warehousing practices are based on rules-of-thumb and simplistic ratios, rather than mathematical and computer models. Much of what was earlier viewed to be tasks performed by humans are now possible to be done by computers, with greater accuracy, higher reliability and continuously without breaks (SSI Schaefer 2018). The conventional techniques are now becoming inefficient to use and many areas see a rise in usage of computer models. One type of these computer models is Machine Learning (ML) techniques. ML is a way of letting computer algorithms find structure, patterns and regularities in data, without explicit intervention, to create models. These models can then be used to react to new data and give predictions (Wenzel et al. 2019). The development of new ML-techniques, greater computing power and exponential growth of available data have resulted in a big interest of applications for ML within Supply Chain Management (SCM), especially the last decade (SSI Schaefer 2018). The amount of research done in the area is increasing for every year (Yani et al. 2019) and according to Gartner (2018) a lot of ML-techniques are already in use within in big companies. However, warehouse management is a sub-area of SCM that has been overlooked (Malik & Jeswani 2018).

Millbyte solutions AB (*Millbyte*) is a logistics software development company that has developed the logistics software platform *Alyson*. *Alyson WMS* is a cloud-based warehouse management system (WMS) included in the Alyson platform which registers data from different sources and actors like warehouse personnel, distributors, 3PL-companies and end-customers, that directly or indirectly interacts with the software. Previous research has shown potential for valuable applications of ML-techniques in warehouse management including for example demand forecasting, predicting backorders, inventory optimization, determining replenishment rules, determining optimal reordering points, control spare parts inventory obsolescence and warehouse automation. *Millbyte* are interested in extending the *Alyson WMS* software with ML functionality to increase the warehouse efficiency for customers using *Alyson WMS*.

Depending on the layout, requirements and conditions, as well as data available there are different possibilities for ML-techniques to make improvements to the applications in warehouse management. All applications are in some way connected to the main goals and decisions which is regarding warehouse planning: what work should be performed, amount of people that is required, where items should be placed and what equipment to use.

1.1. Objective

The aim with this study is to identify and evaluate applications for Machine learning within Warehouse management where there is potential to improve the decision making and planning for existing customers using the warehouse management system *Alyson WMS*.

1.2. Research questions

To complete the aim of this study, the following research questions was formulated:

1) *What applications of Machine learning in warehouse management can be identified to be potentially valuable for the case study?*
2) *What different Machine learning techniques show potential for each of the identified applications?*
3) *What value and benefits can Machine Learning techniques offer compared to each other and to simple benchmark methods, for the identified applications in the case study?*

1.3. Methodology

This chapter describes the research method and the process steps carried out to answer the research questions.

To answer the research questions, this study includes a theoretical framework as a basis for applying machine learning to the case study. The theoretical framework is a review of machine learning in warehouse management and warehouse management theory. The literature was used as guidelines as well as comparisons for the case study. The case study is a quantitative analysis of the identified applications using data from *Alyson WMS*. The warehouse management system *Alyson* acted as a frame of the case, with its' limitations of functions, available users, and user data. A portion of the data and users of the warehouse management system was selected for the quantitative analysis.

A framework of a process model (Figure 1) was used for the case study, with inspiration drawn from the literature (Han & Kamber 2012, Hand et al. 2001, Shearer 2000; Wirth & Hipp 2000; Wenzel et al. 2019).

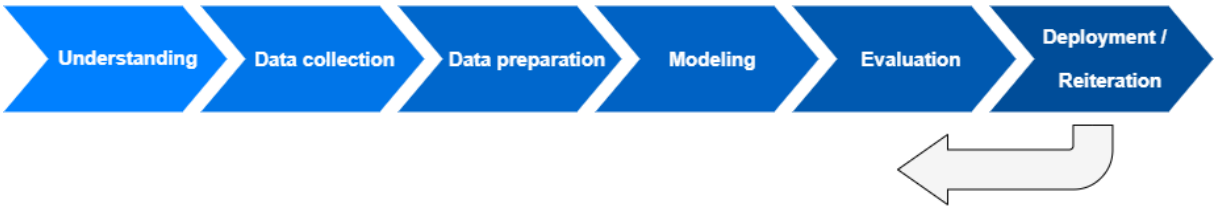

Figure 1: The process model for the case study describing the process steps.

The different steps in the process model are described briefly below:

Step 1: Understanding – Determining business objectives, reviewing the literature, understanding the case, the warehouse management system *Alyson WMS* and describing and exploring the data.

Step 2: Data collection – Further exploration of data, verification of data quality and selection of data attributes and data sets.

Step 3: Data preparation – Pre-processing data through cleaning, formatting, and integration of data.

Step 4: Modelling – Selection of techniques algorithms, building of the models and setting up model environment.

Step 5: Evaluation – Evaluation of the results and determination of the next step.

Step 6: Deployment or reiteration – Many reiterations were done, going back to different steps in the process, depending on insights made from the evaluation. No deployment was done to the *Alyson WMS*.

The literature used in the theoretical framework was collected through searches of relevant keywords in different research databases. The focus was on finding the latest promising ideas regarding Machine Learning because the development in the area has been significantly increasing during the last years. Some useful research papers and books were found that are dated to over 10 or 20 years ago, however their contributions still apply, especially those regarding Warehouse Management in general. Most of the literature found on Machine Learning in Warehouse Management were used to identify Machine Learning applications in Warehouse Management or used as comparisons to this study. The literature was also used to form a better understanding of problem formulations in operation and planning processes in Warehouse Management.

After identifying applications of Machine Learning in Warehouse Management through the literature, initial data was collected to get an understanding of what applications were suitable with the available data in the case study. The amount and structure of data was limited by the *Alyson* WMS and the utilization by the current *Alyson WMS* users. Therefore, all data sets from the different *Alyson WMS* users were explored to find suitable cases for the applications. Examples of suitable structure was data sets with consistent data attributes linked to an order or an item. The data quantity and quality had to be verified and was done according to recommendations in the literature and is further described in chapter *3.1 Data description*. The exploration and understanding of data also included statistical calculations and visualizations to investigate quality factors like consistency, timeliness, reliability, and interpretability. From the applications found in the literature, two applications were considered to fit the case study with respect to the available data: *Demand prediction* and *Return prediction*. Other applications that were considered but rejected because of lack of data or requirements in further processing of data were *Resource prediction*, i.e. forecasting the total work load, *ABC-classification*, i.e. determining optimal item placement and *Replenishment optimization*.

When a sufficient understanding of the data was achieved, it was decided what data sets that was going to be used in the case study. Two data sets were chosen for each application for validation and evaluation purposes. Different data sets result in different models, such that the outcome of the process could be evaluated and compared for different data sets. The selected data sets needed pre-processing to fit within the machine learning algorithms. The data preparation steps included cleaning, integration, reduction, and transformation of data. Data cleaning involved handling of missing values, ignoring dubious entries and removal of noisy data. A more detailed description of the data preparation can be seen in chapter *3.1 Data description*.

The modelling phase consisted of setting up the model environment and selecting and building the models. The selection of categories of ML-techniques depended on the available data and the objectives defined in the understanding-phase. Within each ML-category there are different algorithms and techniques to use, and several was tested and compared. There are different tools existing to implement these algorithms and one or more of these tools can be used depending on what is suitable. Microsoft provides a Machine Learning-toolbox, called *Machine learning studio (ML-studio)*, on their platform Azure which was the preferred choice. This was because of the simplicity for integration with the databases and software since *Alyson WMS* was already using the Azure

platform. The modelling phase also included calibration of the parameters within the ML-techniques used. For that reason, a parameter sweep was performed for each of the techniques and evaluated.

In the evaluation step, analysis of the resulting models and the processes was carried out, with iterations back to previous steps when insights were made. The ML-techniques were compared to each other and the usefulness in warehouse management was evaluated. The latter was done by comparing the results of ML-techniques to other existing methods in literature and naive methods for benchmarking.

1.4. Limitations

Since it was decided to use the toolbox *Machine Learning* studio for building the models, this study was limited to the options available within the toolbox. Though some variations could be made, the ML-techniques used were built-in functions based on certain algorithms which limited this study to those algorithms.

Some of the potentially useful applications found in the literature was not applicable to the case study because of the limitations of existing data. In some cases, similar data did not exist in the *Alyson* WMS, and in other cases, the quantity or quality of data was considered to be insufficient for the application.

The most valuable performance measures, from a business point of view, would be to implement the resulting models in the real-world case and compare them to previous models or current models in use. It could be argued that the most meaningful measurements in the big picture should relate to costs or cost savings, but this study was limited to warehouse management as it has been defined in chapter *1.1 Background* and therefore only considered predictions to use in warehouse planning.

2. Theoretical framework

This chapter will describe previous literature on the subject which this study has used as a basis. Literature relating to warehouse management in general is first described in *2.1 Warehouse management*, followed by a general literature review on Machine Learning in *2.2 Machine Learning* and the last section *2.3 Machine learning in SCM and Warehouse management* describes previous work done on subject most similar to this case.

2.1. Warehouse management

In this chapter, the general concepts and problem formulation of warehouse management will be described, followed by warehouse planning in general and conventional or traditional planning tools and techniques.

2.1.1. Warehouse management and general problem formulation

The fundamental problem in warehouse management is to balance the inventory level with the level of service to customers. If inventories were cost-free, the task would be trivial, but since that is not the case, a trade-off between inventory levels and level of service must be made. This trade-off can be better optimized with better estimates of the customer demand and more efficient processes (Jacobs et al. 2011).

The trade-off can be summarized by the goal of minimizing costs and satisfying the demand, while meeting requirements from customers to stay competitive and sustainable long-term as a business (Pettersson 2015). Karasek (2013) defines that the optimal operation of a warehouse is achieved when "Each customer is satisfied completely according to his order, in due time and when all warehouse and logistic processes are done in the shortest possible time, with minimal cost and optimal utilization of resources under dynamically changing conditions".

There are different types of warehouses and inventories. Traditionally they can be divided into three types: Private, public and contract/dedicated warehouses (Ackerman 1997), but more suitable is to describe them more specific. Bartholdi and Hackman (2019) used the following categorization:

- Retail distribution center (DC), supplying retail stores.
- Service parts DC, holding spare parts.
- Catalog fulfilment or e-commerce DC, typically receiving small orders from individuals.
- Third-party logistics (3PL) warehouse, where warehousing is outsourced and may serve multiple customers.
- Perishables warehouse, for special product requirements and very short shelf life.

The type of warehouse in question will influence the inventory characteristics, throughput and service requirements, and the different costs. Those factors combined will determine what type of warehouse handling system, including selection and organization of equipment and material flow, that is suitable and efficient (Bartholdi & Hackman 2019).

Warehouses are only one part of a larger supply chain which, depending on the structure of the chain, can be operated in many different ways. For warehouses connected to actors within the same company, the integration and sharing of information between the actors will most likely be easier, than for a 3rd party logistics (3PL)-provider who is not sharing organizational structures with its' suppliers and customers. Depending on the organizational structure, business objectives can range from optimization of the whole chain to only consider the warehouse itself. Advanced planning systems (APS) can help integrate, coordinate and share information between actors in the supply chain, regardless of the structure.

To plan, control and coordinate operations within the warehouse, a warehouse management system (WMS) is used. It is often a complex software package that helps warehouses manage information, inventory and planning to ensure efficiency in the warehouse. Ultimately, the WMS help to control and improve operations such that customers get better service while having less inventory (Bartholdi & Hackman 2019). The customer in the view of a warehouse can be actors in the supply chain like the end-consumer of finished goods, a retailer with a vendor managed inventory (VMI) or a distribution center (DC) of another company (Figure 2).

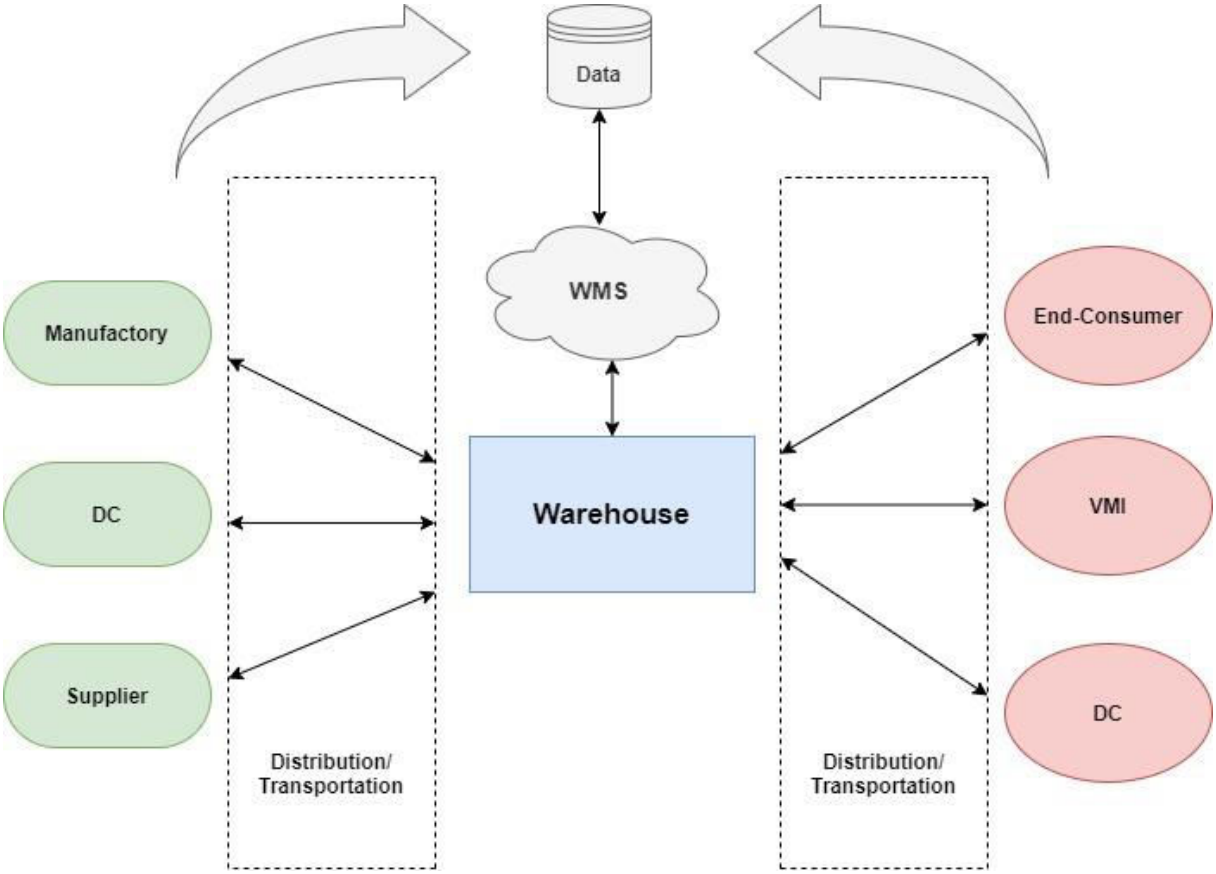

Figure 2: Supply chain in the view of a warehouse and its' directly linked actors, where different suppliers are shown to the left in green, and different customers to the right in red.

In figure 2, the suppliers, customers and distributors can all be stand-alone companies or within the same organization, or in different combinations. Optimally, the warehouse can collect useful data from each actor and within the warehouse itself, to plan and control its' operations in a WMS.

From the trade-off made between level of inventory and level of customer service, a component of cost-optimization can be derived. Generally, having inventory comes with two main categories of costs, ordering costs and holding costs. Ordering costs consists of things tied to placing orders and purchasing items such as utilities, software and salaries for the management. Holding costs include the cost of capital tied up to the inventory including the goods in inventory, rent, warehouse labour and equipment (Muller 2011). These costs are all linked to the elements of warehousing; space, equipment and people, components which needs to be planned and considered on a daily and detailed basis in warehouse management. The most critical part and with the highest potential impact on efficiency improvement is the people component (Ackerman 1997; Bartholdi & Hackman 2019).

The second part of the trade-off stated in the problem formulation is customer service. In the conventional view of Supply Chain Management, customer service together with competitiveness, is the ultimate goal. Customer service can be described by three types of elements: pre-transactional, transactional and post-transactional elements. Pre-transactional elements are activities preceding a customer relationship and includes for example information about products or flexibility to customization. Transactional elements are those concerning the order fulfilment, i.e. availability of products, current status and location, and introduction to the use of a product. Post-transactional elements are related to services provided after the order is fulfilled. That can be services like handling returns, repairing, maintenance, warranties or handling of customer complaints. Warehouse management is mostly involved with the transactional elements and some parts of the post-transactional elements (Stadtler & Kilger 2008). During the last 20 years, WMSs have drastically helped increase the pace of supply chains and thus helped increased the level of customer service (Bartholdi & Hackman 2019).

2.1.2. Warehousing operations

Warehouse operations can be summarized to reorganization and repacking of products. Basically, that means receiving large quantities of products in chunks and redistributing them in smaller quantities. The reorganization of product is done by the in bound processes, receiving and put-away, and the outbound processes, order-picking, checking, packing and shipping (Figure 3) (Bartholdi & Hackman 2019).

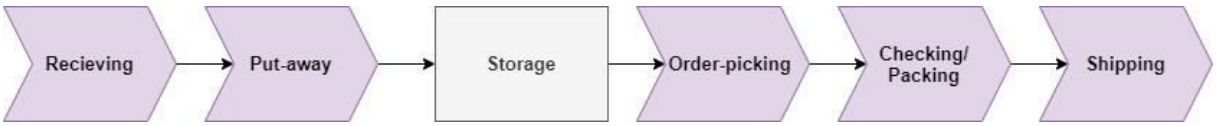

Figure 3: An overview of warehouse operations in order of arrival to departure from the warehouse.

Receiving refers to handling goods at arrival to the warehouse. That typically includes an in-advance notification of incoming delivery, unloading and staging for put-away. It requires scanning and registration, along with inspection of exceptions such as damage, incorrect amounts or wrong descriptions (Bartholdi & Hackman 2019). Receiving stands for about 10% of the operating costs in a typical DC (Frazelle 1996).

Put-away is the act of placing retrieved delivery to storage. It requires exact knowledge of the states of storage locations and planning the put-away destination for efficient handling in following operations (Bartholdi & Hackman 2019). Put-away accounts for 15% of warehouse operating costs (Frazelle 1996).

Order-picking starts upon receiving a customer order, where the warehouse needs to first verify availability followed by planning the order-picking by making picklists. Finally, the order will be picked according to the picklists and schedules. In this step the warehouse must also produce necessary documentation and schedule for shipping, which is all typically done in a WMS. The time spent in each order-picking activity can be broken down to: Traveling (55%), Searching (15%), Extracting (10%) and Paperwork or other activities (20%). Order-picking as a whole process is the largest contributing task to the total warehouse operating costs and accounts for about 55% of the expenses. Since it is the most expensive process, and traveling is the most time-consuming task in the process, much work on design and optimization is typically done to reduce the unproductive time of traveling (Bartholdi & Hackman 2019; Frazelle 1996).

Checking and packing is often done together and can be labour-intensive due to the handling of each customer order. Checking ensures order accuracy and improves the service level and satisfaction of customers. Packed products are finally scanned and registered and set available to shipping, which also enables the ability to track packages. **Shipping** is the last warehouse process, where products are loaded for distribution. Since packages are consolidated in the packing process, shipping handles larger units than in picking (Bartholdi & Hackman 2019).

Additional to above described processes, a recent trend in warehousing is to have some value-adding processes (VAP). Bartholdi and Hackman (2019) describe following VAP-examples:

- Ticketing or labelling
- Monogramming or alterations
- Repackaging
- Kitting (repackaging to form new items)
- Postponement of final assembly
- Invoicing

Another trend increasing the functions and operations of warehouses is the handling of returns, mostly due to the big increase of e-commerce lately (Bartholdi & Hackman 2019). Flexible and responsive networks are important to handle returns which increases the level of customer service (Muller 2011). Returns can also be beneficial in the form of cost reduction and environmental factors. The reverse flow of returns can be directed to different parts of a supply chain and warehouses may be the caretaker of returns depending on the type of return. Products, components and materials may be recycled and used again, if the product is defect and not only returned as a regret of purchase. For warehouses, returns will influence the optimization in planning procedures and add another variable to account for. Predicting returns, especially when supporting e-commerce, is therefore useful for efficient warehouse management (Langevin & Riopel 2005).

2.1.3. Warehouse planning

To plan and optimize inventory procedures efficiently, information about the goods in the inventory need to be accurate. From the physical location and movement of goods to timely records of all item storage and movement. Planning and optimization of inventory should aim to maximize (Muller 2011):

- Use of space, equipment and labour.
- Accessibility and ability to locate items
- Protection from damage
- Flexibility
- Reduction of administrative costs

An important aspect is to define proper planning structures and to manage and control for capacity, therefore, a well-developed demand management is required (Jacobs et al. 2011; Stadtler & Kilger 2008). Information from a demand management system works as input to for example, sales and operations planning (SOP) at the strategical level, master production schedule (MPS) or material requirements planning (MRP) at the tactical level, and to the operational level dealing with decision making on a day-to-day basis (Jacobs et al. 2011, Pettersson 2015). All mentioned levels of planning are present in warehouse management since demand management is tightly linked both ways with customers and suppliers with information flowing from, to and about each of them (Jacobs et al. 2011).

The sub-sections below will describe how planning in a warehouse need forecasts to plan the demand and how to plan warehouse operations for that demand, followed by planning tools and techniques.

2.1.3.1. Forecasting

The main goal with demand management is to create a basis for plans according to estimated states in the future, mainly through forecasts of quantities and timing of customer demand. It is crucial to differentiate between forecasts and plans, where forecasts are predictions of what might happen while an actual plan rather uses forecasts as an input to the planning (Jacobs et al. 2011). Stadtler and Kilger (2008) define a demand planning process in six steps:

1. Preparation of demand planning structures and historic data
2. Computation of statistical forecast
3. Judgemental forecasting
4. Consensus forecasting
5. Planning of dependant demand
6. Release of the forecast

The first step is mainly a collection task where historic data is gathered with former planning, shipments, customer orders and corrections of previous forecasts. The second step is based on the collected data in the first step and the statistical forecasts is commonly done by a computer. The third step involves expert-knowledge by humans that either tune the statistical forecasts by instinct in an unstructured way or combine their knowledge with the statistical forecast and tune the forecasts in a structured way. Structured judgement involves methods for how to tune the forecasts for example assigning weights to statistical forecasts versus the demand planner's forecast. The fourth step is to reach consensus among planners and departments if differences in forecasts are present. The consensus forecast should represent the forecast of finished products that is available to sell, also known as independent demand, and therefore serve as a basis for the fifth step, planning of dependant demand. The fifth step can often require computation of the demand on component level, to be sure of availability from supplier or feasibility with overall demand of key components. Other types of dependent demand specific to a warehouse can be for example equipment and packages. The last step is a release of the forecast to further planning processes (Stadtler & Kilger 2008).

2.1.3.2. Planning warehouse elements

While forecasts are important for many departments in a supply chain, they serve as a foundational input to the warehouse planning. The final plans to establish are concerning the three elements space, people and equipment. Within each of the three elements there are many different considerations to account for and each of the elements are somewhat dependent on each other (Ackerman 1997).

Regarding space planning the main goal is to determine where items and stocks should be placed, for efficient usage of space and capacity. The easiest method is to have fixed slots for all the goods but is evidently a waste of space and in the end very inefficient compared to fluid slots in an adaptable system. Ways to optimize the usage of space is for example to implement location by family groupings, a locator address system, cube utilization, lot size optimization, reduction of aisle losses, reduction of number of aisles and most important: managing the space and storage layout. The space planning can be constrained by requirements of temperature, humidity control or limits to stacking. To make the planning work, a handling system fitting the storage layout, and a layout adapted to the handling system is crucial (Ackerman 1997).

The problem of planning for people or labour can be formulated as quantifying the work to be performed and how many and what people are required. This should include administrative work such as monitoring and supervising. The planning of people benefits greatly from well-defined job descriptions and efficient handling systems. Problems that can occur are empty stocks due to bad communication systems or waiting times if processes are dependent on previous tasks, all which can be minimized or mitigated with good planning and handling systems. Affecting constraints are for example number of facilities compared to the amount of people, space available, equipment available or work environment policies (Ackerman 1997).

Planning of equipment is mainly influenced by equipment available. The planning should consider what equipment available is most suitable to carry out the task with some constraints regarding work environment or regulations (Ackerman 1997).

For both people and equipment planning the essential information needed for planning is stated by Ackerman (1997) as:

- Average order-size as an indication of worktime needed
- Physical characteristics of the goods
- The ratio of receiving units to shipping units and if goods are in bulk or individual units
- Seasonal variations
- Shipping and receiving requirements
- Picking rules

2.1.3.3. Planning tools and techniques

A few standardized planning techniques are mentioned frequently in the literature, especially regarding applications as replenishing and demand forecasting. Depending on which type of inventory you are holding, and which type of demand is present, different replenishment systems are used. When dealing with independent demand, Muller (2011) claims that it is most suitable to determine a re-order point (ROP). ROP can be defined as the minimum available amount in stock and on order before you reorder. ROP is often also referred to as economic order quantity (EOQ) which is the most common approach to calculating ROP. Other basic alternatives are a Min-Max inventory system (Muller 2011) and base stock and distribution requirements planning (DRP) (Jacobs et al. 2011). EOQ is based on calculating optimal quantities to order and inventory level combined with customer service criteria and risk of stocking out. It has the benefits to have a relatively low system complexity where each warehouse is making the decision (Jacobs et al. 2011). The downsides are low flexibility to the assumptions that both the demand and lead time is constant and known, the formula only handles one item at a time and orders arrive in a single batch (Muller 2011).

Regarding forecast techniques, traditional methods described are for example Moving average, exponential smoothing and regression analysis. Moving average (MA) and exponential smoothing are both simple and can easily be implemented and automated. MA typically consider the most recent observations for a mean-value calculation as a prediction. Exponential smoothing instead uses all the historical data but regulates the importance of recency with a weight. Regression analysis is used to predict one variable given one or more correlated variables, explaining a functional relationship between the independent variables and the dependent variable. Linear regression is where the relationship can be explained by a linear function, i.e. a straight line, which is mostly useful for long-term aggregations or aggregations over product families (Jacobs et al. 2011).

The accuracy of forecasts must be measured to be able to evaluate the forecast techniques. All accuracy measures are based on the difference between the forecast and the actual outcome, i.e.

the error. Common measurements are squared error, absolute deviation, absolute percentage error and absolute percentage accuracy. In the end they all should be measured over time to produce a mean value of the accuracy during a certain time horizon (Stadtler & Kilger 2008).

2.2. Machine learning

The following chapter explain the concept of machine learning, examples of categories and techniques of machine learning and the current usage of machine learning within Supply Chain Management and Warehouse management.

2.2.1. Machine Learning in general

Since the emergence of ML in the 1950's, different definitions of ML have been stated by different pioneers, such as Alan Turing, Arthur Samuel and Ray Solomonoff. A commonly used definition is in some shape or form "The ability for computers to learn tasks without explicit instructions". ML is an area within the broader concept of Artificial Intelligence (AI) and is closely related to other fields like statistics, data mining or optimization. ML-models learn and improve with experience from data, called model training, to predict outcomes based upon the historical data (Bell 2014).

Generally, many ML-techniques are based on or are very similar to traditional statistical methods, but some key differences are that ML-techniques have been proven to achieve a higher degree of accuracy, better automation possibility and that they require less human effort (SSI Schaefer 2018; Shahrabi et al. 2009).

In the literature different categorizations are made for the types of existing ML-techniques and the most used categorization is to divide them into Supervised learning and Unsupervised learning (Bell 2014), while some also include Reinforcement learning and Semi-supervised learning (Dey 2016; Wenzel et al. 2019) as separate categories. Supervised learning refers to algorithms which needs external assistance, using labelled data and relate input data to target output data. The input and output data could also be called independent variables and dependent variables respectively. Unsupervised learning is the opposite, where the algorithms investigate the data to discover knowledge without pre-defined, correct answers and no labelling of target values (Bell 2014; Dey 2016; Wenzel et al. 2019). Each category can be further divided to sub-categories depending on which type of tasks it is used for. Bousqaoui et al. (2017) choose to describe the four main tasks as:

- Regression: Supervised learning for predicting continuous data.
- Classification: Supervised learning for predicting discrete data.
- Clustering: Unsupervised learning for dividing data into groups or clusters.
- Association: Unsupervised learning for discovering relations and generate rules between variables.

For each category of ML there are different types of algorithms and some types can be modified to be suitable for applications in more than one category. A short description of some the most common types of algorithms is given below, although many variations exist in literature due to the many applications and use-cases (Bell 2014).

Decision trees:
Decision trees are mainly used in regression and classification models either as single decision trees, or combined together and then called forests. Decision trees aim to predict a value of a target

variable based on a set of input variables. They work as a network linked together where decisions are made from a root node until you reach a leaf, an end-node with a final answer. Decision trees are simple models but still effective, where the simplicity means that they are easy to understand and validate, and effective in the way that they can perform well with relatively small computing power and large amount of data. On the other hand, decision trees can sometimes be very flexible, which can result in large variance and over-fitting.

Bayesian networks :
Bayesian networks works with probabilities to predict outcomes where variables are connected in a network such that values of variables can influence output probabilities of other variables. It will result in a tree of conditional probabilities and it is a useful way to deal with complexity and uncertainty.

Neural networks :
Neural networks (NN), or sometimes artificial neural networks (ANN), are based on the same concept as brain cells, called neurons, that are connected to create a network. A NN is made up of simple but highly connected nodes (neurons) that process input by their dynamic state response. The network can be connected through multiple layers of nodes, also called hidden layers, between the input and output layer. There are different types of NNs relating to both supervised and unsupervised learning.

Association rules :
Association rules are one of the most used ML-methods which tries to identify the strongest relations in a dataset. They are common for example when predicting behaviour. This type of learning is very domain specific and how it will work vary from case to case. One example of a use case can be web page mining for user behaviour and finding strong correlations between elements in the data.

Support vector machines (SVM) :
Support vector machines (SVM) are a useful technique within classification, especially when classifying objects. SVMs work for both one-class classifications and multi-class classifications. A simple example is where the SVM-algorithm can divide data points in a 2D-plot with a linear line, separating one set of data points from another set of data points and in that way classify the data.

K-means algorithms :
The k-means algorithm is a clustering-type of algorithm in unsupervised learning. The k stands for the number of clusters the algorithm will define and within each cluster the centroid (or mean) is used to calculate distances to each object in the data. A calibration task of deciding the number of clusters prior to training the model must be done.

Other types of algorithms are mentioned in literature by Dey (2016): Principal component analysis (PCA), Generative models, Transductive SVM, Boosting, Bagging, k-nearest neighbour; by Bousqaoui et al. 2017: Hyperbox classifier and Gamma classifier; by Wenzel et al. (2019): Fuzzy C-means, Mean-shift, Gaussian mixture, Discriminant analysis.

2.2.2. Bias-Variance trade-off

When considering statistical models like ML-models, an important feature that describes the flexibility of the models is the *Bias-Variance trade-off*. Generally, simpler models have a high *bias* and low *variance* while very flexible and complex models have a low *bias* but a high *variance*. A high bias refers to being resistant to noisy data, with the risk of missing out on important relations between variables in the data. On the contrary, a low bias often means a high variance, which refers to a large

flexibility to fit the training data at the risk of being too adaptable to noise. A low bias model on the other hand, should in theory work better and be usable on entirely new data or new data sets. The goal should be to minimize both bias and variance, but most often as either one decreases, the other one increase. The bias can be decreased by using more local information while the variance can be decreased by averaging over more data observations, data sets or iterations.

2.3. Machine learning in SCM and Warehouse management

Studies reviewing the literature about ML in supply chain management show that the interest and number of published papers on the topic has increased a lot the last few years (Baryannis 2018; Bousqaoui et al. 2017; Wenzel et al. 2019; Yani et al. 2019). The interest and usage of ML in general, even in other fields, has also increased. The reasons behind the recent progress and increased interest can be explained by new inventions of algorithms, an increase in computational power and a higher amount of available data (SSI Shaefer 2018; Wenzel et al. 2019; Yani et al. 2019).

The literature has proven ML to be helpful in many areas of Supply Chain Management (Bousqaoui et al. 2017; Yani et al. 2019). Makkar et al. (2020) describe that ML-methods are currently in use in, for example, Demand and sales forecasting, personalized product recommendations, prize and promotion recommendations, inventory optimization, warehouse throughput optimization, consumer insights, shop-floor yield optimization, predictive maintenance, procurement, best routing options and warehouse automation. Wenzel et al. (2019) also mentions use-cases for selecting supply chain partners, predicting material backorders, detecting false-positive RFID tag reads and determining optimal replenishment rules.

One particular focus in research has been on demand forecasting (Bousqaoui et al. 2017; Wenzel et al. 2019; Yani et al. 2019) but just a few studies reveal applications and focus on warehouse management, though there is a big overlap and correlation between areas. According to surveys done by Gartner (2018), about 50 % of the total use cases are coupled with demand planning but it is unclear what amount can be related and used in warehouse management. Studies exemplifying data driven demand forecasting but in return flows has been done by for example, Toktay et al. (2013) and Cui et al. (2019), of which the latter was stating to have used ML methods. Cui et al. (2019) used variables for sales volumes, time period, product types, production processes, retailer type, multi-product effects and lagged returns to predict the number of returns out of sales per product type.

One specific example of research done on applications of ML in warehouse management mentioned above in this chapter was De Santis et al. (2017) who used ML to predict material backorders. They investigated two classification algorithms, two sampling techniques and three ensemble learning techniques as prediction models. The study found that the Blagging technique was the most promising for adoption in practice and that it has high potential for increasing the level of service. The data used consisted of inventory level, transit times, in-transit quantity, forecasted sales for future months, sales quantity for prior months, earlier supplier performance, amount of orders overdue, minimum amount in stock and six general risk flags. The same objective was tried by Bouganim and Olsson (2019) but with usage of deep learning which is a type of NN technique. Their model was able to predict 73% of backorders (precision) one week before they occurred and 72% of the predicted orders were actual backorders (recall), which can be compared to De Santis et al. (2017) who found a result of 60% precision and 20% recall in the same respective metrics. Therefore, deep learning seems like a better approach to the problem even though the data could account for some differences between the studies. Bouganim and Olsson (2019) had data from 18 variables, 4 related to demand, 5 to inventory, 5 to supply and 4 to item properties. On the other hand, they only had data from 1 year and suggested that the approach could be improved, since deep learning works best on large datasets.

Another empirical study by Inprasit and Tanachutiwat (2018) used NNs to determine optimal reordering points in inventories. They collected data from 38 supply chains in various industries and included 5 variables (lead time, standard deviation of lead time, demand, standard deviation of demand and service level) in their models. The study used and tested three NN-algorithms and best results were achieved by a Bayesian based backpropagation NN.

Further, two studies were found that used ML but not real-world data, and artificial data instead, in problems relating to warehouse management. Priore et al. (2019) used an inductive learning-type of ML-technique to determine the optimal replenishment policy. The data used was acquired through simulation and included seven independent variables to classify one target value with four possible outcomes of policies. Cherukuri and Ghosh (2016) used decision tree algorithms to control spare parts inventory obsolescence, with a better performance in comparison to a simplistic statistical model and to the standard industrial practice model.

A literature review by Xing et al. (2010) did show that most research on AI-applications in reverse logistics was relating to network design, acquisition and transportation for end-of-use or end-of-life products and selection of supplier. However, in today's era of e-commerce, warehouse management must be concerned about handling of returns from purchase regrets and the re-storage of items.

Another literature review of research looked at AI-applications in Supply chain risk management (SCRM), where 14 studies were found using ML and Big Data analysis (Baryannis et al. 2018). Even though the majority of the studies did not relate to warehouse management, a few of them or parts of them could be transferable to the area. One example is Ye et al. (2015) who used SVM-models to determine if companies are prone to demand, supply, product or external disruptions, using economic performance data.

3. Data

The software *Alyson WMS* registers data from operations within the warehouse such as times when operations are performed, quantities of items handled in each operation and storage locations. Data used in this study are historical data existing in the *Alyson WMS* database. A description of a typical warehouse using the software and what the resulting data can look like are described in the sections below.

In general, events in the warehouse starts with orders and this is when the initialization of data recording occurs. The orders can be in the form of either sales-, return- or purchase orders. All types of orders can consist of one or multiple items and data about each order include unit prices, quantities, product numbers and names, order status and timestamps for the creation of the order.

Sales orders most often concern independent demand from either an end-consumer or a vendor. Specific data for sales orders are order type, quantities ordered, shipped or returned, and information about the customer. Return orders are created when customers return goods, and the return order relates to a specific sales order. Specific data for return orders are quantity shipped and returned, along with reason for return. Purchase orders are used for ordering goods to the warehouse from suppliers and have specific data in the form of a supplier, quantity ordered and quantity received.

In addition to orders, decisions and plans made by the staff can also induce events and movements within the warehouse. These events or operations are typical warehouse activities described further in chapter *2.1.2 Warehouse operations*. Data attributes and labels for each of these operations can be seen in Table 1.

Table 1: Warehouse (WH) operations and data about each respective operation.

Operation/event	Data attributes/labels
Receiving	Order number, status, order date, expected delivery date
Put-away	Batch number, location, stock item number, timestamps for task creation and completion
Order-picking	Pick-list number, stock item number, order number, goods item number, employee, warehouse location, timestamps for task creation and completion
Packing	Sales order number, status, quantities of lines and total items, order type
Shipping	Departure number, consignment numbers and quantities, total quantities of items, total weight, status, timestamp for creation, shipping date, transport service provider

With identification numbers, such as goods item numbers or stock item numbers, the different items are traceable during the movements in the warehouse. Between the events and movements, the items are in storage and data about the storage are also available. That type of data includes locations with specific addresses for zone, aisle, place and level, and quantities of items at each location.

One part of identifying useful data was to compare the available data attributes to variables used in previous research. Some examples of data used in other studies are described in chapter *2.3 Machine learning in SCM and Warehouse management*.

3.1. Data description

The choice of applications was done mainly by analysing which application that has a rich and consistent dataset over long periods of time, while also being an application that can improve efficiency for Alyson WMS users. Further reasoning is described in *1.3 Methodology*. The applications chosen were *Demand prediction* and *Return prediction*. Both applications have different input variables and attributes to consider when training the model which is described in a sub-chapter for each application respectively.

The choices made when constructing a data set depend on what the goals are, and what the usage of the predictions are. Since this study is focused on warehouse management, the models should fit the purpose of the actor managing the warehouse. The used data in the case was from e-commerce businesses running their own warehouse.

Two data sets were used for each application to validate the methodology and the process phases. The data sets used for *Demand prediction* are further on referred to as D1 (*Demand data set 1*) and D2 (*Demand data set 2*) while the data sets used for *Return prediction* are referred to as R1 (*Return data set 1*) and R2 (*Return data set 2*). All data sets were first collected as raw data from the *Alyson WMS* database, and then cleaned and pre-processed in *ML-studio*.

3.1.1. Demand prediction

Two data sets were chosen that had enough consistent data for a reasonably long time period. The most important constraint for choosing data set was that it had several years of data in order to include seasonality in the models (Table 2).

Table 2: Summary of the data sets used in Demand prediction.

Data set	Total time period	No Of Orders	No. Of Lines	No. Of Pieces
D1	4 years (161001-200930)	237 240	1 109 187	2 185 597
D2	4 years (160410-200414)	130 409	278 180	340 965

D1 is from a retail company where orders are placed to the warehouse from either the companies own shops or the end-consumer through their web shop. The retail business is generally cyclical which means that the company's sales will somewhat follow the general economic conditions in the society. From trying to understand the business it could be concluded that the company would most likely have a peak in sales around the Christmas shopping.

D2 is from another retail company, mainly selling women's fashion online. Most orders are therefore directly from the end-consumer, but some orders may be from other fashion shops and resellers. The demand should be viewed as somewhat cyclical.

In D2, a subset of the orders was identified as manual adjustments of the inventory rather than actual orders corresponding to the real demand. Those orders where identified with text strings instead of numerical order numbers and could be excluded from the final data set to use in the modelling. A summary of the two data sets after cleaning can be seen in Table 3.

Table 3: Summary of data set D1 and D2 after cleaning.

Data set	Total time period	No of Orders	Avg. Orders/day	St.Dev (orders/day)
D1	4 years (161001-200930)	237 240	162	105
D2	4 years (160410-200414)	129 836	89	81

Further pre-processing of the data is described more in chapters *4.3.2 Pre-processing* for both data set D1 and D2. A detailed summary of the used data variables and characteristics of the variables chosen are described in Table 4.

Table 4: Detailed summary of the data attributes used for D1 and D2.

Date	Description	Type	Values D1	Values D2
Demand	Number of orders/day	int	0-1676	1-1084
Year	Year	cat	2016-2020	2016-2020
Month	Month number	cat	1-12	1-12
Week	Week number	cat	1-52	1-52
Weekday	Day number of the week	cat	1-7	1-7
Weekend	Week with holidays	cat	yes/no	yes/no
Lags 1-14	Computed lag variables	int	0-1676	1-1084

3.1.1.1. Data set D1

The demand could be described in three different ways and for different time periods. The demand can be the number of orders, number of order lines/products or the number of pieces. In the following, the demand refers to the number of pieces, aggregated by day, week or month.

The demand per day has a big variance, especially with high peaks (outliers) on certain days, but also a big variation day to day around the moving average of 30 days (Figure 4).

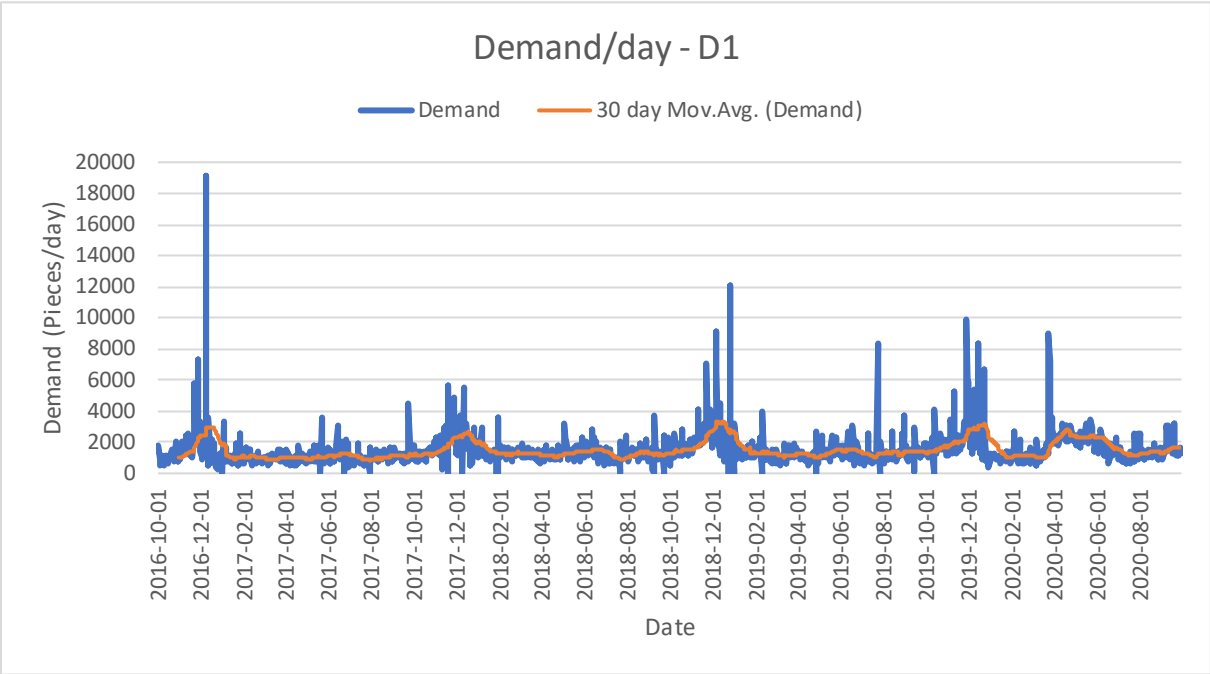

Figure 4: Daily demand (tot. pieces) for D1 in blue and 30 days moving average of the demand in orange.

The spread of demand across the days of the week was mainly characterized by the big difference between data falling into the range of "normal days" and data points that sticks out, i.e. outliers (Figure 5).

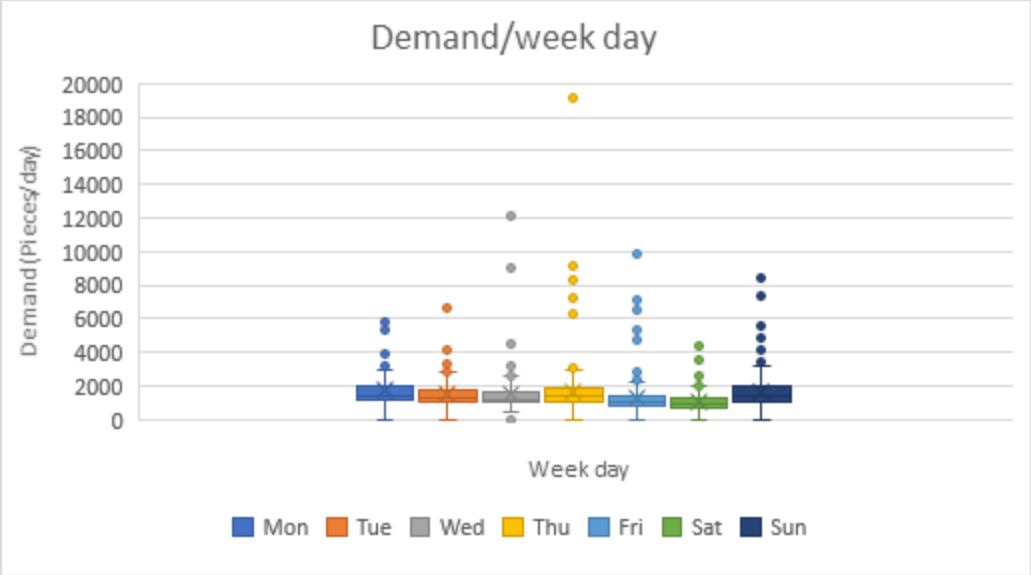

Figure 5: Demand per weekday in boxplots. The median is represented by X, the upper box border is the 3rd quartile (Q3), the lower box border is the 1st quartile (Q1), the horizontal lines are the minimum and maximum which is the interquartile range above or below Q3 and Q1 respectively, and the points are outliers which is if they exceed the minimum and maximum.

The demand on weekdays has a large standard deviation. There were more outliers above the maximum and they existed on all weekdays, while the lowest demand were found on Saturdays and Sundays with 0 demand (Table 5).

Table 5: Statistics of the demand per weekday, where lowest 10 % correspond to the 10th percentile from the minimum and highest 10 % to the 10th percentile from the maximum. Avg. of mid 80 % is the average of the data not in the lowest or highest 10 %. Std.Dev is the standard deviation.

Summary	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Average	1322	1165	1094	1592	942	723	1327
Minimum	216	499	414	315	203	0	0
Maximum	5816	3575	3115	19134	5289	3632	7359
Median	1140	1039	992	1100	791	686	1145
Lowest 10 %	829	783	759	747	560	450	760
Highest 10 %	1608	1639	1494	1888	1202	1009	1840
Avg. of mid 80%	1152	1068	1016	1157	837	705	1178
Std.Dev.	815	514	453	2344	645	465	959

When aggregating the data to demand per week, it was easier to recognize a pattern in the data, with demand peaks each year around week 45-50 (Figure 6). This could possibly be explained by being around the period of Black Friday sales. A clear downshift in the trend can be seen during the last week number 1-13 which has a high probability to be caused by the pandemic caused by the Covid-19 virus, but no data was collected regarding its' effect.

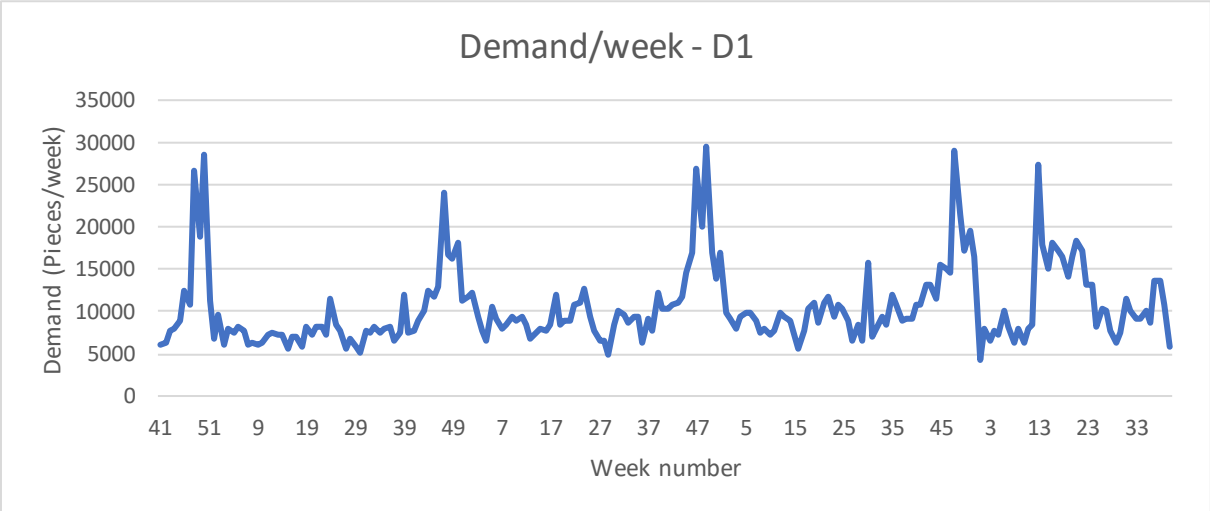

Figure 6: The demand described in pieces per week number in chronological order from year 2016 to 2020, for data set D1.

A clearer understanding of the recurring peaks in the demand present around week number 45-50 every year can be achieved by plotting the variation of demand for each week number (Figure 7).

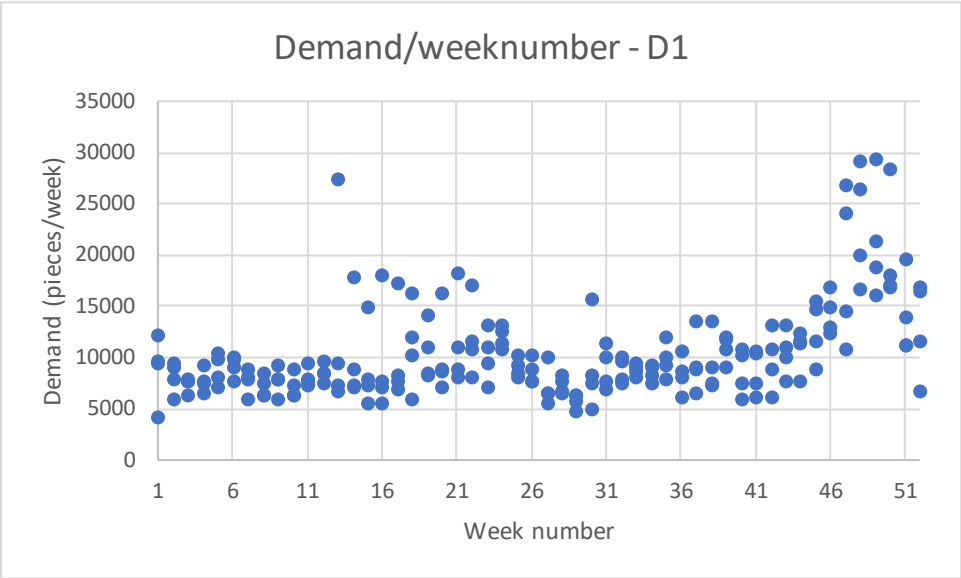

Figure 7: Spread of the demand in pieces per week number for data set D1.

Aggregating the demand per month again shows demand peaks occurring around the period earlier mentioned, but also a slight upwards trend can be detected by plotting a trendline (Figure 8).

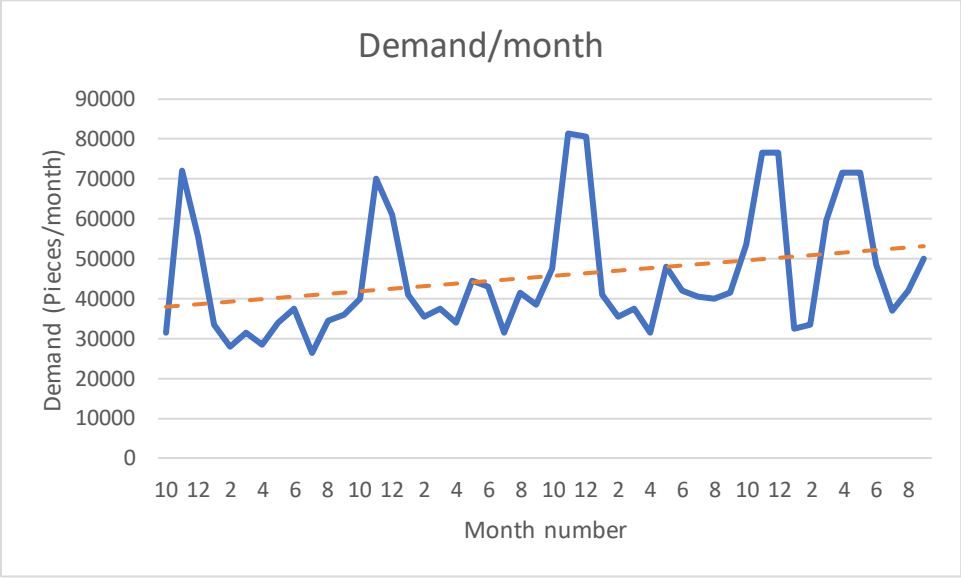

Figure 8: The demand in pieces per month described in month number of the year in chronological order from 2016 to 2020 (blue) and a linear trendline (orange) for the whole period, for data set D1.

3.1.1.2. Data set D2

The daily demand for data set D2 have an even bigger variance than D1. The outliers were more frequent, but also decreasing in size and amount later in the data set (Figure 9).

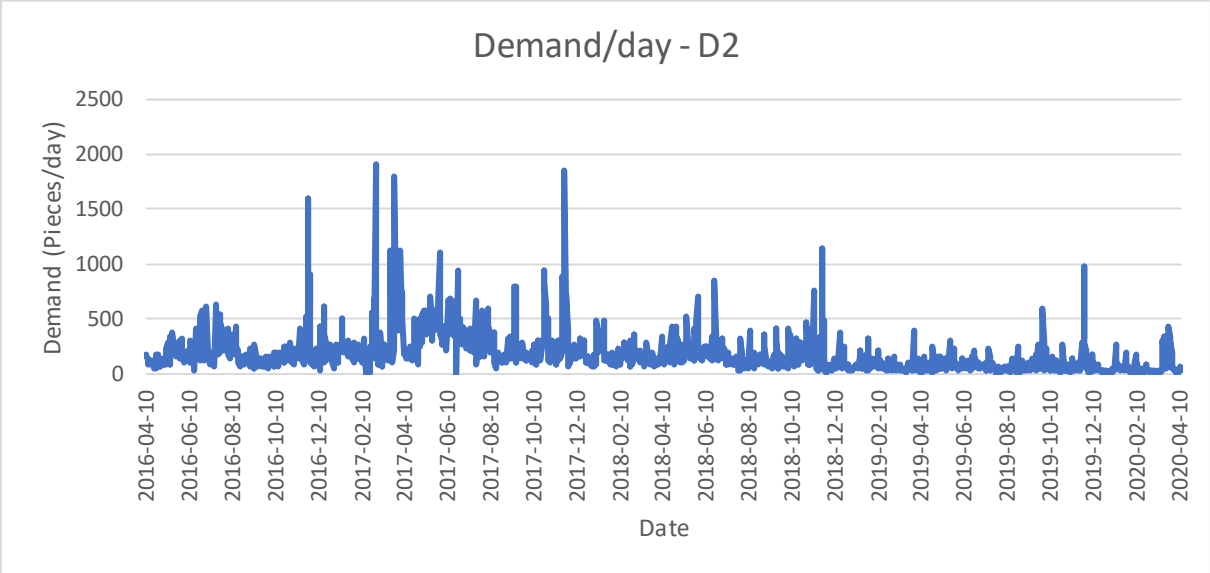

Figure 9: The demand in pieces per day, for data set D2.

The spread of the demand over each weekday was characterized by the outliers, with a large variance for each weekday (Figure 10).

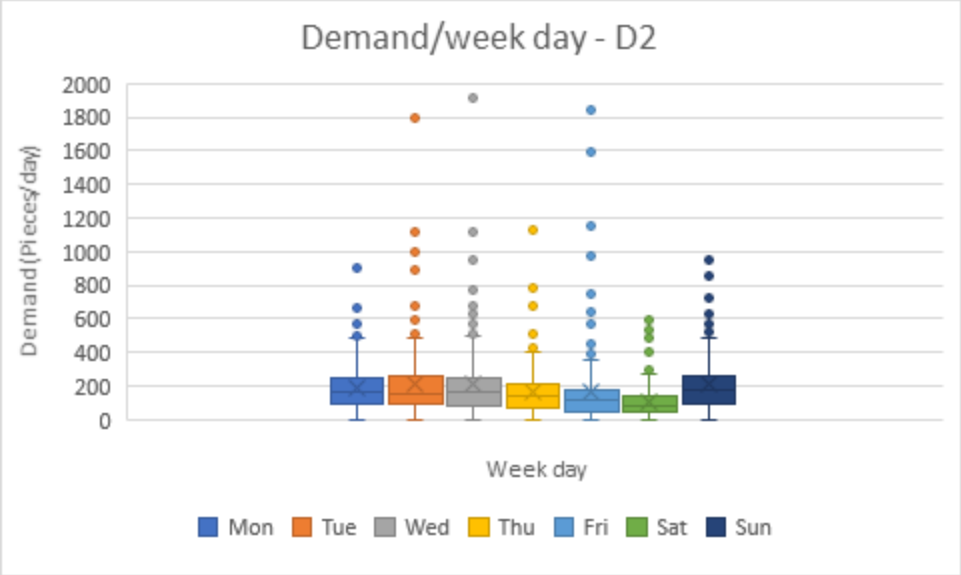

Figure 10: Spread of the demand per weekday for data set D2. See figure 5 for a box-plot description.

Aggregation on the demand per week shows a slight downwards trend time wise (Figure 11).

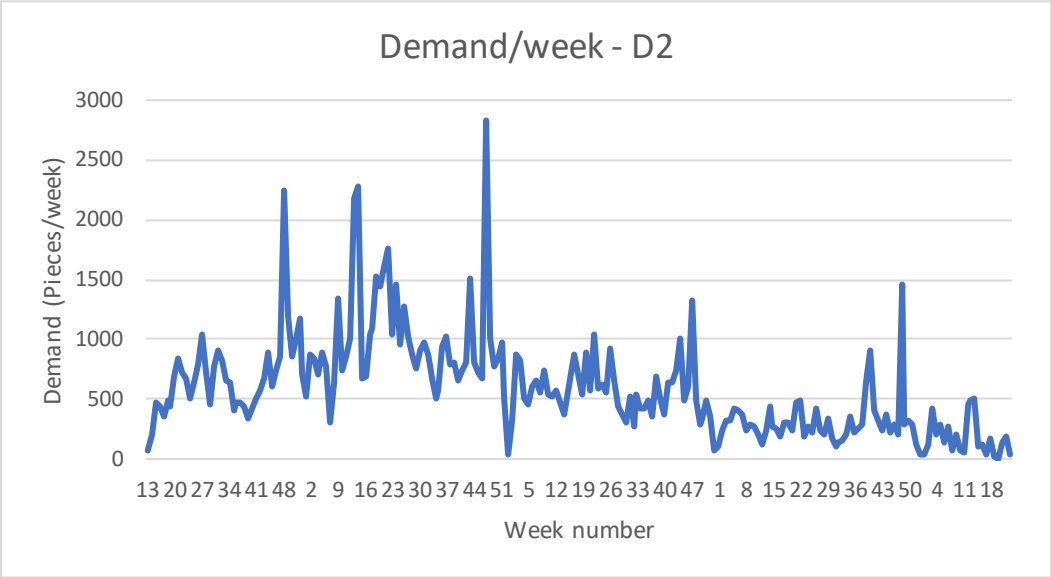

Figure 11: The demand described in pieces per week number in chronological order for the whole time period (2016-2020) for data set D2.

The downwards trend is even more obvious at an aggregation of the demand per month (Figure 12).

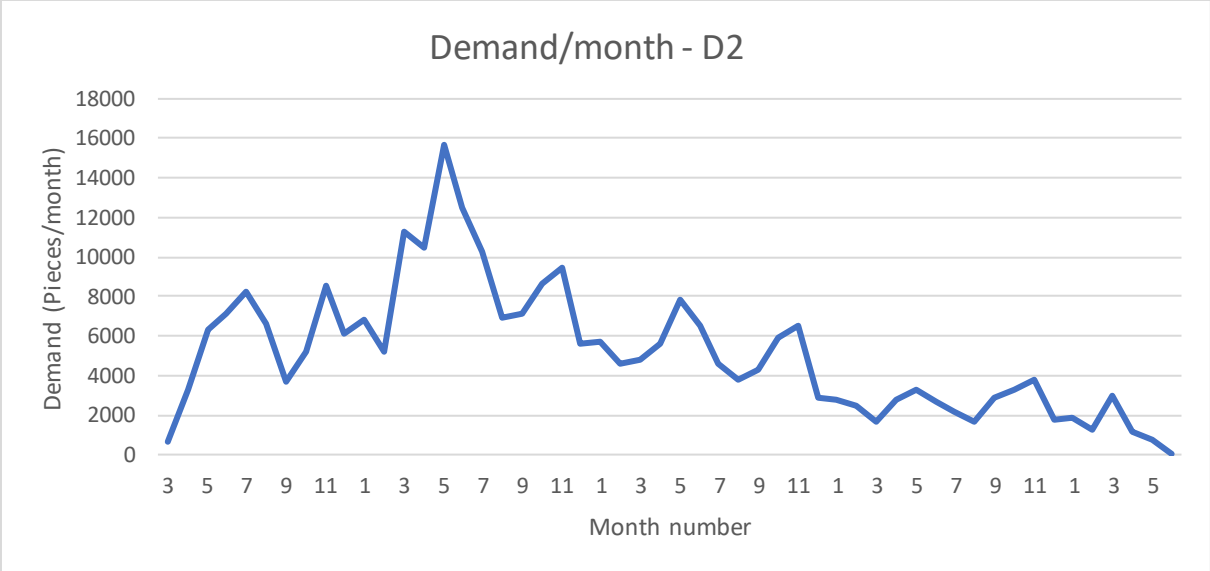

Figure 12: The demand in pieces per month number in chronological order for the whole time period (2016-2020), for data set D2.

3.1.2. Return prediction

Two data sets were selected with consistent data and a sufficiently large amount of data. No criteria were decided for the data to have a minimum time period to be considered, but consistent data for a longer time period was preferred. The two data sets used are referred to as data set R1 and R2 respectively.

Data set R1, is a retail company selling fashion online and all orders are from the end-consumer through their web shop. Data set R2, is a retail reseller selling products from a few different companies through an online web shop. All placed orders are from the end-consumer.

The data includes both the number of shipped orders and which of the orders, or how much products from the order, that have been returned. A summary of the uncleaned data sets is shown in Table 6.

Table 6: Summary of uncleaned, collected raw data sets R1 and R2.

Data set	R1	R2
Shipped orders	1 635 916	3 617 347
Returns	113 450	104 079
Time period start	2018-03-23	2018-01-04
Time period end	2021-04-07	2021-04-07

At the start of the periods from Table 6 above, data was irregular due that it was older, imported data from when *Alyson WMS* was not used by the customers representing R1 and R2. The time periods were therefore shortened to a range where it was concluded that the demand of orders and returns would represent the true demand and return demand when the customers used *Alyson WMS* conclusively. After cleaning the data sets it resulted in a less amount of data but with more consistency (Table 7).

Table 7: Summary of cleaned data sets R1 and R2.

Data set	R1	R2
Shipped orders	1 283 137	2 897 745
Returns	106 734	89 574
Time period start	2019-10-01	2020-02-28
Time period end	2021-02-28	2021-02-28

In addition to the number of orders shipped and the respective return, timestamps for both occurrences could be used as variables. There were some variables for the data that were neglected due to inconsistency and a lack of reliability in the data, such as product types and reasons for return. The variables chosen for the data sets, a description of them and the ranges of values can be seen in Table 8.

Table 8: Characteristics of the data attributes for data set R1 and R2.

Variable	Description	Type	Values R1	Values R2
Quantity Returned	Returned quantity	int	0-680	0-843
Quantity Ordered	Ordered quantity	int	2-21 661	0-37 346
Order date	Timestamp	time		
Return order date	Timestamp	time		
Year	Year number	cat	2019-2021	2019-2021
Month	Month number	cat	1-12	1-12
Week	Week number	cat	1-53	1-53
Day	Day of the month	cat	1-31	1-31
Weekday	Day of the week	cat	1-7	1-7
Weekend	Weekend	cat	1/0	1/0
Order Lags 1-5	Computed lag variables	Int	2-21 661	0-843
Return Lags 1-5	Computed lag variables	int	0-680	0-37 346

The lag-variables computed are further explained in chapter *4.4.2 Pre-processing*.

3.1.2.1. Data set R1

Since the number of returns are affected by both historic orders and returns, it is necessary to investigate data for both factors. A large variance of orders per day can be seen in Figure 13 below.

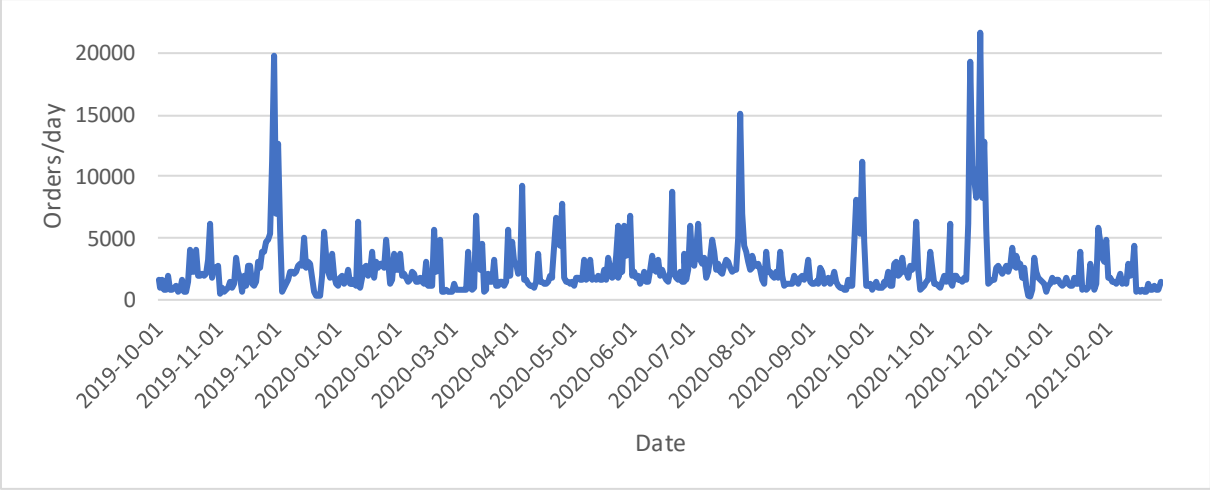

Figure 13: Orders per day for each date, for data set R1.

The variance of returns per day can be seen in Figure 14 below. The returns per day is of lower variance when relating the standard deviation to the average, compared to the variance of orders per day.

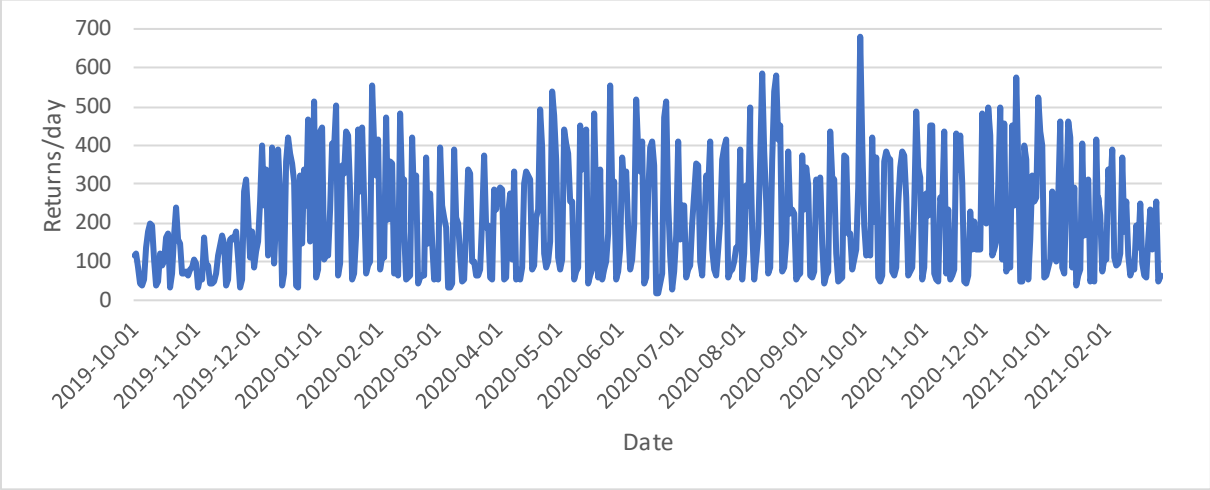

Figure 14: Returns per day for each date, for data set R1.

The average returns per weekday is significantly lower at the weekend but relatively similar during the rest of the days (Table 9).

Table 9: Summary of the returns per weekday for data set R1.

	Returns per weekday						
	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Average	176	316	296	282	230	66	79
Minimum	44	58	55	57	18	18	33
Maximum	521	555	680	576	502	256	263

The variance of returns per weekday is also shown in Figure 15 below. The only outliers in the data occur during the weekend which is because of that most of the data points are close to the average for the other weekdays.

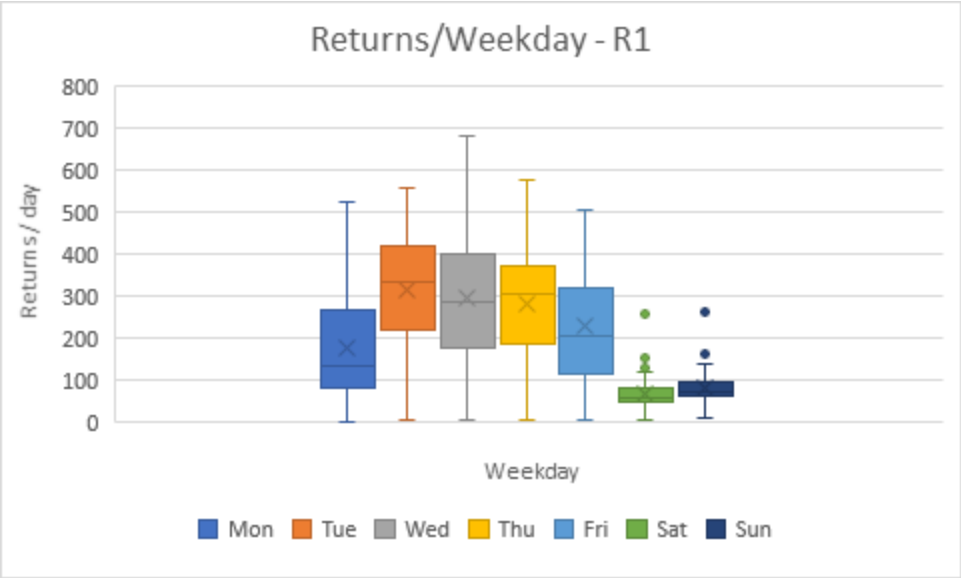

Figure 15: A boxplot of the variance of returns per weekday for data set R1. The boxplot is explained in detail in Figure 5.

Out of all orders in the data, only a subset of them was returned. For the orders that are returned, there is a delay of varying number of days before the return occurs. A comparison of the demand on the order date for the subset of orders eventually going to be returned, and the demand of returns on the return order date, for each week can be seen in Figure 16 below. It shows how orders later being returned are distributed in the following weeks instead of exactly after the same delay in time.

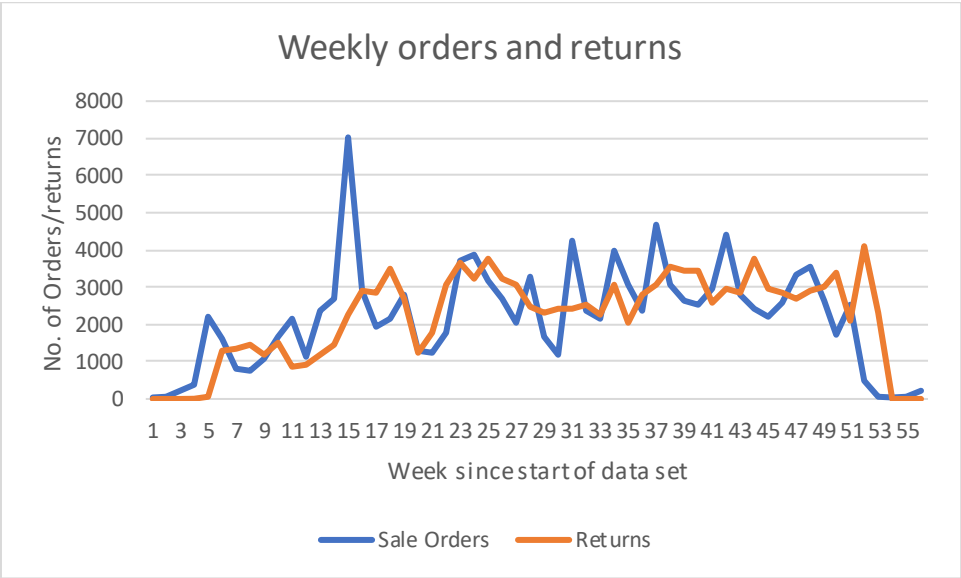

Figure 16: The demand of orders (only orders eventually going to return) and returns per week for a subset of data set R1.

The time between the order and its' return varied from zero days up to approximately two years, but the majority of orders are returned within 30 days, with the largest amount at day 5 (Figure 17).

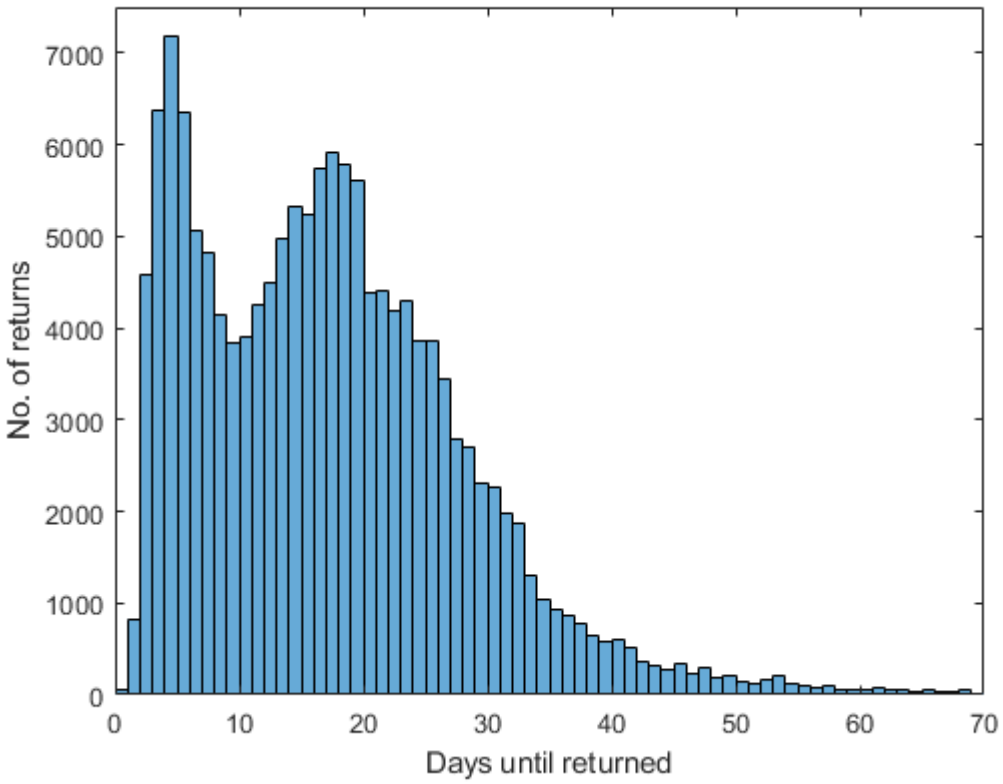

Figure 17: The number of pieces returned after a certain time since the order was shipped (i.e. days until returned) for data set R1.

Grouping the number of days until return in groups of five days gives a clearer view of the distribution and where the majority of orders are being returned (Figure 18).

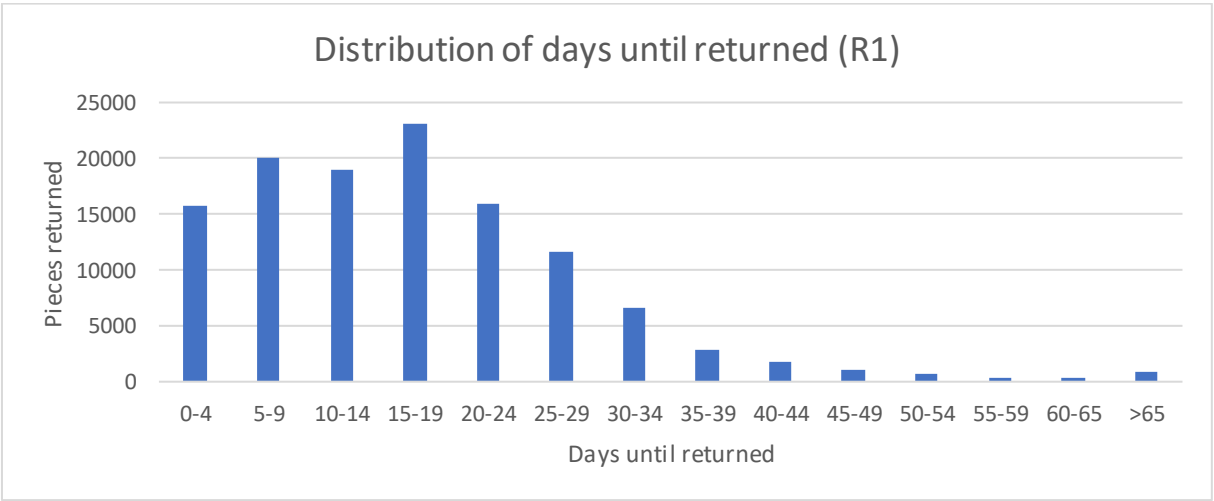

Figure 18: Distribution of days until returned in groups of five, for data set R1.

The return rate is describing the number of orders being returned out of all orders given a certain time period. For data set R1, the return rate per week is ranging from 10% to 28% with an average of 20% (Figure 19).

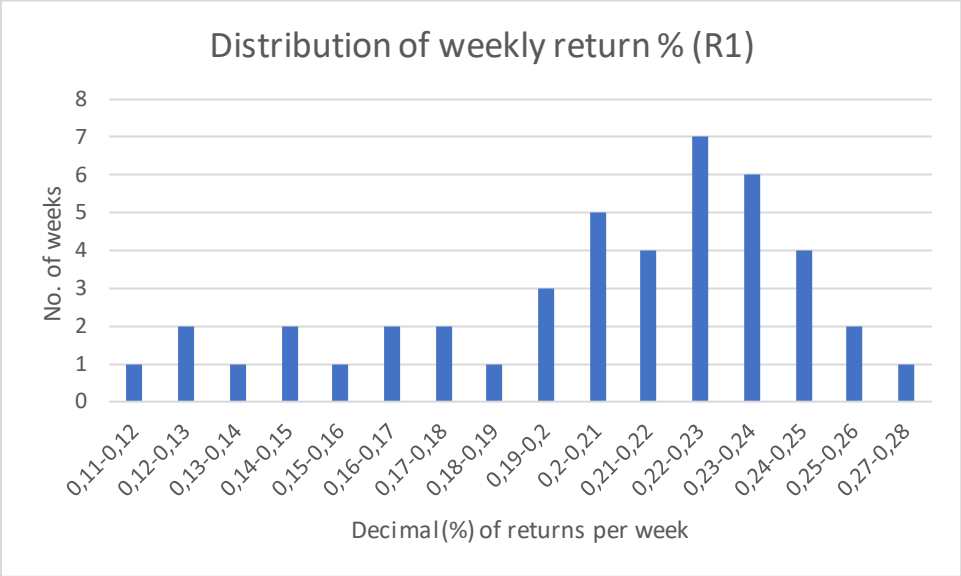

Figure 19: Distribution of weekly return percentage in numbers of weeks it has occurred, for data set R1.

3.1.2.2. Data set R2

The characteristics of orders per day for data set R2 (Figure 20) is clearly different to the same data for R1. The standard deviation in relation to the average is however lower for R2.

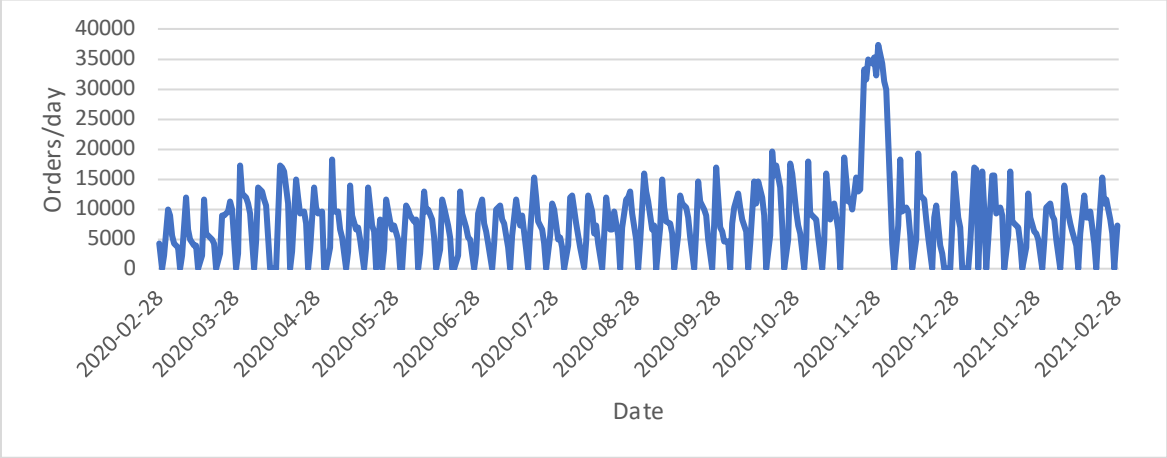

Figure 20: Orders per day for data set R2.

The returns per day follow the same pattern as the orders per day (Figure 21) with the differences that there are rarely days with zero returns, compared to the often recurring days with zero orders. The standard deviation for returns per day, in relation to the average, is lower for R2 compared to R1.

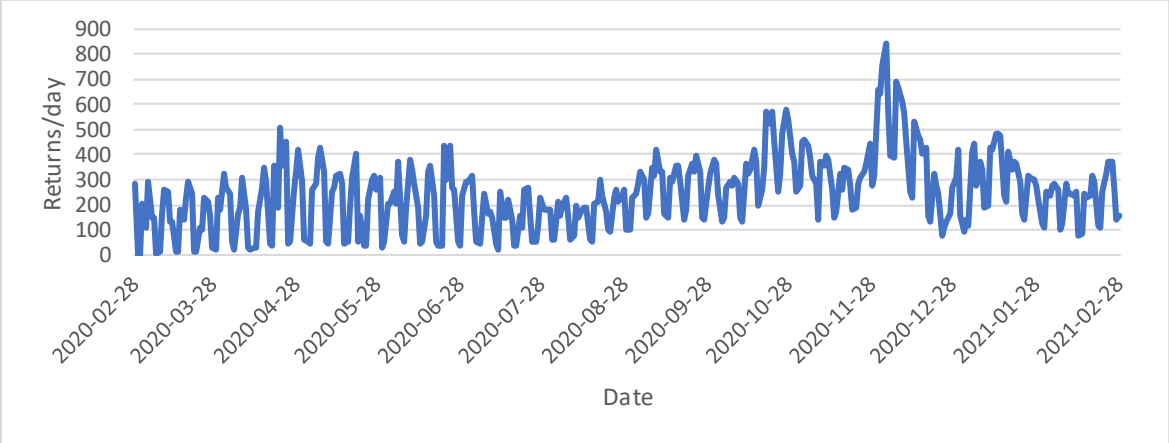

Figure 21: Returns per day for data set R2.

Just like for R1, the average returns per weekday is clearly lower at the weekend (Table 10).

Table 10: A summary of the returns per weekday for data set R2.

	Returns per weekday						
	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Average	285	297	338	309	262	112	111
Minimum	31	101	149	52	32	0	0
Maximum	693	664	757	843	575	397	390

The variance per weekday can also be seen in Figure 22 below.

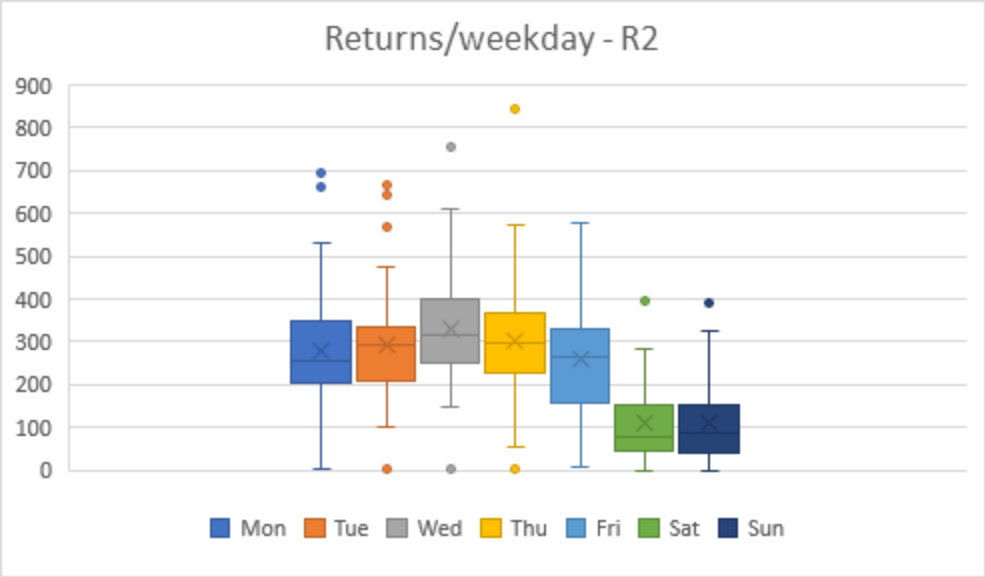

Figure 22: A boxplot of the variance of returns per weekday for data set R2. The boxplot is explained in detail in Figure 5.

The time between the order and its' return varied from zero days up to more than one year in single instances, but the majority of orders was returned within 30 days, with the largest amount at day 9-10 (Figure 23).

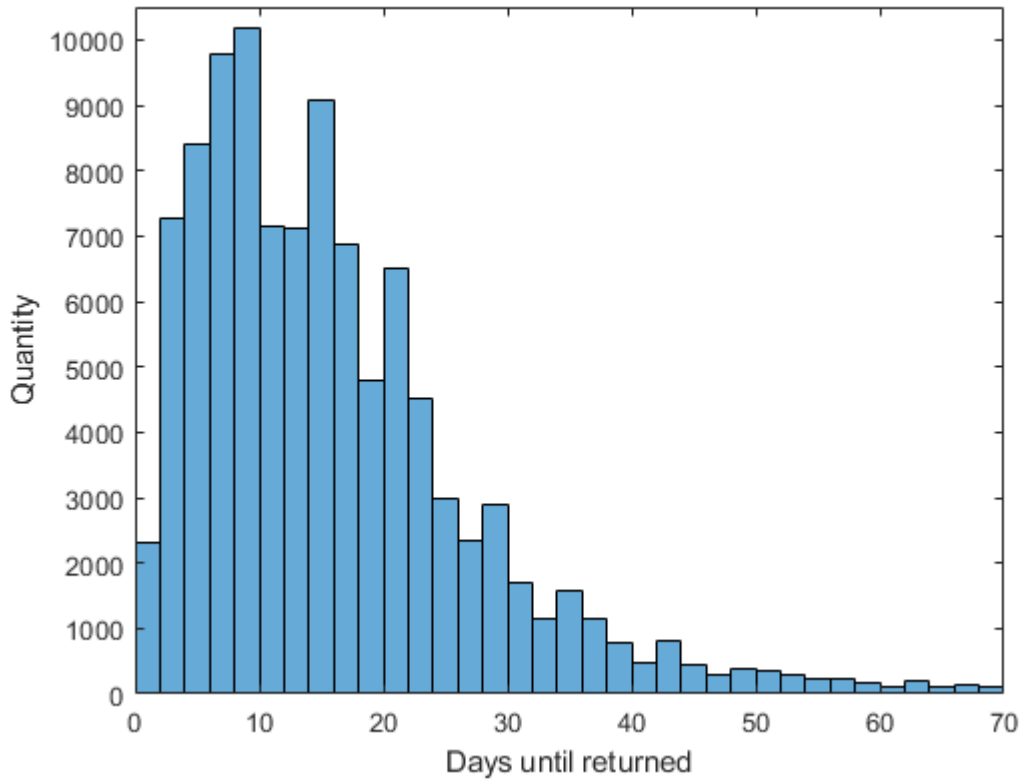

Figure 23: The number of pieces (Quantity) returned on for a number of days after the order date, for data set R2.

The return rates are in majority between 6-9 % per week, but it ranges to 1-10 % for data set R2 (Figure 24), which is clearly lower than the return rates for R1.

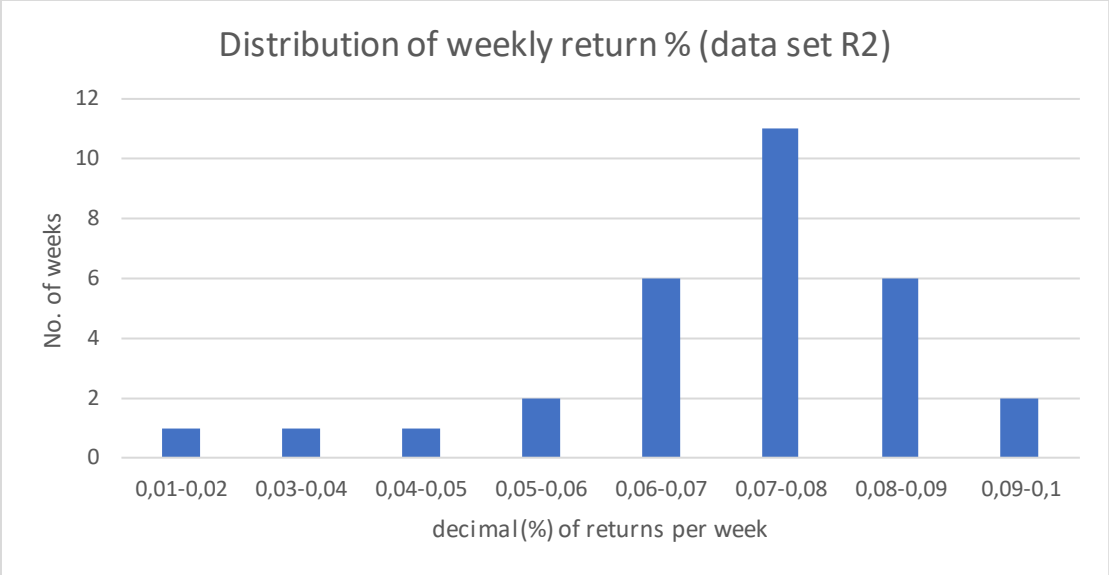

Figure 24: Distribution of weekly return percentage in numbers of weeks it has occurred, for data set R2.

4. Prediction methods

This chapter will explain the machine learning algorithms and evaluation methods used in the study. The first and second sub-section gives an overview of the platform for implementation and the algorithms used in the study. Chapter 4.3 and 4.6 describes benchmarking and evaluation methods, while chapter 4.4 and 4.5 will describe details of the data processing for the two the applications studied in the report.

4.1. Machine Learning Studio

To be able to perform ***Step 4: Modelling*** the platform and framework used was Microsoft's Azure Machine Learning Studio (ML-studio). ML-studio was a preferred platform since *Alyson WMS* is implemented in Microsoft Azure, which makes the integration between the ML-models and *Alyson WMS* more efficient. ML-studio also provides great flexibility to run both simple and more flexible predictive models for many different applications and purposes. Furthermore, ML-studio provides a graphical user interface (Figure 25) and allows both data processing and analysis directly in this interface.

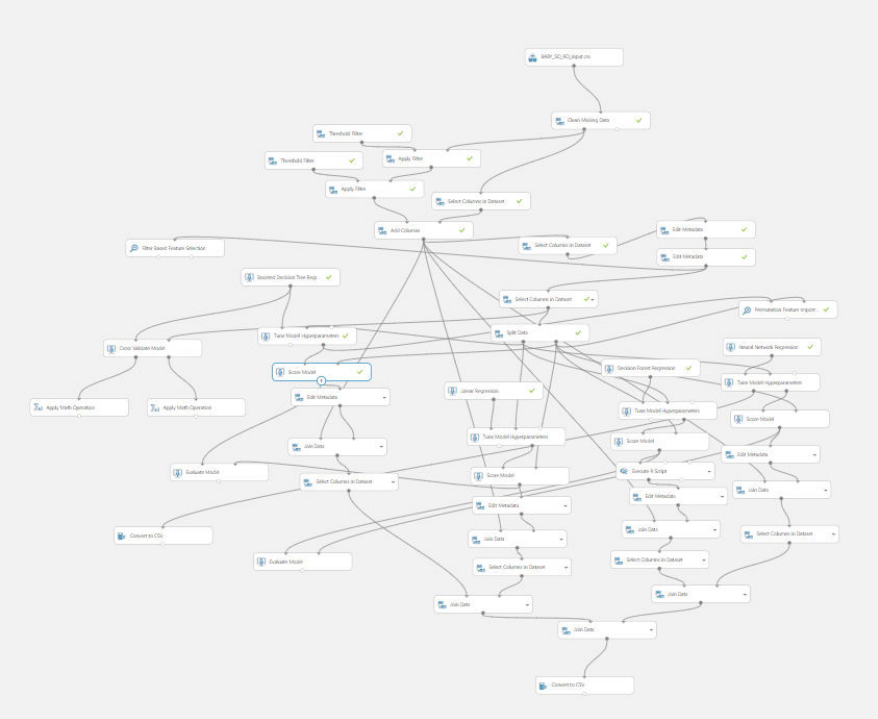

Figure 25: A visual example of a ML-experiment in Azure ML-studio.

ML-studio has support for several algorithms in each of the machine learning categories explained in chapter ***2.2.1. Machine learning in general***.

4.2. Machine learning algorithms

The machine learning algorithms used was selected due to their different characteristics. Four different algorithms were selected for both applications, with varying flexibility, to consider the *bias-variance trade-off* (see *chapter 2.2.2 Bias-Variance trade-off*).

The literature review showed promising results for using *Decision Tree* algorithms to make regression predictions of values of a target variable. In combination with advantages like computational efficiency on large data sets and ease of understanding the models, a *Boosted Decision Tree* model was selected and included in the study. To compare two different types of *Decision Tree*-based models, a similar but more resistant to high variance, version called *Decision Forests* was also included. *Decision Forests* were also mentioned in the literature as a type of algorithm suitable to the kinds of problems in this case study.

To contrast the two *Decision Tree*-based models, one less flexible and one more flexible algorithm was included for comparison. The algorithm with low flexibility was a (Multiple) *Linear Regression* algorithm, which in theory should have a low variance, but high bias. The more flexible algorithm selected was a *Neural Network* algorithm, which was used frequently in the reviewed literature for many different purposes.

One additional motivation for using these four algorithms was that they were compatible with the same input data and data structure. The implementation time can therefore be decreased significantly compared to using different data structures for each model. All the algorithms were regression models using multiple input variables to predict one output, or target, variable. For each algorithm there were different hyperparameters to consider and calibrate for the functionality of the algorithms. The algorithms used are briefly described below with references to a more detailed explanation:

Linear regression (LinearReg)

Linear regression is the most common and the simplest of the algorithms. Therefore, it is a good comparison for the other algorithms. Linear regression finds the minimum error for a linear trendline to the datapoints, often done with the least squares method (James et al. 2013). Linear regression should be able to model for example when the demand is stable around an average, including trends in continuous time. However, the performance might be worse if the daily or seasonal variance is large, and it cannot be explained by a linear relationship between the used features. More details of the algorithm can be found in e.g. James et al. (2013) and Bell (2014).

Boosted Decision Tree Regression (BstDecTree)

Boosted Decision trees are a type of ensemble models. They are relatively easy to understand since you can go through the model by pen and paper and calculate the resulting values. Decision trees are built from one root node down to 2 or more end-nodes called “leaves”. At each node, a split is made depending on if the datapoint either passes a rule or not. In regression problems with integer numbers as the target variable, different weights are applied at each of the splits. When reaching a leaf, a final result is computed by combining the weights for the datapoints’ way through the tree (Figure 26). An in-depth description can be found in e.g. James et al. (2013).

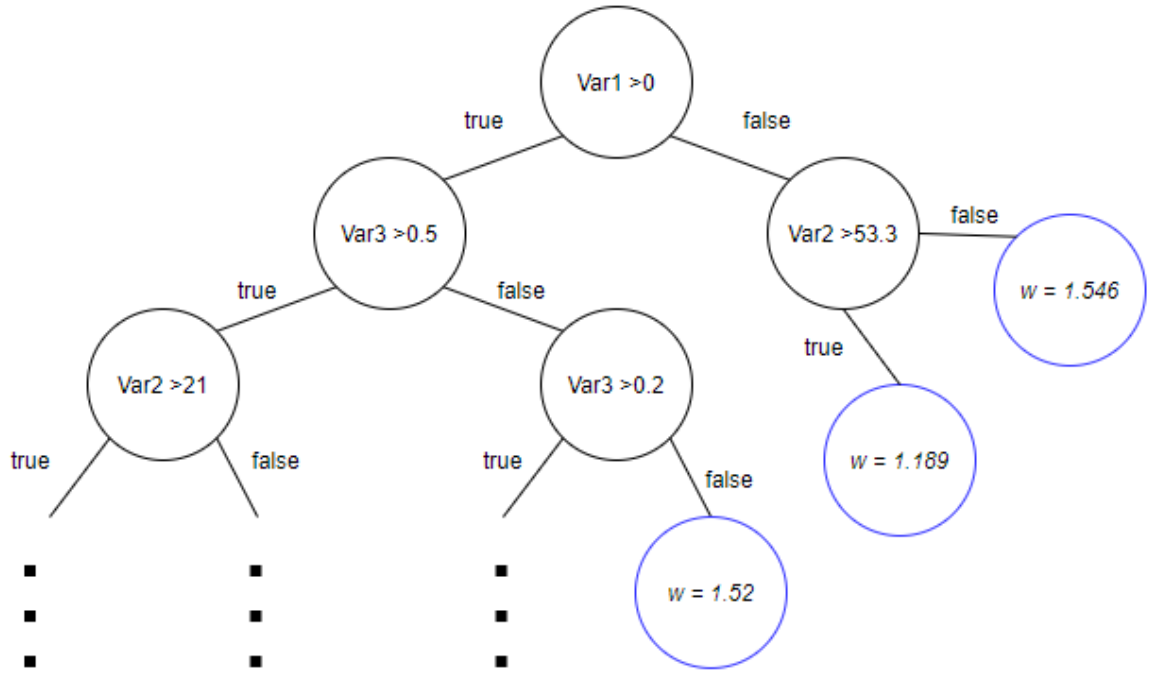

Figure 26: Example of a part of a decision tree with decision nodes (black circles) concerning values of independent/input variables (Var1-3) and leaf nodes (blue circles) resulting in final a final "weight" (w) value for the dependent/output variables in the respective leaf. At each decision node a certain "split gain", depending on what edge is chosen, is added to the weight.

The Boosted Decision Tree method used is based on the MART gradient boosting algorithm (Burges, 2010). The final model is a result of combining weaker models step by step trying to correct itself using the model error residuals. There are four hyperparameters that can be tuned to fit the specific case. The first is the *maximum number of leaves*, which corresponds to the maximum number of end-nodes in a tree. The second is *minimum samples per leaf node* which means the minimum datapoints required to create an end-node. Next, is the *learning rate* parameter, or the step size used for the learner to converge to an optimal solution. Too small learning rate will take long time, while too large learning rate can overshoot the optimal solution or never converge.

Decision Forests regression (DecFore)

Decision forests (Criminisi, 2011) are further development of decision trees, which combines multiple trees to a resulting model. The decision forest algorithm uses bagging (bootstrap aggregation) where the training data set is divided in smaller portions (bags) where datapoints can occur more than once in-the-bag. For a more detailed description, see e.g. James et al. (2013).

The hyper parameters are *number of decision trees*, *maximum depth of decision trees*, *number of random splits per node* and *minimum number of samples per leaf node*. Increased number of decision trees can decrease the bias but will also increase the computation time. The maximum depth of decision trees can increase the precision of the prediction at the risk of higher variance or overfitting. Number of random splits per node refers to a random distribution of features used in each tree. The minimum number of samples per leaf node is deciding how many data points that is needed to form an end node.

Neural Network regression (NN)

Neural network algorithms (Bell 2014) are adaptable to any regression problems but are perhaps mostly suitable when traditional regression models work poorly. The NN algorithm used in this study is of the type a fully connected case, where all input nodes are connected to all nodes in one hidden layer, which in turn are connected to the output layer with one output variable. Input nodes represent input variables, with each edge getting weighted differently when connecting to the hidden nodes in the hidden layer. During training of the model, the hidden nodes combine the input values with the weight and a bias towards each variable in a loss function which results in prediction values for the output layer (Figure 27). The output layer consists of the final target value to be evaluated before back propagating and updating the weights and biases for each input variable. Better predictions favour the input variables with stronger weights and biases when updating, such that the prediction error can converge when the final model is found.

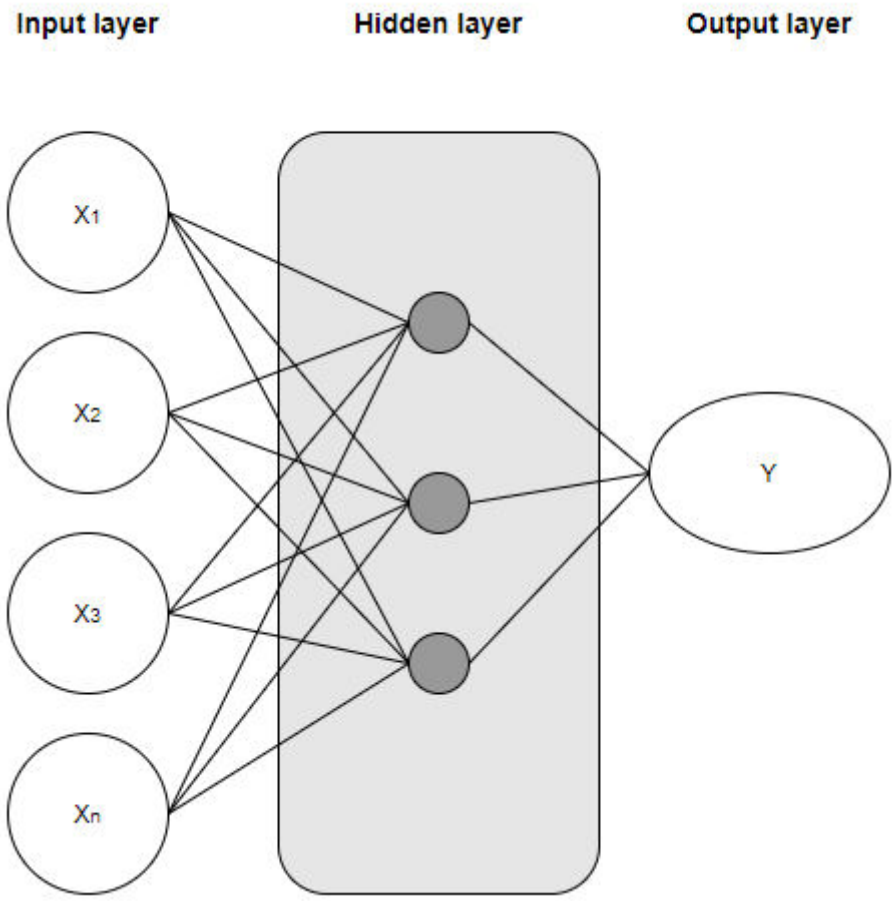

Figure 27: Example of a Neural Network with input variables (X) connected to the output variable (Y) through a hidden layer.

The hyper parameters for the *NN* algorithm used are *number of hidden nodes*, *learning rate, number of iterations*, *initial learning weight*, *momentum* and *normalizer type*. In addition to these, two types of loss functions could be used called *Squared error*- and *Cross entropy*-type functions (Nielsen 2015).

4.3. Benchmark methods

For evaluation of the performance of the algorithms, naive methods were used as benchmarking, along with two simple but common methods for predictions, *Moving Average* (MA) and *Exponential Smoothing* (ES). *MA* and *ES* was only used for the *Return prediction* application to complement the naive methods. Since the demand of returns has a less obvious correlation to its' own history,

considering both the previous demand of orders and returns, the naive methods could potentially be less useful as benchmarking methods.

The naive methods for the demand prediction application were:

1. Daily naive (D-naive) – Use today as a prediction for tomorrow.

2. Weekly naive (W-naive) – Use today as a prediction for the same weekday next week.

3. Seasonal naive (S-naive) – Use today as a prediction for the same day next year (only used for Demand prediction).

4. Two weeks naive (2W-naive) – Use today as a prediction for the same weekday two weeks later. (Only used for return prediction).

The MA benchmark was calculated by using the average of the 7 previous days as a prediction for tomorrow. The ES benchmark was calculated by using the 30 previous days as a prediction, with higher weighting for days closer to the day to predict and lower weighting for days further away.

4.4. Demand Prediction

The following subsections explains the models for the application *Demand prediction* in detail, with the reasoning behind choices made for features, data pre-processing and parameter tuning.

4.4.1. Features

Two types of features were used as input for the models, features based on time and features based on lag. The raw data used were a date and number of orders for that date. From that, characteristics of the date are used to create features. These features were *Year, Month, Week, Weekday, Weekend*, where all of them was configured as categorical features. The lag features were created by using the number of orders for previous days, starting from the first previous day, *Lag1*, and continue for each previous day up to *Lag14*, i.e. the amount of orders 14 days earlier.

4.4.2. Pre-processing

The data cleaning needed is described briefly in chapter *3.1.1 Demand prediction*. The input variables were included and formatted as described in chapter *4.4.1. Features*. Each data set were then divided into one training data set to train the model with, and one test data set, to evaluate the trained models with. 80% of the data was randomly selected as training dataset and the remaining 20% was used as test dataset.

Experiments were run with normalized features since it was suggested in the literature but due to marginal effect and more straightforward interpretation of the final results, it was decided to not use normalized features. Some calibrations of the hyperparameters were done with normalized features to be able to compare the results between data sets, but since the most important comparisons was done on the prediction error of the test data sets, using the mean absolute percentage error (MAPE), it was viewed as a comparable error measurement. As seen in chapter *5.1.2. Results D1* there were also experiments done on data set D1 where outliers in the data was excluded to investigate the effect on the model performance.

4.5. Return Prediction

The following subsections explains the models for the *Return prediction* application in more detail, with the reasoning behind choices made for features, data pre-processing and parameter tuning.

4.5.1. Features

In the raw data for return predictions both the amount of returns and the amount of orders for each date were included. Similar to the demand prediction, same time related features were created from the date (*Year, Month, Week, Weekday, Weekend*), but in addition two types of lag-features were included, one for returns and one for orders. As can be seen in *Chapter 3.1.2.1. Data set R1,* share of orders that has a return time larger than 30 days is quite low, which means that most of the orders will be returned after 30 days. To be able to include this characteristic, while at the same time not creating too many lag features, the lags were aggregated to previous weeks instead of days. In total 10 lag-features, 5 for returns and 5 for orders, where included in the input.

4.5.2. Pre-processing

Data samples with missing values in one or more features were excluded from the data set. Time related features was configured as categorical data, meaning for example that the weekday feature had 7 different categories, 1-7. The lag features were computed by summarizing the orders and returns for the previous 1-5 weeks to order-lag-features and return-lag-features. The selection of training data was 80 % randomly picked data points from the whole data sets, and remaining 20 % was used as test data.

4.6. Evaluation metrics

To complete ***Step 4. Modelling*** and ***Step 5. Evaluation***, one must decide how to measure the success of the algorithms and the final results of the study. The choice of evaluation metrics has a direct effect on both the modelling and the analysis of the results. The algorithms use an evaluation metric to improve itself on the training data from one iteration to the next. Tests with different metric could help get an understanding of the magnitude of the impact or possibly rule out any problems with the model and the data. For doing analysis of the results on the other hand, it is possible to compute a few metrics to compare the different algorithms as well as comparing the applications to each other.

During the model training phase, the chosen metric was **Mean Absolut Error** (MAE) for all algorithms and applications. This was because of simple interpretation what the metric is presenting. An option could be for example the **Root mean squared error** (RMSE), which is perhaps the most commonly used metric. One advantage with MAE is that it is more robust to outliers than RMSE. However, it was found that the choice of evaluation metric did not affect the result significantly, where for example the best found model using RMSE were the second best model using MAE, but with a very small difference (<0,1%).

To evaluate the final result and be able to answer the research questions, a few different metrics were used. Together with earlier mentioned MAE and RMSE, a more comparable metric is when you evaluate the error in terms of percentages. Two ways of looking at that is **Mean Percentage Error** (MPE) and **Mean Absolute Percentage Error** (MAPE). MAPE was used to show the average magnitude of the errors in percentages, whereas MPE helps to understand if the model is generally giving predictions lower or higher than the true values.

5. Results and analysis

The results and analysis for the experiments done in the case study are presented in this chapter. The results are divided into sub-chapters for each application and data sets for each application. A comparison with the chosen benchmarking methods is shown in terms of both training error and test error. The evaluation metric MAPE is highlighted in the tables, together with the value for the best found ML-model.

5.1. Demand forecasting

The results for the demand prediction are presented in the following sub-chapters. Each data set was split to a training data set and a test data set. The training data set was 80% of the data points and the test data set was 20 % of the data points which is along the lines of suggested distributions found in the literature. The split was done randomly so that both training and test data was distributed over the whole time period.

5.1.1. Parameter tuning results

The parameter tuning was done by running a parameter sweep on the training set of the data. First, a full parameter sweep was done for each algorithm. That showed which ranges of parameters values that gave the most promising result, and the range of values could be narrowed down to further tuning. Then tests were done with only one variable parameter at a time, having the other parameters fixed. This gave an understanding of the effect of each parameter and in what ranges it was likely to find the best parameter settings.

When running a full parameter sweep for the BooDecTree algorithm on data set D1, it was found that the error became static at a certain point when varying the parameter *Minimum leaf instances*, and that there were two local minima (right plot in Figure 28). The error decreased as the *Learning rate* decreased (left plot in Figure 29) and the error also decreased as the *Number of trees* decreased (right plot in Figure 29).

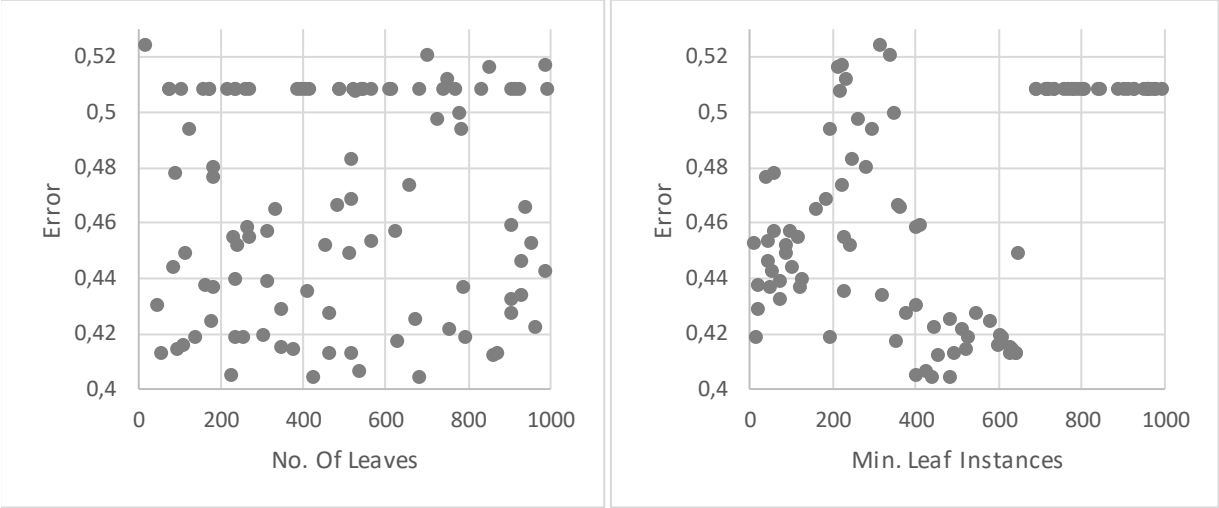

Figure 28: To the left: The normalized training error (MAE) compared to the parameter values of Number of Leaves for the BstDecTree algorithm on D1.
To the right: The normalized training error (MAE) compared to the parameter values of Minimum leaf instances for the BstDecTree algorithm on D1.

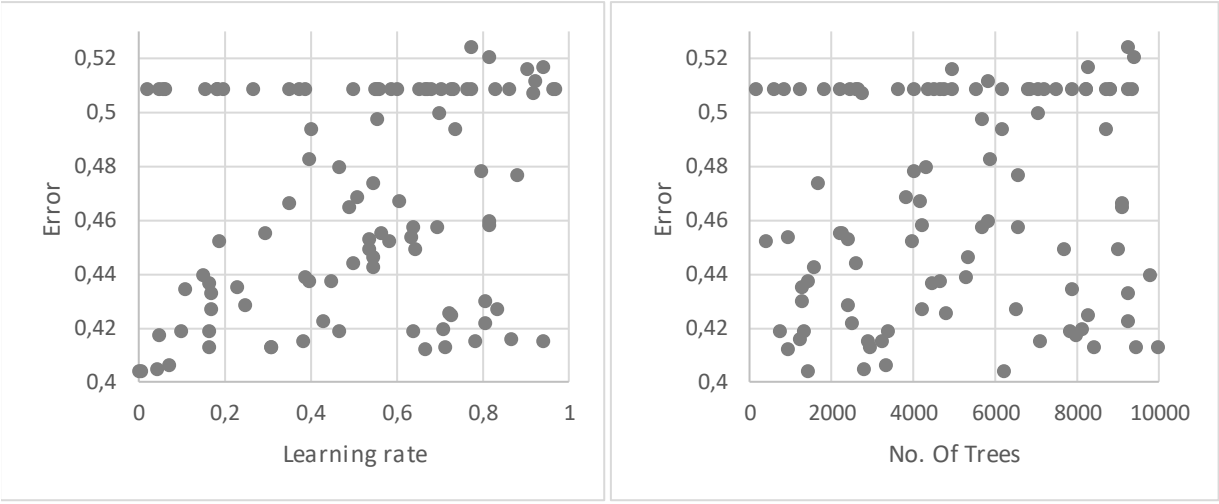

Figure 29: To the left: The normalized training error (MAE) compared to the parameter values of Learning rate for the BstDecTree algorithm on D1.
To the right: The normalized training error (MAE) compared to the parameter values of Number of trees for the BstDecTree algorithm on D1.

The same characteristics of the parameter values were found when doing a full sweep on data set D2, with a decreasing error when decreasing the *Minimum leaf instances* (right plot in Figure 30), *Learning rate* (left plot in Figure 31) and *No. of trees* (right plot in Figure 31).

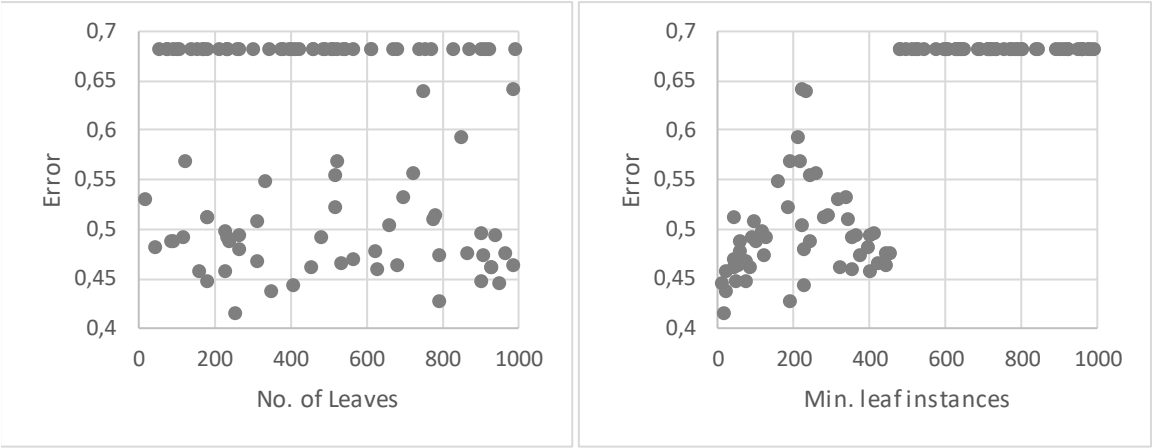

Figure 30: To the left: The normalized training error compared to the parameter values of Number of Leaves for the BstDecTree algorithm on D2.
To the right: The normalized training error compared to the parameter values of Minimum leaf instances for the BstDecTree algorithm on D2.

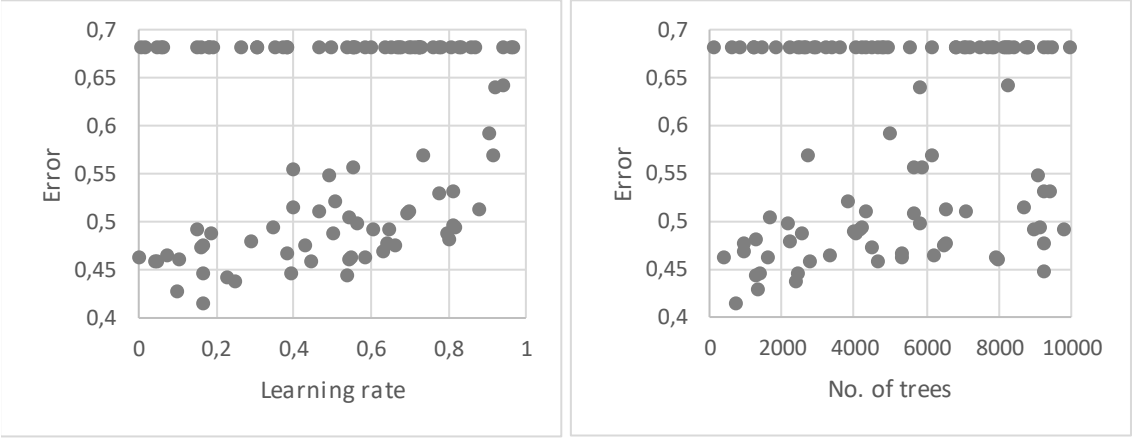

Figure 31: To the left: The normalized training error compared to the parameter values of Learning rate for the BstDecTree algorithm on D2.
To the right: The normalized training error compared to the parameter values of Number of trees for the BstDecTree algorithm on D2.

Similar parameter tuning, with a full parameter sweep followed by new parameter sweeps with smaller ranges of values, was done for all four algorithms.

5.1.2. Results D1

The best results when training the models for each algorithm were determined by minimizing the training error (Table 11). It can be seen that all the ML algorithms perform significantly better on the training dataset compared to the benchmark methods, with *DecFore* and *NN* performing best.

Table 11: Training error, data set D1.

Metric	**Benchmarks**			**Algorithms**			
	D-naive	W-naive	S-naive	BstDecTree	LinearReg	DecFore	NN
MAE	56,3	57,2	64,2	39,9	37,1	16,3	32,0
MPE	23,9%	24,7%	18,9%	12,4%	15,3%	2,3%	5,5%
MAPE	48,0%	51,1%	54,1%	31,7%	33,3%	10,8%	19,8%
RMSE	110,0	113,1	118,5	77,7	72,2	40,4	62,7

The results for each algorithm's performance on the test set can be seen in Table 12 below. The best performance was for the *DecFore*-algorithm with a MAPE equal to 30,2 %, compared to the best benchmark, D-naive, with a MAPE of 34,8 %. All the algorithms performed similar compared to the best benchmark method, but *LinearReg* and *DecFore* were able to beat the D-naive (Table 12). All the ML algorithms performed significantly better than the benchmarks W-naive and S-naive.

By comparing the training error with the test error, we can see that both *DecFore* and *NN* had significantly lower training error compared to the test error, which indicates that the methods have adapted to some of the random variations in the training data, i.e. the models are slightly overfitted. This is not surprising, since the *NN* and *DecFore* methods are the most flexible methods used in the study.

Table 12: Test error, data set D1.

Metric	**Benchmarks**			**Algorithms**			
	D-naive	W-naive	S-naive	BstDecTree	LinearReg	DecFore	NN
MAE	53,1	57,1	66,9	48,4	43,3	39,7	43,4
MPE	11,3%	17,1%	3,8%	11,6%	11,7%	10,1%	15,4%
MAPE	34,8%	42,0%	43,2%	35,4%	33,7%	30,2%	36,2%
RMSE	104,1	118,8	115,7	88,4	86,2	83,7	85,4

The R^2 plot for the best model, *DecFore*, is shown in Figure 32 below. It can be seen that outlier-values were very difficult for the model to predict since the highest value of the predictions was close to 500 while the true number of orders per day were ranging up to 1200 (Figure 32).

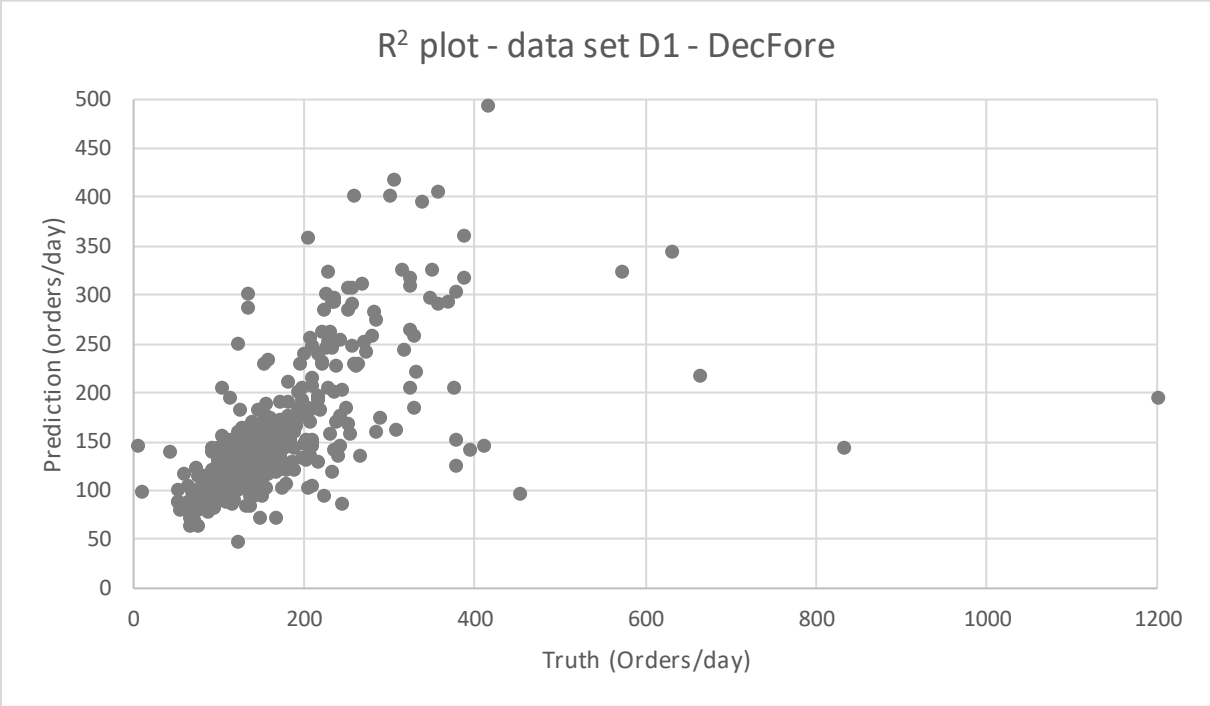

Figure 32: R^2 plot for the best model, DecFore, on the test data for D1.

The differences between the true values and the prediction for the *DecFore*-algorithm can be seen on a timeline in Figure 33 below (Note that because of the randomized split when dividing the data in to training and test sets, it is not a continuous timeline and only 20 % of the dates are included in the figure).

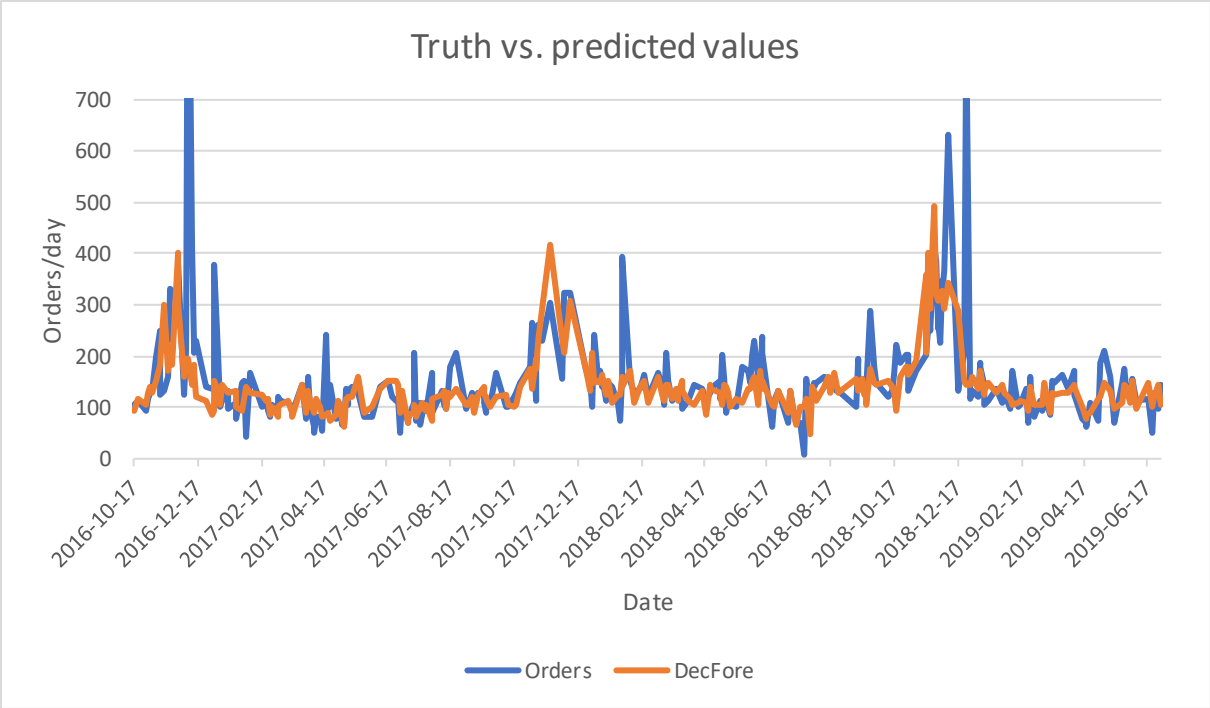

Figure 33: True values (blue) plotted versus predicted values (orange) for the DecFore algorithm, for data set D1.

In *Chapter 3.1.1.1. Data set D1* it was shown that there were significant peaks in some data points, which could in turn affect the training of the models. Therefore, the outliers were excluded from both the training and test set to investigate if the performance of the ML-algorithms could improve. The best performing algorithm was *LinearReg* (Table 13), where the increase in performance when excluding outliers could be explained by the lack of flexibility in the model when including outliers. The results for the other ML-algorithms were just slightly worse than *LinearReg* in MAPE, and *DecFore* had a lower prediction error looking at MAE and RMSE.

Table 13: Test error, data set D1 with outliers excluded.

Metric	**Benchmarks**			**Algorithms**			
	D-naive	W-naive	S-naive	BstDecTree	LinearReg	DecFore	NN
MAE	46,4	45,7	51,6	33,9	28,4	27,9	29,1
MPE	20,8%	20,7%	10,2%	12,0%	10,9%	12,1%	14,7%
MAPE	40,6%	43,3%	43,0%	33,8%	29,8%	30,5%	32,2%
RMSE	103,7	79,5	75,1	45,5	41,3	40,0	42,7

In Table 13, when removing outliers in the data set, it can be seen that all the ML models perform similar and significantly better than all the benchmark methods.

The outliers excluded were only for values ranging much higher than the average but not when ranging in the lower end, towards zero. As seen in the R^2 plot (Figure 34), a few low values of the true orders per day was clearly overestimated and hard to predict for the model.

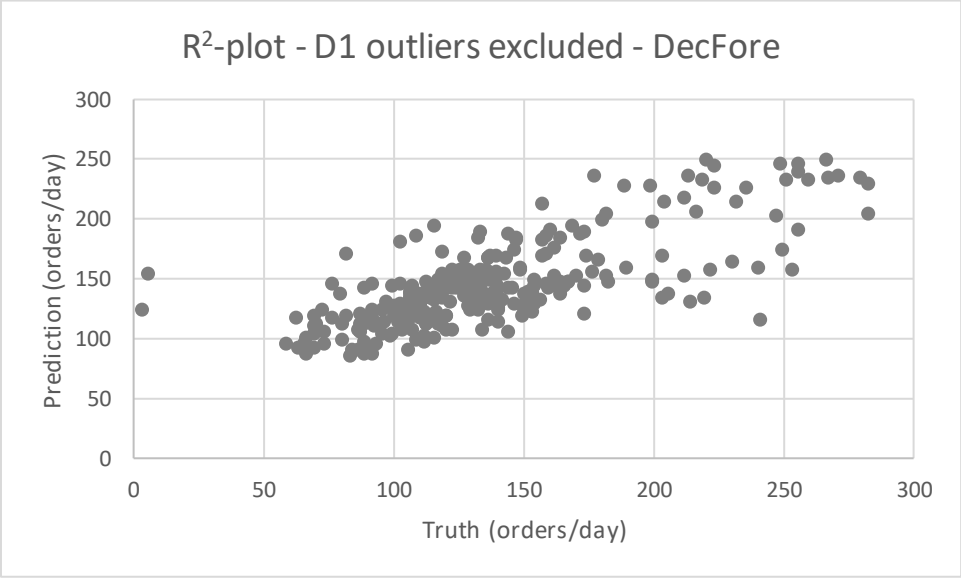

Figure 34: R^2-plot with true values versus predicted values for the DecFore algorithm, for data set D1 with outliers excluded.

5.1.3. Results D2

The training error on dataset D2 varied dramatically for the different ML methods. The training error was close to zero for BstDecTree whereas the training error for LinearReg and NN was very high (Table 14).

Table 14: Training error, data set D2.

Metric	Algorithms			
	BstDecTree	LinearReg	DecFore	NN
MAE	0,0	37,7	17,8	38,3
MPE	0,0%	67,9%	33,0%	94,2%
MAPE	0,0%	102,6%	42,2%	114,6%
RMSE	0,0	70,6	37,9	61,1

The results for the test set of D2 showed a much higher error overall than for data set D1, which is caused by the characteristics of the data set. The results varied more between the ML-algorithms but similar to data set D1, the DecFore and LinearReg algorithms were the only ones that performed better than the best benchmarks (Table 15). The benchmarks also gave significantly worse predictions on data set D2, compared to data set D1.

When comparing the training error with the test error for D2, it can be noted that the extremely low training error for *BstDecTree* was due to overfitting caused by a too flexible model for the amount of training data available. *DecFore* performed similar for training and test set, whereas both *LinearReg* and *NN* performed better on the test set compared to the training set (Table 15).

Table 15: Test error, data set D2.

Metric	Benchmarks			Algorithms			
	D-naive	W-naive	S-naive	BstDecTree	LinearReg	DecFore	NN
MAE	36,7	48,0	55,4	31,9	30,2	27,1	33,1
MPE	24,5%	56,5%	90,7%	28,5%	21,5%	23,3%	50,1%
MAPE	54,3%	94,2%	120,8%	55,8%	49,4%	45,0%	67,5%
RMSE	80,7	95,8	104,6	65,9	67,2	61,7	64,8

Just as for data set D1, data set D2 also had a few outliers which was too hard for the best model to predict (Figure 35).

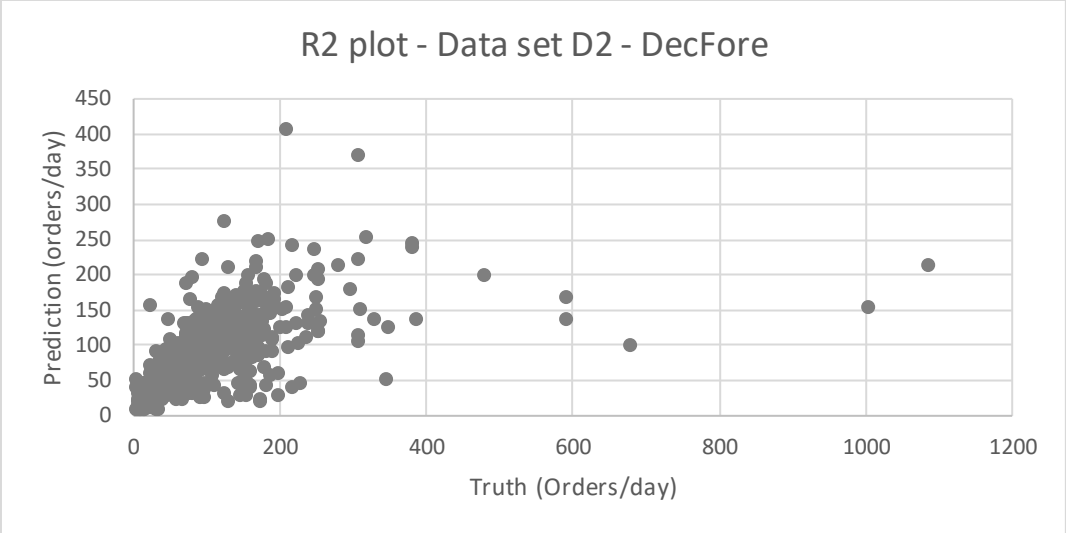

Figure 35: R^2-plot with true values versus predicted values for the DecFore algorithm, for data set D2.

While most predicted values had a relatively low absolute error, the largest error occurred when underprediction of outliers occurred (Figure 36).

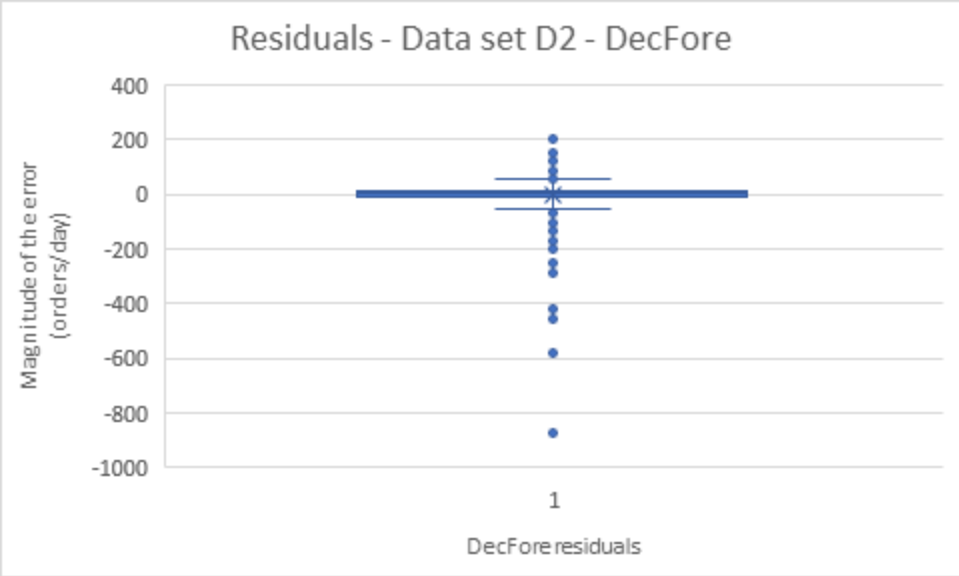

Figure 36: Residuals for the DecFore algorithm, for the test set of D2.

The same plot on a smaller scale shows that the first and third quartile is in the range of approximately +/- 20 orders in prediction error (Figure 37).

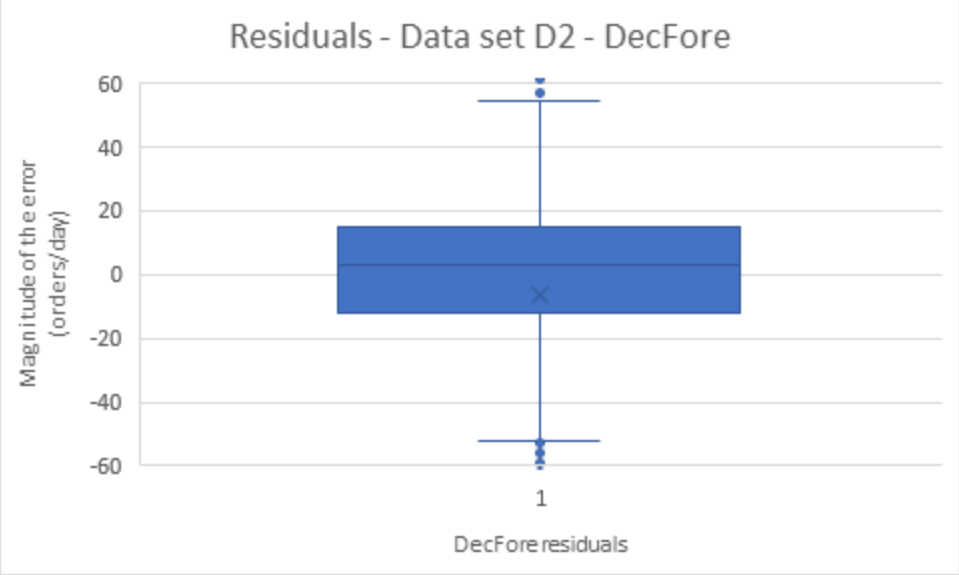

Figure 37: Residuals for the DecFore algorithm, for data set D2.

5.2. Return prediction

The results for the application *Return prediction* will be presented in the following sub-chapters.

5.2.1. Parameter tuning results

Full parameter sweeps were done on both data set R1 and R2 with results of similar characteristics and similar conclusions made about the best value ranges. Therefore, only the results of R1 are presented below as an exemplification.

The conclusions made for the parameter values for the *LinearReg* algorithm was that the *Learning rate* could result in large errors for values between 0.8 and 1, and that very low values on *Number of iteration* and *L2 Regularizer weight* could result in large errors and therefore should be excluded (Figure 38).

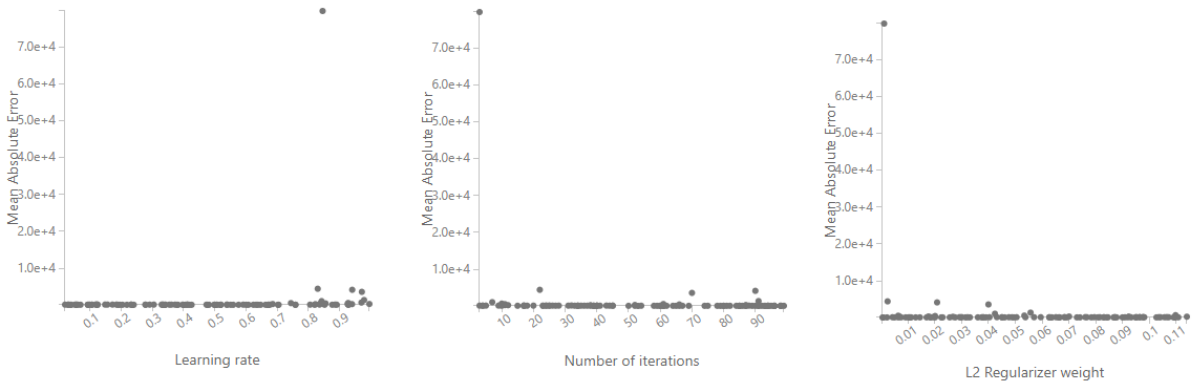

Figure 38: To the left: The mean absolute training error compared to the parameter values of Learning rate for the LinearReg algorithm on R1.
In the center: The mean absolute training error compared to the parameter values of Number of iterations for the LinearReg algorithm on R1.
To the right: The mean absolute training error compared to the parameter values of L2 Regularizer weight for the LinearReg algorithm on R1.

The parameter tuning for the BstDecTree algorithm showed that the error decreased as the *Minimum leaf instances* decreased (right plot in Figure 39), that very low *Learning rate* close to zero could result in large errors (left plot in Figure 40) and that the error decreased as the *Minimum leaf instances* decreased (right plot in Figure 39). The *Minimum leaf* instances in figure 39 only show the best ranges of values whereas a full sweep gave a static error over a certain parameter value, similar to the demand prediction.

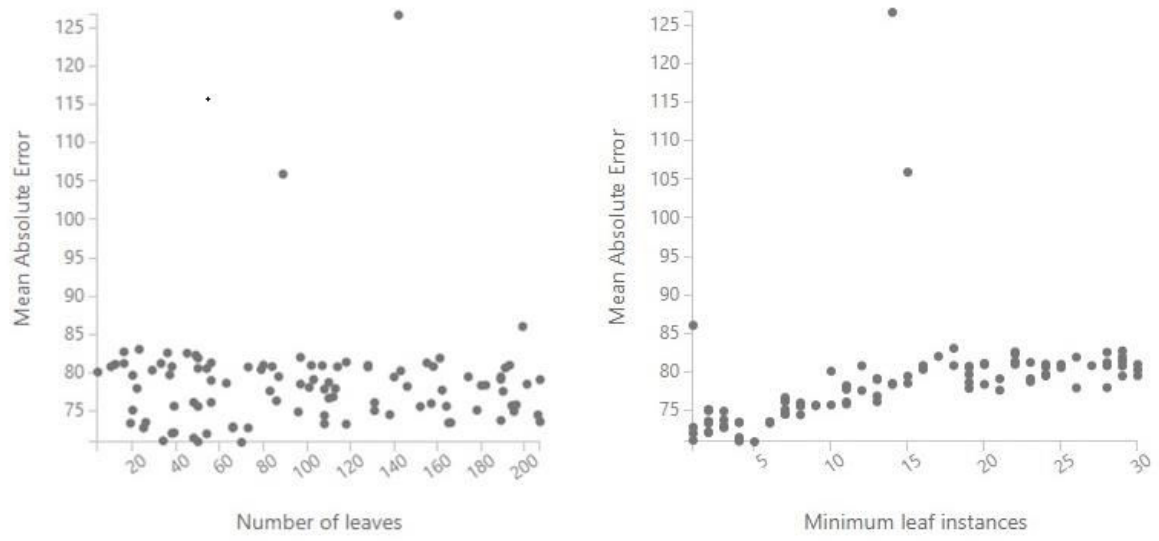

Figure 39: To the left: The mean absolute training error compared to the parameter values of Number of Leaves for the BstDecTree algorithm on R1.
To the right: The mean absolute training error compared to the parameter values of Minimum leaf instances for the BstDecTree algorithm on R1.

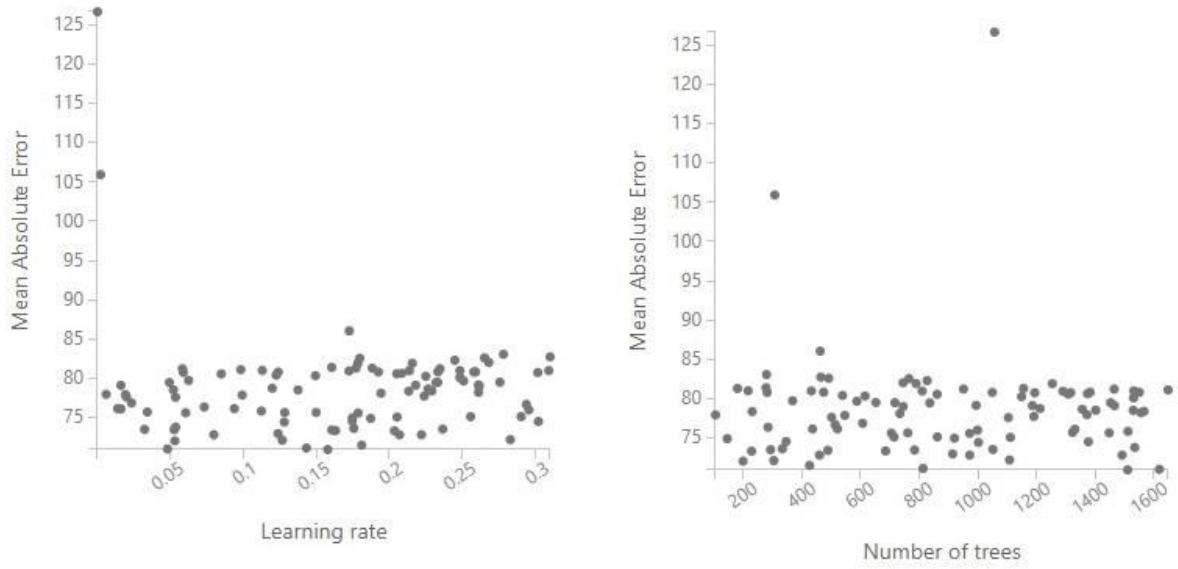

Figure 40: To the left: The mean absolute training error compared to the parameter values of Learning rate for the BstDecTree algorithm on R1.
To the right: The mean absolute training error compared to the parameter values of Number of trees for the BstDecTree algorithm on R1.

The parameter sweeps for the *DecFore* algorithm showed a close to linear relationship between the mean absolute error on the training data and the parameter *Minimum number of samples per leaf node*, where it could be seen that lower values gave lower errors (left plot in Figure 41). The values of *Number of random splits per node* seemed to have no significant impact on the error, except that a very low value close to zero could be causing a clearly higher error (right plot in Figure 41).

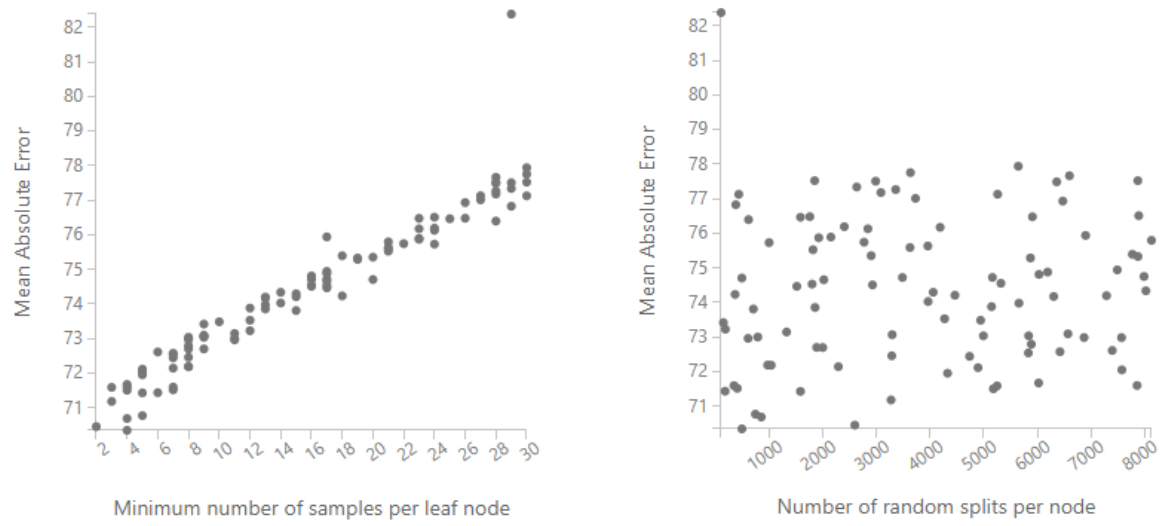

Figure 41: To the left: The mean absolute training error compared to the parameter values of Minimum number of samples per leaf node, for the DecFore algorithm on R1.
To the right: The mean absolute training error compared to the parameter values of the Number of random splits per node for the DecFore algorithm on R1.

No certain conclusions could be drawn about the correlation of the error and parameter values of *Maximum depth of the decision trees* (left plot in Figure 42) and *Number of decision trees* (right plot in Figure 42).

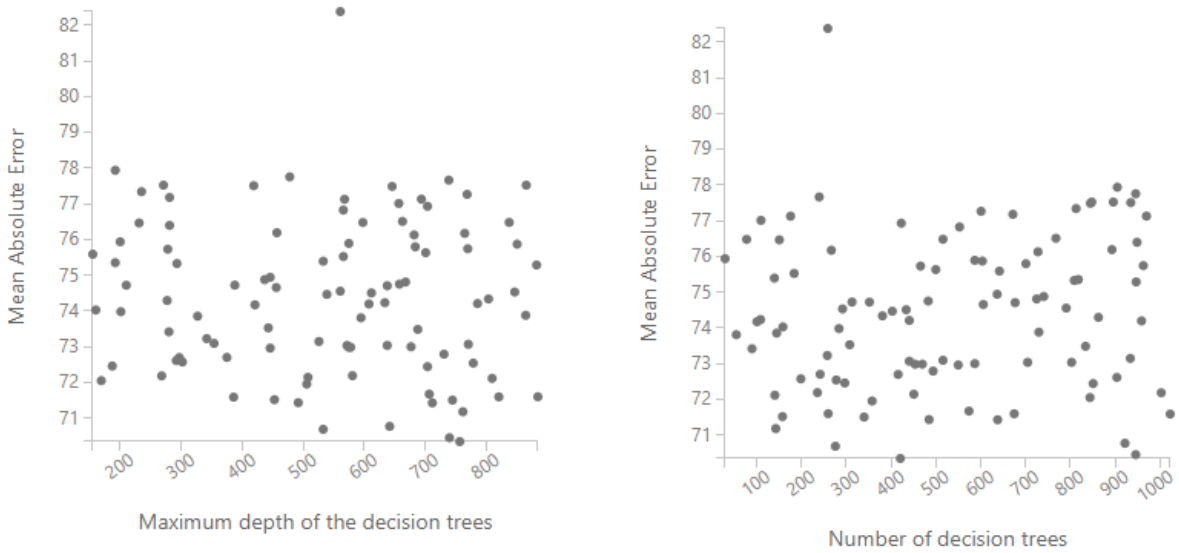

Figure 42: To the left: The mean absolute training error compared to the parameter values of Maximum depth of the decision trees for the DecFore algorithm on R1.
To the right: The mean absolute training error compared to the parameter values of Number of decision trees for the DecFore algorithm on R1.

A comparison of different experiments done with the parameter tuning for the *NN* algorithm can be seen below where Figure 43 and Figure 44 shows the results when the *Number of hidden nodes* was 20 and 5 respectively. Experiments were also done with a higher *number of hidden nodes* but resulted in larger errors. The best models in both cases were found when the *Number of iterations* where slightly above 200, and when the loss function was of the Cross-Entropy type.

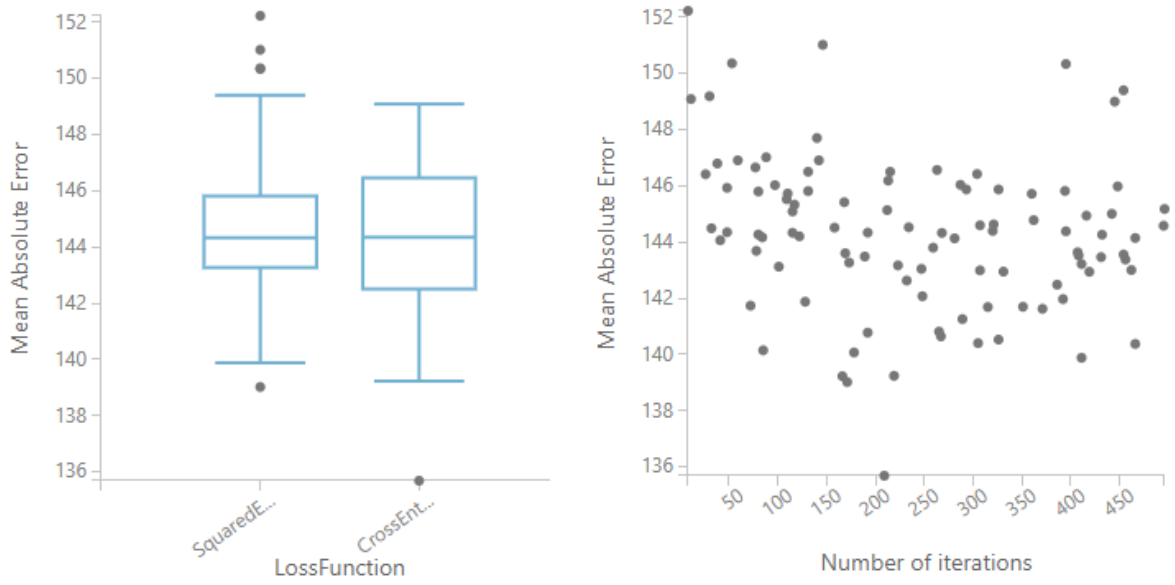

Figure 43: To the left: The mean absolute training error for two different loss functions for the NN algorithm on R1 when the number of hidden nodes was 20.
To the right: The mean absolute training error compared to the parameter values of Number of iterations for the NN algorithm on R1 when the number of hidden nodes was 20.

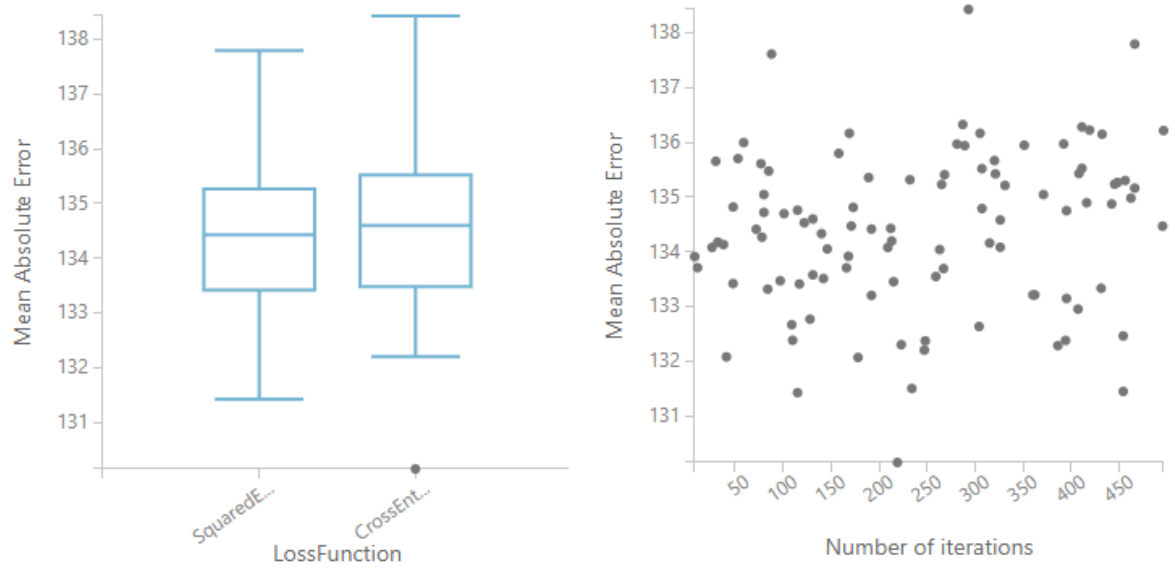

Figure 44: To the left: The mean absolute training error for two different loss functions for the NN algorithm on R1 when the number of hidden nodes was 5.
To the right: The mean absolute training error compared to the parameter values of Number of iterations for the NN algorithm on R1 when the number of hidden nodes was 5.

5.2.2. Results R1

Similar to the D2 dataset, the results for R1 dataset (Table 16) shows very large variations in training error for the different ML models. From extremely small training error for BstDecTree to extremely large training error for NN.

Table 16: Training error, data set R1.

Metric	Algorithms			
	BstDecTree	LinearReg	DecFore	NN
MAE	0,00	54,83	21,45	84,75
MPE	0,0%	119,8%	24,9%	308,0%
MAPE	0,0%	133,0%	30,5%	314,3%
RMSE	0,00	69,18	29,49	103,08

The best performing algorithm on data set R1 for the test set was the *BstDecTree*, tightly followed by *DecFore*, while the *NN* algorithm was clearly the worst (Table 17).

Table 17: Test error, data set R1.

Metric	Algorithms			
	BstDecTree	LinearReg	DecFore	NN
MAE	45,67	55,84	46,52	88,40
MPE	13,0%	32,3%	15,7%	60,9%
MAPE	27,5%	45,8%	27,8%	77,2%
RMSE	62,24	74,17	61,03	109,89

The best benchmark method was W-naive which performed slightly worse compared to the two best ML-algorithms with a MAPE of 35,1 % compared to 27,5 % (*BstDecTree*) and 27,8 % (*DecFore*). The other benchmarks were clearly worse than the W-naive (Table 18).

When comparing the training error with the test error it can be noted that even if the test error was much higher than the training error for *BstDecTree*, it still performed best of all ML methods, although very similar to *DecFore*.

Table 18: Test errors for benchmarking methods, data set R1.

Metric	Benchmarking				
	MA 7	ExpSm.30	D-naive	W-naive	2W-naive
MAE	84,45	137,61	88,68	66,78	87,29
MPE	73,0%	126,6%	23,6%	8,0%	15,9%
MAPE	98,3%	140,8%	58,3%	35,1%	46,5%
RMSE	104,66	171,51	120,74	95,76	122,32

The best model was the *BstDecTree*-algorithm and the prediction plotted versus the truth can be seen in Figure 45 below.

Figure 45: R^2-plot showing the true values versus the predicted values for the BstDecTree algorithm, for data set R1.

5.2.3. Results R2

Also for dataset R2 (Table 19), BstDecTree showed very low training error. However, compared to dataset R1, the training error for the other ML models were significantly lower.

Table 19: Training error, data set R2.

Metric	**Algorithms**			
	BstDecTree	LinearReg	DecFore	NN
MAE	0,69	79,53	39,50	81,84
MPE	0,2%	25,8%	17,5%	26,0%
MAPE	0,5%	57,4%	29,1%	60,2%
RMSE	2,12	104,67	56,54	109,64

The algorithms performed significantly worse on data set R2 compared to data set R1, with a MAPE of 50,2 % for the best model on data set R2, compared to the 27,5 % MAPE on data set R1 (Table 20). However, there were similar characteristics to data set R1 since the same two algorithms were the only ones performing better than the best benchmark, which also was the same as for data set R1, i.e. W-naive (Table 21). *BstDecTree* and *DecFore* performed better than *LinearReg* and *NN* and much better than all of the benchmark methods, except W-naive. Comparing training and test error for dataset R2 again indicates some level of overfitting for *BstDecTree*.

Table 20: Test error, data set R2.

Metric	Algorithms			
	BstDecTree	LinearReg	DecFore	NN
MAE	75,86	84,25	72,54	84,77
MPE	27,2%	28,8%	31,9%	29,9%
MAPE	50,2%	58,8%	52,3%	60,9%
RMSE	110,93	106,27	100,10	111,00

Table 21: Test error for benchmarking methods, data set R2.

Metric	Benchmarks				
	MA 7	ExpSm.30	D-naive	W-naive	2W-naive
MAE	122,53	148,29	113,46	92,27	105,44
MPE	72,3%	132,1%	49,8%	18,8%	32,9%
MAPE	105,2%	150,0%	90,2%	54,2%	69,9%
RMSE	141,15	191,11	160,64	129,27	144,24

The prediction error was higher for data set R2 compared to data set R1, which can also be seen by the spread in Figure 46 below.

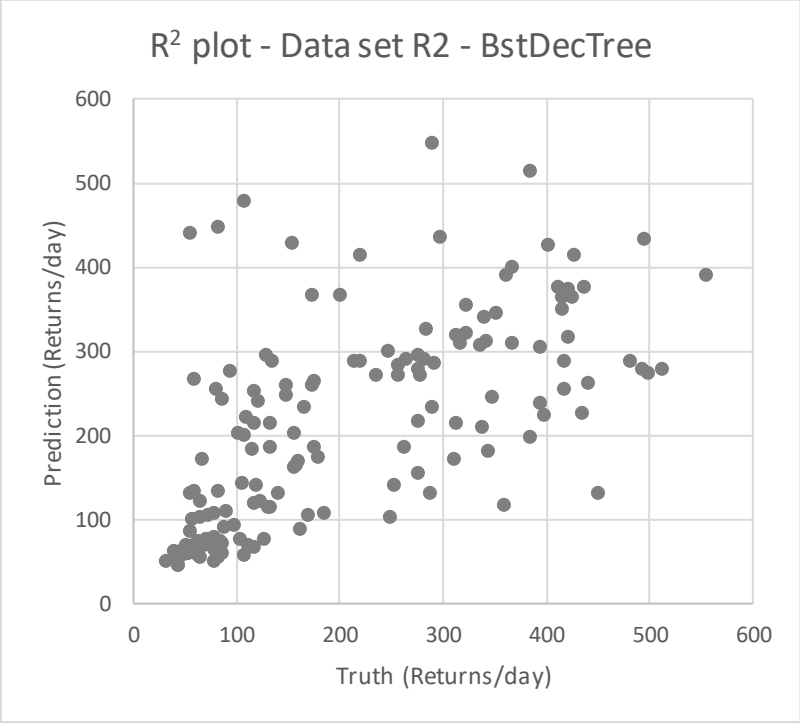

Figure 46: R2-plot showing the true values versus the predicted values for the BstDecTree algorithm, for data set R2.

6. Discussion

The results show that it is possible to gain benefits by using ML in making predictions for both applications tested in this study. The best ML-based models that was tested improves the prediction performance compared to the naive benchmarking methods. However, currently it is not obvious that the improvements of the predictions made, compared to the benchmarking models, in this use case were enough to motivate the work effort of implementing the ML-based prediction models. Which naive prediction method that will give the best performance depends a lot on the data and can be relatively hard to determine though, whereas the decision tree-based models performed reasonably well for all datasets in the use case, which could motivate the use of decision tree-based prediction for the use case. Furthermore, the ML-based models will improve over time when more data to train the models become available, which is also a motivation for using ML-based models over the naive prediction methods. Thus, once the ML models are implemented, it should be relatively easy to maintain and update the models continuously which is why the front-heavy work would be a minor issue in the end.

Both the dataset characteristics as well as the included variables affect the prediction accuracy. As seen in chapter *3.1 Data description* the variance in the data was large for all the data sets, and underlying patterns explained by the included features are hard to find. Furthermore, it is a high probability that some important variables affecting both the demand and return prediction was missing. Those variables could be both be internal, e.g. a lack of data registration, or external, e.g. data like weather, specific events, sales or promotions.

Comparing the data set D1 and D2, it is clear that the data in D1 had less variance in demand (see chapter *3.1.1. Demand prediction*) which could explain the smaller prediction error for that data set. As shown in chapter *3.1 Data description*, aggregations to demand per week has lower variance compared to daily demand which generally gives smaller prediction errors. It would not guarantee better performances though, since the underlying pattern and factors affecting the demand could be more related to daily variables. In addition, it would be less useful to predict the demand per week based on the tactical and operational planning perspective that is needed in warehouse management. The *bias-variance trade-off* would suggest that the *LinearReg* algorithm should have improved more compared to the other algorithms when excluding outliers from data set D1. This is because it should be a low variance model, not being able to find relations and patterns explaining the highest peaks in demand. However, it could be the case that neither of the other algorithms could find those relations either.

When comparing the returns per day for R1 and R2 (See chapter *3.2.1 Return prediction*) it was expected that R2, with less variance, should have resulted in ML-models with less predictions errors. However, the prediction errors were significantly higher for the best algorithms and one explanation could be that the variance per weekday instead was higher. There were no outliers for R1, except for weekends, in contrary to a few outliers for R2 for most weekdays.

Another way to improve the models could be to through further analysis of the variables and to exclude variables which could have a negative impact on the predictions. One way to do this is by doing a *Premutation Feature Importance* analysis where feature values are shuffled and its' effects on the predictions are measured. The least important features could potentially be identified and excluded from the models. In the same way, *Filter based Feature selection*, can be used to ignore features that seems redundant. Limited tests with such selection were done in this study but no clear improvement could be detected. It could be analysed more and tried in more detail in future research.

Product types for orders are available in *Alyson WMS*, but the type of naming of products in the use case would require significant work on pre-processing the data in order to use them in ML models. As previously mentioned in *Chapter 2.3*, product types were included as a predictor variable by Cui et al. (2019) partly because of the rationale that different product types would have different return rates. However, they did not find that product types were conclusively useful for all their models. It would have been interesting to compare model performance including product types in this study with their conclusions. In addition to their reasons, different product types are bought by the end-consumer with very different behaviours, especially when data is coming from online web shops from clothing retailers. Similar to how Cui et al. (2019) used multi-product orders as a variable, it could have been used for this study also.

A few studies mentioned in *Chapter 2.3* did use NN-type algorithms but most of them were focusing on classification problems. Therefore, it is not comparable to this study but such applications could probably have been found in this particular use-case also. Future work on the use-case of ML-applications within *Alyson WMS* could look at classification problems for return flows, similar to the *Return prediction* application in this study. One example is to predict the likelihood of return for individual orders.

A study by Kaneko and Yada (2016) showed that a *Deep Neural Network* method could predict demand with an accuracy of 86 %, and with a better accuracy than *Logistic Regression* for their specific data set. Even though they only predicted a binary increase or decrease in sales the next day, they showed that ML can be useful to demand predictions in other ways. In this study, NN algorithms were performing clearly worse than *Decision tree*-types of algorithms, but Kaneko and Yada (2016) shows that in another type of prediction, it could be more beneficial. Their reasoning was that NN algorithms handle sparse and binary data very well, in contrast to this study where both categorical and integer data variables were used.

The studied applications were the most promising for the use case, but a few other applications would have been possible to investigate further with the data available and could be of interest for further work. These applications include ABC-classification of products, Shortest-path routing and similar to a few studies found in the literature: optimal replenishment policies.

The process model used in this study worked well for the kind of work that is necessary with ML-applications. Step 2 and 3, i.e. data collection and data preparation, from the process model described in chapter *1.3 Methodology* was the most demanding and time-consuming parts of this study, which is generally the case according to the literature. The modelling environment in *ML-Studio* was helpful to shorten parts of these steps as well as step 4, the modelling phase. Many iterations were done after step 5, the evaluation phase, both because of verification of the models and because of experimental purposes when new insights were held.

Previous research found in the literature sometimes add computation times as a factor to be considered in the results. However, in this use case, the amount of data in combination with the prediction update interval did not put any constraints on computational efficiency. It could be of interest to study computational efficiency more if larger data sets or more data sets are used, but for this study it was not a very relevant factor. However, it should be stated that there were clear differences in computation times, but it was mainly during the parameter tuning. Once a smaller range of the parameter values were chosen, the differences in computation times were viewed as unimportant for this particular study.

7. Conclusions

This chapter includes the conclusions made to fulfil the objective of this study and to answer the research questions.

Research question 1: *What applications of Machine learning in warehouse management can be identified to be potentially valuable for the case study?*

The most valuable applications identified are *Demand prediction* and *Return prediction* which relates to the order demand and return demand respectively. Both types of demand are the underlying triggers for all warehouse operations and predictions of them will therefore serve an important role in warehouse planning. *Alyson WMS* includes possibilities for enough data for current users in these applications to enable predictions that are useful for warehouse planning.

Research question 2: *What different Machine learning techniques show potential for each of the identified applications?*

The most promising Machine Learning techniques found in this study was of the same category of algorithms: *Decision trees*. The best technique found for the application *Demand Prediction* was a *Decision Forest*-algorithm, while for the *Return Prediction*, the best technique was a *Boosted Decision tree*-algorithm. Both techniques showed a similar performance, with slightly better results than simple benchmark methods as well as other selected ML techniques, and with further improvements suggested they could be beneficial to use for the case study.

Research question 3: *What value and benefits can Machine Learning techniques offer compared to each other and to simple benchmark methods, for the identified applications in the case study?*

Using Machine Learning techniques for the identified applications needs a lot of rigorous work for implementation, but once implemented, they should be relatively easy to maintain and continuously improve. Once finding a process with promising results, adding more data and updating the models, should give better results with reasonable effort. Though simple benchmark methods, like naive methods, requires minimal work to implement and use, they will not "learn" how to improve over time like Machine learning models could. In theory, Machine Learning models should be able to at minimum produce the same results as naive methods, but with a possibility to be much better with the right conditions, which is also verified in this study. Therefore, predictions can be improved with the help of Machine Learning and serve as an input or supplement to planning for the existing customers using *Alyson WMS*.

One conclusion that can be drawn is that the performance of Machine Learning methods is very dependent on the data for each case. As seen in this study, different data sets and the characteristics of them, will produce big differences in the performance of the Machine learning methods. An implementation to *Alyson WMS* would either require an evaluation of each customer and data set one by one, or include some kind of measurement of the uncertainty in the models that is easy to interpret for the customers.

References

Ackerman, K. (1997). Practical handbook of Warehousing. 4th ed. International Thomson Publishing. ISBN: 0-412-12511-0

Bartholdi, J.J. Hackman, S. (2019) Warehouse & Distribution Science. The Supply Chain and Logistics Institute, Georgia Institute of Technology https://www.warehouse-science.com/book/editions/wh-sci-0.98.1.pdf [2020-02-11]

Baryannis, G. Validi, S. Dani, S. Antoniou, G. (2018). Supply Chain Risk Management and Artificial Intelligence: State of the Art and Future Research Directions. International Journal of Production Research, September 2018. DOI: 10.1080/00207543.2018.1530476

Bell, J. (2014). Machine Learning: Hands-on for developers and technical professionals. John Wiley & Sons 2015. ISBN: 978-1-118-88906-0

Bouganim, J. Olsson, K. (2019). Using Deep Learning to Predict Back Orders – A study in the Volvo Group Aftermarket Supply Chain. Linköping University, Industrial Engineering and Management.

Burges, C. (2010). From RankNet to LambdaRank to LambdaMART: An Overview. Microsoft Research Technical Report MSR-TR-2010-82. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/MSR-TR-2010-82.pdf [2020-09-24]

Criminisi, A. Shotton, J. Konukoglu, E. (2012). Decision Forests: A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning. Foundations and Trends in Computer Graphics and Vision Vol. 7 Nos. 2-3. P. 81-227. DOI: 10.1561/0600000035

Cui, H. Rajagopalan, S. Ward, A.R. (2019). Predicting product return volume using machine learning methods. European Journal of Operational Research 281 (2020), p. 612-627. DOI: https://doi.org/10.1016/j.ejor.2019.05.046

De Santis, R. de Aguiar, E. Goliatt, L. (2017) Predicting Material Backorders in Inventory Management using Machine Learning. IEEE 2017. DOI: 10.1109/LA-CCI.2017.8285684

Dey, A. (2016). Machine Learning Algorithms: A Review. International Journal of Computer Science and Information Technologies, Vol. 7 (3), 2016, 1174-1179. DOI: https://doi.org/10.1016/j.eswax.2019.100001

Frazelle, E.H. (1996). Word-Class Warehousing. Logistics Resources International, Atlanta, 1996.

Frazelle, E.H. (2016). World-Class Warehousing and Material Handling, 2nd ed. McGraw-Hill, 2016. ISBN: 9780071842822

Gartner (2018). Current Use Cases for Machine Learning in Supply Chain Planning Solutions. Gartner, Inc. https://www.gartner.com/en/documents/3875876/current-use-cases-for-machine-learning-in-supply-chain-p [2020-02-07]

Han, J. Kamber, M. (2012). Data Mining: Concepts and Techniques. 3rd ed. Morgan Kaufmann Publishers. ISBN: 9780123814791

Hand, D. Mannila, H. Smyth, P. (2001). Principles of Data Mining. Massachusetts Institute of Technology. ISBN-13: 978-0-262-08290-7

Inprasit, T. Tanachutiwat, S. (2018). Reordering Point Determination using Machine Learning Technique for Inventory Management. 2018 International Conference on Engineering, Applied Sciences, and Technology, 4-7 July 2018. IEEE. DOI: 10.1109/ICEAST.2018.8434473

Jacobs, R. Berry, W. Whybark, C. Vollmann, E. (2011). Manufacturing Planning and Control for Supply Chain Management. 6th ed. McGraw-Hill/Irwin.

James, G. Hastie, T. Tibshirani, R. Witten, D. (2013). An Introduction to Statistical Learning – with Applications in R. Springer Science+Business Media New York. DOI 10.1007/978-1-4614-7138-7

Karasek, J. (2013). An Overview of Warehouse Optimization. International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems, v.2, n.3, p.111-117, nov. 2013. DOI: http://dx.doi.org/10.11601/ijates.v2i3.61

Kaneko, Y. Yada, K. (2016). A deep learning approach for the prediction of retail store sales. 2016 IEEE 16th International conference on data mining workshops. DOI 10.1109/ICDMW.2016.154

Langevin, A. Riopel, D. (2005). Logistics Systems: Design and Optimization. Springer. ISBN: 0-387-24971-0

Malik, S. Jeswani, R. (2018). Literature Review and Techniques of Machine Learning Algorithm used in Business Intelligence for Inventory Management. International Journal of Engineering Sciences and Research Technology, January 2018. DOI: 10.5281/zenodo.1135987

Muller, M. (2011). Essentials of inventory management. 2nd ed. Amacom. ISBN-13: 978-0-8144-1655-6

Nielsen, M.A. (2015). Neural Networks and Deep Learning. Determination press.

Priore, P. Ponte, B. Rosillo, R. de la Fuente, D. (2019). Applying machine learning to the dynamic selection of replenishment policies in fast-changing supply chain environments. International Journal of Production Research, 57:11, 3663-3677. DOI: 10.1080/00207543.2018.1552369

Pettersson, A. (2015). Measurement of Excellence and Cost in a Supply Chain. Luleå University of Technology. ISBN: 978-91-7583-367-5

J. Shahrabi, S. S. Mousavi and M. Heydar, 2009. Supply Chain Demand Forecasting; A Comparison of Machine Learning Techniques and Traditional Methods. Journal of Applied Sciences, 9: 521-527. DOI: 10.3923/jas.2009.521.527

Shearer, C. (2000). The CRISP-DM Model: The New Blueprint for Data Mining. Journal of data warehousing, Vol.5, N.4. 2000. 101communications Publication. The data warehousing institute.

SSI Schaefer (2018). Artificial Intelligence in Logistics: Terms, applications and perspectives. SSI Schaefer whitepaper, oct. 2018. https://www.ssi-schaefer.com/en-be/innovations---trends/whitepaper-artificial-intelligence-513872 [2020-02-03]

Stadtler, H. Kilger, C. (2008). Supply Chain Management and Advanced Planning: Concepts, Models, Software and Case Studies. 4th ed. Springer-Verleg Berlin Heidelberg. DOI: 10.1007/978-3-540-74512-9

Toktay, B. van der Laan, E. de Brito, M. (2003). Managing Product Returns: The Role of Forecasting. ERIM Report Series Research in Management. DOI: 10.1007/978-3-540-24803-3_3

Wenzel, H. Smit, D. Sardesai, S. (2019). A Literature Review on Machine Learning in Supply Chain Management. Proceedings of the Hamburg International Conference of Logistics, Vol. 27. DOI: https://doi.org/10.15480/882.2478

Xing, B. Gao, W-J. Battle, K. Nelwamondo, F. (2010). Artificial Intelligence in Reverse Supply Chain Management. Proceeding of the 21st Annual Symposium of the Pattern Recognition Association of South Africa 22-23, Nov. 2010, pp. 305-310. https://arxiv.org/abs/1012.4046 [2020-01-30]

Yani, L.P.E. Priyatna, I.M.A. Aamer, A. (2019). Exploring machine learning applications in Supply Chain Management. 9th International Conference on Operations and Supply Chain Management, Vietnam, 2019.

Ye, S. Xiao, Z. Zhu, G. (2015). Identification of supply chain disruptions with economic performance of firms using multi-category support vector machines. International Journal of Production Research, 53(10). DOI: 10.1080/00207543.2014.974838