UNIVERSITY OF TWENTE.

Versuni

IMPROVING INVENTORY MANAGEMENT AT VERSUNI: MACHINE LEARNING FOR INVENTORY FORECASTING AND DYNAMIC LOT SIZE MODEL

Gregorius Ugohari Lestario, (Ugohari, Student B-IEM)
S2651505

First Supervisor: dr. Amin Asadi
Second Supervisor: dr. Dennis Prak
Company Supervisor: Alok Dimri

Preface

Dear Reader,

As I sit in my room, reflecting on the journey of the past three years that has led me to this point, I cannot express enough my gratitude and joy in sharing this work with you. This work not only marks the completion of my endeavor at the University of Twente but also signifies the end of an incredible chapter, during which I experienced significant personal growth. The voyage was not always smooth sailing, but the challenges and turbulence made it all the more memorable.

This research would not have been possible without the contribution of amazing individuals. First and foremost, I extend my sincere thanks to my university supervisors, dr. Amin Asadi and dr. Dennis Prak, for their guidance throughout the execution of this research. I am incredibly grateful for the insightful discussions we had, which steered the direction of the research to what it is today. Additionally, I am very thankful for the opportunity given by my company supervisor, Alok Dimri. Your willingness to spend time teaching me about the field, amidst your packed schedule, is something for which I am forever grateful. Finally, I want to thank my family and friends, who have always been by my side, offering constant support and always encouraging me to finish strong. At times where I was confused, stressed, and felt like giving up, your willingness to listen to my frustration and offer advice meant the world to me.

Now, it is time to look forward to what lies ahead and explore the unknown. The sea might be rough, but it is sure to lead to another great chapter.

Godspeed,

Gregorius Ugohari Lestario

Table of Contents

Management Summary

This thesis was conducted to complete a Bachelor of Industrial Engineering and Management, as part of an internship at Versuni, a leading multinational company in the domestic appliance sector. Versuni's product portfolio includes a wide range of household appliances and personal care items.

The company currently faces significant issues with inventory level forecasts, which often deviate from actual inventory levels. These inaccuracies are primarily due to inaccurate demand forecasts caused by dynamic customer trends, economic shifts, and rapid company growth. Additionally, the investigation revealed that multiple products are predicted to have zero future inventory despite forecasted demand, indicating potential stockouts. This investigation demonstrated that Versuni not only has problems with inventory forecasts but also with determining optimal order quantities. Hence, the research question is as follows.

"How can the inventory level forecast accuracy be improved and frequent stockouts be avoided by developing a machine learning model and formulating optimal order quantities?"

An analysis of the current situation revealed that the company uses a static formula to forecast the inventory levels, relying heavily on demand forecasts, which are known to be inaccurate. As a result, the current method fails to counterbalance the inaccurate demand forecasts, leading to inaccurate inventory forecasts. On top of that, it was revealed that the ordering process was done automatically by the ERP system under the supervision of the supply planner, using advanced algorithms. However, this method fails to determine the right order quantity to meet the demand forecasts, as shown by the number of stockouts.

After conducting a literature review, it was decided that several machine-learning models should be developed, and a dynamic lot size model can be implemented to find optimal order quantity, to avoid stockouts. The development of machine learning models follows commonly used procedures, involving data collection, data preparation, hyperparameter tuning, and model evaluation. Thus, machine learning models were developed, and it was found that Linear Regression and Neural Network were the two best-performing models, reducing the errors in the forecasts by a significant margin. In the end, the Neural Network was chosen due to its lower MSE, despite having a slightly higher MAE than the Linear Regression model. All in all, the machine learning model managed to produce more accurate predictions, decreasing the MSE and MAE by 76% and 26% respectively, while increasing the R-squared to 93% from previously 76%.

In determining the optimal order quantity, the chosen method was a deterministic dynamic lot size model, due to the lack of a better option given the available data, despite having stochastic demand. The model was implemented in VBA, using input parameters that were assumed due to the lack of actual parameters. As a result, the model successfully lowered the total inventory costs for 7 out of 9 products and prevented stockouts for all products. The comparison was made between the old and new order quantities, with costs calculated identically for both. Additionally, if stockout costs were factored into the cost calculations, the model would reduce the total costs for 8 out of 9 products

1 Introduction

This thesis was conducted to complete the Bachelor of Industrial Engineering and Management, as part of an internship opportunity provided by Versuni. Since Versuni aims to improve its inventory practices, this research aims to investigate a machine learning model to increase inventory forecasting and determine optimal order quantity to enhance inventory management. In Chapter 1.1, the company is introduced, and then Chapter 1.2 and Chapter 1.3 identify the problems. Lastly, Chapter 1.4 provides, the research design of this graduation assignment.

1.1 Company Introduction

“Turning houses into homes” is the motto of a leading multinational company in the domestic appliance sector, Versuni. In the era of innovation, Versuni aims to deliver comfort and warmth to houses, turning them into homes through products of the highest quality, built sustainably for your home and the planet. The name itself comes from the word “Universe”, reflecting its global presence, with operations in more than 100 countries.

Previously known as Philips Domestic Appliances, Versuni was a part of the market giant Royal Philips. In 2021, the leading health technology company, Royal Philips, decided to sell its Domestic Appliance business to focus on extending its leadership in the highly competitive health technology sector. Versuni was acquired by Hillhouse Capital, which is an investment firm with years of experience in performing digital transformation to drive innovation and business expansion. Despite the transaction, Versuni will continue to sell its products under the brand name Philips for all its sales, manufacturing, and marketing activities globally for the next 15 years, which is subject to the terms of the brand license agreement.

Versuni’s portfolio of products covers a diverse range of household appliances and personal care items. For the kitchen, Versuni offers popular appliances such as coffee makers and air fryers. For floor care, their selection includes cordless vacuum cleaners and robotic options. Versuni also offers a variety of irons and steamers for garment care. For air solution appliances, their selection includes air purifiers and humidifiers. Versuni also provides hygiene products such as shavers and trimmers, along with infant care products designed for new parents.

1.2 Problem Context

This thesis was conducted under the supervision of the inventory management team, part of the Sales and Operations Planning (S&OP) team. The S&OP team is responsible for aligning and integrating different operations within Versuni to ensure a unified strategy that drives the company forward. Their task includes product lifecycle review, demand review, and supply review, before proposing a strategy to the executive team of the company.

The S&OP team is reliant on reliable data to base their decisions, and one of the most important data is the inventory level forecasts. The forecasts are utilized by the operation teams in each market to coordinate future actions, ensuring the inventory level remains optimal. One of the actions, for example, is that if there is a predicted spike in inventory level, the market can then strategize a sales action to ensure that the products do not become obsolete.

The forecasts are created for every finished product currently available on the market and these products are being held at Versuni's warehouses throughout the world. These products are being manufactured either by Versuni's manufacturer or with external manufacturers, before being sent to the warehouses. The products would then be sold to customers either through Direct-to-Consumer (D2C) sales channels such as e-commerce and physical retail stores, or online marketplaces such as Bol and Amazon.

Currently, the inventory management team faces challenges due to the lack of integration within the SAP ERP system. Although the system contains essential data such as demand forecasts, incoming orders, and inventory levels, this information is dispersed across different interfaces. Consequently, the system cannot automatically generate live inventory level forecasts, showing the inventory forecasts from an interface. As a result, the team must manually compile these forecasts weekly.

From the inventory forecast files, it was found that the forecasts are not accurate and often miss the actual inventory level by a huge margin. After an investigation, it was found that the main contributing factor is that the demand forecast accuracy is sub-optimal. Inventory forecast is highly dependent on demand forecasts since it is a product of inventory level subtracted by forecasted demand. The inaccuracy of the demand forecast could be accredited to a variety of factors, which include changes in customer trends, and economic changes, and most relevant to Versuni's case is the rapid growth it is facing. As a result, the S&OP team has been making their decision based on inaccurate inventory forecasts.

Ideally, for this reason, improving the demand forecast accuracy would lead to an improvement in the inventory forecast too. However, this approach is outside the scope of this research, mainly due to the unavailability of weekly historical demand data, limiting the possibility of producing more accurate demand forecasts. Hence, the research will focus mainly on creating a comprehensive tool that can increase inventory forecast accuracy while still inputting the partially accurate demand forecasts. A hypothesis is developing a machine learning model would be able to increase inventory forecast accuracy, by possibly counterbalancing the inaccurate demand forecasts.

Furthermore, after investigating deeper into the inventory level forecasts, it was often noticed that multiple products are predicted to have no future inventory, despite having a positive forecasted demand. This exposes a problem, Versuni is predicted to have a lot of stockouts for various products. Thus, Versuni not only has a problem with its inventory forecasts but also with its method of determining optimal order quantities. It seems that most of the order quantities that have been placed by Versuni will not meet the forecasted demand.

As a result, this research aims to address inventory management malpractices occurring in Versuni, to provide solutions to enhance the whole operation. The focus of this research lies in the development of machine learning to increase the accuracy of inventory level forecasts and to look at relevant inventory management strategies to prevent stockouts. Variables such as order quantities are therefore crucial to solve this problem.

1.3 Problem Identification

In identifying the problem to solve, Chapter 1.3.1 defines the action problem and its evaluation metrics. In Chapter 1.3.2, a problem cluster is created to visualize the relationship between all the problems, and the core problems are analyzed. Lastly, in Chapter 1.3.3, the norm and reality of the evaluation metrics are investigated.

1.3.1 Action Problem

Versuni's inventory level forecast accuracy can be quantified with evaluation metrics such as the Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared. These metrics will be elaborated on in Chapter 4.2**.** But briefly, it is important to understand that a low MSE and MAE, and a high R-squared represent a good forecast accuracy. As for the ordering quantity, its performance can be quantified using the fill rate which is a percentage of customer demand that can be filled from available stock without backorders or delays. Therefore, these quantities would be relevant measures to quantify the action problem which is as follows.

Low inventory forecasts accuracy and frequent number stockouts.

1.3.2 Problem Cluster and Core Problem

Figure 1 depicts a problem cluster to see the relationship between all the problems leading to the action problem. After conducting a root cause analysis, it was found that there are 3 potential core problems. However, only one was chosen as the main core problem of this research. In this chapter, each potential core problem will be analyzed and given reasons as to why it affects the action problem.

1. Low demand forecast accuracy: Demand forecasts are used to forecast inventory levels and to determine the right optimal order quantities. Currently, Versuni uses a static formula (which will be elaborated in Chapter 3.2.2) to forecast its inventory level, which means that it is reliant on accurate demand forecasts to produce accurate inventory level forecasts. Furthermore, as discovered, order quantities for some products are expected to not meet the forecasted demand. The reason is that order quantities are placed by judging how much demand is expected in the future. Hence, Versuni might produce sub-optimal order quantities.

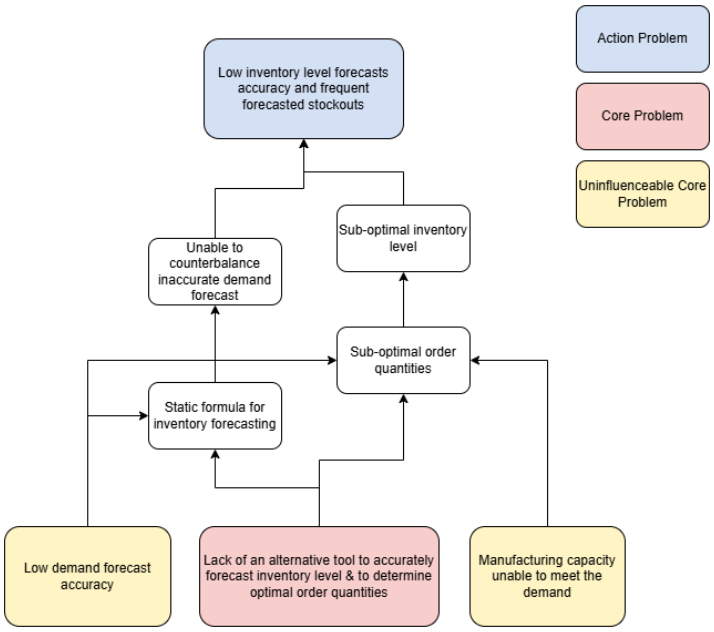

Figure 1: Problem Cluster

Nevertheless, improving demand forecast accuracy is outside the scope of the research, as set by the problem owner, mainly due to data unavailability.

2. Lack of an alternative tool to accurately forecast inventory level & to determine optimal order quantities: Versuni uses a static formula to forecast its inventory level, making it unable to counterbalance inaccurate demand forecast. Unlike a static formula, an alternative tool such as a machine learning model might be able to learn patterns that would result in more accurate inventory-level forecasts. On top of this, it seems that Versuni does not have a tool to optimally ensure all demands are met and to avoid stockouts. Both aspects are within the scope of this research; hence it is chosen as our core problem.
3. Manufacturing capacity unable to meet the demand: This can also be considered a core problem since manufacturing capacity critically affects the inventory level. Without the ability to meet the demand, the most accurate forecast accuracy and the most optimal inventory policy would still not satisfy the customer's needs because they will simply not receive anything. However, this is outside the scope of the research because the graduation assignment was conducted as part of the operation planning team at the headquarters of Versuni, not at one of the manufacturing facilities of Versuni.

From conducting the root cause analysis, it can be concluded that our core problem is as follows.

Lack of an alternative tool to accurately forecast inventory level & to determine optimal order quantity.

1.3.3 Norm and Reality

The norm represents the desired situation whereas the reality represents the current situation. The reality of the action problem is that the accuracy of inventory level forecasts is low and there is a frequent number of stockouts for many products. Firstly, to find the reality of inventory level forecast accuracy, data must be gathered and then analyzed. A dataset was constructed from multiple weeks of inventory-level forecasts files, consisting of these main variables.

- Beginning inventory level at week *t*.
- Incoming orders at week *t*.
- Forecasted demand at week *t*.
- Forecasted closing inventory level at week *t* (refer to the current method of forecasting).
- Actual closing inventory level at week t.

To evaluate the performance of the current forecasting method, we will use the chosen evaluation metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared (R^2). These metrics will be calculated by comparing the forecasted closing inventory levels at week *t* with the actual inventory levels at week *t*. As for the norm, it is hard to quantify since any improvements shown by the evaluation metrics would already satisfy the problem owner. Thus, the norm is set so that the accuracy of the evaluation metrics is improved by 25%.

Furthermore, to find the reality of the current ordering quantity, it is possible to calculate the stockout quantity of each product, which is chosen through a selection process discussed in Chapter 3.3. Therefore, the KPI for this problem is the sum of the stockout quantity of all products for the next 26 weeks. The sum of the stockout quantity of each product for the next 26 weeks should be as close to

zero as possible. However, in real life scenarios where demand is stochastic, it is unlikely to avoid stockouts, hence the norm is set as stockout quantity to be reduced by 75%. Table 1 depicts the reality of the action problem.

	MSE	MAE	R-Squared	Stockout Quantity
Reality	2.419.536	273	70%	11.091
Norm	0	0	100%	0

Table 1: Norm and Reality

1.4 Research Design

In researching to find solutions to solve the core problem, a research design is crucial in ensuring that the research is conducted properly. First, in Chapter 1.4.1, the problem-solving approach and research questions are explained. Then, in Chapter 1.4.2, the scope of the research is stated while in Chapter 1.4.3, the deliverables of the research are described.

1.4.1 Problem Solving Approach & Research Questions

Versuni is facing a problem of low inventory forecast accuracy and frequent stockouts. To solve the problem and bridge the gap between the norm and reality, the development of an alternative tool for inventory forecasting and determining optimal order quantities is needed. Hence, the main research question for this research is as follows.

"How can the inventory level forecast accuracy be improved and frequent stockouts be avoided by developing a machine learning model and formulating optimal order quantities?"

To answer the main research question systematically, a problem-solving approach is needed, consisting of sub-research questions. First, the current situation is analyzed, and then a literature review is conducted to gather relevant information about inventory forecasting and inventory management. Thereafter, the solution is formulated using the gathered information and data available for the research, developing a machine learning model and applying inventory management practices. Lastly, the outcome of the solution is then explained and compared with the current reality of the situation. Below are listed all sub-research questions responding to the sub-sections above.

1. **What is the current method of inventory level forecasting and the inventory policy at Versuni?**

The research question aims to analyze the current situation to find opportunities for improvement. This will be conducted by investigating the data source, the stakeholders, and the current formula used for inventory forecasts. Also, the inventory policy is assessed, including the current ordering process and inventory components (demand, lead time, associated costs) and its performance. Thus, this research question is accompanied by sub-research questions.

a. What is the source of data used for this research?
b. Who are the stakeholders involved in inventory forecasting, and how does Versuni generate its inventory level forecasts?

c. How are the products chosen for this research and what classification method should be used?
d. What are the components of the current inventory policy?
e. How effective is the current ordering policy?

2. What are the most relevant machine learning methods for inventory forecasting and what are the most relevant methods to determine optimal order quantity for this research?

This research question aims to investigate the most relevant methods to answer our main research questions, in the form of a literature review. The findings will be the basis of this research. Firstly, an introduction to machine learning and its application are discussed. Then, the types of machine learning algorithms and the ones most relevant to this research are explored. To assess the performance of the machine learning algorithms, a literature review on relevant evaluation metrics is also conducted. Furthermore, concerning the methods to determine optimal ordering quantity, a chapter will explain fundamental inventory policies. Lastly, the methods to determine optimal ordering quantities that are most relevant to this research are discussed. Thus, this research question is accompanied by these sub-research questions.

a. What is machine learning and what are its applications in the supply chain?
b. What are the types of machine learning algorithms and which ones are most relevant for this research?
c. What are the evaluation metrics to assess the performance of the models?
d. What are the fundamental inventory policies and what are the methods to determine optimal order quantity?

3. What is the procedure to develop a machine learning model and what is the chosen method to determine optimal order quantity?

This chapter of the research aims to generate solutions to address the core problem. First, the procedure for developing a machine learning model is explained. Then, the sources of the data and the data preparation process are discussed. The tools used to prepare and tune the machine learning model for optimal performance are then described. Additionally, the parameters for tuning each model and the rationale behind their selection are elaborated. Finally, we briefly mention the evaluation metrics used for assessing performance.

Next, we focus on determining the optimal order quantity. We explain the rationale behind the chosen method and discuss the input parameters and underlying assumptions. Following this, we present the mathematical model of the method. Lastly, we discuss the tool selected to translate the mathematical model into a functional model. Therefore, the sub-research questions below are to be answered.

a. What are the steps to develop a machine learning model?

b. How to obtain and prepare data to be used for the development of the machine learning model?
c. What are the tools used to tune a machine learning model to obtain the best performance?
d. What are the parameters to tune for each machine learning model?
e. How is the performance of the machine learning model evaluated?
f. What is the choice of method to determine optimal order quantity?
g. What are the input parameters for the method and how is it formulated mathematically?
h. What tool should be utilized to calculate the optimal ordering quantity using the chosen method?

4. **How are the machine learning model and method of determining optimal order quantity performing, in comparison to the current methods of inventory forecasting and ordering process?**

This chapter aims to evaluate the results of the machine learning models and the proposed method of determining optimal order quantity. First, the best parameters for each model are shown and the performance of each model is assessed using the evaluation metrics. Then, the best-performing model is chosen and compared to the current method of forecasting. On top of that, a coefficient analysis of the variables is conducted to know the variables with the most predictive power. Next, in determining the optimal order quantity, the proposed model is assessed by comparing the total costs associated with the quantity of the orders.

a. What are the best-performing parameters for each machine learning model?
b. How is the performance of each machine learning model according to the evaluation metrics?
c. How does the best-performing model compare to the current method of forecasting?
d. What are the most relevant variables for the machine learning model?
e. How does the proposed method of determining optimal order quantities compare to the current method?

5. **What are the conclusions and limitations of the research and what are the recommendations for further research?**

This chapter aims to provide answers to the main research questions through a conclusion of the findings of the research. Additionally, limitations and assumptions that hinder the research from delivering the most reliable outcomes are discussed. Lastly, a chapter provides recommendations for further research to increase inventory level forecast accuracy and to determine optimal order quantities.

1.4.2 Scope

This thesis is conducted as part of an internship opportunity, during which I conducted research while working as an inventory management intern with the S&OP team at Versuni. The research consists of two parts: improving inventory level forecast accuracy and determining optimal order quantity. The data used for both parts is the same, which is the inventory forecasts weekly file, which is explained further in Chapter 3.1. For the inventory forecasts, the research is limited to only using

the inventory forecasts weekly file from the first week of 2024 until the 20th week of 2024, which at the time of collecting data, is the latest available file. The file consists of all products covering various product types and across 7 different markets and located in multiple warehouses around the globe.

As for determining the optimal order quantity, first, a product selection must be done, since the company did not specify the products that should be analyzed. The product selection is done based on ABC-XYZ classification and through a selection process described in Chapter 3.3 The selection is performed only on products that are found in a specific warehouse for a specific market, which is the NL20 warehouse (a warehouse located in the Netherlands). In determining the optimal order quantity, data such as the actual demand of the products is unknown, hence the research is conducted by utilizing demand forecasts and assuming it to be deterministic. This assumption simplifies the problem and allows us to use methods that are relevant to the demand characteristics of the products.

1.4.3 Deliverables

This research aims to improve the inventory level forecast accuracy and determine the optimal order quantity for the selected products experiencing stockouts. To improve the accuracy of the forecasts, this research hypothesized that a machine learning model can increase the inventory level accuracy. Thus, machine learning's application in inventory management is assessed and a machine learning model will be developed. Additionally, to determine the optimal order quantity, we explore and implement the most relevant method using a tool such as Excel, utilizing its VBA programming language for coding. Therefore, to conclude, this research will deliver:

1. A machine learning model to forecast inventory levels by using past inventory forecast files as input.
2. A VBA code to implement a method to determine optimal order quantity.

2 Current Situation Analysis

This chapter aims to analyze the current situation at Versuni regarding their inventory level forecasting method and their inventory policy, to answer the research question "What is the current method of inventory level forecasting and the inventory policy at Versuni?" In Chapter 2.1, the source of data used in this research is explored, and then in Chapter 2.2, the inventory forecasting method is assessed. Lastly, in Chapter 2.3, the inventory policy is investigated, this chapter also discusses the production selection method and relevant inventory components such as demands, lead times, and costs.

2.1 Data Source

For this research, the file used is the weekly inventory forecasts file, which is a file compiled every week by the inventory management team to provide inventory forecasts for the next 26 weeks. The source of data used in this research is Versuni's SAP ERP system, which is an ERP system covering all aspects of business, from financial, human resources, and inventory.

The weekly inventory forecasts file is created by merging data from different interfaces of SAP, specifically SAP Integrated Business Planning (IBP), SAP Analytics Cloud (SAC), and SAC Excel add-ins. The data is then merged into an Excel file.

1. SAP IBP is a comprehensive planning solution that combines demand planning, supply planning, and inventory optimization. In practice, Versuni's inventory management team primarily utilizes the demand planning capabilities of IBP to gather demand forecasts for the next twenty-six weeks.
2. SAP SAC provides a cloud-based analytics solution that offers easy-to-understand visualizations, enabling organizations to make data-driven decisions. Although technically, data found in SAC should also be accessible in SAP IBP, the complexity of the data has made it challenging for the company to consolidate everything to one convenient place. Consequently, SAP SAC is used to gather data on incoming orders for the next twenty-six weeks.
3. SAC Excel add-in is an additional feature on Excel that allows users to access data from SAP SAC in an Excel format. The SAC Excel add-in provides the inventory management team with quantitative variables such as a product's legal price and inventory level, along with descriptive variables such as business unit, plant location, market, and product ID.

Combining all these three sources of data results in the weekly inventory forecasts file, used for this research. The weekly routine of the inventory management team involves manually downloading the data from these sources, merging them into Excel by performing Excel operations and then assessing any anomalies in the data. If everything seems to be correct, the file would then be forwarded to multiple stakeholders in the company.

Important to note, that the data sources show real-time value, meaning the values are dynamic, hence, the weekly inventory forecasts file is compiled from a snapshot of data taken at a particular time, which is every Tuesday morning. According to the inventory manager at Versuni, Tuesday morning is found to be the time with the most reliable data, because, after several experiments, data taken at this time led to the least number of errors according to the stakeholders. The root-cause

analysis for this is beyond the scope of this research, however, a prominent reason could be that on the weekend operational activities stop resulting in less consistent data on the first day of the week. By Tuesday, operations would have already been going for a day, resulting in more consistent data.

The file consists of many variables, but the most relevant ones are as follows.

1. Product ID.
2. Forecasted demand for the next 52 weeks.
3. Incoming orders for the next 26 weeks.
4. Closing inventory level from the previous week/ Opening inventory level at the given week.
5. Closing inventor level projection for the next 26 weeks.
6. Legal price.
7. Lead time.

First, to assess the accuracy of the inventory level forecasts, analyzing a single week's inventory forecasts file is insufficient, as it won't represent the true performance of the current inventory forecasting method. Therefore, in this research, multiple weekly inventory forecast files are merged and utilized, starting from the first week of 2024 until week 20 of 2024. Only the relevant variables are included and utilized in the creation of the machine learning model, which will be discussed in Chapter 5.1.2.

For the assessment of the inventory policy, a file from week 22 is used. This is sufficient since the aim is to evaluate the inventory policy that led to frequently forecasted stockouts. The week 22 file provides the relevant variables needed to conduct this part of the research

2.2 Inventory Forecasting

This chapter discusses the importance of inventory forecasts for the primary stakeholders and investigates the current methods used for generating the forecasts, which have been deemed to be inaccurate.

2.2.1 Stakeholder

The main purpose of the inventory level forecast is to provide an outlook on inventory levels, enabling informed decision-making. Two primary stakeholders utilize the inventory forecasts: the S&OP team and the finance team. Below, the importance of inventory level for each primary stakeholder is discussed.

1. The S&OP (Sales and Operations Planning) team is responsible for aligning and integrating various operations within Versuni to ensure a unified strategy that drives the company forward. They also supervise the performance of individual markets, coordinating with them to address any forecasted inventory gaps or anomalies. Versuni operates in seven different markets, each with its own S&OP team. These teams function similarly to the HQ S&OP team, with their sales, operations planning, and execution units. However, they are reliant on the inventory forecasts file (created by the HQ S&OP team) to assess their respective inventory performance. For example, if the HQ S&OP team identifies future gaps for some products, they notify the respective market to assess and address the issue. Actions taken often

include requesting increased production rates or offering discounts to reduce stock levels, to avoid overstock or stockouts.

2. The finance team uses the inventory forecasts file to monitor the trajectory of expected revenue and profitability, on top of assessing assets owned by the company. The main objective of these actions is to ensure optimal budget allocation is set accordingly, boosting the growth of the company. The finance team also coordinates with the S&OP team to avoid unnecessary risks for products forecasted to have significant amounts of inventory, since it would tie up cash that could be used for more productive activities.

2.2.2 Current Method

The inventory management team creates the inventory level forecasts by using a straightforward formula, as follows.

$$Stock\ On\ Hand_{t+1} = Max\{(Stock\ on\ Hand_t + Incoming\ Orders_t - Forecasted\ Demand_t),0\} \quad (1)$$

Where t : period (week)

Inventory level forecast refers to the stock on hand projected to be available at the end of a week. This straightforward formula aims to estimate the future inventory levels based on three key factors:

1. Stock on Hand: this is the existing inventory at the beginning of the period.
2. Incoming Orders: orders that are expected to be received at the beginning of the period and ready to fulfill demand.
3. Forecasted Demand: this is the predicted demand for the products during the period.

Equation (1) has a max function to ensure that the inventory level does not fall below zero, addressing the issue of negative inventory value.

The straightforward formula fails to accurately forecast the actual inventory levels, due to its reliance on accurate demand. If the actual demand differs from the forecasted demand, the entire inventory projection becomes inaccurate. Therefore, the inaccuracy in inventory forecasts is directly related to the accuracy of the demand forecast. Ideally, increasing the accuracy of demand forecasts would result in inventory forecasts that closely match reality. However, since this is beyond the scope of our research, the aim has been to develop a machine-learning model to capture patterns that are otherwise missed, leading to better inventory forecasting accuracy.

2.3 Product Selection

Our dataset consists of 699 products, too many for each product to be analyzed. Hence, to narrow down the scope of the research, only 9 products will be selected. The selection would be based on a classification method called the ABC-XYZ classification. This method is highly preferred and widely used for cost reduction and process improvement in inventory management because of its simplicity and accessibility (Silver, Pyke, & Thomas, 2017). The classification method is a combination of two classification methods which are ABC classification and XYZ classification. The main purpose is it allows inventory managers to tailor a target service level for each category, in our case fill rate, to enhance the overall inventory process and reduce cost.

ABC classification is a classification method that is based on Pareto's 80/20 rule, which states that 80% of the effects come from 20% of causes. In inventory management, this principle is used to categorize products based on their contribution to the total revenue (Yigit et al., 2019). Following the principle, Category A is filled with products that contribute to 80% of the total revenue; Category B consists of products that contribute around 15%; whilst Category C represents products that contribute only around 5% (Stojanović & Regodić, 2017). However, these percentages are arbitrary and should be tailored to the business's strategy.

As for XYZ classification, it is a method that separates products based on their demand variability or sales patterns. Category X represents a product that has relatively consistent demand/ low demand variability, whereas Category Z represents the opposite (Kumar, 2017). For this reason, forecasting demand for products in Category X would likely result in higher accuracy than forecasting Category Z. To determine the category of a product, a calculation of the variation coefficient needs to be performed. The variation coefficient is a measure of the variability of a dataset in relation to its mean. Usually, the proposed classifications are Category X represents products with a variation of coefficient between 0% and 10%; 10% to 25% for Category Y; and lastly Category Z with a variation coefficient between 25% and infinity (Stojanović & Regodić, 2017). Like the ABC classification, these percentages are arbitrary.

The combined ABC XYZ approach results in a paired comparison matrix to categorize products into their respective classes. The characteristic of each cell of the matrix is as follows (Stojanović & Regodić, 2017).

Part Characteristics in the Combined ABC-XYZ-Matrix [24]

	A	B	C
X	high value, high predictability continuous demand	medium value, high predictability continuous demand	low value, high predictability continuous demand
Y	high value, medium predictability fluctuating demand	medium value, medium predictability fluctuating demand	low value, medium predictability fluctuating demand
Z	high value, low predictability irregular demand	medium value, low predictability irregular demand	low value, low predictability irregular demand

Figure 2: ABC-XYZ Characteristics

By understanding these principles, the products can now be categorized into their respective classes. To do so, first, the total revenue of each product for the next 52 weeks is calculated by summing the weekly demand over the period and multiplying it with the legal price. Then, we sort the products based on their total revenue from high to low. On top of that, we calculated the sum revenue of all the products so that we can find how much each product contributes to the total revenue, shown as a percentage. Then, the cumulative percentage of each product's revenue is calculated. Cumulative percentage measures the proportion of a total that is accounted for by each product up to a given point. Using this statistical measure, we can know which products contribute to 80% of the total revenue. For the ABC classification, we used the percentages for each class that were stated by Stojanovic, 2017, which are as follows.

Class	A	B	C
% of Total Revenue	$\leq 80\%$	$80\% < x \leq 85\%$	$\geq 85\%$

Table 2: ABC Classification Criteria

The result of the ABC classification is as follows.

Class	A	B	C
Item Count	193	38	468
% of All Item	23%	5%	67%

Table 3: ABC Item Count

The Pareto's 80/20 rule can be seen occurring in our dataset, as shown by the fact that 80% of the total revenue comes from 23% of products.

Moving on to the XYZ classification, for every product, the standard deviation and mean of all the weekly forecasted demand are formulated, which are then used to calculate the coefficient variation, shown as a percentage. The formula for the variation coefficient is shown below, where σ is the standard deviation and $\bar{X}$ is the mean.

$$CV = \frac{\sigma}{\bar{X}} \quad (2)$$

From doing so, it was that the coefficient variation ranges from 15% up to as high as 714%, which means that we cannot categorize the products based on the suggested percentages for each class. Hence, we decided the percentage of each class based on the percentile of all the variation coefficients, which are as follows. Category X consists of items with variation coefficients in the 25th percentile, Category Y with variation coefficients in the 50th percentile, and Category Z is filled with items with variation coefficients in the 75th percentile.

Class	X	Y	Z
Variation Coefficient Range (%)	≤39%	$39\% < x \leq 151\%$	>151%

Table 4: XYZ Classification Criteria

The result of the XYZ classification is as follows.

Class	X	Y	Z
Item Count	177	346	176
% of All Item	25%	50%	25%

Table 5: XYZ Classification Item Count

Combining these two classifications results in a matrix as follows.

Class	A	B	C
X	34	9	134
Y	122	21	203
Z	37	8	131

Table 6: ABC-XYZ Classification Item Count

After classifying all products into their respective ABC categories, the products are sorted based on the coefficient of variation (CV) in their demand, in ascending order. The coefficient of variation is a key metric as it indicates the stability of demand; a lower CV suggests that the demand is more consistent. Then, we selected one representative product from each class for detailed analysis. For each selected product, we calculated the total number of forecasted stockouts by comparing the forecasted demand with the quantity of incoming

goods. We then identified the first product whose stockout quantities are large in comparison to their demand. This approach ensures that we focus on products with the greatest number of stockouts, allowing us to address potential issues in inventory management effectively.

2.4 Current Ordering Process

According the inventory management team in Versuni, the ordering process is done automatically by the company's ERP system, SAP Integrated Business Planning (IBP). The system is a cloud-based system for supply chain planning and optimization, covering demand planning and supply planning. SAP IBP consists of multiple complex algorithms to match supply and demand, it does so by forecasting the demand, analyzing the current state of inventory, considering potential supply constraints, and calculating the right optimal order quantity through optimization algorithms. Also, the system provides the opportunity for planners to create and evaluate certain scenarios to analyze different decisions on the supply chain, preparing for uncertainties. Despite being sophisticated, the outcome of the system still gets supervised by supply planners, ensuring any anomalies are addressed.

After determining the ordering quantity, the order is forwarded to the respective manufacturers for production. These manufacturers are connected to the ERP system, ensuring seamless integration and real-time communication. This connectivity allows for efficient communication and transparency throughout the supply chain process. Once the manufacturers receive the order, they initiate the production process, following the specified quantity. The ERP system helps track progress and manage resources, ensuring that production schedules are met. The finished products are then transported to warehouses located across the globe. The locations of these warehouses are strategically planned to optimize storage and delivery times, ensuring that products are ready for sale to meet market demand. The ERP system is still involved in this process, to monitor the inventory level in these warehouses, facilitating a feedback loop to the whole ordering process to ensure the optimal order quantity is determined. This description provides a high-level perspective of the process. The intricate details and coordination of the process are too complex for the scope of the research. Nevertheless, the extensivity of the process is inherent to running a large multinational company.

As for this research, it seems that the outcome of the ERP system fails to determine the optimal order quantities for some products, leading to frequent stockouts. Since the exact ordering policy is unknown to the problem owner, the research makes use of available information, which is the forecast files to analyze the current situation. In the file, there are important variables to forecast inventory as mentioned earlier, including the ordering quantity for the next 26 weeks, formulated by the ERP system.

Therefore, having selected nine products, each from categories of the ABC-XYZ classification, the ordering quantity of each product can thus be analyzed. One of the aims of this research is to determine optimal order quantity to avoid stockouts, hence the first analysis that should be conducted is to know the stockout frequencies of these selected products. As explained in Chapter 3.1, the data source for this research does not have any information about the quantity of stockouts, thus it needs to be calculated using the available data which are the inventory level, incoming orders/ordering quantity, and forecasted demand. The stockout quantity shown in the table below is

the sum of stockouts of each product for the next 26 weeks. The formula for calculating stockouts is straightforward.

$$Stockout = Max\ (Forecasted\ Demand - Stock\ on\ Hand - Incoming\ Orders; 0) \quad (3)$$

As a result, the table below depicts the sum of the stockout quantity of each product for the next 26 weeks.

Product	Stockout Quantity (26 Weeks)
Product AX	100
Product AY	474
Product AZ	2794
Product BX	713
Product BY	214
Product BZ	6133
Product CX	226
Product CY	352
Product CZ	85

Table 7: Sum Stockout Quantity for 26 Weeks

Furthermore, the total inventory cost is analyzed to objectively assess the performance of the current inventory. From investigating the dataset, it seems that the order quantity that has been placed does not quite match the forecasted demand, leading to a lot of forecasted stockouts. Hence, having the order quantity values, we can calculate the total inventory cost, which later will be used to compare with our new proposed ordering quantities. The total inventory cost is the summation of holding cost, setup cost, ordering cost, and stockout cost. Holding cost is the cost of holding one unit of inventory for a specific period. Setup cost is a cost incurred every time Versuni places an order. Furthermore, the ordering cost is the cost of ordering one unit, which will be incurred on top of the setup cost. Lastly, the stockout cost is simply the cost of having stockouts. As a result, the table below depicts the total inventory costs.

Product	Total Inventory Cost (EUR)	Stockout Cost (EUR)
Product AX	1.263.922	40.150
Product AY	871.384	187.641
Product AZ	4.571.854	1.263.503
Product BX	432.736	53.518
Product BY	549.295	123.645
Product BZ	323.891	353.261
Product CX	156.521	11.594
Product CY	410.719	56.285
Product CZ	2.795	9.736

Table 8: Current Total Inventory Cost and Stockout Cost

2.5 Conclusion

This chapter has provided answers to the following question: “What is the current method of inventory level forecasting and the inventory policy at Versuni?” By breaking down the main question into sub-research questions, we have managed to get a good understanding of the current situation.

What is the source of data used for this research? The weekly inventory forecast file is used. This file, compiled every week by the inventory management team, provides inventory forecasts for the next 26 weeks. The data source for this research is Versuni's SAP ERP system, which covers all aspects of the business, including finance, human resources, and inventory.

Who are the stakeholders involved in inventory forecasting? The two main stakeholders are the S&OP team and the financial team who use the inventory forecast files to base their decisions; either to provide unified strategies to handle gaps in inventory or to formulate optimal budget allocation.

How does Versuni generate its inventory level forecasts? The current method of inventory forecasting uses Equation (1) which is a static formula, inhibiting it from counterbalancing inaccurate demand forecasts, which is a crucial aspect of inventory forecasting.

How are the products chosen for this research and what classification method should be used? 9 products were chosen from 699 using the ABC-XYZ classification method, with commonly used thresholds for ABC classification and percentile-based thresholds derived from the variation coefficient for XYZ classification.

What are the components of the current inventory policy and how effective is it? The inventory management team heavily relies on SAP Integrated Business Planning (IBP) for automated ordering and supply chain optimization. Despite its advanced algorithms, the system's effectiveness in determining the optimal order quantity has been deemed sub-optimal, particularly after assessing the stockout quantities of products over 26 weeks.

3 Literature Review

This chapter aims to answer the following research question: “What are the most relevant machine learning methods for inventory forecasting and what are the most relevant methods to determine optimal order quantity for this research?” In Chapter 3.1, general information about machine learning, and its application in inventory management are explained. Additionally, the various machine learning algorithms are explained. Then Chapter 3.2 aims to explain evaluation metrics that are commonly used to assess machine learning model performance. Next, regarding determining optimal order quantity, in Chapter 3.3, fundamental inventory policies are discussed, alongside methods to determine the optimal order quantity and each one’s criteria.

3.1 Machine Learning for Inventory Forecasting

Machine learning is a subset of artificial intelligence that focuses on developing a system that learns from data to produce insightful information, which can be used to help in decision-making (Angra et al., 2017). This is accomplished by using a set of statistical techniques that can identify behaviors of a dataset known as algorithms. Machine learning excels in processing large and unstructured datasets making them perform better in solving supply chain problems than traditional statistical methods. According to Ni et al (2019), machine learning does not rely heavily on historical data, unlike traditional methods such as exponential smoothing, moving averages, or time series. This makes machine learning a great choice for demand forecasting and planning.

Inventory forecasting and demand forecasting are closely intertwined concepts within supply chain management. Forecasting methods used for demand forecasting are also applicable to inventory forecasting. Both rely on data such as historical sales data and market trends. Intuitively, accurate inventory projection is highly dependent on accurate demand forecast. The hypothesis for this research is by training a model on data such as historical sales and historical inventory level, the model can see a relationship between the two. Hence, in this chapter, we will assess the literature not only for inventory forecasting but also for demand forecasting.

A machine learning algorithm is fed with most of the available data, or what is acknowledged as “training data.” The rest of the data unseen by the machine learning algorithm is known as “test data.” As the name implies, the “test data” is used to test the performance of the algorithms that have been trained on the “training data.” Based on the evaluation of the trained algorithms, a reiterative improvement cycle happens where the algorithms are tweaked to achieve the best performance. The data consists of independent and dependent variables.

In the field of supply chain, some variables are often found to be used to develop a machine learning model. Demand values at uniformly spaced intervals and macroeconomic indicators, such as GDP and inflation rate, were chosen as variables because they have correlations to the sales data (Huber, 2020). Similarly, according to Faizabad (2020), macroeconomic indicators should be used because their relationship with customers' shopping behavior is strong. To conclude, the performance of a model is very dependent on the quality of its input, hence it is important to use the most relevant ones.

Machine learning algorithms can be categorized into four most prominent types: supervised, unsupervised, semi-supervised, and reinforcement learning (Rebala et al., 2019).

Supervised learning algorithms identify patterns in data by considering the relationship between independent and dependent variables and are often used for value prediction (Mahesh, 2018). Unsupervised learning algorithms do not have an output variable and are used to identify data behaviors rather than make predictions (Rebala et al., 2019). Semi-supervised learning algorithms use a combination of a small amount of labeled data and a large amount of unlabeled data, refining predictions through iterative training (Rebala et al., 2019). Reinforcement learning is a type of machine learning where the output of the system involves a sequence of actions rather than a single action (Alpaydin, 2010). The focus is on developing a policy, which is a sequence of actions aimed at achieving a goal. For predicting inventory levels, supervised learning is the most suitable approach.

In supervised learning, there are two main types which are classification and regression (Rebala et al., 2019). In classification, the main objective is to classify data based on input variables. So, classification is often used on datasets that have discrete output variables. Whereas, in regression, the aim is to predict numerical continuous values based in line with this research's objective. Hence, regression will be used to predict the inventory level.

For instance, a regression model was developed to forecast the demand for crops to assist farmers in production planning (Prabhu et al, 2020). The model achieved a 98% R-squared error, implying that it performs well. An SVM model to forecast agricultural water demand was developed by (Li Xuemei et al,2020). Their model has good generalization performance and accuracy. KNN and Random Forrest machine learning models were developed to forecast demand in apparel retail stores, utilizing a plethora of variables (Guven et al, 2021). A Neural Network model was developed to forecast demand in the steel manufacturing industry by considering variables such as demands and macroeconomic factors, which achieved an overall accuracy of 89.4% (Feizabadi, 2020). Overall, each model thrives given the right dataset, hence the best way to find the optimal model for our research is to perform experiments with a set of ML models.

3.1.1 Regression Algorithms

As discussed above, the most suitable machine learning algorithm for this research is a regression model. This research requires an algorithm to predict a numerical continuous value by analyzing patterns of past historical data. In this chapter, we will elaborate on the types of regression models and how they differ from one another.

Regression algorithms can be categorized into two, which are linear regression and non-linear regression. A linear regression algorithm assumes a linear relationship between the independent and dependent variables (Rebala et al., 2019). It is a common choice of algorithm due to its simplicity and ease of interpretability (Muller et al., 2016). On the other hand, a non-linear regression algorithm learns from a dataset that does not have a clear linear relationship between its variables (Muller et al., 2016). Despite being able to learn from a more abstract dataset, a non-linear regression algorithm has its downsides. The complexity of the algorithm causes it to require more computing power to run. Non-linear regression is also prone to overfitting and hard to interpret since the algorithm is trained on a more complex dataset than a dataset with linear relationships.

3.1.1.1 Linear Regression

Linear regression is a simple yet powerful algorithm that helps you to understand and predict the relationship between a dependent variable and an independent variable (simple linear regression) or

multiple independent variables (multiple linear regression). It predicts a numerical continuous value by formulating the best-fitting line between all the variables within a dataset. Linear models perform well with either a dataset with a large number of features compared to the number of samples, or a very large dataset (Müller et al., 2016).

The mathematical equations for a simple and multiple linear regression are shown below.

$$\hat{y} = \beta_0 + \beta_1 x + \epsilon \quad (4)$$

$$\hat{y} = \beta_0 + \beta_1 x + \cdots + \beta_n x_n + \epsilon \quad (5)$$

Equation 1: Multiple Linear Regression

The breakdowns of these two equations are as follows.

- $\hat{y}$ is the dependent variable (target).
- x_n are the independent variables.
- β_0 is the intercept of the regression line/plane with the y-axis.
- β_1 is the slope of the regression line, representing the change in $\hat{y}$ for a unit change in x
- $\beta_1, \beta_2, \dots, \beta_n$ are the slopes of the regression plane in case of multiple linear regression, representing the change in $\hat{y}$ for a unit change in x
- ϵ is the error term, showing the difference between the observed and predicted values.

For a linear regression, the most common method to estimate $\beta_1, \beta_2, \dots, \beta_n$ is the method of Ordinary Least Squares (OLS). This method works by minimizing the differences in values between the output of the algorithm and the actual observed values. This method is simple and easy to interpret, however, it often creates models that are overfitting. Hence, to prevent this, methods such as Ridge regression and Lasso regression are used. These methods are extensions of OLS, and they work by adding penalty terms to the OLS cost function to reduce to a certain extent the influence of some variables. This leads to a model that is less likely to be overfitting and can handle multicollinearity between the variables. Also, it improves the generalization of the model, enhancing its performance on unseen data.

3.1.2 Non-Linear Regression

Non-linear regression is a regression algorithm that works well in understanding the relationship between variables that are assumed to be non-linear. It can grasp the complexity of the dataset and offers flexibility in creating relationships between the variables, which includes logarithmic and exponential shapes (Muller et al., 2016). Here, we will only introduce methods that are commonly used and widely available for creating a machine-learning model.

3.1.2.1 Support Vector Machines

Support Vector Machines (SVM) is a powerful supervised learning algorithm that can be used for both classification and regression tasks. It works by formulating the most optimal hyperplane that separates different classes in the feature space (Geron, 2017). A hyperplane is a decision boundary chosen to minimize the difference between observed and predicted values while maximizing the distance of each data point to the hyperplane, known as the margin. Doing so ensures that the model

is accurate whilst also being generalized well enough to perform well on unseen data. This algorithm thrives in capturing the complexity of high-dimensional data (Muller et al., 2016).

3.1.2.2 K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a machine learning algorithm used for classification and regression. It is commonly acknowledged as a lazy learning algorithm because it does not make assumptions about the data during training (Rebala et al., 2019). One of the main advantages of using KNN is it is easy to understand and does not need a lot of adjustment to produce a reasonable model, making it a good starting point before considering more advanced methods. (Muller et al., 2016). However, this method is not often used in practice due to its inability to handle many features and prediction being slow (Muller et al., 2016).

In classification, KNN works by storing all data points and their classification. Then when new unseen data is shown, KNN identifies the target variable that is k-nearest to the unseen data in the feature space. K refers to the number of neighbors that are closest to the new data point. It is slightly different for regression since the expected output is a continuous value, but the principal is the same. In regression, the output of the k-nearest neighbor is aggregated, usually, the average of the values is taken as the output (Rebala et al., 2019). Unlike the previous models, KNN does not have any hyperparameters that generalize the model. Hence, to find the best-performing KNN model, we must experiment with different values of K.

3.1.2.3 Neural Networks

Neural networks are a class of machine learning models that are designed to emulate the way our brain works. They are powerful for a variety of purposes, from classification and regression to image recognition and natural language processing (Muller & Guido, 2017). The structure of neural networks is inspired by the human brain hence it functions like one. It consists of neurons (nodes), layers, weights, biases, and activation functions.

The neurons are the fundamental building blocks of neural networks. They take input from other neurons or external sources. Each input is associated with a weight, reflecting its relative influence compared to other neurons/inputs. An activation function is then applied to each neuron to generate an output. An activation function is a mathematical function that introduces non-linearity into the neural network model, allowing the model to understand complex patterns and relationships in the data. Neurons are organized into layers and layers operate the same way as individual neurons, but at a different level of abstraction. Layers organize neurons to collectively process data through multiple stages, enabling the network of layers to produce predictions.

3.1.2.4 Decision Trees

Decision Trees is a common machine learning algorithm that is inspired by the human decision-making process. It works by breaking down a complex decision into a series of simple decisions. It is often utilized for classification and regression tasks. Decision trees are popular due to their ease of interpretability (Muller et al., 2016). The downside of using this method is that it is prone to overfitting and offers poor generalization performance (Muller et al., 2016)

The structure of decision trees consists of a root node, internal nodes, and terminal nodes. A root node represents the whole dataset and is the starting point of the decision-making process. Once the data is passed through the root node, it will be divided into several internal nodes. Each internal

node represents a specific feature and acts as a decision point. The dataset is continuously split into smaller subsets and internal nodes are created for every decision point. After going through several internal nodes, an outcome is reached, and these outcomes are represented by leaf nodes. All the connections between each type of node are connected by branches.

3.1.2.5 Random Forest

Random forest is referred to as an ensemble machine learning method, meaning that it combines multiple decision trees to produce a more powerful model. Random forest is among the most widely used machine learning methods, known for being powerful, working well without heavy tuning of parameters, and not needing scaling of the data (Muller et al., 2016). The general idea behind random forest is to create a more generalized model by averaging the results of many decision trees that are overfitting the dataset in various random ways (Malik et al., 2016). The randomness results in lower correlations between decision trees, improving the overall performance.

3.1.2.6 Gradient Boosted Regression Trees

Gradient-boosted Boosted Regression Trees is another ensemble machine learning algorithm. This model is known for its incredible performance, used across various industries and by winning teams in ML competitions (Muller et al., 2016). Essentially, Gradient Regression Trees work by creating a series of trees, where each tree tries to correct the errors from the previous one. Each tree provides predictions for a small subset of a dataset, so combining more and more trees iteratively improves performance. Two main parameters for gradient-boosted regression trees are several estimators and the learning rate (Friedman, 2001). These two parameters are interconnected, as a higher number of estimators means that the model is likely to overfit, hence it is usually paired with a lower learning rate to increase generalization, and vice versa.

3.2 Evaluation Metrics

This section aims to introduce different evaluation metrics used to assess the performance of a machine learning model. Evaluation metrics provide insights into the performance of the model, and it helps in comparing the models. They are used to formulate the most well-performing model for your needs through reiterative improvement cycles of evaluating and altering the hyperparameters. Each evaluation metric has distinctive characteristics and there is no one size fits all. Choosing the right evaluation metrics is crucial because each dataset behaves differently.

3.2.1 R(MSE)

Mean Squared Error is commonly used to evaluate machine learning models. This method calculates the average squared differences between predicted and actual values. Since each difference is squared, this method penalizes large errors which emphasize their influence on the model. Hence, this method is sensitive to outliers. A smaller result means a better-performing model. The formula for MSE is as follows.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad (6)$$

As explained, the differences that are calculated by the MSE are squared. This leads to a result that is hard to interpret since it is not in the same unit as the actual values. Therefore, there is an extension of MSE that is known as Root Mean Squared Errors (RMSE). This evaluation method takes the root of

the MSE, providing an error metric in the same unit as the actual values which enhances its interpretability. However, the nature of RMSE is identical to that of MSE regarding its sensitivity to outliers. A lower R(MSE) value represents better model performance.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (7)$$

3.2.2 Mean Absolute Errors

Mean absolute error is a metric that is robust to outliers, unlike the MSE. Mean absolute errors calculate the average absolute difference between the predicted and actual values. Therefore, it is straightforward to interpret. Mean Absolute Errors provide an error metric in the same unit as the actual values. A lower MAE value represents better model performance.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad (8)$$

3.2.3 R-Squared

R-squared, also known as the coefficient of determination, is a statistical measure that indicates the proportion of the variance in the dependent variable that is predictable from the independent variable(s). It is used to assess the goodness of fit of a regression model. An R-squared value ranges from 0 to 1, where 0 indicates that the model does not explain any of the variance in the dependent variable, and 1 indicates that the model explains all the variance.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \quad (9)$$

3.3 Inventory Policies

Reorder policies are crucial in managing inventory effectively, ensuring that stock levels meet customer demand without incurring unnecessary costs. According to Chopra and Meindl (2016), two primary types of reorder policies illustrate the key managerial issues concerning inventory management: continuous review and periodic review. Chopra and Meindl (2016) emphasize that the primary objective of these policies is to support decision-making regarding the quantity to order and the timing of orders. Achieving this objective depends on accurate calculations of the reorder point, safety stock, and order quantity.

1. Continuous Review

In a continuous review system, inventory levels are tracked continuously. An order for a predetermined lot size Q is placed when the inventory level falls to a specific reorder point. This method allows for real-time adjustments to inventory levels, which can be particularly beneficial for businesses with stable and predictable demand.

2. Periodic Review

In contrast, a periodic review system monitors inventory at regular intervals. Orders are placed to raise the inventory level to a set upper limit. This approach is simpler to implement but may not be as responsive to sudden changes in demand as the continuous review system.

The main advantage of utilizing the reorder policies is that they are straightforward and easy to implement, whilst still being able to maintain optimal inventory levels. Using a formula such as the Economic Order Quantity (EOQ) is effective in determining optimal order quantity, and finding the sweet spot between the holding cost, and ordering cost (Schwarz, 2008). However, the downside is that all these benefits can only be actualized given that demand, holding cost, and ordering cost are constant (Schwarz, 2008). A slight fluctuation in demand and lead time could lead to excess stocks or stockouts, which is often the case in real scenarios, especially for a company operating in dynamic markets.

3.3.1 Economic Ordering Quantity

The Economic Order Quantity (EOQ) is a formula used in inventory management to determine the optimal order quantity that minimizes the total cost of inventory. The total cost includes the cost of ordering and the cost of holding inventory. The EOQ model aims to find the optimal point where these costs are balanced, thereby reducing overall expenses. The formula of EOQ is as follows.

$$EOQ = \sqrt{\frac{2*D*S}{H}} \qquad (10)$$

Where:

D : demand.

S : ordering cost.

H : holding cost.

The primary advantage of EOQ is cost minimization; the method determines the ideal order quantity, reducing the total costs associated with both ordering and holding inventory. Therefore, this leads to a more effectively managed inventory level, preventing stockouts and overstocking. However, the effectiveness of the EOQ model relies on several key assumptions.

1. Demand is constant: this may not reflect real-world situations where demand can be very dynamic and unpredictable.
2. Lead time is constant: when a company uses EOQ to determine order quantity, it is at the mercy of its supplier to deliver the orders punctually.
3. Costs are constant.

These assumptions may not be practical in all situations, especially in industries with highly variable demand or volatile supply chains. Nevertheless, the EOQ model remains a proven method for optimizing inventory by finding the optimal quantity.

3.3.2 Dynamic Lot Size Model

The dynamic lot size model, also known as the Wagner-Whitin model, is an inventory management method used to determine the optimal ordering quantity over a finite planning horizon where demand fluctuates (Wagner and Whitin, 2004). Unlike the fixed lot size model, commonly known as the Economic Order Quantity (EOQ), the dynamic lot size model adjusts the order quantity based on dynamic demand, aiming to minimize total cost, which includes holding cost, setup cost, and ordering cost (Wagner and Whitin, 2004).

The dynamic lot size model, also known as the Wagner-Whitin model, is an inventory management method used to determine the optimal order quantity over a finite planning horizon where (Wagner and Whitin, 2004). Unlike the fixed lot size model, also known as the Economic Order Quantity (EOQ) which assumes constant demand, the dynamic lot size model adjusts order quantity based on dynamic demand. The aim of the model is similar to the EOQ which is to minimize total cost, which includes holding cost, setup cost, and ordering cost (Wagner and Whitin, 2004).

There are several assumptions to be met when using the dynamic lot size model which is:

1. Having known demand.
2. No stockouts.
3. Instantaneous replenishment.

While the model does not completely represent real-life scenarios, it is still a more reliable approach than using the EOQ, especially when demand is not constant. Optimizing order quantities and when to put the order, based on actual demand fluctuations, offers a significant improvement in cost efficiency and operational effectiveness compared to the static EOQ model. The formula is as follows.

$$f_t(I) = \min\left[(i_t + x_t - d_t).H + x_t C + S + f_{t+1}(i_t + x_t - d_t)\right] \quad (11)$$

t : current period.

i_t : inventory level at the beginning of the period t.

x_t : order quantity in period t (decision).

d_t : demand quantity in period t.

H : holding cost.

C : ordering cost.

S : setup cost.

$f_t(I)$: minimum cost from period t onwards, given an initial inventory I

The dynamic lot size model is inherently related to dynamic programming because it applies dynamic programming principles to determine optimal order quantities over a finite planning horizon. The model breaks down the problem into smaller sub-problems, each representing a specific period. It calculates the most cost-effective solution for each period and combines these solutions to find the overall optimal strategy. Similarly, dynamic programming solves complex problems by decomposing them into sub-problems and making sequential decisions to find optimal overall solutions.

There are two approaches to the dynamic lot size model: deterministic and stochastic. Understanding the differences is crucial in employing the right solution given a particular situation.

1. Deterministic

The deterministic dynamic lot size model assumes that all relevant inputs (such as demand, lead times, and costs) are known with certainty for each period in the planning horizon. This means that

when using demand forecasts, the model assumes they are 100% accurate. The advantage is the model is not as complex as a stochastic model, allowing for easier understanding and faster computing time. However, assuming the demand forecasts to be 100% accurate can be misleading and it rarely happens in real-life scenarios, leading to less reliable outcomes.

2. Stochastic

On the contrary, the stochastic dynamic lot size model incorporates uncertainty and variables in the input parameters to determine the optimal order quantities. Thus, producing outcomes that are more relevant to real-life scenarios. The model uses probabilistic methods to consider the stochastic nature of the inputs, often using distribution parameters of the inputs (Heijs, 2019). The disadvantage of this method is that it requires a more complex model to compute, hence leading to longer running time. Furthermore, the distribution parameters of the inputs should be known.

3.4 Conclusion

This chapter presents the answer to the following question: "What are the most relevant machine learning methods for inventory forecasting and what are the most relevant methods to determine optimal order quantity for this research?" This was answered through several sub-research questions as summarized below.

What is machine learning and what are its applications in the supply chain? Machine learning is commonly used in supply chain management for its ability to produce data-driven insights from large and complex datasets, enhancing the decision-making process. Unlike traditional methods, machine learning algorithms offer flexibility and accuracy by analyzing relationships between various variables such as demand patterns or macroeconomic indicators.

What are the types of machine learning algorithms and which ones are most relevant for this research? The regression model aims to predict numerical continuous values, relevant to the aim of this research. There are several types of regression models: linear and non-linear. A commonly used linear model is the Ordinary Least Squared Linear Regression Model. As for non-linear models, there are various types which are: Support Vector Regression, KNN, Neural Network, Decision Tree, Random Forrest, and Gradient Boosted Regression Trees.

What are the evaluation metrics to assess the performance of the models? Mean Squared Error, Mean Absolute Error, and R-squared are the evaluation metrics discussed, each assessing the performance of the model differently.

What are the fundamental inventory policies and what are the methods to determine optimal order quantity? The Economic Order Quantity (EOQ) minimizes total inventory costs assuming constant demand, lead times, and costs. For variable demand situations, the dynamic lot size model (Wagner-Whitin) adjusts order quantities dynamically over a planning horizon, leveraging dynamic programming principles.

4 Solution Design

This chapter aims to answer the following question: “What is the procedure to develop a machine learning model and what is the chosen method to determine optimal order quantity?” In Chapter 4.1, the procedure to develop a machine learning model is discussed, including defining the problem, collecting and preparing the data, and lastly developing and evaluating the model. Chapter 4.1.3 and Chapter 4.1.4 provide the tools used to prepare and develop the model, which includes performing hyperparameter tuning. Then, in Chapter 4.1.5, there is a brief explanation of how the model is evaluated. Furthermore, in determining the optimal order quantity, Chapter 4.2.1 provides the reasoning behind the choice of method. In Chapter 4.2.2, the input parameters of the chosen method are elaborated. Additionally, the mathematical formulation and choice of programming language to implement the methods are discussed in Chapter 4.2.3 and Chapter 4.2.4, respectively.

4.1 Machine Learning Model

Developing a machine learning model involves several steps, from defining the problem to deploying the model. In this chapter, we will explain each step and what we have done to come to our results, which will be explained in the next chapter.

4.1.1 Problem Definition

Versuni's weekly inventory forecasts are inaccurate due to Equation (1) which assumes 100% accurate demand forecasts, leading to errors when actual demand differs. While improving demand forecasting is ideal but beyond the research's scope, we hypothesize that a machine learning model can enhance inventory projections with the current accuracy of the demand forecasts. By incorporating various variables, including the forecasted demand, the machine learning model can learn the relationship between forecasted and actual inventory levels, providing more accurate projections.

4.1.2 Data Understanding

Again, we will make use of the weekly inventory forecast files, which consist of multiple variables as explained in the previous section. For the model, however, we will use only a few of these variables since the other variables are not relevant. These variables are the beginning inventory level, the incoming orders, and the forecasted demand for next week only, accompanied by descriptive variables such as the year, week number, business group, business unit, plant location, market, and ID. We consolidated multiple weekly inventory projection files starting from week 1 until week 20 of 2024. From each weekly inventory projection file, we only make use of the forecasted demand and incoming orders for next week.

Also, we added two columns to show the closing inventory level, a column showing the values from the current static formula approach and the other depicting the actual closing inventory level, which we extracted from next week’s inventory projection file. Table 9 is an illustration of how our dataset looks like, showing only the numerical variables. The column “Projected Inventory Level” is showing values from applying Equation (1). Important to note that this variable would not be used as an independent variable for the machine learning model. Instead, the main purpose of it is to compare

the performance of the current approach by comparing the values to the actual inventory level. The second row of Table 9 tries to illustrate how the values for each column might look like.

Column Name	Year	Week	Inventory Level	Incoming Orders	Forecasted Demand	Projected Inventory Level (current approach)	Actual Inventory Level
Example Values	2024	1	10	5	7	$\max\{(10+5-7);0\}=5$	10

Table 9: Variables in the Dataset

4.1.3 Data Preparation

Before feeding our dataset into the machine learning model, we must preprocess it to ensure it produces the best output (Muller et al., 2016). The steps involved in this process are as follows:

1. Handling Missing Values:

Removal of Rows with Missing Values: We removed rows with missing values since most machine learning models require a complete dataset to function properly (Geron, 2019). Although imputing missing values with the median or mean could have been an alternative, we chose not to as it might distort the true nature of the data, leading to misleading results (Muller et al., 2016). Given our dataset's large size relative to the number of rows with missing values, their removal did not significantly affect model performance.

2. Feature Selection:

Relevance of Variables: We performed feature selection to ensure our model trains on the most relevant variables. Through experiments and coefficient analysis in a linear regression model, we determined that the variables ‘business unit’ and ‘business group’ should be excluded. These variables had the lowest absolute coefficients, indicating minimal impact and adding noise, which degraded model performance.

3. Encoding Categorical Variables:

One-Hot Encoding: Regression models cannot process non-numeric data directly, so we converted categorical data using one-hot encoding. We used the `get_dummies` function from the Pandas library, transforming distinct categorical values into columns. If a unit has a specific value, it is marked as 1 in the corresponding column and 0 in others.

4. Splitting the Dataset:

Training and Testing Sets: We split the dataset into training and testing sets using the `train_test_split` function from the sci-kit-learn library, with an 80:20 ratio. This ratio allows the model to train on a substantial portion of the dataset while retaining enough data to test performance, following widely accepted practices.

5. Standardizing Independent Variables:

Standardization: We standardized our independent variables using the `StandardScaler` function from the sci-kit-learn library, ensuring each feature has a mean of 0 and a standard deviation of 1

(Geron, 2019). Standardizing prevents larger features from unequally influencing the model (Muller et al., 2016). Experiments showed that non-standardized datasets resulted in higher errors compared to standardized datasets.

By following these preprocessing steps, we aim to optimize the performance and accuracy of our machine-learning model.

4.1.4 Model Development

In the model development phase, we fed our models with our dataset and made tweaks to them to achieve the best results. This is known as hyperparameter tuning. Each model has different sets of parameters to adjust, giving us the possibility of tuning it according to our objective. Besides achieving the best results, hyperparameter tuning is also beneficial in preventing our model from overfitting or underfitting. In this chapter, we will explain how we adjust the parameters for each model.

We utilized a tool called GridSearchCV, a tool that exhaustively searches through a specified parameter grid, allowing us to test different sets of parameters in one run (Scikit-learn developers, n.d.). For each combination of parameters, it trains the model and evaluates its performance using cross-validation. The evaluation metric that we chose to evaluate every iteration is the negative mean squared scoring method which is common for regression tasks where the objective is to minimize the mean squared error. For most of our models, we specified the parameter CV for GridSearchCV, which is set to 5 k-folds. But, if the cross-validation runs for too long, we decrease it to 3 k-folds. K-folds refer to the number of times the model is trained, each time using k-1 folds for training and the rest of the fold for validation. Below, we will explain the different sets of parameters that we tested on each model.

4.1.4.1 Linear Regression

Linear regression is a relatively straightforward model in comparison to the others hence, no hyperparameter tuning was needed. However, there are many different types of linear regression models, such as ridge regression and lasso regression, for which we can adjust the parameters. We did not use these models.

4.1.4.2 SVR

To develop a Support Vector Regression (SVR) model, there are a few parameters to tune. The main parameters to tune are its kernel type, regularization parameters, and epsilon. However, we experimented with tuning these hyperparameters and the tuning never ended, since the model kept running for a long time. The reason could be the complex nature of SVR and the lack of available computing power. Hence, we use the default parameters for SVR.

4.1.4.3 KNN

For our KNN model, we experimented with different values for two parameters, which are as follows:

- *n_neighbors*: determines the number of neighbors to use by the KNN algorithm, by default, it is set to 5.
    - Our experiment: *n_neighbors from 1 to 15, in increments of 2*.
- *Weights*: determines how much weight is given to the neighbors.

- *Our experiment*: *uniform* or *distance*. Uniform means that all the neighbors are given equal weight, whereas distance means that the closest neighbors are given more emphasis.

These are two of the most important parameters to tune since they determine the complexity of the model and hence, enhance prediction accuracy.

4.1.4.4 Decision Tree

There are more parameters for a decision tree model due to their complexity and the nature of the tree development process. The parameters are as follows:

- *max_depth*: control the model complexity. A higher value for *max_depth* means that the model can map out more complex relationships, but often leads to overfitting.
  - Our experiment : 3, 5, 7, 10, None
- *min_samples_split*: minimum number of samples required to split an internal node, ensuring that nodes have enough data points before splitting, affecting its generalization. A higher value results in more general trees.
  - Our experiment: 2, 10, 20
- *min_samples_leaf*: minimum number of samples required to split a leaf node. A larger value results in general trees.
  - Our experiment: 1, 5, 10
- *max_features*: number of features to consider when looking for the best split.
  - Our experiment: None, sqrt, log2
- *max_leaf_nodes*: max-leaf nodes in the tree, affecting its generalization. A smaller value means a simpler model.
  - Our experiment:: None, 10, 20, 50

4.1.4.5 Random Forest

Random forest is referred to as an ensemble machine learning method, meaning that it combines multiple decision trees to produce a more powerful model (Rebala et al., 2019). Hence, the parameters are similar to a decision tree model. The only additional parameter is the *n_estimator,* which specifies the number of trees in the forest. Each tree is built on a different subset of the training data and the final prediction of the model is usually the average prediction from all the distinct trees. A higher value usually leads to a better performance up to a certain extent, because more trees could capture more complexity of the training data. We experimented with *n_estimator* of 50, 100, 200, 300.

4.1.4.6 XGBoost

Similar to the random forest, XGBoost is another ensemble method, where models (typically decision trees) are built sequentially to correct the errors made by the previous ones. The parameters that we tweaked are as follows:

- *n_estimators*: specifies the number of trees in the ensemble. More trees can lead to a better performance.
  - Our experiment: 100, 200, 300
- *Learning rate (eta)*: determines the contribution of each new tree that is being added to the ensemble model. A lower learning rate results in a more generalized model.

    - Our experiment: 0.01, 0.1, 0.3
- *max_depth*: control the complexity of each tree. A higher value for *max_depth* means that the model can map out more complex relationships, but often leads to overfitting.
    - Our experiment : 3, 5, 6, 7, 10

4.1.4.7 Neural Network

A neural network is a model inspired by how the human brain processes information, through interconnected layers of nodes (neurons) that work simultaneously to solve complex problems. The main parameter to set for a neural network model is its layer configuration referred to as *hidden_layer_sizes* when using MLPRegressor (Geron, 2019). This parameter defines the number and size of the hidden layers (layers of neurons positioned between the input layer and the output layer). More hidden layers can capture more complex patterns, the same applies to the number of neurons. For our model, we experimented with various combinations, which are as follows (the size of the list represents the number of layers and the number within the list is the number of neurons):

- (64, 32)
- (128, 64)
- (32, 16, 8)
- (100)
- (50, 50, 50)

4.1.5 Model Evaluation

The models will be assessed based on the evaluation metrics we have mentioned earlier. These metrics are mean squared error (MSE), mean absolute error (MAE), and r squared.

4.2 Dynamic Lot Size Model

For this research, the dynamic lot size model has been selected as the method to calculate optimal order quantities. The primary reason for this choice is that the forecasted demand, as illustrated in Figure (3), is highly variable and not constant. Figure (3) presents the demand of each product in a standardized manner, to present them on a consistent scale for ease of interpretability. Standardization subtracts the mean of the variable from each data point and then dividing the result by the standard deviation. As demonstrated, this variability contradicts the constant demand assumption of the EOQ method, making the dynamic lot size model a more appropriate choice for our analysis. In the next sub-chapter, the chosen model type and its components are discussed.

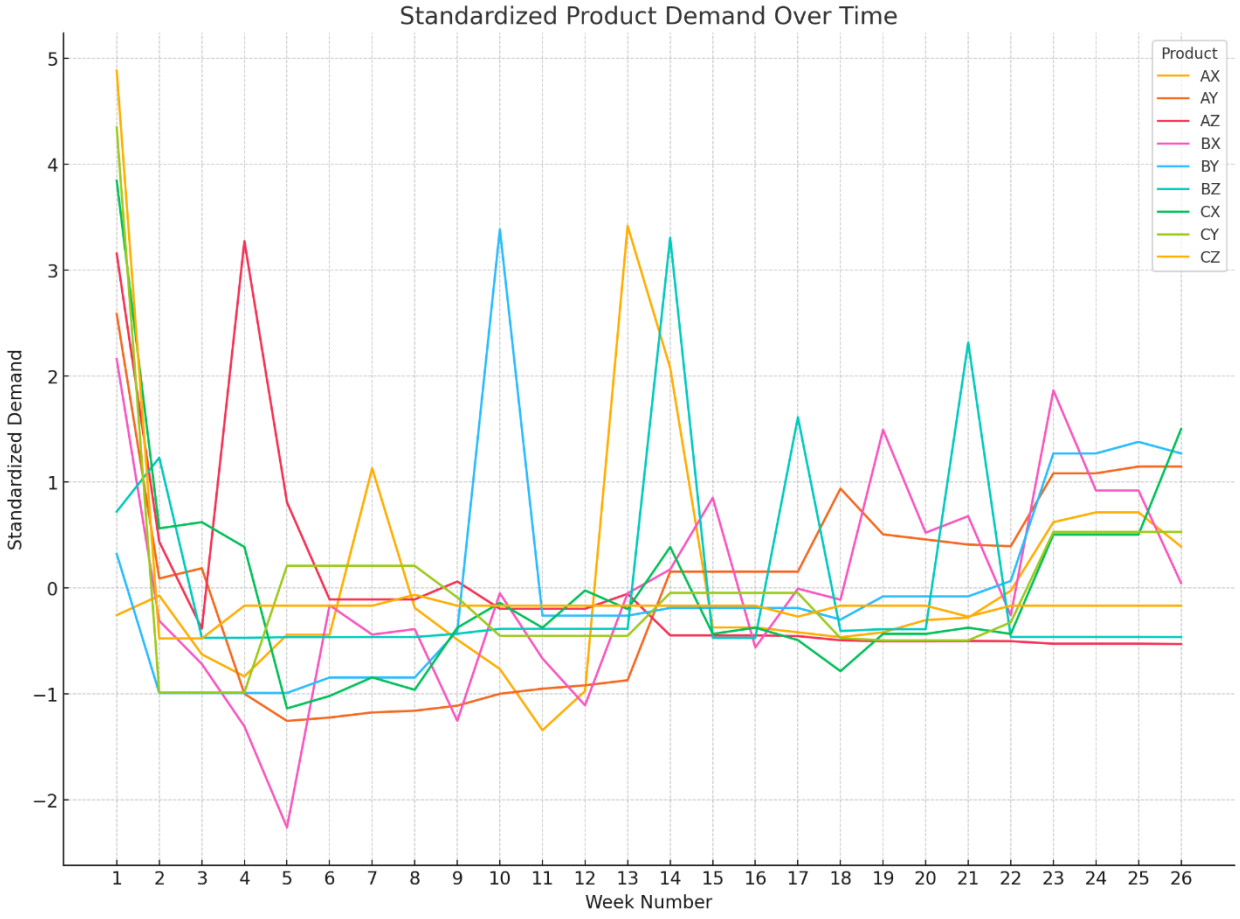

Figure 3: Standardized Demand of Product Over 26 Week Period

4.2.1 Deterministic vs Stochastic

This chapter aims to explain the reason behind the selection of whether to use a deterministic or stochastic dynamic lot size model, considering the available data. First and foremost, the decision is mainly based on the demand characteristic and the assumptions we make about it. Future demand is known, and it is highly fluctuating as seen on Figure 3, but demand forecasts are rarely 100% accurate, as in our case too. However, due to the lack of available actual demand data to evaluate the performance of the demand forecasts, which will give us information about the

distribution of the forecast's errors, we have opted to assume that demand is deterministic. This means that deterministic dynamic lot size model is the most suitable approach, given the limitation.

A stochastic model would produce outcomes that are more relevant to real-life situations, but the distribution parameters of the input parameters need to be known, which is not the case. Furthermore, the stochastic dynamic lot size model would increase the model complexity, increasing the running time of the model. Therefore, the deterministic model is chosen.

4.2.2 Model Input Parameters

This chapter aims to explain the input parameters used for the deterministic model, which are all tailored to meet the assumptions needed to run the model.

4.2.2.1 Demand

The first assumption is demand must be known in advance and assumed to be deterministic (Wagner and Whitin, 2004). The demand data used for the model would be the demand forecasts, available from our data. Hence, with this assumption needing to be met, we assume that demand forecasts are 100% accurate or in other words deterministic.

However, it is important to clarify that demand forecasts are not 100% accurate and rarely are in most cases. This inaccuracy is evident in the inventory level forecasts, which heavily depend on the accuracy of demand forecasts. Furthermore, the accuracy of these demand forecasts is unknown. If accuracy were available, the standard deviation of the forecast errors could be utilized and integrated into the calculation of the optimal order quantity. Consequently, the distribution parameters of the demand forecasts could be modeled, providing data for using a stochastic model. However, this is not the case, hence, we chose to assume demand to be deterministic.

4.2.2.2 Lead Time

The second assumption is lead time must be zero or in other words, replenishment must be instantaneous. Also, the lead time should be deterministic. Hence, we assume that the lead time is zero, despite the actual lead time of 7 weeks. The work around this discrepancy is to place the order quantity, produced by the model, 7 weeks before the actual period, resulting in the company receiving the order as expected by the model.

4.2.2.3 Cost Components

In determining the order quantities using a dynamic lot size model, several costs are needed. These costs are used to find the optimal tradeoffs between different ordering quantities and frequencies of ordering, ensuring inventory levels are managed efficiently. Unfortunately, the team supervising this research does not have access to the actual cost values, as they are not involved in the manufacturing or ordering processes. The only available data is the legal price, from which all costs are derived based on assumptions.

4.2.2.3.1 Holding Cost

In Versuni, a holding cost consists of storage costs such as utilities and maintenance costs, capital costs such as interest rate, insurance cost, and opportunity cost, which is potential revenue lost due to funds tied up in inventory that could have been used elsewhere for more profitable ventures.

According to Waters (2003), the typical inventory holding cost in the industry is 20% of the legal price. Hence, this value is chosen as the percentage of our holding costs.

4.2.2.3.2 Ordering Cost

Ordering cost is the cost incurred every time an order is placed. The cost usually covers among many things, the cost of procurement and the cost of handling the logistics. The ordering cost is highly dependent on the products that are being produced, hence an estimation is hard to get. As a result, after several discussions with the company supervisor, 80% of the legal price is chosen as the ordering cost.

4.2.2.3.3 Setup Cost

The inclusion of a setup cost in the dynamic lot size model aims to find an optimal balance between placing large orders at extended intervals and placing smaller orders more frequently (Muckstadt & Sapra, 2009). When there is no setup cost, the model would not need to consider the trade-off between ordering at specific periods versus ordering continuously, as there would be no financial penalty for frequent ordering.

For lack of a better option and through several experiments, we have chosen to multiply the legal price by 40. The multiplier demonstrates the model's ability to find a trade-off between ordering large sums at extended intervals and placing small orders at a more frequent interval. This number served as a benchmark, derived from analysis of operational expenses in firms with similar manufacturing and inventory processes (Choudhury et al., 2015).

4.2.2.3.4 Stockout Cost

Although stockout costs are not included as part of the dynamic lot size model, they can be useful for assigning costs to the number of stockouts resulting from the current ordering quantities, which we have deemed to be suboptimal. It is chosen to be 200% of the legal price. A stockout cost consists of a lot of aspects, such as loss of profit and damaged brand reputation. For simplicity, we have kept the stockout cost to 200% of the legal price.

4.2.3 Dynamic Programming

Dynamic programming (DP) is a mathematical model for solving complex problems, where decisions are made sequentially (Howard, 1966). The dynamic lot size model can be effectively solved with dynamic programming (Wagner and Whitin, 2004). Below are the components of our dynamic programming model, consisting of stages, states, decisions, state transition, immediate cost, and its recursion formula.

- Stage t : the beginning of the week t ($t \in \{1,2,\dots,26\}$)
- State i_t : the inventory level at the stage t
    - Initialization State i_1 : the initial inventory level for the recursion
        - The initialization state values are the inventory level at the beginning of the first stage. They are obtained from the dataset.

Product	Initialization State i_1
AX	28
AY	0

AZ	0
BX	350
BY	165
BZ	547
CX	0
CY	0
CZ	24

Table 10: Initialization State

- Decision x_t : the order quantity at the stage t
- Transition to the next stage : $i_{t+1} = i_t + x_t - d_t$ *(12)*
- Costs :
  - Holding cost
    - $C(H) = 0.2 * Legal\ price$ *(13)*
  - Setup cost
    - $C(S) = 40 * Legal\ price$ (incurred when $x_t > 0$) *(14)*
  - Ordering cost
    - $C(O) = 0.8 * Legal\ price$ *(15)*
- Constraints
  - $0 \leq x_t \leq \max inventory - i_t,$ $where \max inventory = \sum_{t=1}^{26} d_t$
    - The decision is constrained to a maximum number of items to be produced before the inventory is full.
- Objective function
  - $f_t(i_t)$: the minimum cost in ordering the optimal quantity over a period t given the inventory level of i_t.
- Recursion
  - $f_t(i_t) = min_{x_t \in X_t(i_t)}(C(S) + C(O).x_t + C(H).(i_t + x_t - d_t) + f_{t+1}(i_{t+1}))$ *(16)*
  - The recursion finds the minimal costs associated with all possible decisions. These costs include holding costs, setup costs, and ordering costs. The minimal total costs are achieved by making the optimal decision at each stage.

4.2.4 VBA

We chose VBA for the model because it is a part of Microsoft Excel, which is widely available, facilitating faster adoption by practitioners. However, VBA's old age and lack of add-ons, unlike Python, present some limitations, nonetheless, it remains our chosen implementation language.

Using VBA allows us to create an easy-to-use dashboard, consisting of input and output areas. For the input, a user should put in the legal price and initial inventory level, along with the forecasted demand for the products for the next 26 weeks. The legal price is needed since all cost calculations are based on it, calculating them as a percentage of it. Additionally, the initial inventory level is required to initialize the model, referring to the initial state in the mathematical formulation. Next, the output area consists of multiple rows of information: the optimal order quantity, the beginning inventory, the closing inventory, and the costs of putting the optimal order quantity. The VBA code

prints out these values when the input areas are filled accordingly, and the button called “Optimal Order Quantity” is pressed, basically executing the code.

In translating the mathematical formula into code for implementation, two two-dimensional arrays the size of all possible stages and states are created. These arrays store the optimal costs and the corresponding policies (decisions) for each stage and state. The most optimal costs are found by having a loop that goes over every possible decision for every possible stage and state, and using if statements to check if the cost is already the lowest possible. If that is the case, then the cost and policy are stored in the array. Lastly, a chunk of code is created to print out the outputs as mentioned earlier.

Input	
Legal Price	288,9
Initial Inventory Level	16

Optimal Order Quantity | Clear

Input	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Demand	62	26	26	26	26	30	30	30	42	146	46	46	46	48	48

Output	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Optimal Order Quantity	150	0	0	0	0	132	0	0	0	238	0	0	142	0	0
Beginning Inventory	16	104	78	52	26	0	102	72	42	0	92	46	0	96	48
Ending Inventory	104	78	52	26	0	102	72	42	0	92	46	0	96	48	0
Weekly Cost	52231	4507	3004	1502	0	47956	4160	2427	0	71876	2658	0	49920	2773	0

Figure 4: VBA Dashboard

4.3 Conclusion

This chapter answers the following research questions: “What is the procedure to develop a machine learning model and what is the chosen method to determine optimal order quantity?” Several sub-questions were answered as guidance to answer the main research question.

Steps to Develop a Machine Learning Model: Problem definition, data collection and understanding, data preparation, model development, and model evaluation.

Tools for Preparing Data: The `get_dummies` function from the Pandas library was used to one-hot encode categorical variables. The `train_test_split` function from the sci-kit-learn library was used to split our dataset into training and testing datasets, with an 80:20 ratio. The `StandardScaler` function from the scikit-learn library was used to standardize the data.

Hyperparameter Tuning: We experimented with diverse sets of parameters tailored to each model's characteristics and requirements by using GridSearchCV, a powerful tool for finding the optimal parameters. This resulted in the best-performing models.

Choice of method for determining optimal order quantity: The demand forecasts are very dynamic hence, the chosen method is the dynamic lot size model, specifically a deterministic model. There is a lack of important information about the distribution parameters of the forecasts to utilize the stochastic model, leading us to assume demand is deterministic.

Input parameters and mathematical formulation: The input parameters that need to be considered are demand, lead time, and costs. For lead time, it is assumed that the lead time is 0 and the costs are all assumed, each cost as a percentage of the legal price.

Chosen programming language: VBA was chosen as the programming language to implement our dynamic lot size model due to its availability and ease of use. A dashboard is also provided.

5 Results

This chapter answers the following research questions: “How are the machine learning model and method of determining optimal order quantity performing, in comparison to the current methods of inventory forecasting and ordering process?” In Chapter 5.1, the machine learning models are evaluated, including providing the chosen parameters resulting in the best performance. Then, the machine learning with the best performance is chosen and compared to the current method of forecasting. Additionally, in Chapter 5.2, a feature analysis is conducted to explore the variables with the most predictive power. Then, in Chapter 5.3, the result of the dynamic lot size model is discussed and compared to the current ordering quantity, this includes calculating the total inventory cost and stockout costs.

5.1 Machine Learning Model Evaluation

5.1.1 Parameters Selection and Model Performance

From conducting our hypermeter tuning in Chapter 4.1.4, we found the following to be the best parameters for each model. These are the parameters that deliver the best results based on our training data and a scoring metric of negative mean squared error.

Model Type	Parameters
KNN	{'n_neighbors': 7, 'weights': 'distance'}
Decision Tree	{'max_depth': 7, 'max_features': None, 'max_leaf_nodes': None, 'min_samples_leaf': 10, 'min_samples_split': 2}
Random Forest	{'n_estimators': 100, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 'log2', 'max_depth': None}
XGBoost	{'eta': 0.3, 'max_depth': 7, 'n_estimators': 100}
Neural Network	{‘hidden_layer_size’: (32, 16, 8)}
SVR	{none}

Table 11: Selected Parameters for each Model

As a result, these models produce outputs with performance as follows.

Model Type	MSE	MAE	R-Squared
Linear Regression	585.692	195	93%
KNN	1.046.295	249	88%
Decision Tree	782.581	216	91%
Random Forest	914.423	205	89%
XGBoost	1.540.161	209	82%
Neural Network	565.845	201	93%
SVR	8.578.721	642	2%

Table 12: Model Performance

All machine learning models, except for SVR, performed significantly better than the current inventory projection method. The best-performing models were linear regression and neural networks, showing the lowest MSE and MAE, and the highest r-squared values. Surprisingly, the linear regression model performed as well as the neural network model, which is generally considered more complex and powerful. However, based on these metrics, the neural network is still the best-

performing model due to its lower MSE compared to that of linear regression. Although its MAE is slightly higher, the difference is not significant. Choosing a model with the lowest MSE is preferable because it indicates the model's output has the fewest large errors.

Inventory Projection Method	MSE	MAE	R-Squared
Current Method	2.419.536	273	70%
Neural Network	565.845	201	93%
Difference	76%	26%	23%

Table 13: Comparison between the Chosen ML Model and the Current Method

A machine learning model has been successfully developed, producing significantly more accurate inventory projections than the current method. The model achieved a 76% decrease in Mean Squared Error, reducing it from 2,419,536 to 565,845, and a 26% decrease in Mean Absolute Error. Additionally, 93% of the variance in the dependent variable is predictable by the independent variables, an improvement from what was previously 70%, showcasing the model's better fit than the current method.

The results of the machine learning models prove that the current method of inventory level forecasting at Versuni is not optimal. The result demonstrates that relying on a static formula for forecasting leads to lower inventory forecast accuracy. This approach is insufficient because it fails to capture the complex relationship between various influencing variables. On the other hand, machine learning models can adapt to these relationships, providing more accurate forecasts. This comes especially handy given the fact that the demand forecasts at Versuni are not accurate. The utilization of machine learning demonstrates the potential benefits of integrating data-driven techniques into the inventory management practices at Versuni.

5.2 Feature Analysis

From analyzing the results, particularly the linear regression model, it was found that several variables have more influence on the outcome than others. The analysis was conducted by printing the absolute coefficient of each variable. The absolute value is taken because a large negative coefficient also represents high predictive power. Then, the variables are sorted in descending order based on this value. Consequently, it was found that the top five variables were categorical variables which had been one hot encoded, shown by the Table 14.

The table depicts significantly high absolute coefficients of some variables, representing their predictive power. The reason for the seemingly unreasonably high coefficient could be the fact that the machine learning model is trained on a complex dataset. The complexity was mainly caused by one-hot encoding, which created a variable for every unique value of the categorical variables. The input variables can be seen on Appendix 8.2, along with each one's coefficient. After further investigation, this could be a sign of overfitting, despite already applying 5-fold cross validation to prevent it.

Variables	Absolute Coefficient
Mega Market final_Europe	$5.0 * 10^{15}$
Mega Market final_METAR	$2.9 * 10^{15}$

Mega Market final_NAM	$1.9 * 10^{15}$
Revised Plant_KR20	$1.6 * 10^{15}$
Revised Plant_TH20	$1.5 * 10^{15}$

Table 14: Top 5 Variables with the Highest Absolute Coefficient

The numerical variables, such as stock on hand, incoming order, and forecasted demand are in the 37th,38th, and 39th positions respectively, out of 46 variables. The variables above them are all one-hot encoded variables. The table below shows the coefficients of these numerical variables.

Variables	Absolute Coefficient
Stock on Hand	$2.6 * 10^{3}$
GIT W+1 (Incoming orders)	$2.1 * 10^{2}$
Demand W+1 (Forecasted demand)	$3.4 * 10^{1}$

Table 15: Absolute Coefficient of Numerical Variables

Thus, it can be concluded that one-hot encoded descriptive categorical variables provide the model with useful information. Alongside the numerical variables, the machine learning models managed to enhance the accuracy of the predictions. However, disregarding the numerical variables would not be reasonable, as it would exclude features with values beyond the binary 0 and 1, which are the basis of one-hot encoding.

5.3 Dynamic Lot Size Model Result

5.3.1 Total Inventory Cost

By utilizing the Excel program which has the code to execute our dynamic lot size model, the optimal order quantity for the next 26 weeks for each product has been determined. The model ran for a long time for Product AX and Product AZ, due to their large demands. Below is a graph comparing the total ordering costs over the 26 weeks between the old ordering quantities and the new ordering quantities.

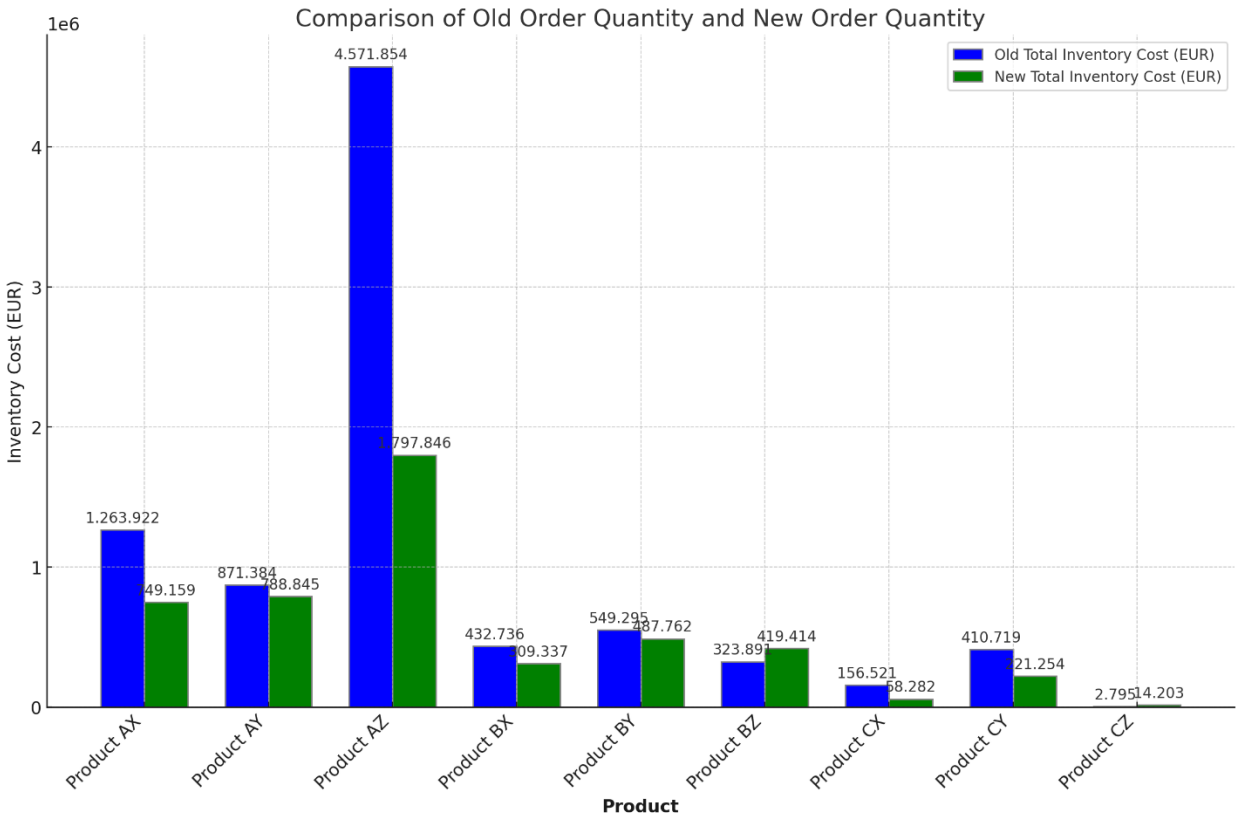

Figure 5: Comparison of Old Order Quantity and New Order Quantity

The comparison was done by calculating the total inventory costs using the same setup (costs, demand, and lead time) data used for the dynamic lot size model, offering an apple-to-apple comparison.

The dataset used for the research includes the order quantities from the old policy, which were produced by the models the company has. The model, as discussed in Chapter 2.4, uses advanced algorithms to formulate the most optimal ordering quantity, however it fails to do so as shown by the number of stockouts. Hence, to compare the old policy with the new policy (dynamic lot size model), we can simply compare the outcome of each model. Hence, under the same assumptions, we can compare the two outcomes.

To reiterate, as mentioned in Chapter 4.2.2, the demand data is assumed to be deterministic, and the lead time is assumed to be zero. Then, the costs are formulated as a percentage of the legal price. The weekly demand and weekly order quantities from the previous policy are known, thus the total associated costs can be calculated. Additionally, the weekly beginning and closing inventory levels can be derived from the data, enabling the calculation of the holding cost, which is included in the total cost.

Our model manages to reduce the total cost by a significant amount for all 9 products. It is important to note that the total cost of the new optimal order quantity includes the stockout cost too, however, since a dynamic lot size model does not allow any stockouts, this value is 0 for all products. The improvement is significant.

The deterministic dynamic lot size model successfully reduced the total inventory cost for 7 out of 9 products, significantly minimizing costs while avoiding stockouts. However, the total inventory cost for Product BZ and Product CZ is higher in the new model. This is because the old ordering quantities for these products were much smaller compared to the optimal quantities determined by the model, leading to originally high stockout values. If the stockout costs were quantified and included in the total cost for comparison, the old model's costs for these products would be higher than those calculated by the dynamic lot size model. As we know, the dynamic lot size model does not allow for any stockouts, implying the stockout cost associated with the optimal ordering quantity is zero.

Product	Stockout Quantity from the Old Order Quantity	Stockout Cost	Total Cost (incl. Stockout Cost) from the Old Order Quantity	Total Cost (incl. Stockout Cost) from the Optimal Order Quantity
BZ	6133	353.261	677.152	419.914
CZ	85	9.736	12.531	14.203

Table 16: Stockout Cost and Total Cost

To illustrate the improvement brought by the dynamic lot size model, a bar chart below compares the total cost of two products. Product CZ is different from the other cases because even with the stockout cost being part of the total cost, the total cost remains higher than the old policy. When analyzing the old order quantity for Product CZ, only one order was put throughout the 26 weeks, causing the setup cost to only be incurred once. This could be one of the reasons why the total cost of the new optimal order quantity is still higher despite having no stockouts. Another reason is the stockout costs might not be large enough to penalize the stockout quantities, since they are only assumptions, could in fact not represent the actual cost accurately. A larger stockout cost would increase the total cost of the old order quantity to the point where the optimal order quantity yields lower total cost. Thus, highlighting the effectiveness of the new optimal order quantity.

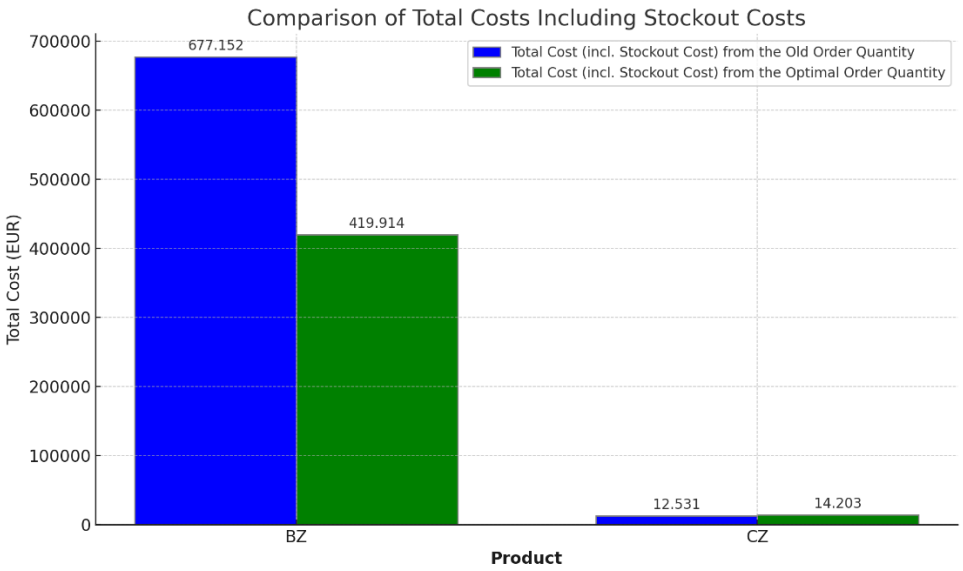

Figure 6: Total Cost of Product BZ and Product CZ

5.3.2 Drawbacks of the Model

Despite providing significant improvements in determining optimal order quantity with the least possible total inventory costs, our model has some flaws, preventing the outcomes from being reliable to use in real-world scenarios. First and foremost, the model, as previously explained, assumes that all input parameters are deterministic. This poses an issue when applying the model in real-life scenarios, in which the input parameters for the model, especially demand forecasts and lead times are rarely deterministic. Consequently, the model produces an optimal order quantity that does not capture the stochastic nature of the input parameters. In practice, ideally, the output of our model should be accompanied by safety stock to buffer against the uncertainty if the parameter for such calculation is available.

The primary reason for not choosing to use a stochastic model instead is the lack of available data. To utilize a stochastic model, the distribution parameters of the demand forecasts should be known. One of the ways to find the distribution is by measuring the forecast errors, by comparing the forecasts and the actual demand data. Unfortunately, the actual demand data is not available to us. If the forecast errors are known, then the standard deviation which represents the variability and uncertainty in the forecast could be formulated. Therefore, the standard deviation could be used to produce the probability distribution of each possible demand forecast, allowing our model to consider this stochasticity, and produce a more robust result.

Not only that but an inventory management practice such as safety stock could also be utilized to buffer against this uncertainty. To calculate the safety stock, first we need to measure the forecast errors, and this can be measured using Mean Absolute Deviation (MAD) as the statistical measure. The equation for MAD is as follows.

$$MAD = \frac{\sum_{i=1}^{n} \sum |Actual\ Demand - Forecasted\ Demand|}{n} \quad (17)$$

Then, the equation for the safety stock is as follows.

$$Safety\ Stock\ (SS) = Z * \sqrt{Lead\ Time} * MAD \quad (18)$$

Z is the z-score which represents the desired service level.

The safety stock will act as a buffer against the forecast errors as illustrated by Figure 7. Hence, the company should adapt their inventory by calculating the safety stock, once the MAD of the forecast errors is known.

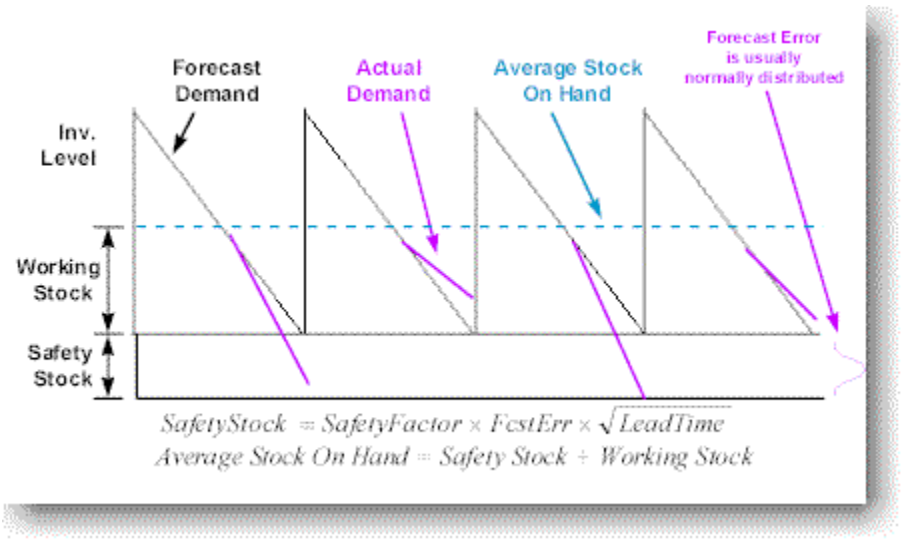

Figure 7: Safety Stock Against Forecast Errors

5.4 Conclusion

This chapter answers the following research question:" How are the machine learning model and method of determining optimal order quantity performing, in comparison to the current methods of inventory forecasting and ordering process?" The main question was answered by answering several sub-research questions, used as guides for the research.

Best Machine Learning Model: It was found that linear regression and neural networks are the best-performing models among the selection of models. The assessment was done by comparing its MSE, MAE, and R-squared against the current method of inventory forecasting and they were found to reduce the errors by a significant margin. In the end, the neural network was chosen as the best-performing model.

Parameters of the Best Machine Learning Model: The parameters of the neural network that produces the most accurate predictions are 'hidden_layer_size': (32, 16, 8), which specifies the configuration of hidden layers of the network.

Most Relevant Variables: The variables with the most predictive power turn out to be the categorical variables that have been one-hot-encoded. The numerical variables such as stock on hand, incoming order, and forecasted demand are on the 37th,38^{th,} and 39th positions respectively, out of 46 variables. This shows that the model was able to capture the relationship between inventory level forecasts and the one-hot encoded categorical variables, producing more accurate results.

Performance of the Dynamic Lot Size Model: The dynamic lot size model manages to determine the optimal order quantity, leading to a reduction in total inventory costs by a significant margin whilst avoiding stockouts. However, the model has some flaws, which include using deterministic demand and lead time, leading to less reliable results.

6 Conclusion, Limitations, and Recommendations

This chapter aims to answer the following question: "What are the conclusions and limitations of the research and what are the recommendations for further research?" The conclusion summarizes the research findings on inventory level forecasting and optimal order quantities. It also addresses the limitations and assumptions that may affect the reliability of these results. Finally, recommendations for further research are provided, aimed at enhancing the accuracy of inventory level forecasts and delivering more reliable results when determining optimal order quantities.

6.1 Conclusion

Following the sub-research question, we have managed to answer the main research question: *"How can the inventory level forecast accuracy be improved and frequent stockouts be avoided by developing a machine learning model and formulating optimal order quantities?"* The main finding is that a machine learning model provides a more data-driven forecast than the current method, which has failed to capture the relationship between available variables. The machine learning model managed to capture the complexity of these relationships and therefore, produced more accurate predictions, decreasing the MSE and MAE by 76% and 26% respectively, while increasing the R-squared to 93% from previously 76%. As for determining the optimal order quantities, it was found that a dynamic lot size model is a powerful method to find the optimal quantity, finding the right balance between all the costs associated with placing an order. As a result, it lowers the total inventory costs whilst avoiding stockouts for all the products.

First, an analysis of the current situation is conducted which revealed that the company uses a static formula, heavily relying on accurate demand forecasts to produce accurate inventory forecasts. In a sense, that is how it should be, however, due to a limitation set by the problem owner, improving the demand forecast accuracy is out of bounds. Hence, the focus was to deliver a tool that could counterbalance the inaccurate demand forecasts to increase the overall inventory level forecasts' accuracy.

Then, a literature review was conducted to familiarize ourselves with machine learning and its application in the supply chain. It was concluded that the research would benefit from implementing regression models. The chosen regression models are Linear Regression, SVR, KNN, Neural Network, Decision Tree, Random Forrest, and Gradient Boosted Tree. Furthermore, we prepare our data by merging multiple inventory forecast files starting from the first week until the 20th week of 2024. We perform data preparation practices such as standardizing, one-hot encoding variables, and splitting the dataset into training and testing datasets.

For each model, a hyperparameter tuning was conducted so that each model performs to the best of its ability. As a result, we found that Linear Regression and Neural Networks performed the best, with the lowest MSE, MAE, and R-squared among all the models. In the end, Neural Network was chosen to be the best model, since it lowered the MSE and MAE by 76% and 26% respectively, while increasing the R-squared to 93% from previously 76%. Lastly, we conducted a feature analysis to see the features with the most predictive power, turns out that the features with the highest absolute coefficient are the main one-hot encoded categorical variables. This proved that by implementing a

machine learning model, relationships between categorical variables and the target variable, that would otherwise be ignored, would enhance the accuracy of inventory forecasting.

Additionally, a situation analysis was also conducted to gather information about the current ordering process. The main finding was that the ordering process was done by the ERP system under the supervision of the supply planner, through the use of advanced algorithms. Despite its sophistication, it fails to determine the right order quantity to meet the demand forecasts. As a result, we conducted a literature review on methods of determining optimal order quantity, finding two relevant methods: Economic Order Quantity and Dynamic Lot Size Model. However, the chosen method was a deterministic dynamic lot size model, due to the nature of our demand which is deterministic and very dynamic. The input parameters for the model were all assumed due to a lack of better options. The model was then implemented in VBA, a widely available programming language that can be found in Excel. A dashboard was also created for ease of use.

As a result, the model was able to reduce the total inventory cost for 7 out of 9 products and avoid stockouts for each product. The comparison was between the old order quantity and the new order quantity, the costs are calculated the same way for both. Furthermore, if stockout costs were included in the calculation of the costs, the model would manage to reduce the total costs for 8 out of 9 products. Nevertheless, this model has some flaws, which brings us to the next chapter, in which we discuss the limitations of the research and recommendations for future research.

6.2 Limitations

The main limitation of this research is the lack of crucial information that would otherwise enhance the quality and reliability of the research. Below, we will list the limitations that we thought hindered the research from delivering the most reliable results. First, for the inventory level forecasting, the limitations are:

1. *Inaccurate demand forecasting:* The model was developed using inaccurate demand forecasting, for which the accuracy is unknown to us. This poses a problem that if the demand forecasting accuracy changes over time, the machine learning model would have a harder time understanding its pattern. On the other hand, if the demand forecasting accuracy was inaccurate yet constant, the machine learning model might have an easier time picking up the pattern, resulting in better predictions.
2. *Lack of actual demand data:* Due to this, we could not conduct our demand forecasting, which if we manage to increase would also result in higher inventory forecast accuracy. This is due to inventory forecasting being heavily reliant on demand forecasting.
3. *Reliability of data:* The files used to be fed to our machine learning models are constructed manually by the inventory management team, which compiles multiple data from different sources into one. This approach is prone to human errors; hence, its validity and reliability should be questioned.
4. *Computing power:* The research is limited to a generic laptop, resulting in less extensive hyperparameter tuning. Performing more extensive hyperparameter tuning could have resulted in a better model.

Additionally, for the dynamic lot size model, the following are some of its limitations:

1. *Lack of actual demand:* Due to a lack of actual demand data, the demand forecast errors are unable to be computed. Thus, the standard deviation of the forecasts is unknown, limiting us from developing a more robust model that considers the stochasticity of the demand. It also hinders us from computing safety stocks to buffer against uncertainty. Hence, the demand forecasts were assumed to be 100% accurate. This means that in practice, the order quantity would be higher than the output of our model, to buffer against the true uncertain nature of the demand, which would increase the total cost. This would contradict our finding, which suggests significantly lower total cost than the old policy. Therefore, the model's validity could be further examined if actual demand data becomes available for future research.
2. *Costs are all assumed:* Due to insufficient information, all cost estimates were assumed. Industry-standard percentages were used for some calculations, such as those based on the legal price. However, other costs, like the setup cost, were determined through trial and error using the dynamic lot size model. The setup cost was selected because it effectively demonstrated the model's ability to identify trade-offs, as discussed in Chapter 5.2.2.

6.3 Recommendations

This research has demonstrated that machine learning can significantly improve inventory forecast accuracy, even when using inaccurate demand forecasts as input. Additionally, it has shown that a dynamic lot size model is effective in determining optimal order quantities. However, some limitations were encountered. Therefore, several recommendations are provided for the company to maximize the benefits of this research and to guide future research.

1. *Re-train the machine learning model every week:* Given that the situation at the company persists, the machine learning model should be re-trained. Since the model in the research was trained on a dataset of 20-week horizon, it is advised that the company do the same by using data from the last 20 weeks. This ensures that the machine learning model is updated with the latest behavior of the inaccurate demand forecasts.
2. *Improve demand forecasting method:* The main reason for inaccurate inventory level forecasting is inaccurate demand forecasting. Improving the current method so that it provides more accurate forecasting would result in better inventory forecasting.
3. *Implement the machine learning model within the ERP system:* Nevertheless, incorporating an additional model specifically for inventory forecasting, alongside the existing demand forecasts, has been shown by this research to enhance the accuracy of inventory forecasts. By having these two models work together, inventory forecasting becomes more precise. Hence, for seamless integration and ease of use, it would be convenient for the machine learning model to be developed inside the ERP system. This would mean that the stakeholders should only worry about the results and not so much about the model, leading to a more efficient operation and adaptation.
4. *Measure the forecasts errors and its MAD:* Doing so allows the company to calculate safety stock to buffer against inaccuracies of forecasts. Hence, it can be added on top of the optimal order quantity produced by the dynamic lot size model, providing a more reliable result.

7 Bibliography

Alpaydin, E. (2010). Introduction to Machine Learning (2nd ed.). The MIT Press.

Angra, S., & Ahuja, S. (2017). Machine learning and its applications: A review. In 2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDACI) (pp. 58-60). IEEE. https://doi.org/10.1109/ICBDACI.2017.8070830.

Choudhury, K.D., Karmakar, B., Das, M., & Datta, T.K. (2015). An inventory model for deteriorating items with stock-dependent demand, time-varying holding costs, and shortages. SEARCH, 52, 55–74. Springer.

Chopra, S., & Meindl, P. (2015). Supply chain management: Strategy, planning, and operation (eBook, Global Edition). Pearson Education. https://books.google.nl/books?id=gPDQCQAAQBAJ.

Feizabadi, J. (2020). Machine learning demand forecasting and supply chain performance. International Journal of Logistics Research and Applications, 25(2), 119–142. https://doi.org/10.1080/13675567.2020.1803246.

Friedman, J.H. (2001). Greedy function approximation: A gradient boosting machine. Annals of Statistics, 29, 1189-1232.

Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems (2nd ed.). O'Reilly Media.

Hanafizadeh, P., Shahin, A., & Sajadifar, M. (2019). Robust Wagner–Whitin algorithm with uncertain costs. Journal of Industrial Engineering International, 15(3), 435–447. https://doi.org/10.1007/s40092-018-0298-y.

Heerkens, H., & van Winden, A. (2017). Solving Managerial Problems Systematically. Noordhoff Uitgevers.

Heijs, É. G. S. G. (2019). Design of an integrated long distance transportation, ordering and inventory quantity optimization tool (Master's thesis). Eindhoven University of Technology.

Howard, R. A. (1966). Dynamic programming. Management Science, 12(5), 317-348. https://doi.org/10.1287/mnsc.12.5.317.

Johnson, L. A., & Montgomery, D. C. (1974). Operations research in production planning, scheduling, and inventory control. John Wiley & Sons.

Kumar, Y. (2017). XYZ analysis for inventory management – A case study of the steel plant. International Journal for Research in Applied Science and Engineering Technology, 5, 46-52. https://doi.org/10.22214/ijraset.2017.2007.

Kumar, Y., Sahu, V., Sahu, D., Khaparde, R. K., Dhiwar, J. S., Dewangan, K., & Dewangan, G. K. (2017). XYZ Analysis for Inventory Management – Case Study of Steel Plant. *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, 5(2), 46-52. http://doi.org/10.22214/ijraset.2017.2007.

Lucey, T. (1992). Quantitative Techniques. London: Ashford Colour Press. 4thedition.

Muckstadt, J. A., & Sapra, A. (2010). Dynamic lot sizing with deterministic demand. In Principles of Inventory Management. Springer Series in Operations Research and Financial Engineering. Springer, New York, NY. https://doi.org/10.1007/978-0-387-68948-7_4.

Osadchy, E., Akhmetshin, E., Amirova, E., Bochkareva, T., Gazizyanova, Y., & Yumashev, A. (2018). Financial statements of a company as an information base for decision-making in a transforming economy. European Research Studies Journal, 21(2), 339-350.

Perez, H.D., Hubbs, C.D., Li, C., & Grossmann, I.E. (2021). Algorithmic approaches to inventory management optimization, Processes, 9, 102. MDPI.

Rebala, G., Ravi, A., & Churiwala, S. (2019). An introduction to machine learning. Springer. https://doi.org/10.1007/978-3-030-15729-6.

Scikit-learn developers. (n.d.). Preprocessing data. Scikit-learn. Retrieved July 4, 2024, from https://scikit-learn.org/stable/modules/preprocessing.html.

Schwarz, L.B. (2008). The Economic Order-Quantity (EOQ) Model.

Silver, E.A., Pyke, D.F., & Thomas, D.J. (2016). Inventory and Production Management in Supply Chains (4th ed.). CRC Press. https://doi.org/10.1201/9781315374406.

Stojanović, M., & Regodić, D. (2017). The significance of the integrated multicriteria ABC-XYZ method for the inventory management process. Acta Polytechnica Hungarica, 14(5), 29-48. https://doi.org/10.12700/APH.14.5.2017.5.3.

Wagner, H. M., & Whitin, T. M. (2004). Dynamic Version of the Economic Lot Size Model. Management Science, 50(12), 1770–1774. http://www.jstor.org/stable/30046142.

Waters, Donald. 2003. Inventory Control and Management, 2nd ed. West Sussex: Wiley.

Yiğit, F., & Esnaf, Ş. (2019). A new fuzzy C-means and AHP-based three-phased approach for multiple criteria ABC inventory classification. In Proceedings of the 10th International Symposium on Intelligent Manufacturing and Service Systems (pp. 633-642). Sakarya University.

8 Appendix

8.1 Linear Regression Code

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Load the data from the uploaded Excel file
excel_file = r"C:\Users\Gregorius\Documents\WITH NAs3.xlsx"
df = pd.read_excel(excel_file)

# Checking for missing values in the dataframe
missing_values = df.isnull().sum()


# Dropping the row with missing ID
df = df.dropna(subset=['ID','Stock on Hand W + 1 (TARGET)'])
df = df.drop(columns = ['Business Group','Business Unit','PROJECTION'])


# One-hot encoding for 'Revised Plant' and 'Mega Market final'
df_encoded = pd.get_dummies(df, drop_first=True)

# Features and target variable
X = df_encoded.drop(columns=['Stock on Hand W + 1 (TARGET)'])
y = df_encoded['Stock on Hand W + 1 (TARGET)']


# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardizing the numerical features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train the linear regression model
model = LinearRegression()
model.fit(X_train_scaled, y_train)

# Predict on the test set
y_pred_test = model.predict(X_test_scaled)

# Predict on the test set
y_pred_train = model.predict(X_train_scaled)

# Calculate evaluation metrics for the test set
mse_test = mean_squared_error(y_test, y_pred_test)
mae_test = mean_absolute_error(y_test, y_pred_test)
rmse_test = np.sqrt(mse_test)
r2_test = r2_score(y_test, y_pred_test)
```

8.2 Absolute Coefficients of Variables

	Feature	Coefficient	Absolute Coefficient
40	Mega Market final_Europe	5.022903e+15	5.022903e+15
44	Mega Market final_METAR	2.965409e+15	2.965409e+15
45	Mega Market final_NAM	1.910183e+15	1.910183e+15
23	Revised Plant_KR20	1.601126e+15	1.601126e+15
31	Revised Plant_TH20	1.526126e+15	1.526126e+15
41	Mega Market final_GRC	1.497207e+15	1.497207e+15
42	Mega Market final_ISC	1.399062e+15	1.399062e+15
8	Revised Plant_BR10	1.266036e+15	1.266036e+15
29	Revised Plant_SG20	1.265651e+15	1.265651e+15
24	Revised Plant_MY20	1.213888e+15	1.213888e+15
7	Revised Plant_AU20	1.149951e+15	1.149951e+15
21	Revised Plant_ID20	1.097285e+15	1.097285e+15
37	Revised Plant_VN20	1.072999e+15	1.072999e+15
38	Revised Plant_VN21	9.762654e+14	9.762654e+14
6	Revised Plant_AR20	9.570799e+14	9.570799e+14
30	Revised Plant_SG21	8.874644e+14	8.874644e+14
10	Revised Plant_CL20	7.496259e+14	7.496259e+14
11	Revised Plant_CL21	6.356111e+14	6.356111e+14
12	Revised Plant_CN20	6.322492e+14	6.322492e+14
18	Revised Plant_HK20	6.059400e+14	6.059400e+14
13	Revised Plant_CN21	5.370808e+14	5.370808e+14
34	Revised Plant_TW20	5.190584e+14	5.190584e+14
0	Year	-5.151897e+14	5.151897e+14
36	Revised Plant_US20	-4.330972e+14	4.330972e+14
9	Revised Plant_CA20	-3.669611e+14	3.669611e+14
25	Revised Plant_NL20	-2.504066e+14	2.504066e+14
19	Revised Plant_HU20	-1.695387e+14	1.695387e+14
26	Revised Plant_NL80	-1.616595e+14	1.616595e+14
14	Revised Plant_FR20	-1.588421e+14	1.588421e+14
27	Revised Plant_PL21	-1.511556e+14	1.511556e+14
20	Revised Plant_HU80	-1.197680e+14	1.197680e+14
43	Mega Market final_LATAM	1.119214e+14	1.119214e+14
35	Revised Plant_UA20	-9.716762e+13	9.716762e+13
22	Revised Plant_IN21	9.431517e+13	9.431517e+13
15	Revised Plant_GB20	-9.268613e+13	9.268613e+13
17	Revised Plant_GR21	-7.976566e+13	7.976566e+13
16	Revised Plant_GB80	-6.975778e+13	6.975778e+13
3	Stock on Hand	2.647901e+03	2.647901e+03
4	GIT W+1	2.121300e+02	2.121300e+02
5	Demand W+1	-3.409279e+01	3.409279e+01
28	Revised Plant_SA20	-2.206478e+01	2.206478e+01
1	Week(W)	1.238027e+01	1.238027e+01
2	ID	1.073184e+01	1.073184e+01
32	Revised Plant_TR20	-1.031752e+01	1.031752e+01
39	Revised Plant_ZA20	-1.019926e+01	1.019926e+01
33	Revised Plant_TR21	-7.277868e+00	7.277868e+00

8.3 KNN Code

```
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Define the parameter grid
param_grid = {
    'n_neighbors': range(1, 15, 2),
    'weights': ['uniform', 'distance']
}

# Initialize the model
knn = KNeighborsRegressor()

# Perform the grid search
grid_search = GridSearchCV(knn, param_grid, cv=5, scoring='neg_mean_squared_error')
grid_result = grid_search.fit(X_train_scaled, y_train)

# Get the best parameters
best_params = grid_search.best_params_
print("Best parameters found: ", best_params)
```

```
Best parameters found:  {'n_neighbors': 7, 'weights': 'distance'}
```

```
# Initialize the final model with the best parameters
import numpy as np
final_knn_model = KNeighborsRegressor(n_neighbors = 1, weights= 'uniform')
final_knn_model.fit(X_train_scaled, y_train)

# Predict on the test set
y_pred_test = final_knn_model.predict(X_test_scaled)

# Predict on the training set
y_pred_train = final_knn_model.predict(X_train_scaled)

# Calculate evaluation metrics for the test set
mse_test = mean_squared_error(y_test, y_pred_test)
mae_test = mean_absolute_error(y_test, y_pred_test)
rmse_test = np.sqrt(mse_test)
r2_test = r2_score(y_test, y_pred_test)

# Calculate evaluation metrics for the training set
mse_train = mean_squared_error(y_train, y_pred_train)
mae_train = mean_absolute_error(y_train, y_pred_train)
rmse_train = np.sqrt(mse_train)
r2_train = r2_score(y_train, y_pred_train)

# Print evaluation metrics for the test set
print("Test Set Metrics:")
print(f"Mean Squared Error: {mse_test}")
print(f"R-squared: {r2_test}")
print(f"Mean Absolute Error: {mae_test}")
print(f"Root Mean Squared Error: {rmse_test}")
```

8.4 Decision Tree

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import GridSearchCV
import numpy as np

# Create a Decision Tree Regressor
dt_reg = DecisionTreeRegressor(random_state=42)

# Define the hyperparameter grid
param_grid = {
    'max_depth': [3, 5, 7, 10, None],
    'min_samples_split': [2, 10, 20],
    'min_samples_leaf': [1, 5, 10],
    'max_features': [None, 'sqrt', 'log2'],
    'max_leaf_nodes': [None, 10, 20, 50],
}
# Set up the GridSearchCV
grid_search = GridSearchCV(estimator=dt_reg, param_grid=param_grid, cv=5, scoring='neg_mean_squared_error', n_jobs=-1)

# Fit the model
grid_search.fit(X_train_scaled, y_train)

# Print the best parameters and the best score
print(f"Best parameters: {grid_search.best_params_}")
print(f"Best score: {np.sqrt(-grid_search.best_score_)}")
```

```
Best parameters: {'max_depth': 7, 'max_features': None, 'max_leaf_nodes': None, 'min_samples_leaf': 10, 'min_samples_split': 2}
Best score: 895.9136608201806
```

```
best_params = grid_search.best_params_
best_dt_reg = DecisionTreeRegressor(**best_params)
# Train the regressor on the training data
best_dt_reg.fit(X_train_scaled, y_train)

# Make predictions on the testing data
y_pred = best_dt_reg.predict(X_test_scaled)

# Calculate Mean Squared Error
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)

# Calculate R-squared
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
print(f"MAE: {mae}")
```

```
Mean Squared Error: 782581.0805368718
R-squared: 0.9105720192214021
MAE: 216.710271915498
```

8.5 Random Forest

```
# Import necessary libraries
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

from sklearn.model_selection import RandomizedSearchCV

param_dist = {
    'n_estimators': [50, 100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2'],
}

# Initialize the Random Forest Regressor
reg = RandomForestRegressor(random_state=42)

# Initialize RandomizedSearchCV
random_search = RandomizedSearchCV(estimator=reg, param_distributions=param_dist, n_iter=100, cv=3, scoring='neg_mean_squared_error', n_jobs=-1, verbose

# Fit RandomizedSearchCV
random_search.fit(X_train_scaled, y_train)

# Get the best parameters and the best score
best_params = random_search.best_params_
best_score = -random_search.best_score_

print(f'Best parameters: {best_params}')
print(f'Best cross-validated MSE: {best_score}')

# Train the model with the best parameters
best_reg = random_search.best_estimator_
best_reg.fit(X_train_scaled, y_train)

# Make predictions
y_pred = best_reg.predict(X_test_scaled)

# Calculate Mean Squared Error
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)

# Calculate R-squared
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
print(f"MAE: {mae}")
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits
Best parameters: {'n_estimators': 100, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 'log2', 'max_depth': None}
Best cross-validated MSE: 1037620.22317314
Mean Squared Error: 914423.5712048847
R-squared: 0.8955059666237939
MAE: 205.14559043256793
```

8.6 XG BOOST

```
from sklearn.model_selection import GridSearchCV, train_test_split
import xgboost as xgb
import pandas as pd


param_grid = {
    'max_depth': [3, 5, 6,7, 10],
    'eta': [0.01, 0.1, 0.3],
    'n_estimators': [100, 200, 300],
}
# Initialize the XGBoost regressor
xgb_reg = xgb.XGBRegressor(objective='reg:squarederror', eval_metric='rmse', use_label_encoder=False)


# Set up the grid search
grid_search = GridSearchCV(estimator=xgb_reg, param_grid=param_grid, cv=3, scoring='neg_mean_squared_error', verbose=1, n_jobs=-1)

# Fit the grid search to the data
grid_search.fit(X_train_scaled, y_train)

# Get the best parameters and score
best_params = grid_search.best_params_
best_score = grid_search.best_score_

print(f"Best parameters found: {best_params}")
print(f"Best cross-validation score: {-best_score:.2f}")

# Evaluate on the test set
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test_scaled)


# Calculate evaluation metrics
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error: {mae}")
print(f"R-squared: {r2}")
```

```
Fitting 3 folds for each of 45 candidates, totalling 135 fits
C:\Users\Gregorius\anaconda3\Lib\site-packages\xgboost\core.py:158: UserWarning: [19:27:13] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-06abd128ca6c1688d-1\xgboost\xgboost-ci-windows\src\learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
Best parameters found: {'eta': 0.3, 'max_depth': 7, 'n_estimators': 100}
Best cross-validation score: 1689975.52
Mean Squared Error: 1540161.425479096
Mean Absolute Error: 209.0916417768484
R-squared: 0.824000950471236
```

8.7 Neural Network

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error

# Define a function to create and evaluate the model
def evaluate_mlp(hidden_layer_sizes):
    mlp = MLPRegressor(hidden_layer_sizes=hidden_layer_sizes, activation='relu', solver='adam', max_iter=500, random_state=42)
    mlp.fit(X_train_scaled, y_train)
    y_pred = mlp.predict(X_test_scaled)
    mse = mean_squared_error(y_test, y_pred)
    print(f'Mean Squared Error with hidden layers {hidden_layer_sizes}: {mse}')
    return mse

# Experiment with different configurations
layer_configs = [
    (64, 32),
    (128, 64),
    (32, 16, 8),
    (100,),
    (50, 50, 50)
]

for config in layer_configs:
    evaluate_mlp(config)
```

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error


# Build and configure the model
mlp = MLPRegressor(hidden_layer_sizes=(32, 16, 8), activation='relu', solver='adam', max_iter=500, random_state=42)

# Train the model
mlp.fit(X_train_scaled, y_train)

# Predict using the model
y_pred = mlp.predict(X_test_scaled)


# Calculate evaluation metrics
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error: {mae}")
print(f"R-squared: {r2}")

# If you want to see the predictions
print(f'Predictions: {y_pred[:10]}')
print(f'True Values: {y_test[:10]}')
```

```
Mean Squared Error: 565845.4540009229
Mean Absolute Error: 201.54012345323608
R-squared: 0.9353390752184593
```

8.8 SVR

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt


# Training the SVR model
svr_regressor = SVR(kernel='rbf')
svr_regressor.fit(X_train_scaled, y_train)

# Making predictions
y_pred = svr_regressor.predict(X_test_scaled)

# Calculate evaluation metrics
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error: {mae}")
print(f"R-squared: {r2}")
a
# Plotting the results
plt.scatter(X, y, color='red', label='Original data')
plt.scatter(X_test, y_pred, color='blue', label='SVR predictions')
plt.title('Support Vector Regression')
plt.xlabel('Feature')
plt.ylabel('Target')
plt.legend()
plt.show()
```

```
Mean Squared Error: 8578721.419257673
Mean Absolute Error: 642.8319488188453
R-squared: 0.01968274819521998
```

8.9 Total Inventory Cost Comparsion

Product	Old Total Inventory Cost (EUR)	New Total Inventory Cost (EUR)
Product AX	1.263.922	749.159
Product AY	871.384	788.845
Product AZ	4.571.854	1.797.846
Product BX	432.736	309.337
Product BY	549.295	487.762
Product BZ	323.891	419.414
Product CX	156.521	58.282
Product CY	410.719	221.254
Product CZ	2.795	14.203