Waste Minimization in a Fully Automated Warehousing System

Jumbo

Geelen, R.

Award date:
2023

Eindhoven University of Technology
Department of Industrial Engineering & Innovation Sciences
MSc Operations Management and Logistics

Waste Minimization in a Fully Automated Warehousing System

Jumbo

by Romée Geelen
Student ID: 0961548

January 26, 2023

Supervisors TU/e:
dr. Ahmadreza Marandi
dr. Karel van Donselaar

Supervisor Jumbo:
Dik Jansen

Abstract

Jumbo, the second-largest supermarket chain in the Netherlands, uses a fully automated warehousing system to fulfill customer orders. With this system, order picking can be performed fast, efficiently, safe and with less sensitivity to errors. However, the system still shows some flaws, as currently more than 1 000 cases fall during the process on a weekly basis. Consequently, Jumbo loses more than €10 000 per week as these cases cannot be sold anymore. On top of that, fallen cases incur additional cleaning costs, potentially result in machine downtime and decrease the service quality of the warehouse. Therefore, this thesis focused on developing a method to minimize the number of fallen cases. A Random Forest and Multilayer Perceptron were used to predict the likelihood of cases falling and to detect potential causes. The results showed that especially the number of partially depalletized pallets adversely affects the number of fallen cases. A bi-objective Mixed-Integer Non-Linear Program was formulated, to find new replenishment strategies with minimal partial depalletizations. Comparing this new replenishment strategy with Jumbo's actual replenishment strategy, proved that a reduction in partial depalletizations indeed resulted in a reduction in fallen cases.

Executive Summary

Jumbo, the second-largest supermarket chain in the Netherlands, was one of the first Dutch retailing companies to own a fully automated warehousing system. With this system, Jumbo is able to fulfill customer orders fully automated, such that order picking can be performed fast, efficiently, safe and with less sensitivity to errors. This automated warehousing system is located in Jumbo's central distribution centre (CDC) in Nieuwegein, where currently 12 000 slow and medium moving Stock Keeping Units (SKUs) are handled in the category of dry groceries. Customer orders are picked within this CDC, which are shipped directly to the Jumbo stores or to another, cross-docking, distribution centre first. The CDC operates (in general) six days a week for 24 hours per day and realized an average weekly volume of more than 1.8 million cases in the first 40 weeks of 2022.

By achieving such volumes, the demand of stores is met in general. However, the automated warehousing system still shows some flaws. Inconveniently, a lot of cases fall due to either manual or machine errors during certain steps in the process. Jumbo lost €131 559 in 2021 due to cases falling, as these cases could not be sold anymore. This estimate is based on manual registration of broken cases and corresponds to more than 200 cases per week. Even though this amounts to only 0.01% of the average weekly volume, it is quite a substantial amount in absolute terms. Moreover, it brings more disadvantages than the direct lost cost of fallen cases.

To start, the fallen cases need to be manually removed from the CDC. This requires additional cleaning time and thus labor costs, especially if the broken cases consider items that are hard to clean properly (e.g. oil or rice). Moreover, SKUs potentially harm the machines when they break, which might result in machine errors and/or machine downtime, which adversely affects the productivity of the CDC and consequently yields (indirect) additional costs as well. Finally, fallen cases worsens the service quality of the CDC. If a case falls closer to the end of the process, the missing case potentially cannot be immediately replaced by a new case of the same SKU. As stores do not receive their ordered SKUs on the desired day, this adversely affects the service quality of the CDC. Concluding, fallen cases bring many disadvantages and therefore the goal of this thesis was to minimize the number of fallen cases.

As indicated in the previous paragraph, Jumbo's current estimates are based on manual registration of fallen cases. However, these manual registrations are often neglected or not performed properly. Consequently, these estimates are likely to underestimate the true number of fallen cases. Therefore, the processes in the CDC were analysed to find more accurate estimates. Order picking of customer demand is done via three different subsystems: OPM, CPS and DPS. In the OPM, nearly all process steps are mechanised, whereas manual labour is required in the other two subsystems. Hence, when a case falls in the CPS or DPS, this is often due to human errors. In contrast, no human interaction is usually required in the OPM, such that the problem is caused by different reasons. The focus of this thesis was to minimize the number of fallen cases in the OPM solely. In particular, all processes up to actual order picking were considered, as preliminary analyses showed that the problem is largest here. To properly understand the concepts of this thesis, it is important to get a better insight in these processes.

SKUs enter the CDC on pallets via an infeed station, where the number of cases on the pallet are measured. The pallets are stored in the highbay warehouse (HBW) before being handled. After some time, the pallets are send to one of the DEPALs, which vacuums each layer separately from the pallet

and puts all cases of that layer on a conveyer belt. A pallet can be either fully or partially depalletized, where in the latter scenario the remaining number of layers is send back to the HBW and called later again for further depalletization. On the conveyer belt, tray-loading occurs at the tray merge, where each case is individually put on a tray. At this point, the inventory is measured again, as the tray merge counts how many cases were actually put on a tray. After tray-loading has taken place, each tray is stored in the tray warehouse (TWH). Work orders are assigned to the DEPALs based on the current stock in this TWH: if the current inventory level of an SKU drops below a certain reorder point, a work order for this SKU is assigned to the DEPALs.

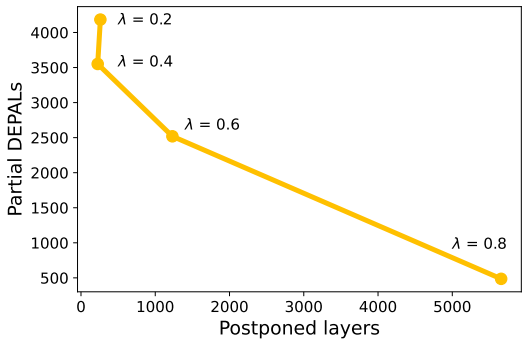
To find new estimates for the number of fallen cases within the described processes, the information obtained by the infeed stations was compared with the information obtained by the tray merge. If there were discrepancies between the measured inventory at these steps, the missing cases were considered as fallen. Hence, the processes from inbound to tray merge were considered as a black box, of which the input and output streams were of particular interest. Using this approach, the number of fallen cases was estimated to be nearly 1 000 cases each week, corresponding to a loss of more than €10 000 per week. Hence, the results confirm the problem of Jumbo's CDC in Nieuwegein and emphasize the need to establish methods to decrease the number of fallen cases.

First, two prediction models were implemented to accurately predict the likelihood of cases falling and to detect potential causes. The problem is highly imbalanced, as much more pallets are handled from which no cases fell than the opposite. Therefore, a method to account for this imbalance was implemented in combination with the prediction models. Many different combinations of models and hyperparameter values were tested, which eventually showed that a Random Forest (RF) on a fully undersampled dataset provided the best results. This model provided the right predictions for 64% of the pallets with fallen cases on an unseen test dataset. Given the fact that the data is likely to contain a lot of noise (e.g. cases fall due to an incidentally badly stacked pallet or just bad luck), the prediction model was still able to find generalizable patterns.

Analysis of the feature importance of the RF model, showed that the number of partial depalletizations adversely affects the number of fallen cases. Additionally, smaller cases and (consequently) more cases on a pallet increase the likelihood of falling. This effect can be devoted to two things: (i) a pallet with many small cases is likely to be more unstable and (ii) pallets with more inventory are more likely to be partially depalletized. Hence, these two findings go hand in hand: pallets with more inventory are more likely to be partially depalletized, which reduces the stability of the pallet and thereby increases the likelihood of falling. To minimize the number of fallen cases, the number of partial depalletizations should thus be minimized and the stability of the pallets should be ensured at all times. The latter can be achieved by judging the pallets at different steps in the process and, if necessary, improving the stability by adding additional stretch wrapping or re-stacking the pallet. To minimize the number of partial depalletizations, a mathematical model was introduced.

This mathematical model was based on a standard (R, s, nQ) inventory management system, as introduced in Section 2.3. This system was used to find initial order quantities (i.e. the demand) for the TWH. Then, the mathematical model adjusts these initial quantities by rounding them up or down to full pallets, to minimize the number of partial depalletizations. However, if too many order quantities are rounded down, the service quality of the TWH would be decreased significantly. As this is an undesired effect, the mathematical model was formulated as a bi-objective MINLP, which minimizes (i) the number of partial depalletizations and (ii) the total unmet demand of the TWH. A further explanation of the MINLP, with adherent constraints, is provided in Section 5.

The MINLP proved to find solutions that decrease the number of partial depalletizations as compared to the actual replenishment strategy, where 4759 pallets were partially depalletized in two weeks time. As the figure below shows, the number of partial depalletizations could already decrease significantly at only a limited number of intentionally postponed demanded layers. Eventually, the preferred replenishment strategy should be based on the desired trade-off between these two objectives: less par-



Results of replenishment strategy of MINLP for two weeks of operation, where λ represents the weight applied to the minimization of partially depalletized pallets and $(1-\lambda)$ represents the weight applied to the minimization of total unmet demand of the THW.

tially depalletized pallets results in less fallen cases, whereas less postponed layers results in a higher service quality of the TWH.

In the current analysis, the MINLP was solved for two days at once. Then, six MINLPs were solved sequentially to obtain a replenishment strategy for two weeks of operation. As each individual MINLP finds an optimal strategy for only two days, it finds short-term optimal decisions. The figure above shows that these models already find promising results, but increasing the time period of one MINLP might even yield more beneficial results on the long-term. However, the formulated MINLP is computationally expensive, such that solving for a longer time period at once might become intractable. Therefore, a suggestion for future research would be to formulate a heuristic for this matter.

Using the replenishment strategy of the MINLP, it was tested whether a reduction in partially depalletized pallets indeed results in a number of fallen cases. Feeding a new dataset based on the MINLP replenishment strategy in the trained RF, showed that a decrease in partially depalletized pallets, decreased the number of predicted pallets with fallen cases. Hence, these results confirmed that minimizing the number of partially depalletized pallets, minimizes the number of fallen cases.

Altogether, there is a potential for Jumbo to decrease the number of fallen cases, by employing a new replenishment strategy that minimizes the number of partially depalletized pallets.

Preface

This thesis marks the end of the master Operations Management and Logistics at Eindhoven University of Technology and thereby it also marks the end of my student life. I would like to thank everyone who was part of it and helped me to get where I am now.

In particular, I would like to thank Ahmadreza Marandi. During the past three years, he helped me not only with this graduation project, but also guided me towards a supervisor and a project for completing my master at Tilburg University. Then, since I started with this project, we have had plenty of meetings where he was able to answer all my questions, provide me with new directions and give quick, constructive feedback. Especially his knowledge on optimization problems and the way he is able to understand them so quickly impressed me enormously. In addition, I really appreciated that he always cared most about my personal life. He even cheered one of my hockey matches! Secondly, I want to thank Karel van Donselaar for his interest in my thesis, taking the time to thoroughly discuss the content, providing detailed feedback and ensuring that I was better able to acknowledge the value of this thesis.

Furthermore, I am grateful for the opportunity to conduct my thesis at Jumbo. My career once started as a re-stocker of shelves at a supermarket and I really enjoyed to be able to look at all the processes further up in the supply chain. Moreover, I could not have been guided in a better way than by my company supervisor Dik Jansen. Thank you for always sparing time to help me with my thesis despite your busy schedule, knowing literally everything about Jumbo, putting effort in understanding the contents, providing trust and caring about my personal life. I really enjoyed our weekly meetings and the fact that we caught ourselves spending more time talking about our weekends rather than about the project.

Lastly, I would like to thank my family and friends from Eindhoven and Den Bosch in making these past seven and a half years memorable. I really made the best out of my student life and could not have done it without them. In particular I want to thank Di, who simultaneously worked on her own thesis. We have spend many hours on our self-claimed study spots at Atlas 8 and even though we struggled a bit sometimes, we always managed to make it better together. Besides, I want to thank Vin for his infinite support and always letting me borrow his much faster laptop when I needed to do many long runs. Without him, my laptop would probably still be optimizing... Furthermore, I want to thank Mats for his feedback and very critical eye, he really helped in making this thesis more clear (although he did not allow me to use that word too much). Finally, I want to thank my family for supporting me in every step on the way and being proud of me no matter what.

With the end of my student life, a new chapter will start. During this thesis, I have not only gained a lot of knowledge about the contents, but also learned a lot about myself. It was definitely though sometimes, but I am proud of what I achieved. For now, I am looking forward to take a few weeks rest and am very curious to see what the future will bring afterwards.

Enjoy reading my work!
Romée Geelen

Contents

List of Tables

List of Figures

List of Abbreviations

ANN	Artificial Neural Network
AUC	Area Under ROC Curve
CDC	Central Distribution Center
CPS	Car Picking System
CSL	Cost-Sensitive Learning
DC	Distribution Center
DEPAL	Depalletizer
DOS	Days of Stock
DPS	Dynamic Picking System
FN	False Negatives
FP	False Positives
FPR	False Positive Rate
HBW	Highbay Warehouse
IR	Imbalance Ratio
KPI	Key Performance Indicator
MINLP	Mixed-Integer Non-Linear Programming
ML	Machine Learning
MLP	Multilayer Perceptron
MSE	Mean Squared Error
OCB	Order Consolidation Buffer
OPM	Order Picking Machinery
RDC	Regional Distribution Center
RF	Random Forests
ROC	Receiver Operating Curve
SKU	Stock Keeping Unit
SMOTE	Synthetic Minority Oversampling Technique
TN	True Negatives
TNR	True Negative Rate
TP	True Positives
TPR	True Positive Rate
UM	Unit of Measurement
TWH	Tray Warehouse

Chapter 1

Introduction

1.1 Company Introduction

In this section, general information about Jumbo Food Group is presented, after which Jumbo Nieuwegein - the distribution center where this research is conducted - is introduced.

1.1.1 Jumbo Food Group

The family business Jumbo was founded in 1921 and initially started as a wholesaler of groceries. The first real supermarket named Jumbo was opened in 1983 and from this point on the company has grown a lot. Over the years, Jumbo acquired multiple Dutch supermarket chains, with Super de Boer (in 2009) and C1000 (in 2012) being the largest takeovers. At the end of 2021, Jumbo was the second-largest supermarket chain in the Netherlands, having a market share of 22% and an annual turnover of €9.91 billion. Both numbers have shown an increasing trend over the past years. In total, Jumbo comprised 705 supermarkets at the end of 2021. Besides the standard brick and mortar supermarket stores, Jumbo also started selling its products online from 2014 onward; a market that cannot be left out in current lifestyles. Moreover, Jumbo added the food market concept to their market channels in 2013, where customers can find the largest food assortment in the Netherlands at the lowest price. In addition to the weekly standard groceries, customers can buy healthy and fresh meals prepared by chefs and specialists here.

The headquarter of Jumbo is located in Veghel. The company has three central distribution centers (CDCs), which distribute items to supermarkets and other distribution centers (DCs) on a national level. In addition, the company has four regional distribution centers (RDCs), which distribute items to supermarkets on a regional level. Finally, the company has three e-fulfilment centers and fifteen hub's which are used to fulfill all online customer orders. In total, the company employed approximately 100000 persons end of 2021. The Jumbo employees contain the so-called Jumbo DNA, meaning that they work according to three key values:

- *Together*: Jumbo employees work together and help each other if necessary.
- *Entrepreneurship*: Jumbo employees see and take opportunities and like to show initiative.
- *Win*: Jumbo employees want to improve every day and are thrilled to find the best solution or idea for customers and colleagues.

Moreover, all employees are committed to the same mission: "Everything for the most pleasant shopping experience". To achieve this, Jumbo works according to the following formula: best service, largest assortment and lowest price. In 1996 Jumbo translated this formula into seven guarantees for its customers: (i) not satisfied? Money back, (ii) your wishes are most important, (iii) for all your groceries, (iv) euros cheaper, (v) fluent shopping, (vi) fresh is really fresh and (vii) service with a smile, which are still lived upon to date.

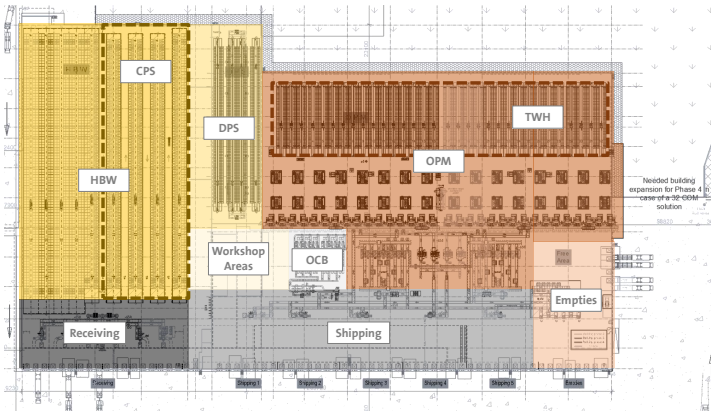


Figure 1.1: Floor map CDC Nieuwegein

1.1.2 Jumbo Nieuwegein

Jumbo Nieuwegein is one of the three CDCs, which handles the slow and medium movers in the category of dry groceries. This means that each Stock Keeping Unit (SKU) handled at the CDC has a shelf life of at least twelve days and at most four complete pallets are redistributed per week. In addition, the CDC in Nieuwegein handles all hazardous goods Jumbo sells. The total number of SKUs currently handled in the CDC is nearly 12000; potentially the CDC could handle up to 16000 SKUs. This CDC was first located in Elst, where every customer order was manually picked. In 2020 Jumbo opened the doors of their new plant in Nieuwegein: a distribution center of 43230 m^2 equipped with a fully automated warehousing system. By opening this plant, Jumbo was one of the first Dutch retailing companies to own such an automated warehouse system. In contrast to the CDC in Elst, customer orders can also be fulfilled fully automated here. The CDC operates six days per week for 24 hours per day; on the seventh day maintenance is performed on the system. In peak weeks, additional operating hours can be made on this seventh day and thus it also provides some slack. With this standard operating scheme, an average weekly volume of more than 1.8 million cases (i.e. one package of an SKU) was achieved in the first 40 weeks of 2022.

The automated warehousing system was built by Witron; one of the worldwide market leaders in the realization of dynamic warehouse- and order picking systems. Witron employees work in-house at Nieuwegein, which operate, repair and maintain the automated warehousing system of the DC. This way, Jumbo employees can focus on all manual activities needed in the remainder of the processes of the CDC. Hence, Jumbo and Witron employees work closely together in Nieuwegein to fulfill all customer orders either with manual or automated order picking. Due to the automation, it is possible to work fast, efficiently, safe and handle more SKUs as compared to the old CDC in Elst. Moreover, the mechanized system is less sensitive to errors (i.e. picking the wrong SKU or in a wrong quantity) than manual order picking.

1.1.3 Distribution Process

This section elaborates on both the manual and mechanical processes conducted at Jumbo Nieuwegein to redistribute SKUs. In addition, it provides insights into the current performance of the site. Figure 1.1 depicts the floor map of the site, demonstrating the location of each of the processes.

The distribution process starts with placing orders for goods by the suppliers, which is done by the headquarter. Those suppliers then ship their goods directly to the CDC on pallets at pre-arranged times. Each pallet is allowed to carry only one particular SKU. The supplier is required to put a barcode on each pallet, containing relevant information such as the total number of cases on the pallet, total number of layers, the dimensions and weight of a case, etc. It is desired to maximize the number of cases on a pallet, as the time required to handle a pallet at inbound is independent of the number of

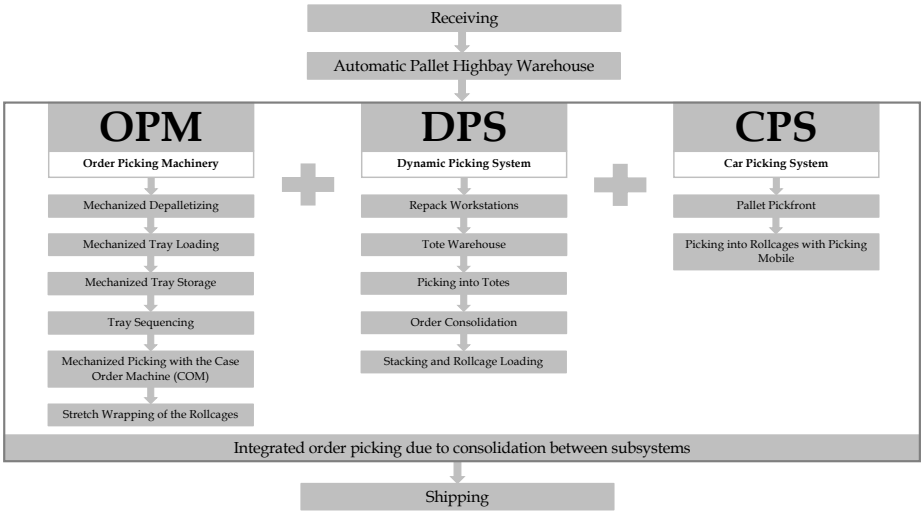


Figure 1.2: Schematic overview of the process flow of SKUs

cases on it. Hence, it is most beneficial to handle as many cases at once as possible. To realize this, it is preferred to order an SKU in full pallets (i.e. with a maximum number of layers set by the supplier). If a full pallet carries more than six weeks of demand (based on historical data) for a particular SKU, then a partial pallet is ordered, containing a certain number of complete layers. In both cases, the SKU is thus ordered in a fixed batch size of either a full pallet or a complete layer. For some SKUs (7.1% of the total number), the demand is so low that even a complete layer corresponds to more than six weeks of demand. For these SKUs, an arbitrary number of cases is ordered instead of a fixed batch size. For all SKUs, the order quantity is chosen such that the forecasted demand can be met till the next (potential) delivery of that SKU. As most SKUs are ordered in batch sizes, the order quantity is set equal to the minimum integer multiple of the batch size to meet this demand.

SKUs that arrive at the CDC all follow the same main processes: it is received, stored, handled and shipped. The steps for receiving, storing and shipping are the same for each SKU, but the way a SKU is handled depends on the type of SKU, as depicted in Figure 1.2. For the receiving process, a supplier's truck driver is assigned to one of the eighteen receiving docks. After truck unloading, the pallets are introduced to the system via one of the six pallet infeed stations. These stations perform height and weight checks, to check if the information of the barcode corresponds to the physical measures. If an infeed station accepts a pallet, an additional Jumbo-owned pallet (with a unique barcode) is placed below the supplier's pallet to ensure high pallet quality and increase stability. Subsequently, the pallets are send to the Highbay Warehouse (HBW) where in total 27637 pallets can be stored. The storage locations can be distinguished in four different heights. Different strategies can be implemented to allocate a certain pallet to a storage location, of which maximizing the used volume (i.e. allocating lower pallets to lower storage locations and higher pallets to higher storage locations) is most often used.

Depending on the demand rate of the SKU, after some time the pallet is collected from the HBW for handling. The CDC receives demand from Jumbo stores and collects all demanded SKUs in roll cages (i.e. the handling process). Each roll cage contains goods for one particular store solely. As Figure 1.2 indicates, the order picking of SKUs can be done via three different subsystems: via the Order Picking Machinery (OPM), the Dynamic Picking System (DPS) and the Car Picking System (CPS). These three subsystems are discussed in more detail in the following subsections.

OPM

In the OPM, nearly all process steps from receiving to shipping are mechanized, as the picking of roll cages is fully automated with the help of Case Order Machinery's (COMs). The handling process starts

Figure 1.3: Example of a sizing matrix for a roll container

with transporting a pallet from the HBW to a stretch wrap removal workstation via a pallet conveyor connection. For each pallet, a pre-defined number of layers are handled at a time, which might be less than the total number of layers on the pallet. An employee is therefore required to only remove the stretch wrapping of the layers that are planned to be processed. After completing this, the pallet is sent to one of the twelve depalletizers (DEPALs). A DEPAL vacuums each layer separately from the pallet and puts all cases of that layer on a conveyer belt. On this belt, tray-loading occurs at the tray merge, where each case is put individually on either a small or large tray. The large tray is twice the size of a small tray and is only used for cases that do not fit on a small tray. If increased stability is required (e.g. for bottles of soda), one of the DEPALs can tilt a case before it is placed on a tray. In addition, two DEPALs can also be used for manual depalletizing in case errors occur due to an incomplete layer, bad stacking of cases, etc.

Each tray has an individual barcode, which is linked to the SKU placed on it at the tray merge. After tray-loading has taken place, each tray is stored in the Tray Warehouse (TWH); in total 533141 small trays can be stored here. The storage racks of the TWH are designed as channels of (generally) four small trays. In each channel, only trays containing the same particular SKU are allowed. Work orders are assigned to the DEPALs based on the current stock in this TWH, which is monitored continuously. If the current inventory level of an SKU drops below a certain reorder point, a work order for this SKU is assigned to the DEPALs. Each work order has a certain priority, based on the timing the SKU is needed to fulfill an order. This means that SKUs demanded on short notice are assigned a higher priority than SKUs demanded at a later point in time. Within the TWH, an optimal allocation principle is used such that (among other things) items with a higher demand rate are located closest to the picking location, to minimize handling transactions and travel distance.

As stated earlier, order picking in roll cages is done per customer order. When an order arrives, the system calculates the best allocation of SKUs over and in the roll cages, based on (i) the store's layout, (ii) the best weight distribution in the roll cage (i.e. no heavy items placed on top of lightweight items) and (iii) the optimal loading and stability of the roll cage. From this calculation, a sizing matrix is derived, such as the example roll cage depicted in Figure 1.3. Based on this calculation and sizing matrix, the trays are collected from the TWH and sent to one of the 31 tray sequencers. Here, as the name suggests, the trays dedicated to a certain roll cage are sequenced such that the items that need to be placed on the bottom arrive first and the items that need to be placed on top arrive last. These sequence buffers enable a sequenced outbound to the COM connected to that tray sequence (i.e. each tray sequence serves one COM). At the COMs, a fully-automated picking process takes place; each COM pushes the cases one-by-one into the designated location in the roll cage. The ready-picked roll cages are removed by one of four transfer cars and transported to one of four stretch wrappers. These transfer cars also replenish the COMs with empty roll cages (empties). After stretch wrapping and labeling, the roll cage is transported to the right shipping section. Figure 1.2 provides a schematic overview of the just described processes.

DPS

The DPS uses an integrated picking and storage system and can especially reach high pick performances with a large number of small volume articles. In the DPS, SKUs are handled which (i) are too small for the OPM, (ii) have packaging that is not suitable for the OPM (e.g. due to its shape) or (iii) have multiple cases packed together, which need manual unpacking before they can be stored separately. Besides, some of the SKUs handled in the DPS could also be handled in the OPM. This offers some flexibility as such SKUs can be moved from one subsystem to the other. This is especially advantageous in peak weeks with high volumes or if the OPM works sub-optimal due to machine errors. Similar to the OPM, the handling process starts with transporting a pallet from the HBW to one of the twelve DPS repack workstations via a pallet conveyor connection. At these workstations, a pre-defined number of layers are manually depalletized; only for these layers the stretch wrapping should be removed by an employee. Then, these layers are manually repacked into multiple empty totes. The replenishment of these empty and the shipping of repacked totes is done fully automatically. After repacking, the totes are transported to the tote warehouse. This warehouse has 77956 storage locations, of which 11956 also serve as picking locations. Order picking thus takes place within the tote warehouse, at 30 workstations on three levels (nearly 400 picking locations per workstation). The picking principle in the DPS is based on a combination of goods-to-man and man-to-goods: orders are manually picked but the allocation of repacked totes to the pick front is done such that the walking distance for the picker is minimized. In addition, fast movers are permanently allocated to a picking location, meaning that if a tote becomes empty, a new tote of the same SKU is replenished to the same or a new picking location. Conversely, slow movers are dynamically allocated, meaning that an empty tote is not necessarily immediately replenished with a new tote of the same SKU. The replenishment of the totes is performed by automated cranes.

At the DPS, the customer orders are collected in totes as well; the system directs an order tote via the workstations where the SKUs needed to fulfill that order are stored. Picking is performed with pick-by-light: if an order tote enters a workstation, the picking locations of all SKUs in the order light up and indicate the number of cases to be picked. The picker turns off the light after picking a particular SKU. As the SKUs allocated to each workstation differ, order totes might cross multiple workstations.

Finished order totes are transported via a conveyor network to one of the six order consolidation buffers (OCB), such that the totes adherent to the same customer order are stored in the same OCB rack. Buffering of finished order totes takes place until an entire roll cage of twelve totes can be palletized. The totes are stacked by a vertical stacker and then mechanically loaded into roll cages. These complete shipping units are labeled and transported to the right shipping section. In case a customer order does not comprise a multiple of twelve totes (i.e. not all roll cages will be complete), the remaining totes are transported to the OPM for further consolidation with other SKUs. These totes then appear at the bottom of a roll cage such that other cases can be placed on top of it by the COMs or an order picker. In Figure 1.2 a schematic overview of the just described processes is depicted.

CPS

In the CPS, so-called 'uglies' are handled: all hazardous goods and all SKUs which are not suitable for OPM, but are too large to effectively be handled in the DPS. SKUs are classified as not suitable for OPM if (i) the packaging material is weak (i.e. quickly opens or breaks), (ii) the packaging is too large to be effectively handled by the OPM, (iii) the packaging is not suitable for the OPM (e.g. due to its shape) or (iv) high associated cleaning costs and time if a case breaks (e.g. for oil products). Similar to the DPS, the CPS is an integrated picking and storage system and is located within the HBW. In Figure 1.4 a graphical representation of the CPS is depicted, showing that the pick fronts are located below the storage locations. This way, the SKUs located at pick front locations are easily accessible by order pickers. The CPS contains 1103 picking locations, distributed over five aisles. Replenishment of the pick front locations is done automatically via cranes.

The picking principle of the CPS is based on a man-to-goods principle: an order picker visits the picking locations and manually collects each order in roll cages. Order pickers drive through the aisles

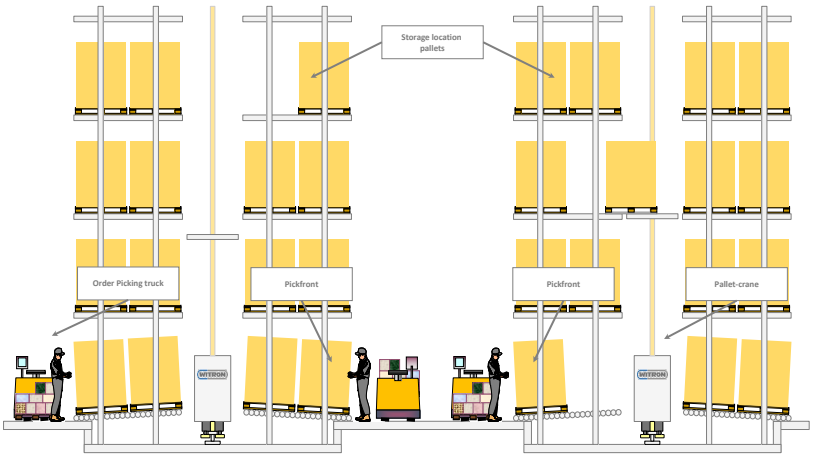


Figure 1.4: Graphic representation of the highbay warehouse and the Car Picking System

on order picking trucks which can carry up to four different roll cages, allowing for parallel order picking. Similar to the OPM and the DPS, each roll cage can only contain cases devoted to one particular store. Hence, an order picker can simultaneously collect orders for at most four different stores. For each work order (comprising one to four roll cages), the order picker follows the same route through the CPS. The SKUs are allocated along this route such that an order picker first passes the hazardous goods and heavy items, which are best placed at the bottom of a roll cage. The pickers carry a pick-by-voice technology, which navigates them to the right picking locations and indicates the number of cases to be picked on which roll cage. After collecting all cases for all roll cages, the order picker manually labels each finished roll cage and transports it to the right shipping section.

At the shipping section, all roll cages devoted to a specific store are stored together at one of the 62 shipping lanes. When the complete store order is collected, the roll cages are redistributed by trucks to either (i) the store directly or (ii) an RDC, depending on the size of a store's order and its location. At an RDC, the roll cages are merged with roll cages collected in the RDC and the combined order is subsequently shipped from the RDC to the store. Hence, the RDC serves as a cross-dock for the CDC roll cages. After shipping the roll cages, the process of redistributing SKUs is completed.

Current Performance

This subsection elaborates on some current figures of the CDC. First, Figure 1.5 shows the weekly outbound volumes (measured in the number of cases redistributed) of the CDC for 40 weeks. On average, more than 1.8 million cases were redistributed each week. Besides the total volume of outbound cases, the figure also highlights by which subsystem the cases were collected. The figure indicates that more than half of all volumes were collected via the OPM and only a small percentage was handled via the CPS. This closely corresponds to the (average) percentage of SKUs that are currently handled by each subsystem, being 11.7%, 31.5% and 56.7% for the CPS, DPS and OPM, respectively.

In Figure 1.6, the utilization rate of each storage location is depicted for the first 40 weeks of 2022. From the figure can be derived that especially the utilization rate of the CPS is remarkably high, as it is nearly equal to the upper limit of 100% in all weeks. In addition, it shows that the utilization of the channels in the TWH used to be close to the maximum value as well, but decreased significantly due to expansion of capacity of the TWH in week 20. The utilization rate of all storage locations in the TWH is the lowest for all periods: on average only 64.2% of all individual storage locations were used. The discrepancy between the channel and location utilization in the TWH is explained by the fact that in each channel (equalling four storage locations in general) only one particular SKU is allowed. Hence, while many channels are occupied, this does not necessarily imply that all locations in each channel are

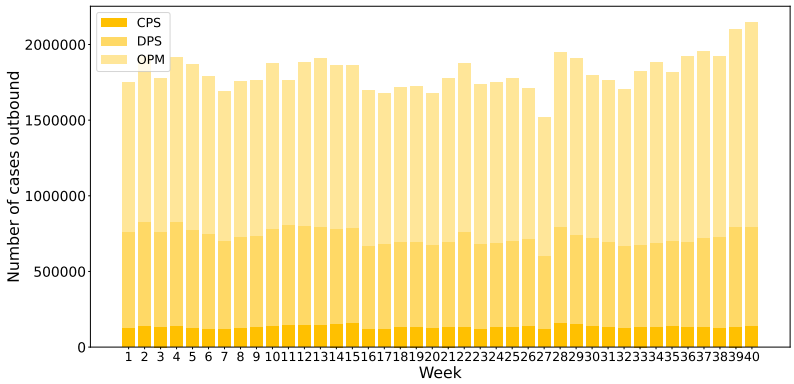


Figure 1.5: The inbound and outbound volumes of CDC Nieuwegein for 40 weeks

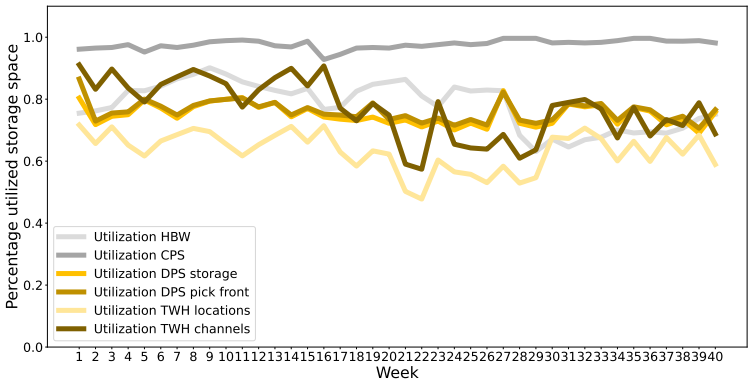


Figure 1.6: The utilization of different storage locations in CDC Nieuwegein for 40 weeks

occupied. Consequently, the utilization of the channels will usually be higher than the utilization of all storage locations. The utilization rates of the DPS storage and picking locations are nearly equal, which is a sensible effect. As more SKUs or more cases of SKUs are transferred to the DPS, it makes sense that they are distributed accordingly over both the picking and storage locations.

1.2 Problem Motivation

As stated in Section 1.1.2, the CDC realized an average weekly volume of more than 1.8 million cases. By achieving such volumes, in general the demand of stores are met. Even though the system works well in general, it still shows some flaws. For example, during certain steps in the process, a lot of cases fall due to either manual or machine errors. In 2021, the total amount of money lost due to cases falling was equal to €131559, as these cases broke and could not be sold anymore. Converting the loss in Euro to cases, this means that each week more than 200 cases break due to falling (using the fact that on average a case costs Jumbo €12.50). This amounts to approximately only 0.01% of the average weekly volume, but in absolute terms it is quite a substantial amount. In fact, besides those broken cases, there are also situations where cases fall but can still be sold. Even though the costs are not lost for these cases, additional registration is necessary to introduce such cases back into the system, requiring additional working hours and thereby yielding indirect costs. Unfortunately, as no proper registration is yet present for such scenarios, it is unknown how many cases did exactly fall in 2021.

Both broken and non-broken cases need to be manually removed from multiple locations in the

warehouse. This requires a lot of additional cleaning time and costs, especially if the broken cases consider items that are hard to clean properly (e.g. oil or rice). Moreover, the proper cleaning of some machines might take several hours. In fact, in the initial design of the warehouse, each day ten regular Jumbo employees (i.e. not specialized in cleaning) were scheduled for cleaning purposes. In the mean-time, this number has increased to 26 employees cleaning the warehouse each day, all hired from a certified cleaning company. Still, one of the main tasks of these employees is cleaning the consequences of broken cases. Hence, part of the additionally incurred labor costs can be appointed to the broken cases.

In addition to the increase in cleaning time and costs, SKUs potentially harm the machines when they break. This might result in machine errors or even machine downtime, which adversely affects the productivity of the CDC and consequently yields additional costs. On top of that, as the problem comprises (among other things) food items, broken cases might attract vermin. This brings even more disadvantages, as vermin might also damage the machinery and is very hard to extrude once present in the warehouse. Moreover, to meet the International Featured Standards to ensure food safety, the probability of vermin should be minimized.

Not only does the problem of broken or fallen cases increase the costs, it also worsens the service quality of the CDC. If a case falls closer to the end of the process, the problem might not be rectifiable (i.e. the missing case cannot immediately be replaced by a new case of the same SKU). In such situations, stores do not receive ordered SKUs on the desired day, which adversely affects the service quality of the CDC.

If a case breaks due to falling, Jumbo Nieuwegein has certain protocols that require proper registration. Because of such registrations, Jumbo knows that in 2021 €131559 was lost due to cases breaking. However, in reality it appears that these protocols are not always performed (properly). If a broken case is not registered, the differences in stock can never be translated back to its direct cause. Moreover, if a case falls but does not break, the case is introduced back into the system without any further registration. Hence, the total number of cases falling is known to be larger than 200 cases per week, but no information is available on the exact number. As it is more or less based on luck whether a case breaks or not after falling, it is important to limit the total number of cases falling, including both the broken and non-broken cases. This way, the direct cause of the problem is tackled.

The above-mentioned problem can be summarized in the following problem statement:

Currently, more than 200 *cases fall every week in Jumbo's fully automated warehouse located in Nieuwegein. It is yet unknown what the exact size of the problem is, as currently no proper registration of fallen cases is performed. Fallen cases adversely influence cleaning, product and employee costs, machine productivity and service quality. Hence, it is desired to minimize the number of cases that fall.*

1.2.1 Research Questions

This section introduces several research questions to tackle the problem as introduced in Section 1.2, which will guide the research. The main research question can be defined as:

How to minimize the total number of cases falling during the process?

This question can be further decomposed into three sub-questions. First, to properly answer the main research question, it is important to gather information about the current (as-is) situation. By gauging the size of the problem, a starting point is defined, from whereon improvements can be implemented to reduce the problem. Hence, the first sub-question is defined as:

1. What is the current size of the problem? Which particular SKUs fall, at which step in the process and in what quantities?

When it is known which SKUs fell in certain quantities in the past, this data can be used to build a prediction model to predict the likelihood of falling for cases handled in the future. From this prediction model, potentially SKU- (e.g. packaging material), process- (e.g. number of calls at DEPAL) or machine (e.g. speed of conveyer belt) related characteristics that caused a certain case falling can be derived. As such, the second sub-question is defined as:

2. How can the likelihood of falling be predicted? What SKU-, process- and machine characteristics influence this likelihood?

The relevant characteristics obtained by answering this sub-question provide the main directions of improvement strategies. Certain SKU- and machine characteristics can potentially be adjusted directly to minimize the number of cases falling, while for other characteristics this is less straightforward. Especially for process-related characteristics, minimizing the number of cases falling might conflict with other objectives (e.g. maximize utilization or minimize costs),which should be kept in mind as well.

One of the important decisions made in the process is the number of layers to be depalletized at a DEPAL or a repack station, as this balances the inventory between (i) the HBW and (ii) the TWH or the tote warehouse. Moreover, if a pallet is not depalletized at once, it makes more movements through the warehouse, leading to more opportunities to drop cases. On top of that, the stability of the pallet decreases after partial depalletization, as some of its stretch wrapping was removed. Therefore, the decision to partially depalletize is believed to be among the factors that adversely affects the number of fallen cases. As the goal is to minimize the number of fallen cases, the third and final sub-question is defined as:

3. How can the number of fallen cases be minimized by influencing the number of layers to be depalletized at once?

1.2.2 Scope

To ensure the feasibility of the project plan given the available resources (e.g. time horizon) and to properly address the research problem, a project scope is defined in this section.

Subsystems

As indicated in Section 1.1.3, the CDC uses three subsystems in handling the SKUs. Picking orders in the CPS and DPS is performed manually, while this is done automatically in the OPM. Hence, when a case falls in the CPS or DPS, this is often due to human errors. In contrast, in the OPM usually no human interaction is required, such that the problem is caused by machine-related errors. Reducing the number of human errors is probably best solved by better training of employees, rather than by analyzing the data and building a prediction model. Therefore, this thesis will focus on minimizing the number of cases falling in the OPM subsystem solely. Consequently, all research questions will be answered for OPM-related processes only.

SKUs

Approximately 56.7% of all SKUs handled by CDC Nieuwegein are currently handled by the OPM, as stated in Section 1.1.3. All of them will be taken into consideration when analyzing the problem.

Processes

To gauge the current size of the problem, physical input and output data are necessary. Inventory checks are performed at the infeed stations, at the tray merge and just before a case enters one of the COMS. If a case passes this last check, it is assumed that the ordered case reaches its dedicated store. Hence, if a case falls after this point, this is not registered by the system. To ensure complete and reliable data, only processes up to this point will be taken into account.

Based on the measurement points, two processes can be distinguished for which input and output data is available: (i) from infeed stations till tray merge and (ii) from tray merge till COMs. Some early analyses for these processes showed that the problem of fallen cases is much greater in the first process than in the second. Therefore, the focus will be on minimizing the number of fallen cases between the infeed stations and the tray merge. The process between tray merge and the COMs will be disregarded in the remainder of this thesis.

In addition, Research Question 3 reflects one decision that adversely affects the number of fallen cases. However, it should be noted that other decisions in the process could have this effect as well (e.g. placement of a case on a roll container or whether the case is tilted or not before put on a tray at tray merge). Such decisions are not addressed in this question and will be disregarded in this thesis.

1.3 Outline

The remainder of this thesis is structured as follows: Chapter 2 discusses relevant literature for answering the research questions. Subsequently, Chapter 3 discusses how relevant data was collected and analysed to gauge the current size of the problem. Then, this data was used to build the prediction models presented in Chapter 4, to be able to predict whether cases will fall off a pallet. Chapter 5 outlines a method to minimize the number of fallen cases by influencing the number of layers to be depalletized at once. Finally, Chapter 6 contains the conclusions, recommendations, limitations and suggestions of future work.

Chapter 2

Theoretical Background

In the previous chapter, CDC Nieuwegein and its distribution processes were introduced and the problem context was discussed. The focus of this chapter is to gain insights into the literature that is relevant in minimizing the number of fallen cases. First, in Section 2.1 the literature devoted to automated warehousing systems is discussed. Then, in Section 2.2, several Machine Learning techniques are provided that can be used to predict whether cases will fall off a pallet. Section 2.3 highlights several Inventory Management Systems, from which an inventory policy can be derived that optimizes the number layers to be depalletized at once. Finally, Section 2.4 discusses the research gap of this thesis.

2.1 Automated Warehousing Systems

The automation of warehouses dates back to 1960 and has known a lot of growth since then. Over the years, more and more technologies aroused to automate several steps in the process and eventually Witron was the first to combine multiple technologies to fully automate the entire process (Azadeh et al., 2019). Due to all these innovations and increased interest, a lot of literature has been devoted to automated warehousing systems. Several review papers summarize parts of this literature. For instance, Boysen et al. (2021) summarizes automated warehousing systems applicable for brick-and-mortar retail chains, whereas Boysen et al. (2019) does the same for e-commerce retailers. Both review papers provide insightful information on the types of available automated warehousing systems, how they operate and in which situation they are most suited. Similarly, Azadeh et al. (2019) review the relatively new developments in the automated warehousing systems literature, thereby indicating directions for future research to further optimize vital warehouse decisions (such as design layout, order batching, picker routing, etc.) with the introduction of a new system.

Besides such review papers, many studies focus on the optimization or improved performance of certain parts of an automated warehouse. Generally speaking, most studies are concerned with:

(i) Optimizing the warehouse design (e.g. Küçükyaşar et al. (2021); Tompkins et al. (2010))
(ii) Optimizing the allocation of SKUs to storage locations (e.g. Yang et al. (2021); Mirzaei et al. (2021))
(iii) Optimizing the order picking performance (e.g. Bertolini et al. (2019); Claeys et al. (2016); Andriansyah et al. (2014); Ko & Han (2022); Boysen et al. (2018))
(iv) Optimization of crane or automated vehicle movements (e.g. Amato et al. (2005); Li et al. (2022); Emde et al. (2021))
(v) Improved performance measurement (e.g. Faveto et al. (2021))

All the optimization policies disregard the fact that inventory losses can occur due to machine malfunctioning (i.e. by dropping cases). Hence, the problem at hand is not discussed in literature yet. Therefore, current literature on automated warehousing systems can be enriched by developing methods to minimize the number of fallen cases. The next sections discuss several concepts that are relevant in developing such a method.

2.2 Machine Learning

The goal of the second sub-question is to predict the likelihood of falling for cases of a certain SKU. Such a prediction model can be built by using Machine Learning (ML) techniques. Therefore, this section elaborates on several ML techniques applicable to the problem under consideration.

ML techniques are tools to better understand a given dataset, by finding generalizable patterns in the data (Nasteski, 2017). The goal of ML is to accurately predict an outcome y for an observation, based on a certain function f of its input features x, i.e. $y = f(x)$ (Athey & Imbens, 2019). In ML, algorithms are built to create good estimates of this function f, without being concerned with finding the true model or estimating the right parameters of this function (James et al., 2013). In contrast, the focus is solely on finding the most accurate out-of-sample predictions (Athey & Imbens, 2019). The main strength of ML techniques is that they can fit flexible and complex functions on a wide variety of data structures (Mullainathan & Spiess, 2017).

Usually, the best estimate of f is obtained in three subsequent steps. First, a set of training (in-sample) data is used as input to build the model and estimate the function f: the training process. Subsequently, the model is validated using a different (validation) dataset, to select the best model and choose the best hyperparameters. Finally, the model is applied to test (out-of-sample) data to evaluate the model's performance (Varian, 2014).

In supervised ML problems, the training, validation and test data used to estimate and evaluate f contains (x, y) pairs; for each observation in the data the outcome is known (Jordan & Mitchell, 2015). In contrast, in unsupervised ML problems, the data only contains information about the input features x, which are grouped into clusters to estimate the best function f. If the outcome feature to be predicted is real-valued, one speaks of regression models; if the goal is to assign it to a pre-defined class, one speaks of classification models (Nasteski, 2017).

The problem at hand as introduced in Section 1.2 can be classified as a binary classification problem, as the goal is to predict whether or not cases will fall from a pallet of a given SKU. Moreover, the first goal of this thesis is to collect data to gauge the size of the problem. That is, the data will contain observations that indicate whether cases fell from a certain pallet and in what quantities. Consequently, supervised ML techniques can be applied. Current estimates suggest that at least 0.01% of Jumbo's weekly volume in CDC Nieuwegein falls, indicating that the outcome feature is extremely imbalanced. To accurately predict an imbalanced outcome feature, standard ML techniques might need to be modified (Chawla et al., 2004). The following subsections discuss two standard supervised ML techniques to predict the outcome feature, which were both implemented to accurately predict the binary classification problem. Moreover, in subsection 2.2.3, several adjustments to these ML techniques are suggested, such that they can better handle imbalanced data.

2.2.1 Random Forests

The first supervised ML technique is very commonly used and is called random forests (RF), which is an extension of the single regression tree algorithm (Breiman, 2001). RF appears to be very effective in ignoring irrelevant features, require relatively little tuning, have a high out-of-sample performance and work especially well on highly nonlinear data (Athey & Imbens, 2019; Varian, 2014). The idea of RF is that it averages over a large number of single regression trees. In a single regression tree, the training sample is split into distinct regions, such that every observation that falls in a certain region receives the same prediction, based on a majority vote (Oshiro et al., 2012). Every split in the tree is based on a single feature exceeding a certain threshold value (James et al., 2013). At each split, the feature and threshold value are selected that improve the split criterion the most (Hastie et al., 2009). Two commonly used split criterions are discussed in Appendix A.

The main problem with using a single decision tree solely is that decision trees have the tendency of overfitting the training data, limiting generalization to new datasets (Varian, 2014; Park & Lek, 2016).

As the goal is to find accurate out-of-sample predictions, it is important to find a model that works well out-of-sample and does not overfit the in-sample data (Mullainathan & Spiess, 2017). Averaging over a large number of single regression trees helps in countering this effect.

In RF, each single regression tree differs from each other due to two things: (i) the tree is grown on a bootstrap sample of the original data (choosing, with or without replacement, a sample of size n (Breiman, 1996)) and (ii) at each split made in the tree, only a random subset m of all features could be chosen, which changes at every split (Breiman, 2001). Each tree is grown independently until a forest of T trees is obtained. The final prediction $\hat{y}_i$ for each observation i equals the majority vote over all trees, which is obtained as follows (Probst & Boulesteix, 2017):

Let $\hat{y}_{i,t}$ be the predicted value for observation i for tree $t = 1, \ldots, T$. Averaging over the predictions of all trees yields a probability:

$$\hat{p}_i = \frac{1}{T} \sum_{t=1}^{T} I(\hat{y}_{i,t} = 1), \tag{2.1}$$

where $I(\hat{y}_{i,t} = 1)$ is the indicator function which takes value 1 if $\hat{y}_{i,t} = 1$ and 0 otherwise. The final prediction for observation i is based on the majority vote, defined as:

$$\hat{y}_i = \begin{cases} 1 & \text{if } \hat{p}_i > 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

To assess the performance of an RF, the predicted outcomes $\hat{y}_i$ are compared to the actual outcomes y_i. In section 2.2.3, several relevant performance measures are highlighted.

RF significantly improves the out-of-sample predictive power as compared to a single decision tree. However, this increase comes at the expense of interpretability. As the RF is an average of many trees, it is impossible to make a graphical representation of the forest and easily interpret its results. Still, an RF can be interpreted utilizing the feature importance. As stated, at each split in a tree the feature and threshold value are selected that improve the split criterion the most (Hastie et al., 2009). The size of the split criterion improvement is attributed to this particular feature. A feature's importance is then defined as the sum of the split criterion improvements over all trees. Clearly, the higher the importance, the more important a feature is in predicting the outcome feature. In that sense, the feature importance can be used to perform feature selection, as it reveals the subset of important features to predict the outcome feature (Varian, 2014).

To prevent the model from overfitting and to find the most accurate out-of-sample predictions, some hyperparameter tuning is required. An RF requires relatively little tuning as compared to other ML techniques, but still a few hyperparameters need to be set by the user. Several important hyperparameters are highlighted in Appendix A.

2.2.2 Multilayer Perceptron

The second type of ML that is applied is called a multilayer perceptron (MLP), which is a feed-forward artificial neural network (ANN) (Kavzoglu & Mather, 2003). ANNs can automatically learn underlying rules from a given set of examples (Jain et al., 1996) and proved to work very well for representing highly nonlinear relationships between a large number of input features and the outcome feature (Qi et al., 2019; Abraham, 2005). However, the technique requires more tuning of the parameters of the model relative to other ML techniques such as RFs (Athey & Imbens, 2019).

An MLP contains three kinds of layers, all consisting of a certain number of neurons: an input layer, an output layer and several so-called hidden layers between these two (Choi et al., 2020). Figure 2.1 illustrates an MLP with one hidden layer, with three, four and two neurons in the input, hidden and output layers, respectively. In feed-forward ANNs, the signal flow is strictly fed forward from input to output neurons: no recursive feedback connections are present (Abraham, 2005). In an MLP, all neurons in neighboring layers are connected via weights (i.e. a fully connected network), which represent the importance of an input to an output (Nielsen, 2015). These weights are free parameters of the

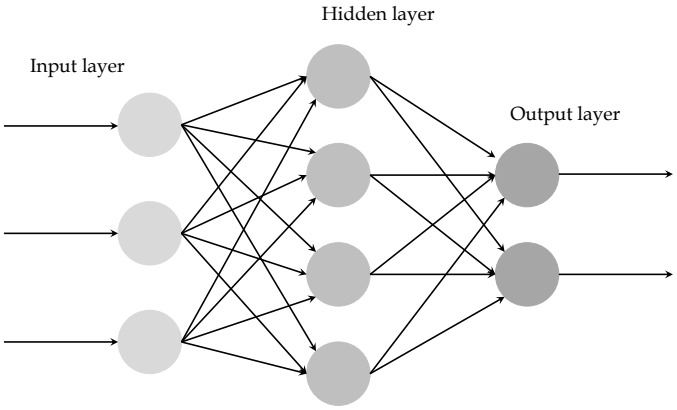


Figure 2.1: Example of a multilayer perceptron (Abraham, 2005)

model, which is trained such that the out-of-sample performance is maximized (Qi et al., 2019). The number of neurons in the input layer usually corresponds to the number of input features p of the dataset (Park & Lek, 2016). The number of hidden layers and the number of neurons in each hidden layer are hyperparameters of the model that require tuning (Abraham, 2005). The neurons in the output layer correspond to the classes of the problem; a binary classification problem has two output neurons (Kavzoglu & Mather, 2003).

Modelling the neurons

The neurons in an MLP are often modeled as perceptrons or sigmoid neurons. A perceptron has multiple input neurons (x_j for all input neurons j) and all yield a single binary output (Nielsen, 2015). The output of a single perceptron is calculated as the weighted sum of the input values $\sum_j w_j x_j$ (or $\mathbf{w} \cdot \mathbf{x}$ in vector terms) plus an overall bias term b (Abraham, 2005). Its final output is modified by an activation function, being the step-function for perceptrons:

$$\text{output perceptron} = \begin{cases} 0 & \text{if } \mathbf{w} \cdot \mathbf{x} + b \leq 0 \\ 1 & \text{otherwise} \end{cases} \tag{2.3}$$

That is, if the output is below a certain threshold $-b$, the observation is classified as 0 and if it exceeds the threshold it is classified as 1 (Nielsen, 2015). As all outputs are binary, the perceptrons in the hidden layer can be interpreted as partial classifications. The perceptrons in the output layer subsequently combine all partial classifications from the hidden layer (by applying new weights and biases) to find a final classification for an observation (Krogh, 2008).

The problem with perceptrons is that a small change in the weights or bias terms (being the free parameters of the model) can cause a major change in the behavior of the entire network, where the prediction for many observations is flipped from 0 to 1 or vice versa (Nielsen, 2015). This results in an unstable network, which is an undesirable effect. Moreover, the step function shown in Equation (2.3) is discontinuous and thus not differentiable, which turns out to be a problem when the model is trained (Jain et al., 1996). To overcome this problem, a different activation function, being continuous and differentiable, can be applied. The most commonly used activation function for MLPs is the sigmoid function, defined as (Yonaba et al., 2010):

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.4}$$

Instead of a binary output, the output of the sigmoid function is allowed to take on any real value between 0 and 1, where the extreme values can only be reached asymptotically (Rojas, 1996). To find the output of a sigmoid neuron, $z = \mathbf{w} \cdot \mathbf{x} + b$ is inserted in the sigmoid function, as illustrated in Equation (2.5).

Due to the smoothness of the sigmoid function, small changes in the weights and bias terms of neurons only cause small changes in the output values, resulting in a more stable network (Nielsen, 2015).

$$\text{output sigmoid neuron} = \frac{1}{1 + \exp(-\mathbf{w} \cdot \mathbf{x} - b)} \tag{2.5}$$

As indicated earlier, a binary classification problem has two output neurons. Using Equation (2.5), each of these neurons yields an output between 0 and 1. These outputs can be captured in a two-dimensional vector $\hat{\mathbf{a}}_i$, representing the predicted values for observation i. The final classification given to this observation is based on the highest element in this vector (Nielsen, 2015). The performance of the MLP can be assessed by comparing the predicted output $\hat{\mathbf{a}}_i$ with the actual output y_i. The actual output can also be represented with a two-dimensional vector, where $\mathbf{y}_i = (1, 0)^T$ and $\mathbf{y}_i = (0, 1)^T$ represent a negative and positive sample, respectively.

The goal is to find a model that accurately predicts this outcome feature y. Maximizing the model's performance requires finding the optimal combination of parameters of the MLP (being the weights and biases), which together best approximate the true function of y (Rojas, 1996). To find the optimal values, training of the MLP is required, by iteratively presenting new observations with known classifications. Then, for each observation, the MLP's output $\hat{\mathbf{a}}_i$ is compared with the actual output $\mathbf{y}_i$ and the weights and biases are updated such that the performance of the model is improved (Krogh, 2008). This process is known as the backpropagation algorithm (Rumelhart et al., 1986).

Backpropagation algorithm

First, let w^l_{kj} denote the weight from the j^{th} neuron in layer $l-1$ to the k^{th} neuron in layer l, such that $\mathbf{W}^l$ denotes the $k \times j$ weight matrix of layer l containing all these weights. In addition, let b^l_k denote the bias of the k^{th} neuron in layer l, such that $\mathbf{b}^l$ denotes the k-dimensional bias vector of layer l. Note that l takes on values between 2 (the first hidden layer) and L (the output layer). As indicated, the individual weights w^l_{kj} and biases b^l_k are the free parameters of the model and should be optimized by the model to accurately predict the outcome feature y (Rojas, 1996).

The backpropagation algorithm starts with randomly initialized weights w^l_{kj} and biases b^l_k, such that it yields a first output for all observations fed into the model (Rojas, 1996). Based on the initial weights and biases, the output neurons in the model yield a first prediction for each observation i. To assess the performance of the model, the values of the output neurons $\hat{\mathbf{a}}_i$ are compared to the actual outputs $\mathbf{y}_i$ by applying a cost function. The mean squared error (MSE)[1] is a commonly used cost function, which is defined as the sum of the squared difference of predicted and actual outputs (Nielsen, 2015):

$$C(\mathbf{x}_i, \mathbf{W}^l, \mathbf{b}^l) = \frac{1}{2n} \sum_{i=1}^{n} ||\mathbf{y}_i - \hat{\mathbf{a}}_i||^2 \tag{2.6}$$

where n represents the total number of training samples and $||\mathbf{v}||$ represents the norm of the vector $\mathbf{v}$. Note that all predicted outputs $\hat{\mathbf{a}}_i$ are dependent on the input features of an observation $\mathbf{x}_i$, the weight matrix $\mathbf{W}^l$ and the bias vector $\mathbf{b}^l$ of each layer $l = 2, \ldots, L$. Clearly, the smaller the MSE, the better the performance of the model (Krogh, 2008). Hence, the model should set the weights and biases such that this cost function is minimized.

To reach a (global or local) minimum of the cost function, the gradient descent rule is applied. That is, the partial derivative of the cost function with respect to each weight in $\mathbf{W}^l$ and each bias in $\mathbf{b}^l$ is calculated, to obtain the gradient vector:

$$\nabla C(\mathbf{W}^l, \mathbf{b}^l) = \left(\frac{\partial C(\mathbf{x}_i, \mathbf{W}^l, \mathbf{b}^l)}{\partial \mathbf{W}^l}, \frac{\partial C(\mathbf{x}_i, \mathbf{W}^l, \mathbf{b}^l)}{\partial \mathbf{b}^l} \right)^T \tag{2.7}$$

[1] Note that the MSE is a function of the weight matrix $\mathbf{W}^l$ and bias vector $\mathbf{b}^l$ of each hidden and output layer $l = 2, \ldots, L$.

A change in the cost function $\Delta C(\mathbf{x}_i, \mathbf{W}^l, \mathbf{b}^l)$ is defined as the product of the gradient vector $\nabla C(\mathbf{W}^l, \mathbf{b}^l)$ and the vector of (small) changes in weights and biases $(\Delta \mathbf{W}^l, \Delta \mathbf{b}^l)^T$:

$$\Delta C(\mathbf{x}_i, \mathbf{W}^l, \mathbf{b}^l) \approx \nabla C(\mathbf{W}^l, \mathbf{b}^l) \cdot (\Delta \mathbf{W}^l, \Delta \mathbf{b}^l)^T \tag{2.8}$$

Now, the small changes in weights $\Delta \mathbf{W}^l$ and biases $\Delta \mathbf{b}^l$ should be set such that the change in the cost function is negative (i.e. $\Delta C(\mathbf{x}_i, \mathbf{W}^l, \mathbf{b}^l) \leq 0$, causing a decrease in $C(\mathbf{x}_i, \mathbf{W}^l, \mathbf{b}^l)$). To ensure this, the changes in weights and biases can be set as follows (Nielsen, 2015):

$$(\Delta \mathbf{W}^l, \Delta \mathbf{b}^l)^T = -\eta \nabla C(\mathbf{W}^l, \mathbf{b}^l) \tag{2.9}$$

where η represents the learning rate, being a positive (hyper)parameter of the model. The learning rate determines the step size of the adjustments in weights and biases, such that the model moves towards the global minimum of the cost function (Kavzoglu & Mather, 2003). Inserting Equation 2.9 in Equation (2.8), gives $\Delta C(\mathbf{x}_i, \mathbf{W}^l, \mathbf{b}^l) \approx -\eta ||\nabla C(\mathbf{W}^l, \mathbf{b}^l)||^2$, which indeed ensures that $\Delta C(\mathbf{x}_i, \mathbf{W}^l, \mathbf{b}^l) \leq 0$ as both η and $||\nabla C(\mathbf{W}^l, \mathbf{b}^l)||^2$ are positive terms.

After adjusting the weights and biases using Equation (2.9), the model calculates a new prediction for each observation i. Then, the performance of the model is evaluated again by inserting these new predictions in the cost function, yielding a new gradient vector. Based on the new gradient, the weights and biases are adjusted again to reach a minimum of the cost function. This process of iteratively evaluating the performance of the model and adjusting the weights and biases is repeated until a certain stop criterion is reached (Kavzoglu & Mather, 2003).

As for an RF, hyperparameter tuning for an MLP is required to prevent the model from overfitting and to find the most accurate out-of-sample predictions. The most important hyperparameters for an MLP are highlighted in Appendix A.

2.2.3 Imbalanced data

Imbalanced data refers to a dataset where the observations are disproportionately distributed over the classes, such that, in a binary classification setting, you have a clear majority and minority class (Yijing et al., 2016). This problem has many applications, such as fraud detection, risk management and medical diagnosis (Chawla et al., 2004), but also applies to Jumbo's problem of cases falling. The problem with standard ML techniques is that the goal is to maximize the predictive accuracy, being the number of correctly classified observations among all observations (Provost, 2000; Kaur et al., 2019). However, applying such techniques to an imbalanced dataset will very likely result in a model that classifies all observations in the majority class (Loyola-González et al., 2016). Consider for example the case at Jumbo, where only 0.01% of the observations are known to be in the minority class. A model that guesses the majority class for all observations yields a predictive accuracy of 99.99%. However, all minority observations were misclassified, while these observations have high associated costs. Therefore, a model is required that has a fairly high accuracy in classifying the minority class.

A model is selected and evaluated based on certain performance measures. Binary classification problems can be evaluated by using a confusion matrix, as depicted in Table 2.1 (Chawla et al., 2002). In the remainder of this thesis, the Positives refer to the minority class and the Negatives refer to the majority class and these terms are used interchangeably. The TN (TP) indicate that the actual negative (positive) observations are also predicted as negative (positive). On the other hand, the FN (FP) indicate that actually positive (negative) observations are misclassified as negative (positive).

Using the confusion matrix, several performance measures can be calculated. As indicated, predictive accuracy might not be the best performance measure in imbalanced data settings, such that models need to be selected based on other measures. In Equations (2.10) till (2.13), several other frequently used performance measures are presented (Kaur et al., 2019). The TPR (also known as sensitivity or recall) shows the fraction of positive observations that are correctly classified by the model. Similarly, the TNR (also known as specificity) shows the fraction of negative observations that are correctly classified by the model. As indicated, in imbalanced settings the goal is to find a model that

Table 2.1: Confusion Matrix (Chawla et al., 2002)

	Predicted Negative	Predicted Positive
Actual Negative	True Negatives (TN)	False Positives (FP)
Actual Positive	False Negatives (FN)	True Positives (TP)

accurately predicts the minority class, i.e. a model with a high TPR. To achieve this, a slightly lower TNR is accepted as misclassifying majority observations generally results in lower associated costs (Chawla et al., 2002).

In a Receiver Operating Curve (ROC), the trade-offs between accuracy on positive observations and errors on negative observations are depicted. The area under the ROC curve (AUC) indicates whether the model can discriminate well between the two classes: a larger AUC indicates a better discrimination (Probst & Boulesteix, 2017). In Equation (2.13), FRP represents the False Positive Rate, being the fraction of negative observations that are misclassified by the model (Kaur et al., 2019).

$$\text{Predictive Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.10}$$

$$\text{True Positive Rate (TPR)} = \frac{TP}{TP + FN} \tag{2.11}$$

$$\text{True Negative Rate (TNR)} = \frac{TN}{TN + FP} \tag{2.12}$$

$$\text{AUC} = \frac{1 + TPR - FPR}{2} \tag{2.13}$$

As stated, many standard ML techniques try to maximize the total accuracy of the model. To improve the other performance measures to accurately classify the minority class, the ML approaches need some modifications. Two commonly used approaches are based on (i) Resampling and (ii) Cost-Sensitive Learning (CSL) (Chawla et al., 2004).

Applying a resampling method is part of the data preparation and thus independent of the ML technique (López et al., 2013). These methods rebalance a given sample before it enters the model, to diminish the skewed distribution over the two classes (Haixiang et al., 2017). Three kinds of resampling methods can be applied:

(i) Over-sampling methods: add new minority class samples to the dataset. The two most commonly used techniques are to randomly duplicate existing minority class samples (i.e. sample with replacement) and the Synthetic Minority Oversampling Technique (SMOTE). The latter approach introduces *"synthetic examples along the line segment joining any/all of the k minority class nearest neighbors"* (Chawla et al., 2002, p.328). That is, new synthetic samples are generated that are closely related, but not necessarily equal, to already existing samples in the minority class.

(ii) Under-sampling methods: remove majority class samples from the dataset. The most effective method appears to be random undersampling, which randomly selects the majority class samples to be eliminated (Tahir et al., 2009).

(iii) Hybrid methods: a combination of an over- and under-sampling method (Haixiang et al., 2017).

All of the above resampling techniques can resample the data to any desired ratio; which ratio works best is dependent on the data and the model (Haixiang et al., 2017; L. Zhou, 2013). Consequently, both the resampling method and the ratio of resampling can be considered a hyperparameter of the implemented model. If the dataset contains hundreds of minority observations, undersampling the data is preferred as it reduces the computational time significantly. Otherwise, SMOTE appeared to work better than random oversampling (Napierala & Stefanowski, 2016; Loyola-González et al., 2016). This method is also better able to ignore the noise in the data.

Instead of using a resampling method as a preparation technique, CSL adjusts the cost function of a ML algorithm. A cost term is added, such that the model is penalized more for misclassifying a

minority class observation than for misclassifying a majority class observation (Chawla et al., 2004). That is, a misclassification cost term c_k is added to the split criterion function for RF (Equation (A.2) or (A.3)) and the cost function for an MLP (as a two-dimensional vector in Equation (2.6) or as term in Equation (A.6)), with $c_1 > c_0$.

Setting the misclassification costs right appears to be a difficult task and should be based on expert opinions (Krawczyk et al., 2014). As a first estimate, C. Castro & Braga (2013) set the misclassification costs for the majority class equal to 1 ($c_0 = 1$) and the misclassification costs for minority class equal to the ratio of imbalance (IR):

$$c_1 = \text{IR} = \frac{\text{Number of observations in majority class}}{\text{Number of observations in minority class}} \tag{2.14}$$

Still, the misclassification cost for the minority class is a hyperparameter that should be optimized.

2.3 Inventory Management Systems

The goal of Research Question 3 is to minimize the number of fallen cases by influencing the number of layers to be depalletized at once. The number of layers to be depalletized affects both the inventory in the HBW and the TWH, as it moves the stock from the first location to the second. One pallet occupies only one location in the HBW, whereas each case on this pallet occupies a single location in the TWH. Customer orders (i.e. demand from Jumbo stores) are picked from the TWH; if the inventory here is insufficient to meet all demand, the lead time of an order increases significantly as first a new pallet should be depalletized. Therefore, the division of inventory over these two warehouses reflects an important trade-off: inventory stored in the TWH requires much more storage space as compared to the HBW, but the stock is immediately accessible for order picking.

For both storage locations, an inventory management system is used to ensure high service quality and simultaneously use the storage space efficiently. This section discusses commonly known inventory management systems, which will serve as a starting point for modeling the problem. The Lecture Notes for the course Stochastic Operations Management (Van Donselaar & Broekmeulen, 2017) were used as the main reference within this section, which serves as a supplement to standard textbooks regarding Inventory Management.

First, it is important to note that replenishment decisions are based on the inventory position $IP(t)$ of a system at a given time t. The inventory position equals the total inventory on hand $I(t)$ minus the outstanding backorders $BO(t)$ plus the inventory in transit $IT(t)$ (inventory ordered at an earlier point in time that did not yet arrive). In the current situation, demand from stores that cannot be met from stock is not backordered. Conversely, the stores are required to re-order unmet demand at a later point in time. That is, the unmet demand is visible in the demand levels of succeeding days. Hence, the backorders at the TWH equal zero in all periods and any unmet demand is lost for that period.

Inventory management systems are generally classified based on two things: (i) either periodic or continuous review of the inventory systems to make replenishment decisions and (ii) whether the replenishment quantity is fixed or is allowed to be chosen arbitrarily. Based on these classifications, four inventory management systems can be distinguished, as depicted in Table 2.2.

Table 2.2: Classification of Inventory Management Systems

	Periodic Review	Continuous Review
Fixed replenishment quantity	(R, s, nQ)	(s, nQ)
Variable replenishment quantity	(R, s, S)	(s, S)

At each review period R, the (R, s, nQ) system checks whether or not the inventory position dropped below the reorder level s. If this is true, the system chooses the smallest integer n such that n times a fixed batch size Q raises the inventory position back to or above the reorder level s. If the inventory position is still above the reorder level s at a review period, no replenishment decision is made. The (s, nQ)

system works similarly, although the replenishment decision can be made at any point in time due to continuous review of the inventory position. In a similar way, the (R, s, S) policy makes a replenishment decision at each review period R if the inventory position drops below the reorder level s. Conversely, this system chooses the order quantity such that the inventory position equals the order-up-to level S after ordering; the replenishment quantity is thus allowed to be chosen arbitrarily. The (s, S) policy works the same but as for the (s, nQ) system, the replenishment decisions can be made at any point in time instead of at dedicated time periods only. Both continuous review systems can be modeled as their corresponding periodic review system by setting the review period to a very small number.

The inventory management system applicable to the HBW is the (R, s, nQ) policy, as each supplier has a pre-arranged time interval between deliveries. The stock of an SKU can only be replenished during such deliveries and thus only needs to be reviewed periodically. All SKUs of the same supplier have the same review period; the review periods of distinct suppliers might differ. Moreover, it is most common to order complete layers or full pallets of an SKU, indicating that the batch size Q is fixed (being the number of cases per layer or the number of cases per pallet). For each review period, the minimum number of complete layers or pallets n should be chosen such that the inventory position of the HBW is raised to or above the reorder level s. The (s, nQ) system applies to the TWH, as the inventory of the TWH is continuously reviewed and orders to one of the DEPALs can be sent at any point in time. Moreover, the DEPAL only depalletizes complete layers by default. Hence, the batch size Q corresponds to (a multiple of) one layer and for each work order the minimum number of such batches n should be chosen such that the inventory position for an SKU in the TWH is raised to or above the reorder level s. As stated above, the (s, nQ) can be modeled in the same way as the (R, s, nQ) system by setting the review period R to a very small number.

The HBW and TWH correspond to a two-stage serial inventory system, where customer demand arises at stage 1 (the TWH) and stage 2 (the HBW) replenishes stage 1, which itself is replenished from outside suppliers (Chen & Zheng, 1994). In (two-stage) serial systems, the reorder levels and batch sizes are preferably set such that the total inventory management system results in a system-wide optimal cost (Shang et al., 2009). Hence, the optimal policy parameters (being the review periods R, reorder levels s, and batch sizes Q) for each single stage are preferably based on centralized decisions.

A lot of literature has been devoted to the control of serial inventory systems. Especially systems where each stage follows an (s, nQ) policy received a lot of attention. Most studies focused on either evaluation of this inventory management system, optimization of the policy parameters (i.e. s and Q) or developing a heuristic for that matter. For instance, Chen & Zheng (1994) evaluated the (s, nQ) policy for a serial system and derived steady state inventory levels for each stage. Chen (2000) formulated a method to find optimal reorder levels for serial systems with fixed batch sizes, whereas Shang & Song (2007) provided approximations to optimize both the reorder levels and the batch sizes for the entire system, by optimizing these parameters individually for each single stage. Furthermore, Axsäter & Rosling (1993) evaluated the difference between centrally setting reorder levels and batch sizes (i.e. such that the system wide cost is minimized) and optimizing them for each stage solely, and found that a system wide solution is superior in general. Instead of a continuous review policy, Shang & Zhou (2010) studied a serial system where each stage orders according to an (R, s, nQ) system. They proposed an optimal and heuristic solution to find batch sizes and review periods that minimize the total average cost per period.

The problem at hand is a valuable extension on the current literature on two-stage serial inventory systems. Even though most studies are concerned with finding the optimal policy parameters R, s, and Q, these parameters are considered to be fixed in the current analysis. Then, the logic of the standard (s, nQ) system will be used to determine the order quantity of the TWH at the HBW for a given SKU. Using these order quantities as a starting point, a method can be formulated that adjusts these order quantities such that less pallets will be partially depalletized, as this is believed to reduce the number of fallen cases. Hence, the goal is still to find an optimal balance of inventory over the two stages, but the approach is slightly different. To the extent of my knowledge, no other study targeted this problem

in a similar way. Therefore, the current study is a valuable addition to the literature on two-stage serial inventory management systems.

As stated earlier, the number of layers to be depalletized represents a trade-off between storing the inventory at the HBW (requiring fewer locations) and storing it at the TWH (being accessible for order picking). This trade-off should be acknowledged when modeling the problem: the model should be prevented from pushing too much inventory to the TWH. Simply fixing the batch size to full pallets instead of (a multiple of) one layer for all SKUs will not result in a satisfactory solution, as the available capacity in the TWH will not be used efficiently. Therefore, Chapter 5 introduces a mathematical model, which optimizes which pallets should be partially depalletized and which pallets should not.

2.4 Research Gap

Since the problem at hand is not discussed in recent literature yet (as indicated in Section 2.1), a straightforward research gap is present. Solving the problem by developing methods to minimize the number of fallen cases will enrich the current literature on automated warehousing systems. In addition, predicting whether cases will fall off a pallet is a new application of standard prediction theories and represents a new setting of imbalanced data. Finally, the order quantities obtained by a standard inventory management policy will be reviewed and potentially altered to ensure more pallets will be fully depalletized at once, thereby minimizing the number of fallen cases. This new approach to balance inventory in a two-stage serial inventory system is believed to be a valuable addition to the current literature on inventory management systems.

Chapter 3

Current size of the problem

The goal of Research Question 1 is to gauge the size of the problem and to build a dataset containing observations that indicate whether one or more cases of a certain SKU fell off a pallet between inbound and tray merge. This dataset was used as input for the prediction models discussed in the next chapter. As the performance of a prediction model is only as good as the data used to train it (a phenomenon known as "garbage in, garbage out"), the data was selected carefully and several data preparation techniques were applied to clean the data (Brownlee, 2022). The data preparation techniques are listed in Appendix C, whereas the data collection and preliminary data analyses are discussed in the following sections.

3.1 Data Collection

As indicated in Section 1.2, the current data on fallen cases is inaccurate. If a case falls, the missing case should be manually registered in the system. However, this registration is often neglected in current practice, yielding incomplete and unreliable data. Therefore, other data sources were used to estimate the number of fallen cases between inbound and tray merge.

As explained in Section 1.1.3, SKUs enter the CDC on pallets containing only that particular SKU. They are introduced to the system at one of the infeed stations, which measures the height and weight of a pallet. This information is used to calculate the total number of cases and number of layers on a pallet. If the physical measurements correspond to the data known by the system (within some tolerable limit), the pallet is accepted and send to the HBW. Based on replenishment calls from the TWH, the pallet will be sent to one of the DEPALs. After depalletization, each case is individually put on a tray at tray merge. This is where the second physical check of a pallet takes place, as a height check is performed for each case. This way, the tray merge counts how many cases of a particular SKU coming from a particular pallet are put on a tray.

From the previously described processes, a black box can be derived where especially the input and output of the system are of interest, whereas the inner workings of the system can be slightly disregarded (Cuadra & Katter, 1967). In Figure 3.1 a visual representation of a black box is depicted. In the current

Figure 3.1: Visualisation of the black box method

context, the information from the infeed stations serves as input and the information from tray merge serves as the output. No physical measurements are conducted between these two steps and therefore all processes performed here are considered as a black box. The difference between input and output gives an indication of the number of cases that were lost somewhere in this black box. This number is used as an estimate for the number of fallen cases between inbound and tray merge.

Besides the tray merge, one additional output stream requires consideration. Some "missing" cases (i.e. cases that do not pass a tray merge) were sent to a clearing station instead. From this clearing station, the missing cases are re-introduced to the system, either independently on trays (such that they are sent to the TWH directly) or on a pallet, with multiple cases of the same SKU (such that it is sent back to HBW). This stream is used for multiple reasons, such as badly stacked layers for pallets or an unintended tilted case for trays. In such circumstances, the system or an employee sends the cases to a clearing station, where a Witron operator decides if and how the case should be re-introduced to the system. This decision is registered in a report called *IN32*. Based on the decision, the subsequent steps adherent to the decision made are performed manually. As these streams are potentially used for different reasons than a fallen case, they are considered as output of the black box as well, as depicted in Figure 3.1.

Data was collected for a time frame of fifteen weeks. Pallets were included in the dataset according to the following criteria:

- Only pallets for which inbound information was present in report *INB14* were included in the data. This report contains real information on the number of cases and the number of layers on a pallet. As pallets usually spend some time in the HBW before being handled for the first time, information from this report is included during the stated time frame and two weeks in advance. These time frames are visualized in Figure 3.2. If a pallet entered the system more than two weeks before the time frame of interest, it was thus neglected.
- Only pallets for which outbound information was present in report *TL10* were included in the data. This report contains the number of cases put on trays from a certain pallet. Information from this report was only included during the stated time frame.
- Pallets are not necessarily depalletized at once: they can be sent back to HBW and called again at later points in time for further depalletization. To have complete output data, only pallets were considered which were fully depalletized (at once or in several times) within the stated time frame. This information was extracted from a report called *OP48a*, which contains all DEPAL transactions and reveals how many layers were picked at each transaction. Information from this report was only included during the stated time frame. Despite the restriction, still more than 21 000 partially depalletized pallets were included in the dataset. This was believed to suffice for the analysis on partial depalletization.

Some assumptions were needed regarding the alternative output streams, where cases are re-introduced to the system on pallets or trays. For both streams, information is present on the type of SKU, the number of cases of that particular SKU and the time at which the re-introduction took place. However, it is not known from which pallet the SKU originated, as the complete pallet is usually not present at the clearing station. Since both input and output data were checked for each pallet individually, the re-introduced cases should be linked to a particular pallet to get a complete overview. To do this, the time period in which a case of a particular SKU should be re-introduced to the system was constrained. This way, it is believed that the re-introduced cases originate from the pallet that was handled a given period earlier.

Pallet re-introduction typically takes longer than tray re-introduction. For both streams two different periods were tested, indicated with the parameters t_{tr} and t_{pal} for trays and pallets, respectively, both expressed in hours. Two scenarios are considered: Scenario 1 with values $t_{tr} = 12$ and $t_{pal} = 24$ hours and Scenario 2 with values $t_{tr} = 24$ and $t_{pal} = 48$ hours. As re-introduction streams can take place up to two days after the pallet was handled at a DEPAL, information from *IN32* is included till two days after the stated time frame, as depicted in Figure 3.2. Clearly, the number of cases on a new

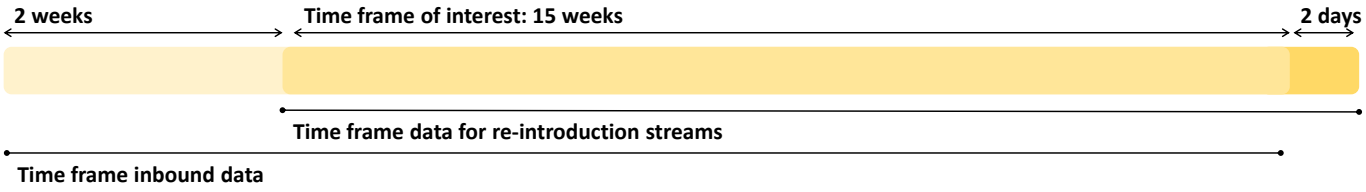


Figure 3.2: Time frame data collection

(re-introduced) pallet should be less than or equal to the number of cases missing from the origin pallet.

The data collected according to the previously mentioned criteria and assumptions is referred to as raw data (Brownlee, 2022). The raw data contains information about pallets entering and leaving the system (i.e. passing the tray merge), including the number of cases that fell during this process. Each row in the data represents an observation, which corresponds to a particular pallet in the system. Each column represents a certain feature of the observations, including information about the pallet, the SKU and the machine settings. The relevant features were selected along with Witron and Jumbo employees and are discussed in Appendix B. The features originate from three data sources: (i) features from Witron Master data (mostly machine related features), (ii) features from Jumbo Master data (mostly SKU-related features) and (iii) features from inbound or tray merge, based on physical measures (pallet-specific features).

Besides the features extracted from existing data sources, some new features were added. Some of these features were added to include more information about the process between HBW and the tray merge. The inner workings of the black box were thus not fully disregarded, as the processes within the black box were analyzed to obtain these features (e.g. *Number of Calls DEPAL* and *DEPAL i*, using report *OP48a*). These features and how they are defined are listed in Appendix B as well. The feature *Fallen cases Binary* reflects if cases fell off a pallet or not and is the feature of interest. That is, the goal is to predict this feature accurately using the ML techniques discussed in Section 2.2. The column containing this information is referred to as the output feature, whereas all other columns are referred to as input features.

3.2 Data Analysis

Data was collected for a period of fifteen weeks according to the criteria listed in Section 3.1. Before the collected raw data can be used as input for ML techniques, it usually needs to be transformed by applying certain data preparation techniques. Preparing the raw data well is very important and potentially has a big influence on the performance of a prediction model (Hadley & Whitin, 1963). For instance, many algorithms require that all input features are transformed into numeric features, even though they appear as categorical, ordinal or binary features in the raw data. Moreover, some ML techniques have specific requirements or characteristics, such as performing worse when two features are highly correlated (Brownlee, 2022). Appendix C lists all data preparation techniques applied to the dataset of interest.

As stated, two different scenarios were tested for the time-periods of the re-introduction streams; the obtained estimates for the number of fallen cases are listed in Table 3.1. The datasets contained 120 626 observations, cases fell off 4.2% and 4.1% of these pallets for Scenario 1 and 2, respectively. Moreover, these estimates indicate that each week, nearly 1 000 cases fell between inbound and tray merge, which corresponds to 0.12% and 0.11% of the inbound volume (*Cases Pallet*) for Scenario 1 and 2, respectively. The monetary value of these fallen cases equalled €164 735 and €163 880 for Scenario 1 and 2 respectively, corresponding to more than €10 000 per week. Note that Jumbo's current estimates, based on manual registration, suggest that each week approximately 200 cases fall in the entire warehouse (including all OPM, CPS and DPS processes). Due to inaccurate manual registration, Jumbo highly underestimates the real number of fallen cases.

Table 3.1: Number of fallen cases for Scenario 1 ($t_{tr} = 12$ and $t_{pal} = 24$ hours) and Scenario 2 ($t_{tr} = 24$ and $t_{pal} = 48$ hours)

	Fifteen weeks of data	
Cases Pallet	12 903 103	
Cases Tray Merge	12 886 339	
Difference	16 764	
	Scenario 1	Scenario 2
Re-introduced cases on trays	1 487	1 654
Re-introduced cases on pallets	319	225
Fallen cases	14 958	14 830
Fallen cases Binary = Yes	5 045	4 960

For both scenarios, first the cases re-introduced on trays were added to the output and subsequently the cases re-introduced on pallets were added. As one would expect, the number of re-introduced cases on trays is higher for Scenario 2 than for Scenario 1, as a longer period is allowed for the re-introduction. A similar effect would be expected for the number of re-introduced cases on pallets, but the reverse is true. Apparently, cases for some SKUs are re-introduced both on trays and pallets in Scenario 2. After adding the re-introduced trays to the output for these SKUs, the remaining number of missing cases appears to be less than the number of cases on a re-introduced pallet. Consequently, the latter stream is not added to the output and the remaining number of cases for these SKUs are considered as fallen. Conversely, in Scenario 1 the re-introduced trays were not added to the output as the re-introduction took longer than 12 hours. As a result, the remaining number of missing cases is less than or equal to the number of cases on a re-introduced pallet, such that this stream could be added to the output. This situation shows that the applied method is not completely accurate. Still, the re-introduction streams cannot be redirected to a given pallet in any other way, meaning that there is no possible way to overcome this inaccuracy. In the remainder of this thesis, all analyses will be performed using the dataset from Scenario 2.

To analyse the data, the dataset was separated based on the *Fallen cases Binary* feature, to visually check if pallets with fallen cases (i.e. minority class) have different characteristics from pallets without fallen cases (i.e. majority class). That is, the distribution of all features was analyzed for the majority and minority classes separately. The figures visualising those distributions are depicted in Appendix D.

From these figures can be derived that cases of SKUs in the minority class tend to be small, such that more cases fit on a single pallet. That is, the distribution of all features indicating the size of a single case (e.g. *Length Case*, *Height Case*, etc.) is more shifted to the left for the minority class than for the majority class. Furthermore, the distribution of features related to the number of cases on a pallet (e.g. *Cases Pallet*, *Layers*) is more shifted to the right for the minority class. Consequently, pallets with fallen cases appear to have more cases on them when they enter the warehouse. This is a sensible effect, as a pallet with many small cases is likely to be more unstable than a pallet with fewer large cases.

In addition, the *Number of Calls DEPAL* feature appeared to be greater for minority class observations than for majority class observations. Apparently, making more movements through the warehouse results in more fallen cases. This finding is closely related to the previous finding, as pallets with more inventory on it, are more likely to be partially depalletized. Hence, besides the fact that smaller cases could result in more unstable pallets, they also result in more partial depalletizations and thus more movements through the warehouse, increasing the risk of falling even more.

Not surprisingly, the distributions show that minority class observations pass DEPAL 6 and DEPAL 7 more often than majority class observations. These DEPALs can also be used for manual depalletization. Therefore, pallets with errors (e.g. due to fallen cases) are often sent to these DEPALs. Finally, the largest percentage of minority class observations originated from the Sodas (Dutch: Frisdrank) group,

which is not true for the majority class observations. Apparently, SKUs in this group have a higher risk of falling.

The data collection method described in this chapter appeared to be effective in finding an estimate for the number of fallen cases, given the available data sources. These estimates indicate that each week, nearly 1 000 cases fell between inbound and tray merge solely, which is much higher than Jumbo's current estimates. These fallen cases cost Jumbo more than €10 000 on a weekly basis, and bring additional cleaning cost, potential machine downtime and decreased service quality. As such, the collected data confirmed the problem of Jumbo's CDC in Nieuwegein and emphasizes the need to establish methods to decrease the number of fallen cases.

As indicated, the applied method is not completely accurate, due to the discrepancy in the re-introduction streams. To increase the accuracy of the data, additional information could be added to report *IN32*, to better clarify the reason of a re-introduction. In the current report, no information is present that clearly distinguishes one re-introduction stream from another. Therefore, additional assumptions were needed to re-direct re-introduced cases to particular pallets. By better clarifying the reason of a re-introduction, the re-introducted cases can be easily linked to a pallet and no additional assumptions are needed.

Chapter 4

Prediction model

The goal of Research Question 2 was to build a prediction model that accurately predicts whether cases will fall off a new pallet entering the system (i.e. for out-of-sample data). Moreover, the goal is to find SKU-, process-, and machine related characteristics that influence the likelihood of falling.

To build the prediction models, the dataset discussed in the previous chapter was used, containing approximately 120 000 observations. Of this dataset, 80% of the observations was used to train the model, 10% was used to validate the model and select the best hyperparameter values, and the remaining 10% was used to test the model. The samples in each dataset were randomly selected, whilst ensuring a constant ratio of imbalance among all datasets. The goal of the prediction models is to accurately predict the *Fallen Cases Binary* feature. As this feature was derived from the *Fallen Cases* feature, this latter feature is a perfect estimator of the output feature. Therefore, this feature was removed when training the models. As such, a dataset was left with 120 (transformed) input features.

Two ML techniques were applied to find accurate out-of-sample predictions: an RF and an MLP. Moreover, in Section 2.2.3, Resampling and CSL were introduced as methods to account for the imbalanced dataset, to improve the predictions of the ML techniques. Both Resampling as CSL were applied in combination with RF and MLP, to find the best model for the current application. Sections 4.1 and 4.2 discuss how the RF and MLP models were build, respectively, and how they performed. The performance of both methods is compared in Section 4.3.

4.1 Random Forest

In Section 2.2.1 and Appendix A the inner workings of an RF and the important hyperparameters were discussed, respectively. Based on this information, several RF models were build with different hyperparameter values, which are listed in Table 4.1. First, Oshiro et al. (2012) proved that the number of trees T in an RF is not necessarily a hyperparameter that needs to be optimized: choosing a large number of trees appears to work well in general. Hence, this parameter was set equal to 1000 trees for all models. The size of the random subset m was set equal to $\sqrt{p}$, with p being the number of input features. This is a commonly used value used for classification problems and appeared to work well in practice (Bernard et al., 2009). For each model, the entire training dataset was used (i.e. full sample). The samples used in each single tree in a forest were drawn with or without replacement from the training sample (potentially after resampling was performed). Three values were tested for the maximum depth of the tree d, to measure its effect on the performance of an RF and prevent overfitting. The Gini function was used in all models, since Tangirala (2020) showed that the split criterion function does not affect the performance of an RF. Lastly, the scaling method was added as a hyperparameter to the RF models, as it highly depends on the dataset whether standardization or normalization of the numerical features works better (Brownlee, 2022).

First, an RF was fit for all above-mentioned hyperparameters combinations without accounting for the imbalanced dataset. Subsequently, the effects of Resampling the data beforehand and CSL (i.e. adding a minority class cost to the split-criterion function) were tested. For Resampling, both

Table 4.1: Hyperparameter values tested for RF

Hyperparameter	Number of values	Tested values
Number of trees T	1	1000
Size of random subset m	1	$\sqrt{p}$
Sample size n	1	Full sample
Sampling with replacement	2	With and without
Maximum depth d	3	10, 25, 50
Split-Criterion	1	Gini index
Scaling method	2	Standardization, normalization
No method to account for imbalance	1	
Resampling method	2x3	Random undersampling, SMOTE with fractions $\frac{1}{3}$, $\frac{2}{3}$, 1
Cost Balance	3	Minority cost equals 0.75IR, IR, 1.25IR

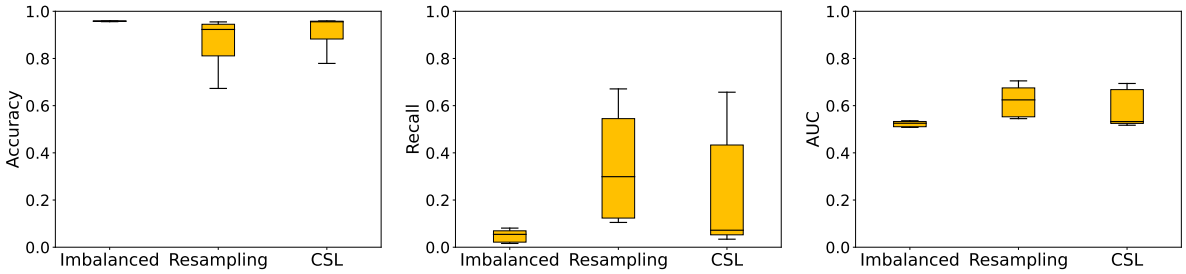
undersampling of the majority class (using Random Undersampling) and oversampling of the minority class (using SMOTE) were tested, with three different fractions of resampling. For CSL, three different values were tested for the misclassification cost of the minority class. In C. Castro & Braga (2013), this cost was set equal to the IR. To evaluate the effect of this parameter, 25% below and above this ratio was tested as well.

A separate model for each combination of hyperparameters was implemented in Python using the scikit-learn library, which allows for quick and easy implementation of an RF. That is, 12 models were tested without accounting for the imbalance in the data, 72 models were tested with Resampling (using the imblearn library) and an additional 36 models were tested with CSL. All models were trained on the training dataset and evaluated on the validation dataset. From all these models, the model that performed best on the validation dataset was selected. Moreover, the feature importance of the best performing models was analysed, to select the subset of features that were most relevant in predicting the output feature. Subsequently, the best model was fit again on this subset of features, to check if the performance improves without the presence of irrelevant features. Based on these results, the best model and best subset of features were selected to evaluate the performance on the test set.

Results

Figure 4.1 visualizes the performance on the validation dataset of all models with different hyperparameter combinations. Three performance measures are depicted: the overall accuracy, the recall (TPR) and the AUC. The goal is to find a model that scores reasonably well on all three performance measures. To achieve this, the model with the highest AUC should be selected, as this measure indicates whether the model is able to discriminate well between the minority and majority class (Probst & Boulesteix, 2017). That is, the AUC score represents a trade-off between recall and accuracy. As such, higher levels of the AUC could result in higher levels for the other two measures as well, which is a desired effect. Conversely, in imbalanced data settings, higher accuracy scores could result in lower recall scores, and vice versa.

This effect is visualised in Figure 4.1. All models on the imbalanced dataset achieved an accuracy of 96%, but scored very badly on recall. Apparently, these models were not able to distinguish the minority class observations. Consequently, the AUC performance is close to its minimum value (0.50) as well. As the goal is to accurately predict those observations (and thus achieve a high recall and AUC) these models were not applicable for the current context. The different hyperparameter settings had little influence on the performance of these models: due to the imbalanced dataset all models were not able to distinguish the minority class observations.



(a) Accuracy **(b)** Recall **(c)** AUC

Figure 4.1: Box plots of three performance measures for results on validation dataset, for RF models without accounting for the imbalance in the data (Imbalanced), with Resampling or with CSL.

In contrast, several models using Resampling or CSL were better able to predict the minority class: these models achieved a maximum AUC of 71% and 69% with an adherent recall of 66% and 66%, respectively. Figure 4.1 clearly shows the trade-off between these two measures and the accuracy: to achieve higher recall and AUC performances, a slight drop in the accuracy should be accepted. The figure shows that on average, the Resampling models scored better on recall and AUC than the CSL models. Still, the performance of the best model of both methods is comparable.

To understand the effect of the hyperparameter settings, the performance of all models using Resampling or CSL was visualized in pairwise plots, which are depicted in Appendix E. That is, in each pairwise plot the levels of two hyperparameters were changed, whereas the values of all other hyperparameters were kept constant. From these pairwise plots could be derived that the scaling method and whether the samples were drawn with or without replacement only had little influence on the performance of all models. In general, normalization of numerical features performed slightly better in terms of recall and AUC than standardization and sampling without replacement performed slightly better than sampling with replacement.

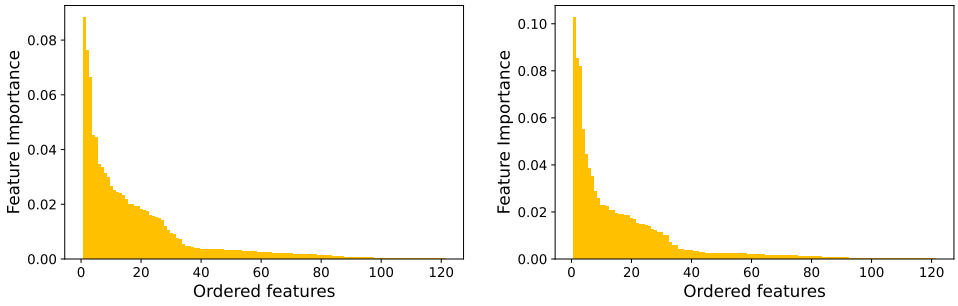
For the models using Resampling, the maximum depth and whether the dataset was over- or undersampled greatly influenced the performance of the models. All models with undersampled data achieved higher recall and AUC performances than all models with oversampled data, for all different tree depths and resampling fractions. Moreover, completely under- or oversampling the dataset (i.e. a resampling fraction of 1) appeared to work better than a lower resampling fraction. Finally, RFs with more shallow trees resulted in better recall and AUC scores on all oversampled datasets, while this was only true if the dataset was fully undersampled. Otherwise, the ability of the model to predict the minority class decreased as the maximum depth of the tree decreased.

In all models using CSL, RFs with more shallow trees outperformed RFs with higher depth trees. For every value of the minority class cost, the trees with more depth were not able to distinguish the minority class observations well. In addition, the recall and the AUC of the RFs with a maximum depth of ten increased as the minority class cost increased. Hence, a higher minority class cost proved to work better than a lower minority class cost.

As stated, the final goal is to find a model with a reasonable score on all three performance measures. To achieve this, the best models were selected based on the highest AUC. For the models with resampling, the best Resampling model was a completely undersampled model (i.e. resampling fraction of 1) with a maximum depth of 10 splits. The best CSL model had a maximum depth of 10 splits and a minority class cost of 30 (i.e. 1.25 times the IR). Both models had normalized numerical features and all samples were drawn without replacement. The performance of these models are listed in Table 4.2, from which can be derived that the best Resampling model slightly outperforms the best CSL model: having the same accuracy, the Resampling model is able to achieve a higher recall and AUC score.

Table 4.2: Performance of best RF models on the validation and test dataset for different sets of features, with a maximum depth of 10 splits, sampling without replacement, normalized numerical features and (i) trained on a fully undersampled majority class (Resample model) or (ii) with a minority class cost of 30 (CSL model).

	Accuracy	Recall	AUC	Number of features
Results best Resampling model				
Full model	0.74	0.67	0.71	120
Feature importance above 0.001	0.74	0.66	0.70	85
Feature importance above 0.01	0.75	0.65	0.70	29
Feature importance above 0.02	0.75	0.61	0.68	16
Results best CSL model				
Full model	0.74	0.65	0.70	120
Feature importance above 0.001	0.74	0.64	0.69	77
Feature importance above 0.01	0.77	0.61	0.69	30
Feature importance above 0.02	0.76	0.62	0.69	14
Results on test set				
Resampling 0.01 model	0.72	0.64	0.68	29



(a) Best Resampling Model **(b)** Best CSL Model

Figure 4.2: Ordered feature importance of 120 input features of the best RF models

The feature importance of the best Resampling and CSL models are visualized in Figure 4.2. These figures show that the importance decreases drastically in the number of features; only a few features have high importance values as compared to all other features. Moreover, the importance of a large number of features is close to zero in both models. Consequently, a large set of features seems to be irrelevant in predicting the outcome feature.

Based on these figures, three cutoff values for the feature importance were derived to find three subsets of relevant features: a higher cutoff value excludes more features from the subset. The best Resampling and CSL models were fit again on these subsets of features: the obtained performances are listed in Table 4.2. The table shows that for both models, the recall and AUC performances slightly decreased as the number of features decreased, but the performance is still comparable to the performance of the full models. Apparently, fitting both models on only a small subset of all features still yields reasonable results. Moreover, the subsets of most important features were nearly equal for both models.

Considering the results listed in Table 4.2, the Resampling model on the subset of features having an importance above 0.01 (Resampling 0.01 model) was selected as the final best model. This model has a negligible difference in performance as compared to the full model, whereas the number of features in the model was reduced with more than 75%. Moreover, the recall and AUC performance of this

model is equal to the performance of the full CSL model, whilst achieving a better accuracy. Hence, the performance of this model is higher than the performance of all CSL models. The subset of features used for this model is listed in Figure 4.6 and will be discussed in more detail in Section 4.3.

To assess the final performance of the RF, the Resampling 0.01 model was fit on the test set, which performance is listed in Table 4.2. The table shows that the performance on all three measures on the test set is slightly lower, but still comparable to the performance on the validation set. Consequently, the model is believed to perform reasonably well on out-of-sample data and does not overfit on in-sample data.

4.2 Multilayer Perceptron

The way an MLP learns and the adherent important hyperparameters were introduced in Section 2.2.2 and Appendix A, respectively. In addition to this information, several results from the RF models were used to build the MLP models. First, the best RF models showed that the performance was not significantly affected when the model was fit on only a subset of relevant features. Therefore, the best RF model (Resampling 0.01 model) was used as feature selection method and all MLP models were fit on the subset of features listed in Table 4.6. Additionally, RF models with normalized numerical features performed slightly better than RF models with standardized numerical features. To limit the number of hyperparameter values, the numerical features were normalized in all MLP models.

All other hyperparameter values are listed in Table 4.3. Only one hidden layer was used in all models, as such configurations appeared to be able to approximate a large set of functions (Athey & Imbens, 2019; Lippmann, 1987). The number of neurons included in this hidden layer is an important parameter, as it affects the performance and the complexity of the model (Abraham, 2005). To measure the effect of the number of neurons in this layer, three values were tested: $2p/3 \approx 19$ (Wang, 1994), $2p = 58$ and $3p = 87$ (Kanellopoulos & Wilkinson, 1997) neurons. To properly train the model, the minimum number of training samples should be five times the number of free parameters in the model (Messer & Kittler, 1998). As the entire dataset contains approximately 120 000 observations, this was true for most hyperparameter settings. However, if a dataset is undersampled, the number of training samples decreases significantly. Consequently, the undersampled datasets were too small for the largest configuration (i.e. 87 neurons in the hidden layer). Therefore, only two configurations were tested for the undersampled datasets: 19 and 58 hidden neurons.

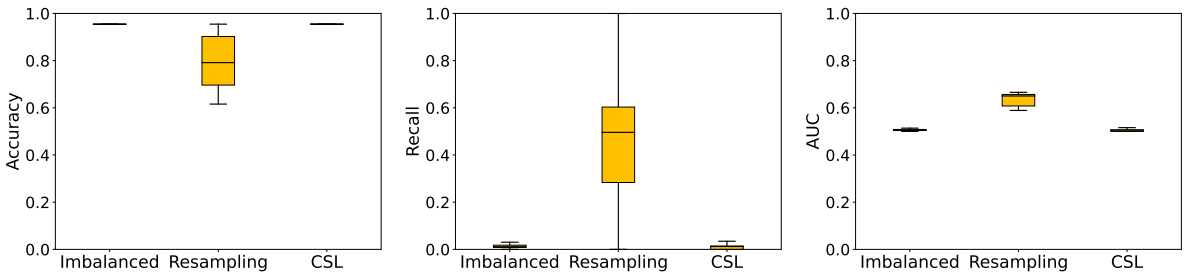
The sigmoid function was used as an activation function in all models. This is the most commonly used function and appears to work well in many contexts. The weights and biases were initialized as normally distributed random variables, as depicted in Table 4.3. For comparison, the same initialization (i.e. starting point) was used in all models. However, as was explained in Section 2.2.2, the exact starting point potentially influences the performance of the model. To account for this, ten different initializations were tested for the best model. Following the approach of Nielsen (2015), the order of magnitude of the learning rate η was obtained by trial and error; a learning rate between 0.001 and 0.01 appeared to work well for the current data and configurations. A similar approach was used for the regularization parameter λ, which showed that a maximum value of 10 worked well. To test the effect of regularization, no regularization was applied in some models (i.e. $\lambda = 0$).

As designated in the previous subsection, the model with the highest AUC should be selected, as this performance measure takes both the accuracy and the recall of the model into account. For similar reasons, the AUC was used in the stopping criterion, which was set such that the algorithm terminates if the AUC on the validation set did not improve in the last 50 epochs. Finally, to speed up the process, the stochastic gradient descent rule was used in the backpropagation algorithm. An explanation of this rule is included in Appendix F.

Similarly as for the RF models, a separate MLP was fit for all above-mentioned hyperparameter combinations, together with (i) no method to account for the imbalanced dataset, (ii) Resampling or (iii) CSL. The tested values for Resampling and CSL were equal to those tested for the RF models.

Table 4.3: Hyperparameter values tested for MLP

Hyperparameter	Number of values	Tested values
Number of hidden layers	1	1 layer
Number of neurons in hidden layer k	3 (2)	19, 58, 87
Activation function	1	Sigmoid function
Initialization of weights and biases	1 (10)	Weights $\sim \mathcal{N}(0, \frac{1}{\sqrt{n_l}})$ Biases $\sim \mathcal{N}(0, 1)$
Learning rate η	3	0.001, 0.005, 0.01
Regularization λ	3	0, 5, 10
Cost function	1	Cross entropy function
Stop criterion	1	No AUC improvement in the last 50 epochs
Scaling method	1	Normalization
No method to account for imbalance	1	
Resampling method	2x3	Random undersampling, SMOTE with fractions 0.33, 0.67, 1
Cost Balance	3	Minority cost equals 0.75IR, IR, 1.25IR



(a) Accuracy **(b)** Recall **(c)** AUC

Figure 4.3: Box plots of three performance measures for results on validation dataset, for MLP models without accounting for the imbalance in the data (Imbalanced), with Resampling or with CSL.

Consequently, 27 models were tested without accounting for the imbalance in the data, 54 and 81 models were tested for undersampled and oversampled datasets, respectively (using the imblearn library), and 81 models were tested with CSL. All models were trained on the training dataset and evaluated on the validation dataset. The model that performed best on the validation dataset was fit on the test dataset to assess the final performance of an MLP.

Results

In Figure 4.3 the performance of all models with different hyperparameter combinations are summarized in box plots. The figure shows that all models on the imbalanced dataset and with CSL performed badly. These models achieved a high accuracy, but scored extremely low on recall and AUC. Hence, these models were not able to distinguish the minority class observations well and are therefore not applicable for the current context. Apparently, the MLP configurations tested here only worked well if the data was resampled beforehand, as the figure shows that the models with Resampling performed much better on average. It appears that the different hyperparameter settings influenced the performance significantly, as some models achieved a recall of 0% while others achieved 100%. To understand the effect of the hyperparameter settings for the models with Resampling, pairwise plots were used, which are listed in Appendix G.

From the pairwise plots could be derived that especially the fraction of resampling greatly influ-

enced the performance of the MLPs. Especially for all models where the minority class was oversampled, no other hyperparameter significantly influenced the performance. For these models, the recall scores increased as the resampling fraction increased, whereas the AUC score remained relatively stable. Hence, these plots clearly show that the AUC reflects a trade-off between accuracy and recall: at the same level of AUC, a higher accuracy score results in a lower recall score. Therefore, a model with a high AUC is desired, as this results in a higher value for the other two measures as well.

If the majority class was undersampled instead, the learning rate η also played a role in the model's performance. Again, the recall and AUC scores of these models increased as the resampling fraction increased, but also as the learning rate η increased. Especially $\eta = 0.001$ resulted in a low performance, whereas the performance appeared to be indifferent for the higher values. For all sampling fractions, oversampling the minority class worked better than undersampling the majority class.

The MLP model with the highest AUC score was a completely oversampled model (i.e. resampling fraction of 1), with learning rate $\eta = 0.01$, no regularization (i.e. $\lambda = 0$) and 87 neurons in the hidden layer. As suggested in the beginning of this section, ten different weight and bias initializations were tested for this model, to test whether a particular initialization would affect the performance of the model. The average performance of these ten initializations on the validation set is listed in Table 4.4. All runs achieved the same AUC score of 66%, an accuracy between 67% and 72% and a recall score between 60% and 65%. Hence, these results reflect once again that the AUC score represents a trade-off between accuracy and recall. As the algorithm was terminated if the AUC score did not improve, it seems that an AUC of 66% was the maximum score this MLP configuration could attain. As the different weight and bias initializations resulted in some variations in accuracy and recall score, the final performance on the test set was also evaluated for ten different initializations. The average performance of these models is listed in Table 4.4. These results show that the AUC on the test set increased as compared to the performance on the validation set. The same holds for the recall score, whereas the average accuracy slightly decreased. All runs achieved an AUC score of 67%, an accuracy between 64% and 68% and a recall between 66% and 71%. Consequently, the accuracy performance is slightly lower than on the validation set, whereas the recall performance is slightly higher. Apparently, even though no regularization was applied in the MLP, the model was still prevented from overfitting on the in-sample data.

Table 4.4: Average performance of best MLP model on validation and test dataset, trained on a completely oversampled minority class, with learning rate $\eta = 0.01$, no regularization and 87 neurons in the hidden layer, for ten different weight and bias initializations.

Model	Accuracy	Recall	AUC
Average score Validation set	0.69	0.62	0.66
Average score Test set	0.65	0.69	0.67

4.3 Performance of prediction models

The goal of the prediction models was to accurately predict the minority class observations, whilst simultaneously achieving a reasonable accuracy and AUC score. In Table 4.5 the performance of the best RF and MLP models are summarized, which shows that both models achieved a similar score on AUC, whereas the best RF model scored better on accuracy and the best MLP scored better on recall. In addition, it should be noted that an RF requires little tuning, while for an MLP many parameters need to be set by the user. Considering that an RF is easier to implement and the fact that the best RF model achieved a higher AUC value, this technique is preferred over an MLP.

Remarkable is that for the RF models, undersampling the majority class proved to work better, while for the MLP models oversampling the minority class gave better results. Yet, for both models complete under- or oversampling appeared to work better than models with a lower resampling fraction.

Table 4.5: Performance of best RF and MLP model.

	Accuracy	Recall	AUC
Best RF model	0.72	0.64	0.68
Best MLP model	0.65	0.69	0.67

Moreover, CSL with a relatively high minority class cost worked well for the RF models, but not at all for the MLP models, as these results were similar to the models without CSL.

Besides accurately predicting the minority class observations, the goal of the second subquestion was to find SKU-, process- and machine related characteristics that influence the likelihood of falling. The feature importance of the best RF model was used to find the subset of relevant features, which are listed in Table 4.6. Many of the features in this list showed a significant difference in distribution for the majority and minority class, as depicted in Appendix D. To start, the most important feature appeared to be *Number of Calls DEPAL*, indicating that a partial depalletization (and thus more movements through the CDC) increases the likelihood of falling. Furthermore, the SKU-related characteristics mainly relate to the size of a single case or the number of cases on a pallet. Apparently, smaller cases (and consequently more cases on a pallet) have a higher risk of falling. As indicated in the previous chapter, this can be devoted to the fact that a pallet with many small cases is probably more unstable than a pallet with fewer large cases, or to the fact that pallets with more inventory on it are more likely to be partially depalletized. In the latter scenario, the *Number of Calls DEPAL* is the indirect true cause of the increased likelihood of falling.

Whether a given pallet was send to *DEPAL 6* or *DEPAL 7* for depalletization appears to be important as well. However, pallets with problems (such as cases falling) are often send to these DEPALs as they allow for manual depalletization. As such, it could be true that these DEPALs are not the cause of falling but are used to solve the issue instead. Still, the reason of falling can also be due to malfunctioning of these DEPALs and therefore these features were kept in the dataset.

Furthermore, Table 4.6 shows that *Weight Difference Pallet* and *Height Difference Pallet* had a high importance in predicting the outcome feature as well. Nevertheless, looking at the distribution of these features in Appendix D, no significant difference could be found for the majority and minority class. Still, these two features are one of the few features in the dataset that are pallet specific, as they are based on physical measures. That means that the values for these features are unique for each observation, which potentially explains why they are important in predicting the outcome feature.

A higher demand rate also resulted in a higher risk of falling. This is a sensible effect, as more pallets were handled in the stated time frame, indicating more possibilities for cases to fall. A similar line of reasoning can be applied to the features indicating the size of the supplier (*Weekly Pallets* and *SKUs Supplier*): a larger supplier ships more pallets, which also increases the risk of falling. Finally, the most important machine related features appeared to be *Fix next layer*, *Skirt Touch Height* and *Vacuum Pressure*. These three features also show small differences in distribution for the majority and minority class.

Concluding, the *Number of Calls DEPAL* feature appeared to be most important in predicting the likelihood of falling. Consequently, minimizing the number of partial depalletizations will minimize the number of fallen cases as well. To achieve this, a mathematical model is introduced in the next chapter. Moreover, many important features relate to the size of a single case and the number of cases on a pallet. As these pallets are more likely to be partially depalletized (and thus make more movements through the warehouse) and are relatively unstable, they should be handled with more care. At different steps in the process (i.e. the infeed stations, each visit at a DEPAL, etc.), the stability of the pallet should be judged by an operator and, if necessary, improved by adding additional stretch wrapping or re-stacking the pallet. That way, the number of fallen cases can be limited as much as possible.

Table 4.6: Subset of features having a feature importance above 0.01 in best RF model

Feature	Importance	Feature	Importance
Number of Calls DEPAL	0.088	Weight Tolerance	0.020
Volume Case	0.076	Height Tolerance	0.020
Cases Pallet	0.066	Height Width Ratio	0.019
DEPAL 6 Yes	0.045	Fix next layer	0.019
Cases layer	0.044	Weekly Pallets	0.018
Length Case	0.035	Length Width Ratio	0.018
Width Case	0.034	SKUs Supplier	0.017
Weight Case	0.031	Maximum layers	0.016
Weight Difference Pallet	0.030	Skirt Touch Height	0.015
Height Case	0.027	DEPAL 7 Yes	0.015
Weight Pallet	0.025	Layers	0.015
Height Difference Pallet	0.024	Demand rate Sign	0.014
Height Pallet	0.024	Vacuum Pressure	0.012
Demand rate	0.023	Distance between products	0.010
Pallet Area Fill rate	0.022		

Finally, as higher values for the *Weight Difference Pallet* and *Height Difference Pallet* features appeared to increase the likelihood of falling, it might be valuable to use lower values for the *Weight Tolerance* and *Height Tolerance* features. That way, pallets with a higher difference between physical- and expected measures are not immediately allowed to enter the system, but require a manual check beforehand. These additional checks can be used to judge the stability of the pallets and check if the physical inventory really corresponds to the information in the system. As lowering these tolerance levels will result in an increase in manual labour, the tolerance levels should be set such that a desired balance is obtained between performing too little (possibly resulting in fallen cases) and too much checks (labour-intensive).

Chapter 5

Minimize Partial Depalletizations

The goal of this chapter is to establish a method to minimize the number of partial depalletizations (i.e. reduce the feature *Number of Calls DEPAL*). Visually analysing the data in Section 3.2 already revealed that this feature adversely affects the number of fallen cases, as pallets in the minority class appeared to be partially depalletized more often (52.7% of all pallets in this class) than pallets in the majority class (25.6% of all pallets in this class). Not surprisingly, the prediction models introduced in Chapter 4 showed that this feature is most important in predicting the *Fallen Cases Binary* feature. As such, the number of fallen cases can be reduced by minimizing the number of partial depalletizations.

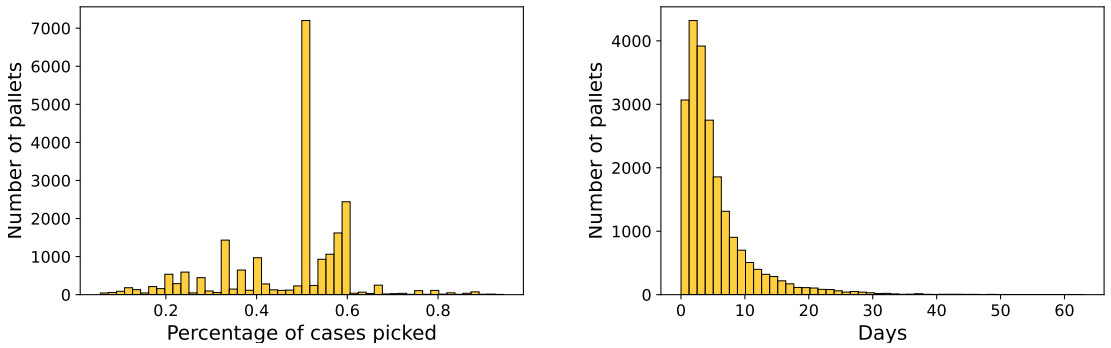
The next section sheds a light on the current replenishment strategy for the TWH and the resulting distribution of the *Number of Calls DEPAL* feature. Then, in Section 5.2 a mathematical model to minimize the number of partial depalletizations is presented, followed by the implementation and the obtained results. In Appendix J, a summary is included with all variables, parameters and sets used within this chapter.

5.1 Current Replenishment Strategy

As stated, the inventory management system in the TWH corresponds to an (s, nQ) system. That is, the stock in the TWH is continuously monitored such that if the current stock for a given SKU falls below the reorder level, a new replenishment is triggered. Each reorder level is based on a certain batch size, which are both dynamically set and differ per day for each SKU. The currently used definition for the reorder levels is included in Appendix H. The batch size $Q_{i,t}^{ca}$ for an SKU i on day t is based on the forecasted customer demand $F_{i,t}^{ca}$ for the upcoming days, using the following rule:

> **Rule Batch Size:** If the number of cases on a full pallet corresponds to less than RFP days of forecasted demand, the batch size equals the number of cases on a full pallet. Otherwise, the batch size equals the maximum of (i) the number of cases on one layer and (ii) the maximum number of layers such that the total number of cases corresponds to less than RPP days of forecasted demand.

Here, RFP reflects the maximum days of stock on a full pallet and RPP reflects the maximum days of stock on a partial pallet. The current strategy uses $RFP = 5$ and $RPP = 3$ days. Now, suppose the forecast for the following five days equals 10 cases per day for a given SKU i. Moreover, suppose that the number of cases on a full pallet FP_i^{ca} equals 60 cases, having $P_i^{ca} = 12$ cases per layer. As $FP_i^{ca} = 60$ corresponds to more than five days of forecasted demand (i.e. $\sum_{i=1}^{5} F_{i,t}^{ca} = 50$), the batch size is not set equal to a full pallet. Following the second part of the rule, the batch size is set equal to two layers instead ($2\,P_i^{ca} = 24$). That way, the maximum batch size is chosen, such that it corresponds to less than three days of forecasted demand ($\sum_{i=1}^{3} F_{i,t}^{ca} = 30$). Note that if the forecasted demand in the following three days would be less than 12 cases in total (i.e. $\sum_{i=1}^{3} F_{i,t}^{ca} < 12$), the batch size would still be set equal to one layer, as this is the minimum batch size by definition.



(a) Percentage of cases picked in the first call

(b) Number of days between the first and the last call

Figure 5.1: Empirical analysis of *Number of Calls* feature

Mathematically, this rule can be expressed as:

$$Q_{i,t}^{ca} = \begin{cases} FP_i^{ca} & \text{if } \sum_{t=1}^{RFP} F_{i,t}^{ca} \geq FP_i^{ca} \\ \max\{P_i^{ca}, \lfloor \frac{\sum_{t=1}^{RPP} F_{i,t}^{ca}}{P_i^{ca}} \rfloor\} & \text{otherwise,} \end{cases} \tag{5.1}$$

where $\lfloor x \rfloor$ means rounding x down to the nearest integer. Concluding, the batch size $Q_{i,t}^{ca}$ of SKU i on day t corresponds to the number of cases on (i) one layer, (ii) a multiple of complete layers or (iii) a full pallet, where the latter is the preferred option. Consequently, the largest share of pallets is fully depalletized at once.

In standard (s, nQ) systems, the replenishment quantity equals nQ, where n corresponds to the minimum number of batches Q required to increase the inventory position above the reorder level. Jumbo uses a similar system, with an additional rule that favors the replenishment of half of a pallet over other partial pallets. Consequently, replenishment quantities of only a few layers are rounded up to half pallets, such that the entire pallet is depalletized in only two calls instead of more calls. The explanation of the rule is included in Appendix H and the effect is visible in Figure 5.1a. This figure shows the percentage of cases picked in the first call (excluding fully depalletized pallets) and clearly visualises that this percentage often lies between 50% and 60%. That is, the replenishment quantity in the first call corresponds to (nearly) half a pallet for many observations.

Even though Jumbo already implemented several rules to favor full depalletization over partial depalletization and to limit the number of partial depalletizations, there is room for further improvement. Figure 1.6 shows that the utilization in the TWH is on average 77.5% and 64.2% for channels and locations, respectively, indicating that additional capacity is available to shift more inventory to the TWH and partially depalletize less. Furthermore, the current system still shows some inefficiencies. To start, Figure 5.1a also shows that the number of cases picked in the first call equals up to 95% for some observations, indicating that some pallets were send back to the HBW with only one layer left. Especially for such replenishments, a partial depalletization seems to be redundant and should be prevented. Besides, Figure 5.1b shows the number of days between the first and the last call, which equals only a few days on average. As the remaining inventory on a pallet is picked within short notice, shifting to a full depalletization at once seems to be more efficient. The following section introduces a mathematical model that minimizes the total number of partial depalletizations, to minimize the number of fallen cases and simultaneously overcome such inefficiencies.

5.2 New Replenishment Strategy

The mathematical model was build to make decisions on a tactical level (e.g. which SKUs should be replenished in what quantities during period $|\mathcal{T}|$?) rather than on an operational level (e.g. which pallet

should be send to which DEPAL at what time?). The model was based on historical data, such that the solutions could be compared with the actual replenishment strategy. In reality, the replenishments of the TWH are based on the (s, nQ) logic, where a replenishment is immediately triggered if the inventory position drops below the reorder level. However, as the mathematical model makes replenishment decisions on a daily level, the (R, s, nQ) logic was followed instead (explained in Section 2.3), using a review period of $R = 1$ day. This logic was used to calculate initial replenishment quantities (say: order quantity) nQ. Then, the model adjusts these order quantities such that the total number of partial depalletizations is minimized. Preferably, the order quantities are rounded up to full pallets, but a replenishment quantity lower than the order quantity is allowed as well. Due to limited TWH capacity, the model might decide to postpone (part of) the order quantity for one SKU, such that pallets of other SKUs can be fully depalletized at once. As a lower replenishment quantity potentially worsens the service quality of the TWH, the model also minimizes the number of postponed layers.

5.2.1 Assumptions

The mathematical model was based on the following assumptions:

Replenishments from HBW to TWH have zero leadtime. In the model, the inventory on hand is measured at the end of the day, triggering replenishment decisions. Even though the replenishment decisions are made at the end of the day, it is assumed that they are evenly spread throughout the day (which is true in reality) and can be finished at the end of the day. That way, the replenished quantities can be immediately added to the inventory on hand of the next day.

Order of decisions. The inventory on hand is checked at the beginning of day t, for both the HBW and the TWH. Customer demand takes place during the day. At the end of the day, a replenishment decision is triggered if the inventory position (being the inventory on hand at the beginning of the day minus the demand during the day and the outstanding backorders) dropped below the reorder level. The replenishment quantities are immediately added to the inventory of the TWH for the next day and immediately fulfill outstanding backorders. Moreover, new inventory delivered by external suppliers during a given day, is added to the inventory of the HBW at the beginning of the next day.

Customer demand is deterministic. To properly compare the replenishment decisions made by the model with the actual replenishment decisions made by the system, historic (deterministic) demand data was used.

Customer demand can be fulfilled on the same or following day and is backordered instead. In reality, customers are supposed to re-order an SKU on a subsequent day if the current stock is insufficient to meet all demand. That is, the demand is not backordered but considered as lost sales instead. However, as the model is based on historic (real) demand, the demand data already reflects these corrections: if a demand could not be fulfilled from stock on day t, it is not visible in the data on day t. Consequently, the demand data reflects what was actually picked on a given day. To ensure no demand is lost and to mimic the behaviour of the actual system, the demand that could not be met from stock immediately or with a delay of at most one day was added to the demand of the following day. This corresponds to a system with backorders.

TWH demand can only be replenished during the same day and is lost instead. The TWH demand refers to the order quantities obtained by the standard (R, s, nQ) logic. If the TWH demand cannot be met due to insufficient inventory in the HBW or because the model decides to postpone (part of) it due to limited capacity, the unfulfilled demand is not backordered. That is, a new replenishment decision will be triggered in a subsequent day for the postponed layer(s).

Reorder levels and batch sizes are dynamically set. The logic presented in Section 5.1 and Appendix H was used to set all batch sizes and reorder levels. These parameters are used to calculate the order quantities using the standard (R, s, nQ) logic.

FIFO handling of pallets is not a requirement. This is not necessarily true in reality, as most products have an expiration date. However, CDC Nieuwegein only handles products that have a shelf life of at least twelve days, so this relaxation can still be applied.

Each SKU has a fixed number of cases per layer. This assumption holds in the actual replenishment strategy as well.

A fraction of α of the DEPAL and TWH capacity is reserved for high priority replenishment orders. If the actual demand appears to be much higher than forecasted demand (and the stock in the TWH appears to be insufficient to fulfil all demand), sufficient capacity should be available to immediately depalletize pallets for these SKUs.

A fraction of β of the channel capacity is lost due to inefficient use of the channels. All other channels are used efficiently (i.e. only one partial channel per SKU).

Only one SKU is allowed per channel. This is generally true in practice. Occasionally, a channel contains more than one SKU due to certain re-allocations, but this is not desired.

Each week has six operation days. This corresponds to CDC Nieuwegein's standard operating scheme. However, occasionally customer orders are picked and TWH replenishments are performed on the seventh day. Demand picked during that day is neglected in the model.

5.2.2 Mathematical model

This section introduces the mathematical model with all its parameters, variables and constraints. The indices used in defining the variables all belong to a specific set, which are listed in Table 5.1. Moreover, the superscripts indicate whether the variable is expressed in cases (all variables relating to the TWH) or in layers (all variables relating to the HBW).

Table 5.1: Description of sets used in MINLP

Sets	Description
$\mathcal{I}$	Set of SKUs i that have a batch size less than a full pallet and are included in the MINLP.
$\mathcal{I}_{fp}$	Set of SKUs i that have a batch size equal to a full pallet during the entire period of interest and are excluded from the MINLP.
$\mathcal{T}$	Set of days t considered in one period.
$\mathcal{X}_i$	Set of pallets x of SKU i present in the CDC during time period $\lvert\mathcal{T}\rvert$

The first objective of the model is to minimize the total number of partially depalletized pallets during a given time period of length $|\mathcal{T}|$. The decision variable $\Theta_{i,t,x}$ is modeled as a binary variable, which equals 1 if pallet x from SKU i was partially depalletized on day t and 0 otherwise.

$$\min \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}} \sum_{x \in \mathcal{X}_i} \Theta_{i,t,x} \tag{5.2}$$

The second objective is to minimize the total unmet demand of the TWH at the HBW (i.e. the "lost sales") during a given time period of length $|\mathcal{T}|$. That is, the objective is to minimize the total number of layers $LS^l_{i,t}$ that were ordered to replenish SKU i in the TWH but were not sent due to insufficient inventory at the HBW or limited capacity in the TWH (such that the model decided to postpone the demand).

$$\min \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}} LS^l_{i,t} \tag{5.3}$$

Any feasible solution to the problem complies with the following set constraints. To start, Constraint (5.4) initializes the inventory on hand at the TWH ($I^{ca}_{i,t}$) for day $t = 1$ as the start inventory in the TWH SI^{ca}_i.

$$I^{ca}_{i,1} = SI^{ca}_i \qquad \forall i \in \mathcal{I} \tag{5.4}$$

Constraint (5.5) represents the inventory balance equation for the TWH for all other days. The inventory on hand on day $t+1$ equals the inventory on hand of the current day ($I^{ca}_{i,t}$), minus the customer demand during that day ($D^{ca}_{i,t}$), minus the unmet demand of the previous day ($BO^{ca}_{i,t-1}$) plus the unmet demand of the current day ($BO^{ca}_{i,t}$) and plus the replenishment quantity of that day. As the replenishment quantity ($\sum_{x \in \mathcal{X}_i} RE^l_{i,t,x}$) is expressed in layers, it is converted to cases by multiplying with the number of cases per layer (P^{ca}_i).

$$I^{ca}_{i,t+1} = I^{ca}_{i,t} - D^{ca}_{i,t} - BO^{ca}_{i,t-1} + BO^{ca}_{i,t} + P^{ca}_i \sum_{x \in \mathcal{X}_i} RE^l_{i,t,x} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{5.5}$$

Constraint (5.6) ensures that demand of day t (being the actual demand $D^{ca}_{i,t}$ plus the unmet demand of the previous day $BO^{ca}_{i,t-1}$) is backordered if and only if the inventory on hand ($I^{ca}_{i,t}$) is insufficient to meet all demand. The variable $\Omega_{i,t}$ is modeled as a continuous variable with values between 0 and 1 and reflects the fraction of demand that is truly unmet.

This constraint also reflects that demand can be fulfilled on the same or following day. That is, $\Omega_{i,t}$ and $BO^{ca}_{i,t}$ can both have value zero, even though the inventory on hand is insufficient to fulfill all demand (i.e. $D^{ca}_{i,t} + BO^{ca}_{i,t-1} > I^{ca}_{i,t}$). Constraint (5.5) ensures that this behavior can only happen if the replenishment quantity can immediately be used to meet the remaining demand[1].

$$\Omega_{i,t} \left(D^{ca}_{i,t} + BO^{ca}_{i,t-1} - I^{ca}_{i,t}\right) \geq BO^{ca}_{i,t} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{5.6}$$

It should be noted that the current constraint on the number of backorders is different than traditional formulations. Oftentimes, this constraint is formulated as $BO^{ca}_{i,t} \geq D^{ca}_{i,t} + BO^{ca}_{i,t-1} - I^{ca}_{i,t}$ (and defining $BO^{ca}_{i,t}$ as a non-negative variable). However, given the current formulation of the problem (i.e. the fact that the model minimizes the total unmet demand of the HBW), this formulation potentially resulted in solutions were (parts of) demand was backordered, even though there was ample inventory on hand[2]. That way, higher levels of inventory on hand could be attained, triggering less replenishment decisions. As this behavior is not desired in practice, the constraint was modified to Constraint (5.6), such that demand was backordered if and only if the inventory on hand was insufficient to meet all demand.

Constraint (5.7) initializes the number of backorders of day 0 (i.e. the day before replenishment decisions are made by the model) at zero. That is, the model starts with no outstanding backorders.

$$BO^{ca}_{i,0} = 0 \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{5.7}$$

[1] Suppose $D^{ca}_{i,t} = 13$, $BO^{ca}_{i,t-1} = 2$ and $I^{ca}_{i,t} = 10$. As such, the current inventory on hand is insufficient to meet the total demand. Now, suppose the replenishment quantity equals 30 cases (i.e. $P^{ca}_i \sum_{x \in \mathcal{X}_i} RE^l_{i,t,x} = 30$). This quantity can immediately be used to fulfill the unmet demand of the current day. That way, the model is allowed to set $\Omega_{i,t} = 0$ and $BO^{ca}_{i,t} = 0$, such that the inventory on hand of the following day equals $I^{ca}_{i,t+1} = 10 - 13 - 2 + 0 + 30 = 15$ cases. If the replenishment quantity would be less than 5 cases, the model is forced to assign a positive value to $\Omega_{i,t}$ and $BO^{ca}_{i,t}$, to ensure $I^{ca}_{i,t+1}$ remains non-negative.

[2] Suppose $D^{ca}_{i,t} = 15$, $BO^{ca}_{i,t-1} = 0$ and $I^{ca}_{i,t} = 20$. Then $D^{ca}_{i,t} + BO^{ca}_{i,t-1} - I^{ca}_{i,t} = -5$, but $BO^{ca}_{i,t}$ can still be positive.

Constraint (5.8) determines the number of occupied (full and partial) channels ($I^{ch}_{i,t}$) in the TWH for each SKU. Each channel comprises four locations and each case of SKU i takes up L_i locations.

$$I^{ch}_{i,t} \geq \frac{L_i \, I^{ca}_{i,t}}{4} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{5.8}$$

Constraint (5.9) ensures that the total number of occupied channels does not exceed the TWH capacity (K^{ch}). From this capacity, α% is reserved for high priority replenishments, whereas β% of the capacity is lost due to inefficient usage of the channels (i.e. multiple partial channels per SKU).

$$\sum_{i \in \mathcal{I}} I^{ch}_{i,t} \leq (1-\alpha)\,(1-\beta)\,K^{ch} \qquad \forall t \in \mathcal{T} \tag{5.9}$$

Constraint (5.10) determines the order quantity (in cases) of the TWH placed at the HBW based on the standard (R, s, nQ) logic. The total order quantity corresponds to a multiple $n_{i,t}$ of the current batch size $Q^{ca}_{i,t}$. An order is placed if the inventory position (being equal to $I^{ca}_{i} - D^{ca}_{i} - BO^{ca}_{i,t-1}$) at the TWH drops below the current reorder level $s^{ca}_{i,t}$.

$$n_{i,t}\, Q^{ca}_{i,t} \geq s^{ca}_{i,t} - I^{ca}_{i,t} + D^{ca}_{i,t} + BO^{ca}_{i,t-1} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{5.10}$$

Constraint (5.11) defines the unmet demand of the TWH as the difference between the order quantity $n_{i,t}\, Q^{ca}_{i,t}$ (converted to layers) and the total replenishment quantity ($\sum_{x \in \mathcal{X}_i} RE^{l}_{i,t,x}$). As the objective is to minimize the total unmet demand (i.e. minimize the sum of all $LS^{l}_{i,t}$), the unmet demand will equal zero if the replenishment quantity is greater than the ordered quantity.

$$LS^{l}_{i,t} \geq \frac{n_{i,t}\, Q^{ca}_{i,t}}{P^{ca}_{i}} - \sum_{x \in \mathcal{X}_i} RE^{l}_{i,t,x} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{5.11}$$

Similar as for the TWH, Constraint (5.12) initializes the start inventory for the HBW for day $t = 1$. The inventory in the HBW ($I^{l}_{i,t,x}$) is expressed in layers and defined for each pallet $x \in \mathcal{X}_i$ separately, as each pallet potentially has a different number of layers on it.

$$I^{l}_{i,1,x} = SI^{l}_{i,x} \qquad \forall i \in \mathcal{I}, x \in \mathcal{X}_i \tag{5.12}$$

Constraint (5.13) reflects the inventory balance equation for the HBW. The inventory on hand on pallet x at day $t+1$ is defined as the number of layers on a pallet on the current day ($I^{l}_{i,t,x}$), minus the replenished layers to the TWH during that day ($RE^{l}_{i,t,x}$), plus new inventory (in layers) delivered by external suppliers ($IT^{l}_{i,t,x}$). If a new pallet x enters the HBW on a given day $\hat{t} > 1$, then $IT^{l}_{i,\hat{t},x} > 0$ and $IT^{l}_{i,t,x} = 0$ for all other $t \in \mathcal{T}$. Moreover, as pallet x was not present at the HBW before day $\hat{t}$, the constraint ensures that $I^{l}_{i,t,x} = 0$ for all $t < \hat{t}$ and $I^{l}_{i,t,x} \geq 0$ for all $t \geq \hat{t}$.

$$I^{l}_{i,t+1,x} = I^{l}_{i,t,x} - RE^{l}_{i,t,x} + IT^{l}_{i,t,x} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T}, x \in \mathcal{X}_i \tag{5.13}$$

Constraint (5.14) ensures that the replenishment quantity from pallet x ($RE^{l}_{i,t,x}$) is less than or equal to the total number of layers on that pallet x ($I^{l}_{i,t,x}$). The variable $\Phi_{i,t,x}$ is modeled as a binary variable which equals 1 if pallet x was send for depalletization and 0 otherwise. Hence, if pallet x was not send for depalletization, the replenishment quantity from that pallet equals zero.

$$RE^{l}_{i,t,x} \leq \Phi_{i,t,x}\, I^{l}_{i,t,x} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T}, x \in \mathcal{X}_i \tag{5.14}$$

Constraint (5.15) ensures that $\Theta_{i,t,x}$ equals 1 if a partial depalletization has taken place (i.e. $RE^{l}_{i,t,x} \leq I^{l}_{i,t,x}$) and 0 otherwise.

$$\Theta_{i,t,x} \geq 1 - \frac{RE^{l}_{i,t,x}}{\Phi_{i,t,x}\, I^{l}_{i,t,x}} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T}, x \in \mathcal{X}_i \tag{5.15}$$

Constraints (5.16) till (5.18) represent the maximum capacity of the DEPALs, as they can only depalletize a given number of cases (C^{ca}), layers (C^{l}) and pallets (C^{p}) per day. Note that $\alpha\%$ of the capacity is reserved for high priority replenishments.

$$\sum_{i\in\mathcal{I}}\sum_{x\in\mathcal{X}_i} P_i^{ca}\, RE_{i,t,x}^{l} \leq (1-\alpha)\, C^{ca} \qquad \forall t\in\mathcal{T} \tag{5.16}$$

$$\sum_{i\in\mathcal{I}}\sum_{x\in\mathcal{X}_i} RE_{i,t,x}^{l} \leq (1-\alpha)\, C^{l} \qquad \forall t\in\mathcal{T} \tag{5.17}$$

$$\sum_{i\in\mathcal{I}}\sum_{x\in\mathcal{X}_i} \Phi_{i,t,x} \leq (1-\alpha)\, C^{p} \qquad \forall t\in\mathcal{T} \tag{5.18}$$

Finally, Constraints (5.19) till (5.22) define all decision variables as (i) a continuous variable, (ii) a binary variable or (iii) a non-negative integer.

$$\Omega_{i,t} \in [0,1] \qquad \forall i\in\mathcal{I}, t\in\mathcal{T} \tag{5.19}$$

$$\Theta_{i,t,x},\quad \Phi_{i,t,x} \in \{0,1\} \qquad \forall i\in\mathcal{I}, t\in\mathcal{T}, x\in\mathcal{X}_i \tag{5.20}$$

$$I_{i,t}^{ca},\quad I_{i,t}^{ch},\quad BO_{i,t}^{ca},\quad LS_{i,t}^{l},\quad n_{i,t} \in \mathbb{N}_0 \qquad \forall i\in\mathcal{I}, t\in\mathcal{T} \tag{5.21}$$

$$I_{i,t,x}^{l},\quad RE_{i,t,x}^{l} \in \mathbb{N}_0 \qquad \forall i\in\mathcal{I}, t\in\mathcal{T}, x\in\mathcal{X}_i \tag{5.22}$$

5.2.3 Implementation

The model presented in Section 5.2.2 is formulated as a Mixed-Integer Non-Linear Programming (MINLP), as most decision variables are formulated as either an integer or binary variable. The model was implemented on a system with an Intel(R) Core(TM) i5-6200 CPU processor and 8.00 GB RAM memory. The implementation was done in Python using Gurobi, a well-known optimization solver that is able to handle MINLPs. However, Gurobi is not able to handle constraints that include a multiplication of three decision variables or division by a decision variable. Therefore, Constraint (5.15) needed to be slightly adjusted. In fact, a dummy variable $Z_{i,t,x} \in \mathbb{N}_0$ was introduced, such that Constraint (5.15) could be replaced by the following two constraints:

$$Z_{i,t,x} = \Phi_{i,t,x}\, I_{i,t,x}^{l} \qquad \forall i\in\mathcal{I}, t\in\mathcal{T}, x\in\mathcal{X}_i \tag{5.23}$$

$$\Theta_{i,t,x}\, Z_{i,t,x} \geq Z_{i,t,x} - RE_{i,t,x}^{l} \qquad \forall i\in\mathcal{I}, t\in\mathcal{T}, x\in\mathcal{X}_i \tag{5.24}$$

The remaining problem still contains quadratic constraints, which Gurobi handles by the use of McCormick envelopes. That is, the solver applies convex relaxation by replacing each bilinear term (here: $\Phi_{i,t,x}\, I_{i,t,x}^{l}$ and $\Theta_{i,t,x}\, Z_{i,t,x}$) with a new variable and four linear inequality constraints (P. Castro, 2015). As both bilinear terms involve binary variables, the relaxation still provides an exact reformulation to the problem.

The introduced MINLP is formulated as a bi-objective problem. This problem can be solved by applying a pre-defined weight to both objectives, which adds up to one:

$$\min \lambda \sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}}\sum_{x\in\mathcal{X}_i} \Theta_{i,t,x} + (1-\lambda) \sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}} LS_{i,t}^{l} \tag{5.25}$$

If $\lambda = 1$, the model is focused on minimizing the number of partial depalletizations solely. This model results in a solution where only full pallet replenishments are accepted and all other replenishments are denied. Consequently, the solution to this problem results in a minimum number of partial depalletizations and a maximum total unmet demand of the TWH. Conversely, setting $\lambda = 0$ results in a solution where nearly each pallet is partially depalletized. Hence, this solution corresponds to a maximum number of partial depalletizations and a minimum total unmet demand of the TWH. As the solutions to both extrema are not desired, λ should be tuned such that reasonable values for both objectives can be attained. As both objectives have a different unit of measurement, they should be normalized by dividing them by there maximum values.

In total, 6801 SKUs are currently handled via the OPM. Depending on the period length $|\mathcal{T}|$ and the number of pallets that were kept in the HBW during this period for each SKU i (i.e. $|\mathcal{X}_i|$ for all i), the number of decision variables can become extremely large. In fact, up to 70 pallets of the same SKU can be simultaneously present in the HBW. Consequently, solving an MINLP such as the one formulated in Section 5.2.2 generally takes a lot of time and might even become intractable. To limit the computational time of the problem, the model was terminated if an optimality gap of at most 3% was reached. That way, the problem was limited from very slow convergence to optimality, but close to optimal solutions could still be attained. Furthermore, all integer decision variables were relaxed into non-negative continuous decision variables, except for the number of batches $n_{i,t}$ and the number of occupied channels I^{ch}. This way, it was still guaranteed that the ordered quantity at the TWH was expressed in full layers and the occupied channels were assigned correctly. Due to the fact that all $n_{i,t}$ variables were kept integer, the values for all other decision variables were often expressed as integers as well. Still, some decision variables were expressed as floats. As this is not a possibility in reality (e.g. depalletize half a layer or store half a case), the results of the model require some pre-processing to ensure feasibility of the non-relaxed model. To achieve this, all floats in the obtained solutions were rounded to the nearest integer, and all decision variables for subsequent days were re-defined based on the rounded values.

Besides this modification to limit the computational time, the model size of the problem was limited as well. As shown in the previous section, the batch sizes $Q^{ca}_{i,t}$ used in the replenishment logic preferably correspond to a full pallet. All SKUs with a batch size corresponding to a full pallet on day t, will only have fully depalletized pallets during that day. Hence, pallets of these SKUs already attain the minimum value of the objective presented in Equation (5.2). Therefore, it was decided to leave these SKUs (represented with the set $\mathcal{I}_{fp}$) out of scope of the mathematical model. For this set of SKUs, the standard (R, s, nQ) logic was followed to find replenishment quantities. No replenishment was postponed to a later day (provided that there is sufficient inventory in the HBW). Consequently, for these SKUs the minimum value for the objective presented in Equation (5.3) is attained as well. It should be noted that even though these SKUs attain their individual minima for both objectives, this does not necessarily imply that this solution also corresponds to the global minimum. It might be the case that postponing the order quantity of one of these SKUs results in less partial depalletizations for several other SKUs, yielding a better global solution. Still, removing these SKUs from the model decreases the computational time significantly, such that this decision was still made.

Replenishment for the SKUs $i \in \mathcal{I}_{fp}$ requires TWH and DEPAL capacity. The used capacity for these replenishments was subtracted from the parameters K^{ch}, C^{ca}, C^{l} and C^{p} (after subtracting α% and β% of capacity), to find the remaining capacity for all other replenishments.

Preferably, the MINLP is solved for a long time period $|\mathcal{T}|$, such that it reflects long term steady-state solutions. However, given the computational complexity of the problem, the MINLP was solved for a limited time period of $|\mathcal{T}| = 2$ days at once. Then, the model was sequentially solved for six periods of two days (corresponding to two operating weeks), such the effect over time could still be measured. Hence, the model was implemented as a rolling horizon, where the final inventory in the TWH and HBW of a previous period was used as the start inventory for the following period. Moreover, the final backordered customer demand of a previous period was added to the demand for the following period. As the end inventories and backordered demand differ per λ, they should therefore be set accordingly. In Appendix I, a pseudo-code is included for the implementation of this rolling horizon.

Even though the rolling horizon allows to measure the effect of the model over time, solving the MINLP for a time period of $|\mathcal{T}| = 2$ days will result in short-term optimal solutions. These solutions can be used to check whether the number of partially depalletized pallets can be (significantly) decreased and how it relates to the total unmet demand of the TWH. However, considering a longer time period, will more likely result in better solutions on the longer term. To evaluate the effect of lengthening the time period, the MINLP was solved for a time period of $|\mathcal{T}| = 4$ days as well. To guarantee that this MINLP could still be solved within a reasonable amount of time, it was implemented on a small subset of 10% of all SKUs solely.

Lastly, it should be noted that the objective reflected in Equation 5.3 considers the total unmet demand of the HBW. That is, it considers (i) the part of demand that could not be met due to insufficient stock in the HBW (i.e. $LS^l_{i,t} > 0$ because $\sum_{x \in \mathcal{X}_i} I^l_{i,t,x} < n_{i,t}\, Q^{ca}_{i,t}$) and (ii) the part of demand that is postponed by the model due to limited capacity in the TWH (because other pallets were fully depalletized at once). As the number of postponed layers are of particular interest in the current analysis, the part of demand that could no be met due to insufficient stock was subtracted from the objective value. The remainder of this section will focus on the number of postponed layers solely, unless stated otherwise.

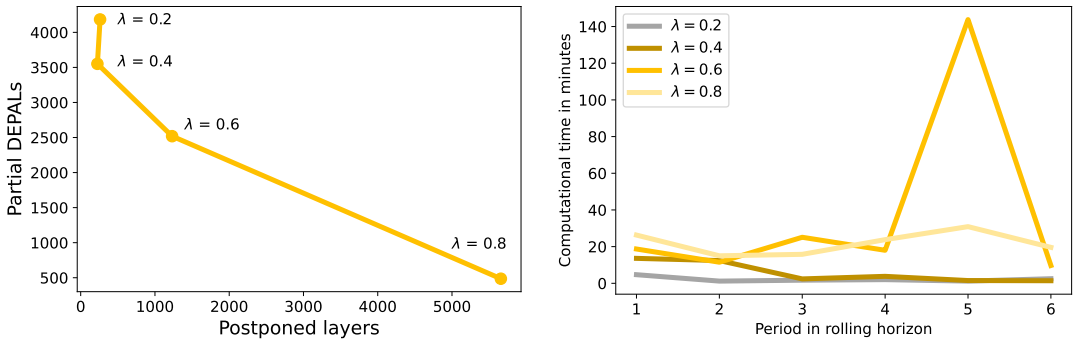
5.2.4 Results

For the baseline model on the entire set $\mathcal{I}$ and $|\mathcal{T}| = 2$, the following parameter settings were used: $RFP = 5$ days, $RPP = 3$ days, $\alpha = 0.05$ and $\beta = 0.03$. These values correspond to currently used settings or are believed to be reasonable values. The effect of all parameters was tested and will be discussed later in this section. Furthermore, a separate MINLP was solved for $\lambda \in [0, 0.2, 0.4, 0.6, 0.8, 1]$. With $RFP = 5$, the batch size of 1940 SKUs corresponded to a full pallet during the entire period of interest (i.e. $|\mathcal{I}_{fp}| = 1940$). These SKUs were kept out of the scope of the MINLP and the standard (R, s, nQ) logic was applied to find the daily replenishment quantities. The pre-processed solutions of the MINLP were added to these solutions to find the complete solution for all SKUs $i \in \mathcal{I} \cup \mathcal{I}_{fp}$.

Efficient Frontier Baseline Model

Figure 5.2a depicts the results of the baseline model for different values of λ, for the complete two week operating scheme (i.e. twelve subsequent days). The values for $\lambda = 0$ and $\lambda = 1$ are omitted in the figure, as the extrema were far out of range of all other solutions. The figure clearly shows the trade-off between the two objectives: to achieve a lower value for one objective, a higher value for the other should be accepted. Furthermore, changing $\lambda = 0.6$ to $\lambda = 0.8$ yields a very large difference in solutions: the number of partial depalletizations decreases drastically, but also causes a large increase in the number of postponed layers. Exploring more solutions on this line might be interesting, to better reflect the trade-off between the two objectives.

Noteworthy, the figure also shows that the solution of the MINLP with $\lambda = 0.4$ outperforms the solution with $\lambda = 0.2$, as it achieves a lower value on both objectives. This effect can be explained by the fact that the figure visualises the number of postponed layers (as explained in the previous section), whereas the mathematical model minimizes the total unmet demand. In fact, the total unmet demand is 2.8% higher for the MINLP with $\lambda = 0.4$ than for the one with $\lambda = 0.2$. Apparently, the MINLP with $\lambda = 0.4$ makes slightly different choices, which eventually results in a lower number of postponed layers. The figure shows that at this small increase in unmet demand (and thus a small decrease in



(a) Efficient Frontier

(b) Computational Time

Figure 5.2: Results of MINLP with Baseline parameter settings.

postponed layers), the number of partial depalletizations could be decreased with 15.1%. Hence, at a small increase (decrease) in unmet demand (postponed layers), the number of partial depalletizations can already decrease significantly.

Considering this effect, it would make sense to solely minimize the number of postponed layers instead of the total unmet demand, as this is the variable of interest. However, to achieve this, the variable $LS^l_{i,t}$ should be splitted in two distinct variables, where one reflects the postponed layers of a given pallet (i.e. additional three-dimensional variables) and the other reflects the part of demand that could not be met from stock (i.e. additional two-dimensional variables). Moreover, additional constraints (and additional binary variables) would be required, to ensure that the parts of TWH demand that are not met are assigned correctly to these variables. Hence, modeling the problem this way increases the complexity of the problem (and so the computational time) enormously. Therefore, it was decided to minimize the total unmet demand instead. The solutions of the $\lambda = 0.2$ and $\lambda = 0.4$ are closely related, so that this behavior could happen. However, for all other solutions, the number of postponed layers and the total unmet demand increases as λ increases, being in line with expectations.

In the actual replenishment strategy, 4759 pallets were partially depalletized during the two weeks of operation. Consequently, all solutions obtained by the MINLP reduced the number of partial depalletizations as compared to the actual replenishment strategy. The actual number of postponed layers could not be determined, as the reorder levels used in the MINLP were approximations of the (unknown) reorder levels used in the actual analysis (as explained in Appendix H).

Besides the efficient frontier, Figure 5.2b depicts the computational time of solving each individual MINLP, for six periods in total. As stated earlier, the program was terminated if an optimality gap of at most 3% was reached. The figure shows that the computational time increases as λ increases. Apparently, shifting to a solution with less partial depalletizations is more computationally expensive than a solution with less postponed layers. Still, the optimality gap prevented slow convergence of the model, as all models could be solved within 45 minutes.

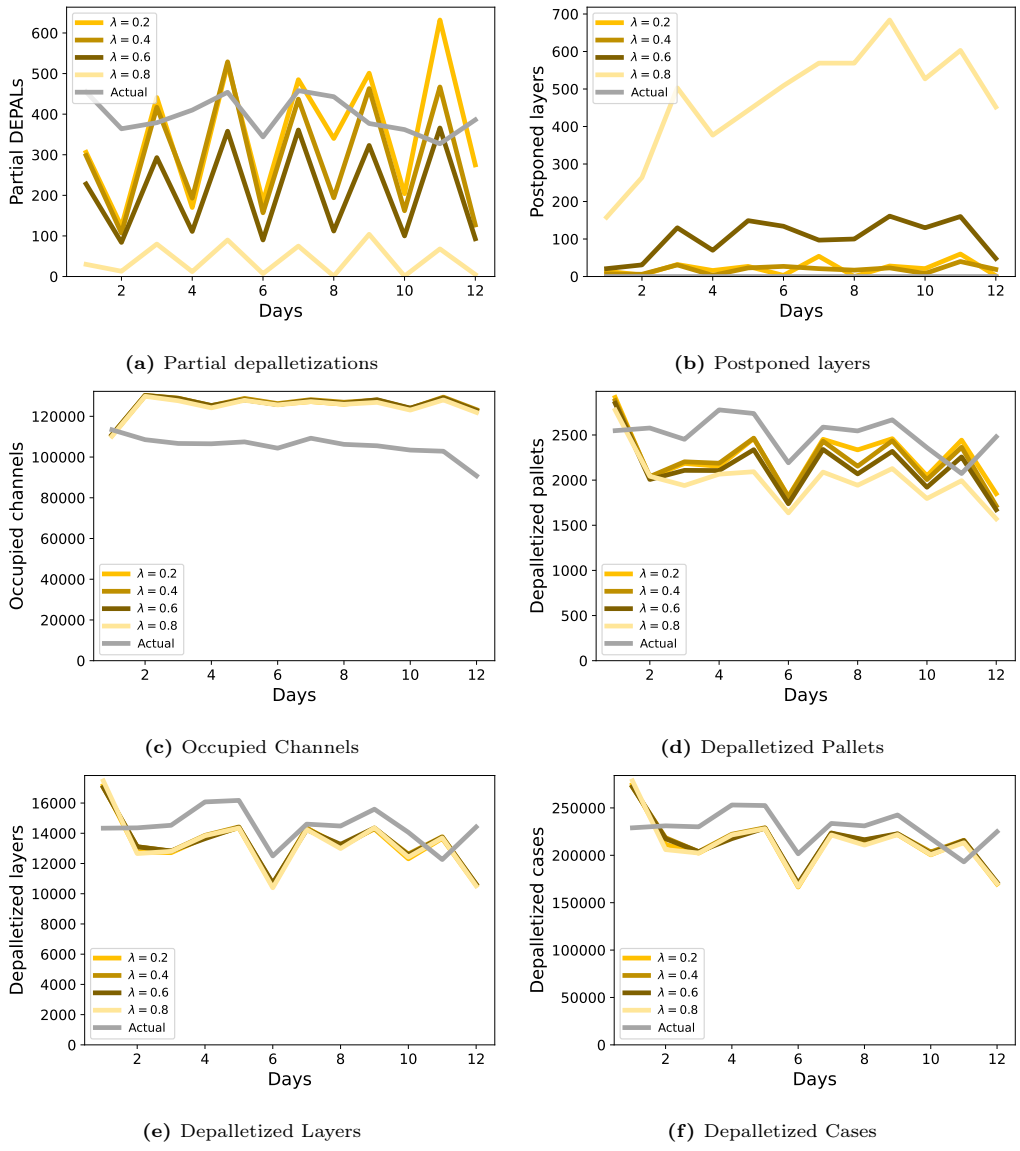
Key Performance Indicators

Figure 5.3 shows the behavior of the actual replenishment strategy and the MINLP solutions over time, for six Key Performance Indicators (KPIs). To start, Figure 5.3a depicts the number of partial depalletizations per day. The MINLP finds solutions where most pallets are partially depalletized at the beginning of the period and few pallets at the end of the period. This is a sensible effect, as a partially depalletized pallet in the beginning of a period can be fully depalletized at a following day. In contrast, the number of partial depalletizations remains more stable throughout the days in the actual replenishment strategy. Increasing the number of days in one period (i.e. increase $|\mathcal{T}|$), will likely result in a more smooth pattern as well, as the partial depalletizations will probably be distributed more evenly throughout the period.

Figure 5.3b shows the number of postponed layers per solution and clearly depicts the large difference between the solution of the MINLP with $\lambda = 0.8$ and all other solutions. For this model, the number of postponed layers appears to increase over time, while this is not true for all other models. Hence, this solution keeps postponing layers to later days, such that some replenishments might be postponed endlessly. This behaviour is not desired and therefore the solution to this MINLP is not applicable in reality.

Figures 5.3c till 5.3f show the used TWH and DEPAL capacity of the solutions. Not surprisingly, the MINLP pushes (at least) one of these capacities to its maximum value, as the capacity constraints limit the MINLP from fully depalletizing all pallets. All solutions show that the TWH capacity was pushed to its maximum value on all days. Figures 5.3d till 5.3f show that a large number of replenishments are performed on the first day, such that the inventory in the TWH (Figure 5.3c) reaches its maximum value immediately on the second day. From that point onward, the capacity in the TWH remains close to the maximum value, whereas the number of replenished pallets, layers and cases decrease to a lower rate. Remarkably, the behavior of the MINLP with different values of λ is nearly the same for all KPIs, except for the number of depalletized pallets. As λ increases, less pallets were send for

depalletization, but more pallets were fully depalletized. As such, all solutions still depalletize a similar number of layers and cases. Finally, these figures show that the actual replenishment strategy uses less TWH capacity (as was already clear from figure 1.6), but in many periods more DEPAL capacity. Due to the fact that the MINLP pushes a lot of inventory towards the TWH on the first day, less replenishments could be performed on subsequent days. Hence, the optimal solution of the MINLP for the first period, turns out to be sub-optimal over the entire period. That is, a lot of (slow moving) inventory was sent to the TWH on the first day, that limits the model from replenishing pallets on later days. If the number of days in one period $|\mathcal{T}|$ increases, the optimal solutions obtained in each individual period will probably be more optimal on the long-term as well.



(a) Partial depalletizations

(b) Postponed layers

(c) Occupied Channels

(d) Depalletized Pallets

(e) Depalletized Layers

(f) Depalletized Cases

Figure 5.3: Effect over time of MINLP with Baseline parameter settings, for six KPIs.

Sensitivity Analysis

To test the effect of the parameters used in the MINLP, four additional models were tested with different settings, as depicted in Table 5.2. To start, the RFP was increased to six days, such that the batch size of more SKUs was fixed to full pallets. In fact, 334 additional SKUs were left out of scope of the MINLP in this setting. Then, a model was tested were the batch size of each SKU equalled either (i) a full pallet or (ii) one layer, by setting RPP to zero days. Furthermore, the effect of increasing α and β, and thus

decreasing the TWH and/or DEPAL capacity, was tested. The obtained results of the MINLPs with different parameter settings are visualized in Figure 5.4.

From the figure can be derived that the model with an RFP equal to six days outperforms all other models. That is, fixing the batch size of more SKUs to full pallets resulted in both a decrease in partial depalletizations, as well as a decrease in postponed layers, as compared to the baseline model. In contrast, increasing α and β provided worse results for both objectives. This is a sensible effect, as both parameters limit the TWH capacity, which was binding in all solutions with the baseline settings. Conversely, a reduction in (one of) these values will likely result in better solutions. Finally, the model with an RPP of zero days performed comparable or slightly better than the baseline model.

Model	RFP	RPP	α	β
Baseline	5 days	3 days	0.05	0.03
Test RFP	6 days	3 days	0.05	0.03
Test RPP	5 days	0 days	0.05	0.03
Test α	5 days	3 days	0.08	0.03
Test β	5 days	3 days	0.05	0.05

Table 5.2: Parameter settings for different MINLPs

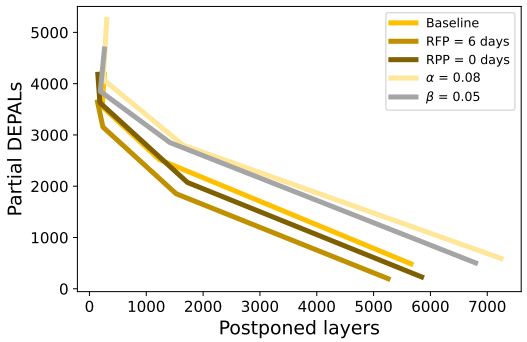


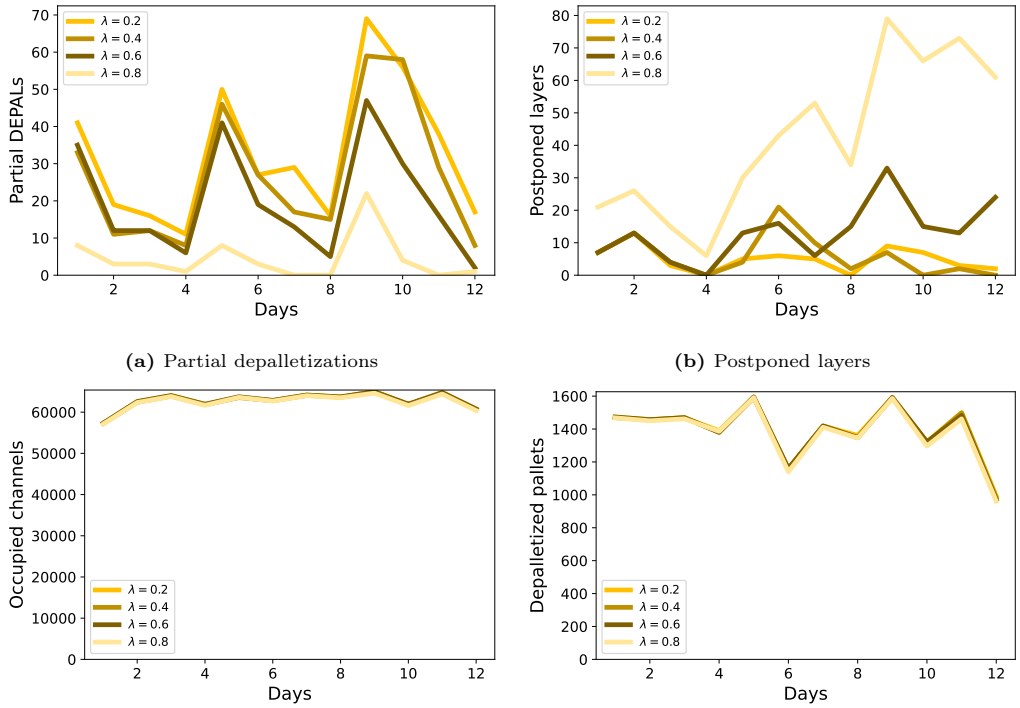
Figure 5.4: Efficient Frontiers of MINLPs with different parameter settings

Besides the effect of these parameters, the behavior of the MINLP was analysed over a longer period of time of $|\mathcal{T}| = 4$ days. As indicated in the previous subsection, this model was solved for a random subset of 10% of all SKUs. To obtain the solutions for a two week operating scheme, a separate MINLP was solved for three consecutive periods of four days, using the rolling horizon algorithm depicted in Appendix I. As only 10% of all SKUs were included in the model, the TWH and DEPAL capacities were adjusted with the same ratio. The new MINLP was implemented using the parameter values of the baseline model. Figure 5.5 shows the effect of four KPIs over time. Similar as for the MINLPs with $|\mathcal{T}| = 2$, most pallets were partially depalletized at the beginning of a period (i.e. on day 1, 5 and 9) and fewer pallets at the end of a period. The number of postponed layers also shows a similar behaviour, as they increase over time in the solution obtained by $\lambda = 0.8$. In this solution, the same seems to happen in the solution obtained by $\lambda = 0.6$, but this could also be due to the particular SKUs in the random subset.

In line with expectations, Figure 5.5c shows that the MINLP pushes the TWH capacity immediately to its maximum value, which remains constant on all days. This behavior is similar as in the baseline model, shown in Figure 5.3c. However, Figure 5.3d showed that a lot of pallets were replenished on the first day, such that fewer pallets could be replenished on later days. Figure 5.5d shows that this behavior can be prevented with a larger time period $|\mathcal{T}|$. That is, even though the model again immediately pushes the TWH capacity to its maximum value, it makes different (and better) replenishment choices on the first day, such that the replenishment rate can remain more stable. Hence, this indicates that using a larger $|\mathcal{T}|$ results in a more beneficial long term solution.

Prediction model

Finally, the MINLP was introduced to minimize the number of partial depalletizations, as the prediction models in Chapter 4 showed that this feature adversely affects the number of fallen cases. To test whether a reduction in partial depalletizations indeed results in less predicted pallets with fallen cases, the actual values for the *Number of Calls DEPAL* feature were substituted with the values obtained by the MINLP with $\lambda = 0.4$ or $\lambda = 0.6$ and the parameter settings of the baseline model. First, a new (test) dataset was build for the two weeks of operation according to the criteria listed in Section 3.1. Additionally,



(a) Partial depalletizations

(b) Postponed layers

(c) Occupied channels

(d) Depalletized pallets

Figure 5.5: Effect over time of MINLP with Baseline parameter settings, for six KPIs, for a small subset of SKUs and a larger time period $|\mathcal{T}|$.

pallets that were not replenished in both MINLP solutions were removed from the dataset, to allow for accurate comparison between the three strategies. Ultimately, a dataset with 14 377 observations was left, of which 3 419 pallets were partially depalletized in the actual replenishment strategy. The number of calls for each pallet $x \in \mathcal{X}_i$ of SKU $i \in \mathcal{I}$ from the MINLP solutions was obtained as follows:

$$\textit{Number of Calls DEPAL MINLP} = \sum_{j \in \mathcal{J}} \sum_{t \in \mathcal{T}} \Phi_{i,t,x} \tag{5.26}$$

where $\mathcal{J}$ refers to the set of periods for which the model was consecutively solved. Hence, the total number of calls were summed over the entire period of twelve days. For each pallet in the dataset of a given SKU i, a pallet $x \in \mathcal{X}_i$ in the MINLP solution was randomly assigned, where each pallet could only be assigned once. Substituting the actual *Number of Calls DEPAL* with the *Number of Calls DEPAL MINLP* values, decreased the number of partial depalletizations significantly, as shown in Table 5.3. This large decrease can be devoted to the fact that the dataset contains only pallets that were fully depalletized (in one or more calls) during the time period of two weeks. As such, these numbers show that the actual replenishment strategy contains many partially depalletized pallets that were fully depalletized on quite short notice (i.e. within two weeks). In contrast, the time between subsequent calls in the solution of the MINLP appears to be much greater in general, as much fewer pallets were fully depalletized in multiple calls within this period.

To obtain the predictions, the best RF model was trained on the same training dataset as in Chapter 4. Then, the new test dataset with the actual *Number of Calls DEPAL* feature was first fed into the model, to obtain corresponding predictions. Subsequently, the dataset with the substituted *Number of Calls DEPAL MINLP* feature for both $\lambda = 0.4$ and $\lambda = 0.6$ was fed into the model, to obtain new predictions for comparison. The number of predicted pallets with fallen cases are listed in Table 5.3, which shows that a decrease in partial depalletizations indeed results in a 15.9% and a 17.1% decrease in positive predictions for $\lambda = 0.4$ and $\lambda = 0.6$, respectively. Hence, this confirms that a decrease in partial depalletizations also results in a decrease in fallen cases.

Table 5.3: Predictions obtained by best RF model on dataset with actual *Number of Calls DEPAL* feature and substituted *Number of Calls MINLP* features

	Number of partial depalletizations	Predicted pallets with fallen cases	Improvement
Number of Calls DEPAL	3419	3870	
Number of Calls DEPAL MINLP $\lambda = 0.4$	941	3256	-15.9%
Number of Calls DEPAL MINLP $\lambda = 0.6$	772	3207	-17.1%

Table 5.4: Performance of actual replenishment strategy and MINLP with Test RFP parameter settings and $\lambda = 0.4$ or $\lambda = 0.6$

	Number of partial depalletizations	Number of postponed layers
Actual Replenishment strategy	4759	?
Strategy of MINLP model with $\lambda = 0.4$	3154	237
Strategy of MINLP model with $\lambda = 0.6$	1853	1526

Conclusion

To conclude, the introduced MINLP proved to find solutions with fewer partial depalletizations than the actual replenishment strategy. Even at only a limited increase in postponed layers, the number of partial depalletizations could already decrease significantly. However, the solution from the MINLP with $\lambda = 0.8$ showed that a model that focuses too much on minimizing the number of partial depalletizations is not desired, as the number of postponed layers then increases over time, which in turn decreases the service level of the TWH. Furthermore, the efficient frontiers depicted in Figures 5.2a and 5.4 showed that MINLPs with $\lambda = 0.4$ outperformed the solutions obtained by $\lambda = 0.2$, as the number of partial depalletizations decreases significantly at only a small increase in unmet demand (and potentially even a decrease in postponed layers).

Furthermore, Figure 5.4 showed that the MINLP with $RFP = 6$ days was able to decrease both the number of partial depalletizations as well as the number of postponed layers as compared to the baseline model. Considering the solutions obtained by this model, especially the MINLPs with $\lambda = 0.4$ or $\lambda = 0.6$ found reasonable solutions, as they both decreased the number of partial depalletizations drastically, against a limited increase in postponed layers, as shown in Table 5.4. The choice between these solutions depends on the desired trade-off between the two objectives: less partially depalletized pallets results in less fallen cases, whereas less postponed layers results in a higher service quality of the TWH.

Solving the model for a longer time period $|\mathcal{T}|$ indicated to find better long term solutions. However, this comes at the cost of increased computational time. Even though the models with $|\mathcal{T}| = 2$ days find short-term optimal solutions, they already showed that the number of partial depalletizations can be significantly decreased without yielding a high increase in the number of postponed layers. Moreover, these MINLPs could be solved within a reasonable amount of time for the entire set of SKUs $i \in \mathcal{I}$. Therefore, these models are believed to provide valuable starting points.

Finally, Table 5.3 showed that by solely decreasing the *Number of Calls DEPAL* feature, the predicted pallets with fallen cases was already decreased by more than 15%. These results thus confirm that minimizing the number of partial depalletizations is effective in minimizing the number of fallen cases.

Chapter 6

Discussion

Within this final chapter, each research question is addressed separately. For each research question, the findings and the conclusions are discussed, some recommendations for Jumbo are provided, several limitations of the study are highlighted and directions for future work are suggested.

6.1 Research Question 1: Current size of the problem

Chapter 3 described a data collection method to estimate the number of fallen cases between inbound and tray merge, given the available data sources. The system was considered as a black box, where its input and output streams were compared to estimate the number of fallen cases. These estimates indicated that nearly 1 000 cases fell between inbound and tray merge, costing Jumbo more than €10 000 on a weekly basis. On top of that, additional cleaning cost were incurred to fix the consequences of the fallen cases. Moreover, the fallen cases potentially caused machine downtime and a decreased service quality. The estimates also indicate that Jumbo currently highly underestimates the number of fallen cases, as Jumbo's current (manual) registrations indicated that 200 cases fell on a weekly basis in the entire CDC. The collected data thus confirmed the problem of Jumbo's CDC in Nieuwegein and show that the problem is much greater than initially thought. Hence, the results emphasized the need to establish methods to decrease the number of fallen cases.

The distribution of all features was analysed in Chapter 3, for the minority and majority class separately. These analyses showed that pallets in the minority were partially depalletized more often than pallets in the majority class. Moreover, SKUs in the minority class tend to be small, such that more cases fit on a single pallet. These findings are closely related, as pallets with more inventory on it are more likely to be partially depalletized. Therefore, the data analyses confirmed that this step in the process is important to consider when minimizing the number of fallen cases.

The data collection method still shows some flaws, due to the discrepancy in the re-introduction streams. However, given the available data sources, it was not possible to overcome these flaws. Cases are re-introduced to the system at different steps in the process, for different reasons, even though this is currently not visible in the data. The accuracy of the data collection method can be improved by adding this information to report *IN32*, such that it is clarified why and where cases are re-introduced to the system. That way, the re-introduced cases can easily be linked to a pallet, such that the number of fallen cases can be estimated more accurately.

The data collection was focused on estimating the number of fallen cases between inbound and tray merge solely. However, a similar approach, where part of the system is considered as a black box and physical measures are used as input and output streams, could be applied to other processes in the CDC as well. As such, the number of fallen cases could be estimated between tray merge and the COMs, between the HBW and the repack stations (i.e. the manual depalletizers for the DPS subsystem) and between the repack stations and order picking in the DPS as well. Consequently, all estimates would be based on data generated by the system, instead of relying on manual registrations, which appeared to

highly underestimate the number of fallen cases. Using this approach, Jumbo can better monitor how many cases fell and what the effect of potential migitation strategies is.

6.2 Research Question 2: Prediction Model

In Chapter 4, two prediction models were introduced to accurately predict whether cases will fall off a new pallet entering the system: an RF and an MLP. For both prediction models, several hyperparameter values were tested to find the best out-of-sample predictions. In addition, Resampling and CSL were introduced as methods to account for the imbalanced dataset, to further improve the predictions of the ML techniques. The best performing RF models were used to perform feature selection, to find the subset of features most relevant in predicting the outcome feature. Moreover, this subset of features was analysed to find SKU-, process-, and machine related characteristics that increase the likelihood of cases falling.

The results emphasized the importance of implementing a method to account for the imbalanced data, as both the RF as MLP models were not able to distinguish the minority class observations otherwise. Moreover, applying a Resampling method to the dataset beforehand proved to work better than incorporating CSL in both prediction models. Furthermore, the best RF model (fit on a fully undersampled dataset) was preferred over the best MLP model (fit on a fully oversampled dataset). First, the RF model achieved a higher accuracy and AUC value than the MLP model. In addition, an RF requires little tuning, whereas many parameters of an MLP need to be set by the user. Therefore, the RF is easier to implement an preferred over an MLP.

The best RF model showed that using only a small subset of all features still yields reasonable predictions. From this subset could be derived that especially the *Number of Calls DEPAL* feature is important in predicting the likelihood of falling. Additionally, many features in this subset relate to the size of a single case and the number of cases on a pallet. Apparently, smaller cases and (consequently) more cases on a pallet increase the likelihood of falling. As stated in Chapter 4, this effect can be devoted to two things: (i) a pallet with many small cases is likely to be more unstable and (ii) pallets with more inventory are more likely to be partially depalletized. Hence, these two findings go hand in hand: pallets with more inventory are more likely to partially depalletized, which reduces the stability of the pallet and thereby increases the likelihood of falling. To minimize the number of fallen cases, the number of partial depalletizations should thus be minimized. Moreover, the stability of such pallets should be ensured at all times. That is, the stability of the pallet should be judged at different steps in the process and, if necessary, improved by adding additional stretch wrapping or re-stacking the pallet. Finally, the *Weight Difference Pallet* and *Height Difference Pallet* appeared to increase the likelihood of falling. A potential improvement could be to use lower values for the *Weight Tolerance* and *Height Tolerance* features, such that pallets with higher differences between physical- and expected measures are more often manually checked. With these additional checks, stability of the pallets and correctness of the data can be ensured. However, as lowering these tolerance levels results in an increase in manual labour, a balance should be found between performing too little (i.e. possibly resulting in fallen cases) and too much (i.e. labour-intensive) checks.

From the results could be derived that both ML techniques were not able to achieve an AUC score above 68% on the test set, such that both the recall and accuracy could also retain a higher level. Potential reasons for this could be the level of noise in the data. Occasionally, cases fall off a pallet due to an incidentally badly stacked pallet or just bad luck. Such situations are a one-time occasion, for which no particular reason will be present in the data. These observations are hard to filter from the data, but also limit the models to properly learn the underlying rules that do affect the probability of cases falling. Yet, this emphasizes the need to ensure stability of the pallets, by potentially performing additional manual checks.

Besides the noise in the data, another reason could be that many features in the dataset are ob-

tained from master data and SKU specific. That means that the values for these features are equal for all pallets of that particular SKU. Now it is true that for all SKUs having pallets from which cases fell off, other pallets of the same SKU were handled without cases falling off. In other words, all SKUs having observations in the minority class, also have observations in the majority class. As such, the dataset contains several very similar observations with different outcome labels, making it harder for the ML techniques to assign these observations to the right class. Consequently, the presence of double labels worsens the ability of the models to learn the underlying rules. One potential way to handle this could be to assign all SKUs to the class where most observations belong to and remove all observations of that SKU in the other class. However, for the current dataset, only 3% of all SKUs with observations in both classes have more observations in the minority class than in the majority class. Removing all observations with double labels, would result in an even more imbalanced dataset. Therefore, all observations were kept in the dataset.

In total, 120 RF and 243 MLP models were implemented, with different hyperparameter combinations. However, the number of potential hyperparameter combinations was not exhausted, and many more models could be tested. Especially considering an MLP, the effect of adding additional hidden layers, using different activation or cost functions or a different stop criterion could be tested. Besides different hyperparameter values, different methods to account for the imbalanced dataset could be tested as well. For instance, a combination of several oversampling and undersampling methods could be implemented, to test whether the performance of the ML techniques can be improved even further.

Another possibility would be to fit the prediction models on more different subsets of features. The best RF model was fit on the full model and on three subsets of relevant features of different size. In contrast, the MLP was only fit on the subset of features listed in 4.6. However, it is yet unknown whether different subsets of features would result in a higher performance for both models. Therefore, using different feature selection methods to select subsets of relevant features might be interesting. Lastly, the problem was currently modeled as a binary classification problem. Alternatively, more classes could be distinguished (e.g. no fallen cases, few fallen cases and many fallen cases) or the the problem could be targeted as a regression problem, to check if such prediction models are better able to predict the (number of) fallen cases.

6.3 Research Question 3: Minimize Partial Depalletizations

As the results of the previous two research questions showed that the number of partial depalletizations adversely affects the number of fallen cases, Chapter 5 introduced a mathematical model to minimize the number of partial depalletizations. The mathematical model was build to make replenishment decisions of the TWH on a daily basis and used the (R, s, nQ) logic to find initial order quantities. Then, the model adjusts these order quantities, such that the total number of partial depalletizations is minimized. Preferably, the order quantities are rounded up to full pallets, but a lower replenishment quantity is allowed as well. In such circumstances, the model decides to postpone several ordered layers, such that other pallets can be fully depalletized. However, as a decreased order quantity potentially worsens the service quality of the TWH, the model also minimizes the total unmet demand of the TWH (including the number of postponed layers).

The resulting mathematical model was formulated as a bi-objective MINLP. To limit the computational time of the model, the set of SKUs $\mathcal{I}_{fp}$ having a batch size equal to a full pallet during the entire period of interest was left out of scope of the MINLP. The standard (R, s, nQ) logic explained in Section 2.3 was followed to find replenishment quantities for these SKUs, which were eventually added to the solution obtained by the MINLP. In addition, the MINLP was solved for a time period of $|\mathcal{T}| = 2$ days at once. Then, the model was sequentially solved for six periods of two days using the rolling horizon algorithm depicted in Appendix I, such that the effects over time could still be measured. Even though the individual MINLPs find short-term optimal solutions, the results still provide valuable insights in

whether the number of partial depalletizations can be decreased and what the effect is on the total unmet demand of the TWH.

The MINLP proved to find solutions with fewer partial depalletizations as compared to the actual replenishment strategy. Even at a limited number of intentionally postponed layers, the number of partial depalletizations could already decrease significantly. However, the results also showed that putting too much weight on the minimization of partial depalletizations is not desired as well, as this results in a high number of postponed layers over time. Eventually, this will decrease the service level of the TWH drastically. Especially the MINLPs with $\lambda = 0.4$ and $\lambda = 0.6$ found reasonable solutions, as Table 5.4 showed that the resulting replenishment strategies significantly decreased the number of partial depalletizations against a limited increase in postponed layers. The preferred solution should be based on the desired trade-off between these two objectives: less partially depalletized pallets results in less fallen cases, whereas less postponed layers results in a higher service quality of the TWH. To better explore this trade-off, more solutions could be obtained with $0.4 < \lambda < 0.6$.

Furthermore, a sensitivity analysis was performed where different values were tested for all parameter values used in the MINLP. These results showed that increasing the RFP appeared to be beneficial, as the solutions to these MINLPs achieved lower values on both objectives. Conversely, an increase in α and β provided worse results for both objectives. This is a sensible effect, as both parameters limit the TWH capacity, which was binding in all solutions with the baseline settings. Hence, lower values for these parameters would likely yield more beneficial results. Alternatively, the capacity of the TWH might be used more efficiently by removing ample stock of slow movers. That way, more space is available to reduce the number of partial depalletizations and the number of postponed layers for all SKUs with higher demand rates. It should be noted that all parameters were only individually changed and the effect of changing multiple parameters at once was not tested. To further explore the effect of these parameters and/or better tune them, a more extensive sensitivity analysis would be needed.

The results also showed that MINLPs solved for a longer time horizon $|\mathcal{T}|$ yield better long-term solutions. Due to the complexity of the model, a time period of two and four days was tested solely. However, in the ideal situation, an even longer period would be considered, to find better steady-state solutions. One potential way to model this, would be to aggregate t to multiple days or weeks, such that the length of set $\mathcal{T}$ does not increase significantly. Alternatively, besides removing the pallets with a batch size equal to a full pallet, slow movers could be left out of scope of the MINLP as well, as the order quantity for these SKUs often corresponds to only one layer. Then, the MINLP could be used to minimize the number of partial depalletizations for all other SKUs, for a longer period of time.

To test if a reduction in the number of partial depalletizations indeed results in a decrease of fallen cases, the *Nr of Calls DEPAL* feature was substituted with the number of calls obtained by the MINLP solutions. The substituted dataset was fed into the best prediction model of Chapter 4, which showed that the number of positively predicted observations decreased with more than 15% by solely reducing the number of partial depalletizations. Hence, this confirmed that the number of fallen cases can be minimized by minimizing the number of partial depalletizations. Moreover, the findings of the previous chapters indicated that some of the relevant features for predicting the outcome variable might interact. For instance, the results showed that smaller cases (and consequently more cases on a pallet) have a higher risk of falling. However, pallets with more inventory on it are also more likely to be partially depalletized. As such, the importance of these features potentially aroused from the same cause, being a partial depalletization. Hence, if the number of partial depalletizations are reduced, the importance of these other features might be diminished as well. In the current analysis, the *Number of Calls DEPAL* feature was substituted solely, whereas the values for all other features was kept constant. However, when the prediction model would be trained on a new dataset with fewer partial depalletizations (and consequently fewer fallen cases), the decrease in positive predictions might be amplified.
Finally, there are several suggestions for future work:

- As indicated, solving the MINLP is computationally expensive and might become intractable when solved for a large number of SKUs or a long time period. Therefore, it might be valuable to

formulate a heuristic that finds a feasible solution which closely approaches to the optimal solution of the MINLP. Then, the heuristic can be used to find good solutions for larger sets of SKUs and longer time periods.

- The MINLP was formulated to make tactical decisions: for each day, it decides which pallets to replenish in what quantities. In reality, such decisions are made at an operational level: at which point in time should a certain pallet be depalletized? To find such decisions, the index t could be reduced from days to a couple of hours. Regarding the unmet demand of the TWH, it might be beneficial to switch to a system with backorders. That way, a small delay (e.g. one time period) can be allowed in fulfilling the TWH demand, in a similar fashion as for the customer demand (which can be fulfilled on the same or following day in the current formulation). However, to switch to a system with backorders, the variable $LS_{i,t}^{l}$ should become pallet specific and requires addition constraints, which increases the size and the complexity of the problem.
- The current MINLP was based on an (R, s, nQ) inventory management system, where the batch size Q was equal to either (i) a full pallet, (ii) a multiple of layers or (iii) one layer. Alternatively, the model could be based on an (R, s, MOQ, IOQ)-policy, where MOQ represents a minimum order quantity and IOQ represents an incremental order quantity (Hill, 2006). That is, if the inventory position drops below the reorder level s, the order quantity equals $MOQ + n\,IOQ$ with n being the minimum number of additional IOQ-batches to increase the inventory position above the reorder level. Especially for SKUs with a batch size corresponding to a multiple of layers, this policy might be more effective. For instance, suppose $Q = MOQ = 3$ layers, and four layers are needed to raise the inventory position above the reorder level again. Then, following the (R, s, nQ) logic, the order quantity would be six layers. However, following the (R, s, MOQ, IOQ) logic with $IOQ = 1$ layer (being the minimum batch size by definition), the order quantity would be set to exactly four layers. Hence, this logic might result in lower order quantities and therefore lower values of unmet TWH demand. As such, the MINLP might likely find a different and potentially more beneficial replenishment strategy. However, the disadvantage of implementing this policy, is the fact that it would increase the complexity of the MINLP, as additional variables and constraints would be needed. Therefore, the increase in complexity should be balanced by a substantial increase in performance.
- Instead of solely considering pushing more inventory to the TWH, an alternative would be to order less layers on a pallet at the supplier. That way, full pallets carry less inventory, requiring less storage space in the TWH. Consequently, more pallets could be fully depalletized. The drawback of this option is that many suppliers only deliver on full pallets. Moreover, the storage space of the HBW would be used less efficiently, as potentially more pallets (with less cases on it) need to be ordered, requiring more capacity.
- The current formulation of the MINLP is based on the assumption that FIFO handling of pallets is not a requirement of the model. As several SKUs have a limited shelf life, the effect of relaxing this assumption could be analysed.
- The MINLP uses historic, deterministic demand, such that the solutions could be compared with the actual replenishment strategy. However, the actual demand will not be known beforehand in reality. Therefore, implementing the model using stochastic demand might provide interesting insights.

References

Abraham, A. (2005). Artificial neural networks. *Handbook of measuring system design*.

Amato, F., Basile, F., Carbone, C., & Chiacchio, P. (2005). An approach to control automated warehouse systems. *Control Engineering Practice*, *13*(10), 1223–1241.

Andriansyah, R., Etman, L., Adan, I. J., & Rooda, J. E. (2014). Design and analysis of an automated order-picking workstation. *Journal of Simulation*, *8*(2), 151–163.

Athey, S., & Imbens, G. W. (2019). Machine learning methods that economists should know about. *Annual Review of Economics*, *11*, 685–725.

Axsäter, S., & Rosling, K. (1993). Installation vs. echelon stock policies for multilevel inventory control. *Management Science*, *39*(10), 1274–1280.

Azadeh, K., De Koster, R., & Roy, D. (2019). Robotized and automated warehouse systems: Review and recent developments. *Transportation Science*, *53*(4), 917–945.

Bernard, S., Heutte, L., & Adam, S. (2009). Influence of hyperparameters on random forest accuracy. In *International workshop on multiple classifier systems* (pp. 171–180).

Bertolini, M., Esposito, G., Mezzogori, D., & Neroni, M. (2019). Optimizing Retrieving Performance of an Automated Warehouse for Unconventional Stock Keeping Units. *Procedia Manufacturing*, *39*, 1681–1690.

Bottou, L., et al. (1991). Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, *91*(8), 12.

Boysen, N., de Koster, R., & Füßler, D. (2021). The forgotten sons: Warehousing systems for brick-and-mortar retail chains. *European Journal of Operational Research*, *288*(2), 361–381.

Boysen, N., De Koster, R., & Weidinger, F. (2019). Warehousing in the e-commerce era: A survey. *European Journal of Operational Research*, *277*(2), 396–411.

Boysen, N., Fedtke, S., & Weidinger, F. (2018). Optimizing automated sorting in warehouses: The minimum order spread sequencing problem. *European Journal of Operational Research*, *270*(1), 386–400.

Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*(2), 123–140.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Brownlee, J. (2022). *Data Preparation for Machine Learning*.

Castro, C., & Braga, A. (2013). Novel cost-sensitive approach to improve the multilayer perceptron performance on imbalanced data. *IEEE transactions on neural networks and learning systems*, *24*(6), 888–899.

Castro, P. (2015). Tightening piecewise McCormick relaxations for bilinear problems. *Computers & Chemical Engineering*, *72*, 300–311.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321–357.

Chawla, N. V., Japkowicz, N., & Kotcz, A. (2004). Special issue on learning from imbalanced data sets. *ACM SIGKDD explorations newsletter*, *6*(1), 1–6.

Chen, F. (2000). Optimal policies for multi-echelon inventory problems with batch ordering. *Operations research*, *48*(3), 376–389.

Chen, F., & Zheng, Y.-S. (1994). Evaluating echelon stock (R, nQ) policies in serial production/inventory systems with stochastic demand. *Management Science*, *40*(10), 1262–1275.

Choi, R. Y., Coyner, A. S., Kalpathy-Cramer, J., Chiang, M. F., & Campbell, J. P. (2020). Introduction to Machine Learning, Neural Networks, and Deep Learning. *Translational Vision Science & Technology*, *9*(2), 14–14.

Claeys, D., Adan, I., & Boxma, O. (2016). Stochastic bounds for order flow times in parts-to-picker warehouses with remotely located order-picking workstations. *European Journal of Operational Research*, *254*(3), 895–906.

Cuadra, C. A., & Katter, R. V. (1967). Opening the Black Box of 'Relevance'. *journal of Documentation*.

Emde, S., Tahirov, N., Gendreau, M., & Glock, C. H. (2021). Routing automated lane-guided transport vehicles in a warehouse handling returns. *European Journal of Operational Research*, *292*(3), 1085–1098.

Faveto, A., Traini, E., Bruno, G., & Lombardi, F. (2021). Development of a key performance indicator framework for automated warehouse systems. *IFAC-PapersOnLine*, *54*(1), 116–121.

Hadley, G., & Whitin, T. M. (1963). *Analysis of Inventory Systems* (Tech. Rep.).

Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., & Bing, G. (2017). Learning from class-imbalanced data: Review of methods and applications. *Expert systems with applications*, *73*, 220–239.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). Random forests. In *The elements of statistical learning* (pp. 587–604). Springer.

Hecht-Nielsen, R. (1987). Kolmogorov's mapping neural network existence theorem. In *Proceedings of the international conference on neural networks* (Vol. 3, pp. 11–14).

Hill, R. M. (2006). Inventory control with indivisible units of stock transfer. *European journal of operational research*, *175*(1), 593–601.

Jain, A. K., Mao, J., & Mohiuddin, K. M. (1996). Artificial neural networks: A tutorial. *Computer*, *29*(3), 31–44.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning* (Vol. 112). Springer.

Janitza, S., Binder, H., & Boulesteix, A.-L. (2016). Pitfalls of hypothesis tests and model selection on bootstrap samples: causes and consequences in biometrical applications. *Biometrical Journal*, *58*(3), 447–473.

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255–260.

Kanellopoulos, I., & Wilkinson, G. G. (1997). Strategies and best practice for neural network image classification. *International Journal of Remote Sensing*, *18*(4), 711–725.

Kaur, H., Pannu, H. S., & Malhi, A. K. (2019). A systematic review on imbalanced data challenges in machine learning: Applications and solutions. *ACM Computing Surveys (CSUR)*, *52*(4), 1–36.

Kavzoglu, T., & Mather, P. M. (2003). The use of backpropagating artificial neural networks in land cover classification. *International journal of remote sensing*, *24*(23), 4907–4938.

Kingsford, C., & Salzberg, S. L. (2008). What are decision trees? *Nature biotechnology*, *26*(9), 1011–1013.

Ko, D., & Han, J. (2022). A rollout heuristic algorithm for order sequencing in robotic compact storage and retrieval systems. *Expert Systems with Applications*, 117396.

Kolen, J., & Pollack, J. (1990). Back propagation is sensitive to initial conditions. *Advances in neural information processing systems*, *3*.

Krawczyk, B., Woźniak, M., & Schaefer, G. (2014). Cost-sensitive decision tree ensembles for effective imbalanced classification. *Applied Soft Computing*, *14*, 554–562.

Krogh, A. (2008). What are artificial neural networks? *Nature biotechnology*, *26*(2), 195–197.

Küçükyaşar, M., Ekren, B. Y., & Lerher, T. (2021). Energy efficient automated warehouse design. In *Solving urban infrastructure problems using smart city technologies* (pp. 269–292). Elsevier.

Kuhn, M., & Johnson, K. (2019). *Feature Engineering and Selection: A Practical Approach for Predictive Models*. CRC Press.

Kuhn, M., Johnson, K., et al. (2013). *Applied Predictive Modeling* (Vol. 26). Springer.

Li, C., Zhang, L., & Zhang, L. (2022). A route and speed optimization model to find conflict-free routes for automated guided vehicles in large warehouses based on quick response code technology. *Advanced Engineering Informatics*, *52*, 101604.

Lippmann, R. (1987). An introduction to computing with neural nets. *IEEE Assp magazine*, *4*(2), 4–22.

López, V., Fernández, A., García, S., Palade, V., & Herrera, F. (2013). An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information sciences*, *250*, 113–141.

Loyola-González, O., Martínez-Trinidad, J. F., Carrasco-Ochoa, J. A., & García-Borroto, M. (2016). Study of the impact of resampling methods for contrast pattern based classifiers in imbalanced databases. *Neurocomputing*, *175*, 935–947.

Martínez-Muñoz, G., & Suárez, A. (2010). Out-of-bag estimation of the optimal sample size in bagging. *Pattern Recognition*, *43*(1), 143–152.

Messer, K., & Kittler, J. (1998). Choosing an Optimal Neural Network Size to aid a Search Through a Large Image Database. In *Bmvc* (pp. 1–10).

Mirzaei, M., Zaerpour, N., & de Koster, R. (2021). The impact of integrated cluster-based storage allocation on parts-to-picker warehouse performance. *Transportation Research Part E: Logistics and Transportation Review*, *146*, 102207.

Mullainathan, S., & Spiess, J. (2017). Machine learning: an applied econometric approach. *Journal of Economic Perspectives*, *31*(2), 87–106.

Napierala, K., & Stefanowski, J. (2016). Types of minority class examples and their influence on learning classifiers from imbalanced data. *Journal of Intelligent Information Systems*, *46*(3), 563–597.

Nasteski, V. (2017). An overview of the supervised machine learning methods. *Horizons. b*, *4*, 51–62.

Nielsen, M. A. (2015). *Neural networks and deep learning* (Vol. 25). Determination press San Francisco, CA, USA.

Oshiro, T. M., Perez, P. S., & Baranauskas, J. A. (2012). How many trees in a random forest? In *International workshop on machine learning and data mining in pattern recognition* (pp. 154–168).

Park, Y.-S., & Lek, S. (2016). Artificial neural networks: Multilayer perceptron for ecological modeling. In *Developments in environmental modelling* (Vol. 28, pp. 123–140). Elsevier.

Probst, P., & Boulesteix, A.-L. (2017). To tune or not to tune the number of trees in random forest. *The Journal of Machine Learning Research*, *18*(1), 6673–6690.

Probst, P., Wright, M. N., & Boulesteix, A.-L. (2019). Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: data mining and knowledge discovery*, *9*(3), e1301.

Provost, F. (2000). Machine learning from imbalanced data sets 101. In *Proceedings of the aaai'2000 workshop on imbalanced data sets* (Vol. 68, pp. 1–3).

Qi, X., Chen, G., Li, Y., Cheng, X., & Li, C. (2019). Applying Neural-Network-based Machine Learning to Additive Manufacturing: Current Applications, Challenges, and Future Perspectives. *Engineering*, *5*(4), 721–729.

Ripley, B. D. (1993). Statistical aspects of neural networks. *Networks and chaos—statistical and probabilistic aspects*, *50*, 40–123.

Rojas, R. (1996). The backpropagation algorithm. In *Neural networks* (pp. 149–182). Springer.

Rumelhart, D. E., Hinton, G. E., McClelland, J. L., et al. (1986). A general framework for parallel distributed processing. *Parallel distributed processing: Explorations in the microstructure of cognition*, *1*(45-76), 26.

Segal, M. R. (2004). Machine learning benchmarks and random forest regression.

Shang, K. H., & Song, J.-S. (2007). Serial supply chains with economies of scale: Bounds and approximations. *Operations research*, *55*(5), 843–853.

Shang, K. H., Song, J.-S., & Zipkin, P. H. (2009). Coordination mechanisms in decentralized serial inventory systems with batch ordering. *Management Science*, *55*(4), 685–695.

Shang, K. H., & Zhou, S. X. (2010). Optimal and heuristic echelon (r, nQ, T) policies in serial inventory systems with fixed costs. *Operations Research*, *58*(2), 414–427.

Strobl, C., Boulesteix, A.-L., Zeileis, A., & Hothorn, T. (2007). Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC bioinformatics*, *8*(1), 1–21.

Tahir, M. A., Kittler, J., Mikolajczyk, K., & Yan, F. (2009). A multiple expert approach to the class imbalance problem using inverse random under sampling. In *International workshop on multiple classifier systems* (pp. 82–91).

Tangirala, S. (2020). Evaluating the impact of GINI index and information gain on classification using decision tree classifier algorithm. *International Journal of Advanced Computer Science and Applications*, *11*(2), 612–619.

Tompkins, J. A., White, J. A., Bozer, Y. A., & Tanchoco, J. M. A. (2010). *Facilities planning*. John Wiley & Sons.

Van Donselaar, K. H., & Broekmeulen, R. (2017). Lecture Notes for the course Stochastic Operations Management: Stochastic inventory models for a single item at a single location.

Varian, H. R. (2014). Big data: New tricks for Econometrics. *Journal of Economic Perspectives*, *28*(2), 3–28.

Wang, F. (1994). The use of artificial neural networks in a geographical information system for agricultural land-suitability assessment. *Environment and planning A*, *26*(2), 265–284.

Witten, I. H., & Frank, E. (2002). Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. *Acm Sigmod Record*, *31*(1), 76–77.

Yang, D., Wu, Y., & Ma, W. (2021). Optimization of storage location assignment in automated warehouse. *Microprocessors and Microsystems*, *80*, 103356.

Yijing, L., Haixiang, G., Xiao, L., Yanan, L., & Jinling, L. (2016). Adapted ensemble classification algorithm based on multiple classifier system and feature selection for classifying multi-class imbalanced data. *Knowledge-Based Systems*, *94*, 88–104.

Yonaba, H., Anctil, F., & Fortin, V. (2010). Comparing sigmoid transfer functions for neural network multistep ahead streamflow forecasting. *Journal of hydrologic engineering*, *15*(4), 275–283.

Zheng, A., & Casari, A. (2018). *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. " O'Reilly Media, Inc.".

Zhou, L. (2013). Performance of corporate bankruptcy prediction models on imbalanced dataset: The effect of sampling methods. *Knowledge-Based Systems*, *41*, 16–25.

Zhou, S., & Mentch, L. (2021). Trees, forests, chickens, and eggs: when and why to prune trees in a random forest. *arXiv preprint arXiv:2103.16700*.

Appendix A

Description hyperparameters ML techniques

A.1 Random Forest

The number of trees T in the forest. Breiman (2001) proved by the Law of Large Numbers that the error rate (being the mean number of misclassifications) of an RF converges to a limit (depending on the dataset) as the number of trees in the forest becomes large. Considering several other relevant performance measures and their (non-)monotonicity, Oshiro et al. (2012) showed that using a large number of trees (whilst staying computationally feasible) is sufficient for (binary) classification problems and further tuning of this hyperparameter is not necessarily required.

The size of the random subset m of features to be chosen at each split in a tree. A low value leads to very distinct trees, which ensures stability of the results when averaging over many of such trees. However, all individual trees will have a lower performance on average, as trees are potentially grown on unimportant features. Hence, the choice for this hyperparameter is a trade-off between stability and accuracy (Probst et al., 2019). For classification problems, a commonly used size of the random subset equals $m = \sqrt{p}$, with p being the number of input features (Hastie et al., 2009). This appears to be a reasonable value in practice (Bernard et al., 2009).

The sample size n for each tree and whether each sample is drawn with or without replacement. Similar to the size of the random subset of features, selecting the sample size is a trade-off between stability and accuracy. Using a smaller sample size leads to more distinct trees, but usually decreases the performance since fewer observations are used to grow each tree (Probst et al., 2019). Moreover, a lower sample size reduces the runtime (Martínez-Muñoz & Suárez, 2010). In Breiman (2001)'s original formulation of the RF, the samples of observations were drawn with replacement. However, Janitza et al. (2016) and Strobl et al. (2007) showed that for specific cases, this leads to lower performance as compared to sampling without replacement. Hence, both the sample size as sampling with or without replacement are hyperparameters of an RF.

The maximum depth d of the tree, being the maximum number of splits in every single tree. This hyperparameter is strongly related to several other hyperparameters limiting the size of the tree, such as the minimum number of observations in each node and the maximum number of nodes in each tree (Probst et al., 2019). Setting this parameter to a lower value decreases the runtime for large datasets significantly, without causing a substantial loss in the model's performance (Segal, 2004). Moreover, it limits every single tree from overfitting on the training data (S. Zhou & Mentch, 2021).

The split criterion used. This is not necessarily considered a hyperparameter, but more as a characteristic of an RF (Probst et al., 2019). For classification problems, the Gini index and Entropy are the most commonly used split criteria (Kingsford & Salzberg, 2008; Tangirala, 2020). Suppose $\mathcal{Q}_z$ denotes the set of observations at node z, including in total n_z observations. Moreover, let p_{zk} denote the

proportion of class k observations in node z, which is defined as:

$$p_{zk} = \frac{1}{n_z} \sum_{y \in \mathcal{Q}_z} I(y = k), \tag{A.1}$$

where $I(y = k)$ is the indicator function which takes value 1 if $y = k$ and 0 otherwise. Note that for binary classification problems, k is either 0 or 1. Then, the Gini index and Entropy functions are defined as:

$$\text{Gini} = 1 - \sum_k p_{zk}^2 \tag{A.2}$$

$$\text{Entropy} = -\sum_k p_{zk} \log(p_{zk}) \tag{A.3}$$

Tangirala (2020) showed that the choice of split criterion does not affect the performance of a single decision tree; both functions gave the same results.

A.2 Multilayer Perceptron

The number of hidden layers. Usually, an MLP comprises one input layer, one or two hidden layers and one output layer (Kavzoglu & Mather, 2003). Many applications showed that an MLP with only a single hidden layer can already approximate a large set of functions, especially for classification problems (Athey & Imbens, 2019; Lippmann, 1987).

The number of neurons in a hidden layer k. This parameter has a large influence on the performance and complexity of the model (Abraham, 2005). An adequate number of neurons is needed to be able to separate the data well over the different classes and achieve a reasonable performance. However, if the number of neurons is set too large, the model will be less able to generalize to new data (i.e. overfitting the data), which decreases the out-of-sample performance (Kavzoglu & Mather, 2003). Hence, the number of neurons in the hidden layer(s) should be set deliberately.

Several heuristics have been proposed to estimate the optimal number of neurons, but none has been universally accepted. Several rules of thumb used in practice are based on the number of input features p (Kavzoglu & Mather, 2003), such as: $2p$ or $3p$ (Kanellopoulos & Wilkinson, 1997), $2p + 1$ (Hecht-Nielsen, 1987), $2p/3$ (Wang, 1994) or $p/2 + 1$ (Ripley, 1993).

The maximum number of neurons in the hidden layer is dependent on the amount of available training data. To accurately estimate all free parameters, the training samples should at least be five times the number of free parameters in the model (Messer & Kittler, 1998). As the number of free parameters in the MLP increases in the number of neurons in each hidden layer, the amount of available training samples defines an upper bound on the number of hidden layer neurons.

The activation function. As stated earlier, the sigmoid function is the most commonly used activation function in MLPs. Still, other activation functions are applicable as well, such as the symmetrical sigmoid $S(z)$ (Rojas, 1996) or the rectified linear neuron $R(z)$ (Nielsen, 2015) functions, defined as:

$$S(z) = \frac{1 - e^{-z}}{1 + e^{-z}} \tag{A.4}$$

$$R(z) = \max(0, z) \tag{A.5}$$

Based on the problem at hand, one activation function might outperform another in terms of computational time and prediction performance.

Initialization of weights and biases. As indicated in the previous paragraph, the backpropagation algorithm starts with randomly assigned initial weights and biases. This is a starting point, from whereon the model should move as quickly as possible towards a global minimum of the cost function (Kavzoglu & Mather, 2003). Hence, each initialization requires a different route to this global minimum. However, several routes may contain local minima, where the algorithm gets stuck. Consequently, the gradient descent method is not guaranteed to find the global minimum from all starting points (Krogh, 2008). As this affects the performance of the model, it can be beneficial to test the same network configuration whilst using multiple initializations of the weights and biases (Kolen & Pollack, 1990).

In addition, Nielsen (2015) stated that the pace of the learning process can change significantly for different starting points. Especially for large initial weight values, the learning process is likely to saturate quickly. To prevent this behavior, Nielsen (2015) proposed to initialize the weights as normally distributed random variables with mean 0 and standard deviation $1/\sqrt{n_l}$, with n_l being the number of input neurons to the l^{th} layer. The biases can be initialized as normally distributed random variables with mean 0 and standard deviation 1 as these initial values appear to have little influence on the learning pace of the model.

The learning rate η. As stated, the learning rate determines the step size of the adjustments in weights and biases, such that the model moves towards the global minimum of the cost function (Nielsen, 2015). It can be considered one of the most important hyperparameters of an MLP because it influences both the performance and the speed of learning (Kavzoglu & Mather, 2003). If the learning rate is set too large, the local or global minima will be overstepped constantly, causing a slow convergence of the model. If the learning rate is set too low, the model will improve very slowly, requiring a high number of iterations to yield reasonable results (Abraham, 2005). Moreover, it increases the likelihood of getting stuck at a local minimum (Kavzoglu & Mather, 2003).

As the best value for η is highly problem dependent, Nielsen (2015) suggests finding the best learning rate by trial and error: try different orders of magnitudes and select the highest learning rate at which the costs decrease during the first few epochs. Then, the learning rate can be further optimized by trying several values in this order of magnitude.

The regularization parameter λ. To prevent the model from overfitting, Nielsen (2015) proposed to add an L2 regularization term to the cost function $C(\mathbf{x}_i, \mathbf{W}^l, \mathbf{b}^l)$:

$$\frac{\lambda}{2n} \sum_{l=2}^{L} \sum_{\mathbf{W}^l} (w_{k,j}^l)^2$$

where $\lambda \geq 0$ denotes the regularization parameter, n denotes the total number of training samples and $\mathbf{W}^l$ denotes the weight matrix for layer l.

The regularization term penalizes the model for selecting large weights, as this adversely affects the stability of the network and the ability to generalize to new data. That is, a model having large weight values is more likely to change if the input changes and thus more likely to overfit the training data. As the regularization term reflects a preference for smaller weights, this behavior is prevented. Moreover, due to the penalization, the exact initialization of the weights plays a less crucial role, as it decreases the probability of getting stuck at a local minimum.

The regularization parameter λ defines a trade-off between the minimization of the cost function and the preference for a more stable network (by penalizing larger weights). Similarly as for the learning rate η, the best value for λ highly depends on the problem context and the values for all other hyperparameters. Therefore, the best order of magnitude should be selected by trial and error for each setting. Subsequently, the parameter can be further refined by trying different values in this order of magnitude.

The cost function $C(\mathbf{x}_i, \mathbf{W}^l, \mathbf{b}^l)$. As the goal of the backpropagation algorithm is to minimize the cost function, the way an MLP model learns is highly dependent on the choice of this function. The MSE

introduced in Equation (2.6) is a very commonly used cost function, which minimizes the total error of the model. As an alternative, the cross-entropy cost function can be implemented, defined as (Nielsen, 2015):

$$C(\mathbf{x}_i, \mathbf{W}^l, \mathbf{b}^l) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=0}^{1}[y_{i,k}\ln(\hat{a}_{i,k}) + (1-y_{i,k})\ln(1-\hat{a}_{i,k})] \tag{A.6}$$

where $y_{i,k}$ is the actual outcome and $\hat{a}_{i,k}$ is the predicted outcome of observation i for the k^{th} output neuron. As stated earlier, binary classification problems have two output neurons (i.e. k is either 0 or 1). Even though the MSE is used more in practice, the cross-entropy function turns out to perform better in many contexts, as it is less sensitive to the exact starting point of the weights and biases and the (Nielsen, 2015). That is, the algorithm is less likely to get stuck at a local minimum of the cross entropy cost function.

The stop criterion. As indicated above, the backpropagation algorithm continues training the model until a certain stop criterion is reached. The number of iterations or training epochs should be set carefully, as terminating the algorithm too early results in lower performance (underfitting), whereas the algorithm should also be prevented from training too long such that overfitting occurs (Park & Lek, 2016).

A clear sign of overfitting is no improvement (or even decrease) in the performance of the model on the validation dataset, whilst the performance on the training dataset still increases (Kavzoglu & Mather, 2003). To prevent such overfitting, the algorithm should thus be terminated when the performance on the validation set does not improve over an earlier epoch. However, the performance of the model may stabilize for a while, after which it improves again (Nielsen, 2015). To capture this behavior, the learning algorithm can be terminated if the performance on the validation set has not improved during the last X epochs. This formulation reflects the trade-off between terminating the algorithm too soon and waiting too long for improvements.

Appendix B

Variable Description

Features from Witron masterdata

Feature	Definition	Type	Values
Complete Auto DEPAL	Indicates whether the complete pallet is automatically depalletized or if the last layer needs manual depalletization	Binary	0: Last layer manual or 1: Entire pallet automatically
DepalFlag	Indicates whether an item should by handled by the depalletizer with manual option	Binary	0:mDep (No) or 1:hDep (Yes)
Distance between products	Indicates whether there is a significant distance between products within a case	Float	$\mathbb{R}^+$
Fix next layer	Indicates whether or not a different mode is needed on the last layer	Integer	$\in \{0, 4, 9\}$, no unit
Gaps Cases	Indicates whether there are gaps between cases on a pallet	Binary	0:No or 1:Yes
Lid	Indicates whether the cases contain a loose lid	Binary	0:No or 1:Yes
Pem Flag		Binary	Yes or No
Profile MHE	Mechanical Handling Equipment profile; indicates whether the crane should handle a pallet slower than default pace	Categorical	Three classes: 1:Default pace, 2:50% slower pace (in all dimensions) or 3:90% slower pace (in all directions)
Sectional Vacuum	Indicates whether the top layer is complete, such that only pressure can be applied to the parts where cases are	Binary	0:Complete or 1: Incomplete

Features from Witron masterdata

Feature	Definition	Type	Values
Skirt Pressure	Indicates how much the side pillows of a depalletizer need to inflate to hold a layer of the pallet	Integer	$\in \{60, 120, 230, 300\}$, in bar
Skirt Touch Height	Indicates how much the side pillows of a depalletizer should be lifted to lift the cases correctly	Integer	$\in \{-25, -15, -10, 0, 10, 20, 49\}$, in mm
SlipSheet	Indicates whether a slipsheet is present between layers and how it should be removed	Categorical	Three classes: No special handling, Manual removal or Automatic removal
Stripper Plate	Indicates whether the vertical movements of the pallet at the DEPAL should be slower than default	Binary	0:No or 1:Yes
Suction Plate Offset	Indicates how much the suction head should hover above the pallet	Integer	$\in \{0, 20, 40\}$, in mm
Tilt	Indicates whether a case should be tilted after depalletizing, before loaded on a tray at tray merge	Binary	0:No or 1:Yes
Vacuum Pressure	Amount of pressure the depalletizer applies when it vacuums a layer	Integer	$\in \{25, 35, 50, 60, 65, 80, 85\}$, in bar

Features from Jumbo masterdata

Feature	Definition	Type	Values
Demand rate	Average number of cases sold during time period	Float	$\mathbb{R}^{+}$, in cases
Content Type	Classifies items based on package of products within case	Categorical	Four classes (e.g. Glass)
Crushability	Classifies items based on how crushable they are	Categorical	Six classes (e.g. Medium loadable)
Demand rate Class	Classifies items based on demand rate	Categorical	Three classes: A (fast), B (medium) and C (slow)
Demand rate Sign	Ordinal classification of SKUs based on demand rate	Ordinal	Ten values: $1-3$ for Class A, $4-6$ for Class B and $7-10$ for Class C
Height Case	Height of one case	Float	$\mathbb{R}^{++}$, in mm
Height Width Ratio	Height divided by width of one case	Float	$\mathbb{R}^{++}$, no unit
Pallet Length	Length of a supplier pallet	Integer	$\in \{800, 1000\}$ in mm
Length Case	Length of one case	Float	$\mathbb{R}^{++}$, in mm
Length Width Ratio	Length divided by width of one case	Float	$\mathbb{R}^{++}$, no unit
Package Shape	Classifies items based on packaging	Categorical	Twelve classes (e.g. Cardboard box)
Pet	Indicates whether a case includes pet or not	Binary	0:No or 1:Yes
UM	Unit of measurement, indicates how the SKU is packed	Categorical	Twelve classes (e.g. DSL)
Group	Indicates the group the SKU is categorized in	Categorical	22 classes (e.g. Cosmetics)
Supplier Pallet Type	Type of supplier pallet used (by default)	Categorical	Eight classes, of which four small pallets and four large pallets
Volume Case	Volume of one case	Float	$\mathbb{R}^{++}$, in l
Weight Case	Weight of one case	Float	$\mathbb{R}^{++}$, in g
Weight Class	Classifies items based on their weight	Categorical	Three classes: 1, 2 and 3
Width Case	Width of one case	Float	$\mathbb{R}^{++}$, in mm

Pallet specific features from inbound or tray merge

Feature	Definition	Type	Values
Cases Layer	Number of cases per layer on a pallet	Integer	$\mathbb{Z}^+$, in cases
Cases Pallet	Total number of cases on a pallet	Integer	$\mathbb{Z}^+$, in cases
Cases Tray Merge	Total number of cases put on a tray at tray merge	Integer	$\mathbb{Z}^+$, in cases
Height Difference Pallet	Difference between measured height at an infeed station and expected height based on master data	Float	$\mathbb{R}$ in cm
Height Pallet	Height of a pallet as measured at an infeed station	Float	$\mathbb{R}^{++}$, in cm
Height Tolerance	Range by which the measured height is allowed to vary from the expected height to be accepted in the system	Float	$\mathbb{R}^{++}$, in cm
Layers	Total number of layers on a pallet	Integer	$\mathbb{Z}^+$, in layers
Maximum Layers	Maximum number of layers on a pallet, as set by the supplier	Integer	$\mathbb{Z}^+$, in layers
Weight Difference Pallet	Difference between measured weight at an infeed station and expected weight based on master data	Float	$\mathbb{R}$ in kg
Weight Pallet	Weight of a pallet as measured at an infeed station	Float	$\mathbb{R}^{++}$, in kg
Weight Tolerance	Range by which the measured weight is allowed to vary from the expected weight to be accepted in the system	Float	$\mathbb{R}^{++}$, in kg

New features added to the dataset

Feature	Definition	Type	Values
DEPAL i	Indicates whether or not the pallet was send to DEPAL i for depalletization, with $i \in \{1, ..., 12\}$	Binary	0:No or 1:Yes
Error Area Fill rate	Indicates whether the fill rate of the area of the pallet was above 1 due to inaccurate information in the master data	Binary	0:No or 1:Yes
Error Height Fill rate	Indicates whether the fill rate of the height of the pallet was above 1 due to inaccurate information in the master data	Binary	0:No or 1:Yes
Number of Calls DEPAL	Indicates the number of times in which a pallet is depalletized	Integer	$\mathbb{Z}^+$, in number of calls
Fallen Cases Binary	Indicates whether one or more case fell from a pallet	Binary	0:No or 1:Yes
Fallen cases	Number of cases that fell off a pallet	Integer	$\mathbb{Z}_0^+$, in cases
Pallet Area Fill rate	Indicates the used area of a pallet	Float	$\in$ (0,1], fraction
Pallet Height Fill rate	Indicates the used height of a pallet at inbound	Float	$\in$ (0,1], fraction
SKUs Supplier	Total number of SKUs suitable for OPM the supplier of a particular SKU ships	Integer	$\mathbb{Z}_0^+$, in SKUs
Top layer	Indicates whether the top layer equals a complete layer or not	Binary	0:No or 1:Yes
Weekly Pallets	Mean number of pallets the supplier of a particular SKU ships every week	Float	$\mathbb{R}^+$, in pallets

Appendix C

Data Preparation

As indicated in Section 2.2, a prediction model should be built on a training set, the model should be validated on a validation set and its performance should be tested on a test set (Varian, 2014). When observations from the validation or test set would also be used to train the model, one speaks of so-called (direct) data leakage (Brownlee, 2022). If the model is then evaluated on the same data, the performance of the model will likely be overestimated due to the advantage of making better predictions (Zheng & Casari, 2018). Running the same model later on a dataset that was not known beforehand will probably yield worse results. Similarly, applying all data preparation techniques to all available data simultaneously could result in (indirect) data leakage (Kuhn & Johnson, 2019). For instance, scaling or normalizing a numerical feature requires the mean or global minimum and maximum of a feature, respectively. These statistics potentially yield different values for the training, validation and test dataset. Hence, if this transformation is applied to the entire dataset, features have been transformed by the global statistical values and some information about the validation or test set is leaked into the training dataset. To prevent this, some data preparation techniques should be performed on the training, validation and test set independently (Brownlee, 2022). The remainder of this chapter discusses several data preparation techniques that should be performed on the entire dataset or the training, validation and test set independently.

Preparation techniques for entire dataset

Missing values Some of the input features originate from Jumbo master data (report *MA57*) and generally regard SKU-specific information. However, some (newly introduced) SKUs do not appear in this report yet, resulting in missing values for the pallets containing these SKUs. As missing values are not allowed in ML techniques, these observations (rows) are removed from the dataset.

Errors For each pallet, the number of cases entering the system (input) is compared with the number of cases leaving the system (output). If these numbers differ, it is assumed that the cases fell somewhere in the process. However, for some pallets this difference yields a negative value, indicating a surplus in cases. Clearly, this observation is impossible in reality. Therefore, pallets with negative differences were considered to be errors and removed from the dataset.

Two other features in the dataset appeared to have unfeasible values. By definition, the *Pallet Area Fill Rate* and *Pallet Height Fill Rate* should not have values exceeding 1, as the dimensions of the pallet then exceed its maximum value. Yet, the dataset contains observations where either one of these fill rates exceeds 1. The *Pallet Height Fill Rate* is calculated using the *Maximum Layers* feature, which value is specified by the supplier and known for each SKU separately. However, this information might have become outdated, such that in reality additional layers could potentially be fit on the pallet. A similar line of reasoning can be applied for the *Pallet Area Fill Rate*. If this value exceeds 1, this might be due to incorrect information in the master data. This fill rate is calculated by using the *Length Case*, *Width Case*, *Cases per Layer* and *Pallet Length* features. All these features are included in the master data,

but are subject to changes over time, potentially resulting in a *Pallet Area Fill Rate* above 1. Hence, the observations with a fill rate above 1 contain incorrect information for some features. Usually, such observations should be removed from the data. However, in the current context, the incorrectness of the data potentially results in system errors as some calculations are based on these values. In turn, these system errors could result in cases falling. If the observations with incorrect data would be removed from the dataset, the model is limited from detecting this as a cause of cases falling. Therefore, instead of removing these observations, two binary features were added to the dataset indicating for either fill rate whether its value was feasible; *Error Height Fill Rate* and *Error Area Fill Rate*. If not, this implies that the current information in the master data was incomplete. In addition, the infeasible values for *Pallet Height Fill Rate* or *Pallet Area Fill Rate* above 1 were set equal to 1.

Outliers Raw data potentially contains outliers for some features, which are rare or distinct values that are outside the range of all other values (Brownlee, 2022). Detecting these outliers is important, as ML techniques potentially perform better when such outliers are removed from the dataset. By visually checking the dataset, outliers were detected for two features. First, in *Maximum Layers* some outliers were present. For three SKUs, the *Maximum Layers* value equaled $70, 80$ or 160 layers, whereas this value is below 34 for all other SKUs. Considering the *Layers* feature of these SKUs, the actual number of layers for these SKUs was always less than or equal to $7, 8$ or 16 layers for the respective SKU. Therefore, the outliers are believed to be a typo and changed to the latter values.

Second, the *Fallen cases* feature seems to have some outliers. For most pallets, the value for *Fallen cases* is between 0 and 25, but for some observations the value exceeds 100 cases. However, occasionally a complete pallet falls in HBW, resulting in a high number of fallen cases. Therefore, these outliers are not considered errors and are not removed from the data.

Transforming features All binary and categorical features need to be transformed into numeric features. All binary features are transformed into labels, where each observation having a value *"Yes"* or *"True"* is assigned a 1 and each observation having a value *"No"* or *"False"* is assigned a 0. For categorical features one-hot encoding is used, meaning that each value of the feature becomes a unique column. That is, for each value a dummy binary feature is added to the dataset. In total, the dataset has 23 binary features (excluding the output feature) and 10 categorical features. By encoding these features into numeric features, the dataset increases from 66 features to 120 features.

Correlated features Not surprisingly, the features *Cases Pallet* and *Cases Tray Merge* are perfectly correlated. As only 0.01% of the inbound volume is lost due to fallen cases, these features have nearly the same values for all observations. Therefore, the *Cases Tray Merge* feature could be removed from the dataset. In addition, the features *Pallet Length* and *Supplier Pallet Type* appear to be extremely related. This is a sensible effect, as the *Pallet Length* feature is derived from the *Supplier Pallet Type*. Consequently, the *Pallet Length* feature could be removed from the dataset as well.

Selecting features The number of features present in the data adversely affects the computational time of the prediction models. Moreover, it is unlikely that all features are relevant for predicting the output feature; including these features in the model might decrease the performance of the model, as it adds uncertainty to the predictions (Kuhn et al., 2013). Hence, the goal is to select the best subset of all features to predict the outcome feature. The feature importance listed by a RF can be used for this matter, by selecting the subset of features having the highest importance (Brownlee, 2022; Varian, 2014). That is, first a RF can be fit on the data including all features. Then, based on the feature importance in this model, a new RF can be fit on a subset of all features.

Preparation techniques for each separate dataset

Transforming numerical features The numeric values in the dataset have different units. For instance, the dimensions of cases are expressed in mm, whereas the weight is expressed in g and the volume in l. Moreover, some features are dimensionless, whereas others are expressed in number of cases. As many alternative units are present in the data, the mean values of the features might have very different scales. This increases the complexity of the problem and might adversely influence the model's performance (Brownlee, 2022). To counter this, the data should be scaled by means of standardization or normalization. If standardization is applied, each value is subtracted by the mean and divided by the standard deviation of the feature:

$$x_{\text{new}} = \frac{x_{\text{old}} - \mu}{\sigma} \tag{C.1}$$

This way, the distribution of the feature is shifted towards a normal distribution having a mean of zero and a standard deviation of one (Witten & Frank, 2002). If normalization is applied instead, all features are scaled to have a value between zero and one by subtracting the minimum and dividing by the range of the feature:

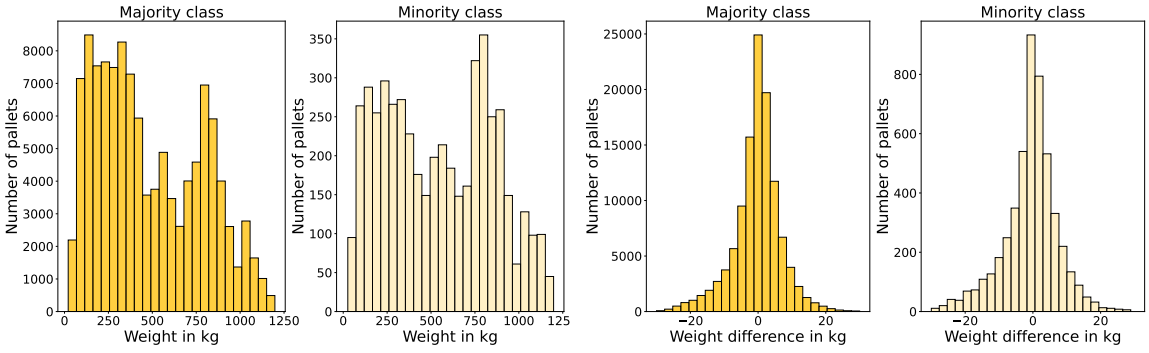
$$x_{\text{new}} = \frac{x_{\text{old}} - \min}{\max - \min} \tag{C.2}$$

Whether standardization works better than normalization or vice versa highly depends on the dataset (Brownlee, 2022). Therefore, the effect of both scaling methods was tested.

Resampling Many pallets in the dataset have zero cases missing. Consequently, the output feature is highly imbalanced, meaning that the number of observations in the majority class (no fallen cases) is much greater than the number of observations in the minority class (fallen cases) (Yijing et al., 2016). By using one of the resampling methods introduced in Section 2.2.3, the skewed distribution over the two classes can be diminished (Haixiang et al., 2017). That way, the model will be better able to predict the minority class. Note that this preparation technique should only be applied to the training dataset and not to the validation and test datasets.

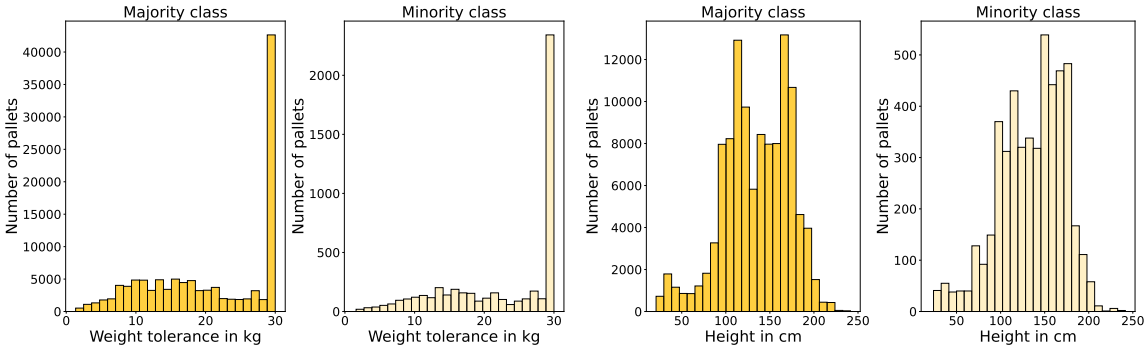
Appendix D

Data analysis



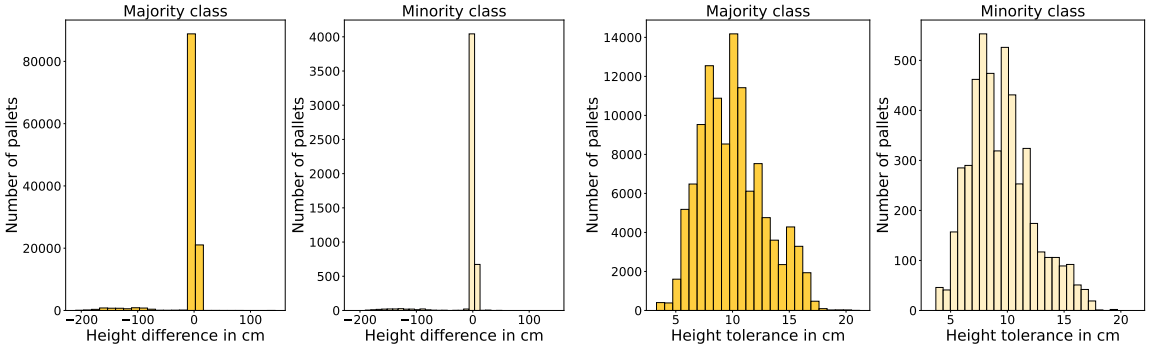
(1) Weight Pallet

(2) Weight Difference Pallet



(3) Weight Tolerance

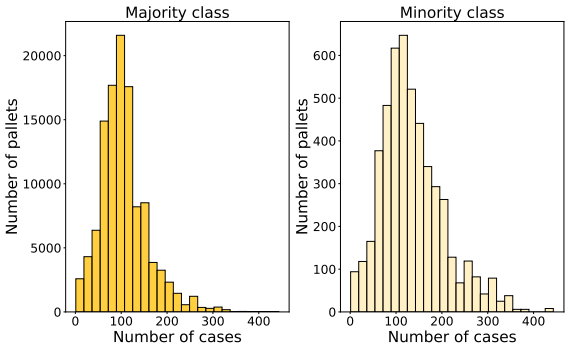
(4) Height Pallet



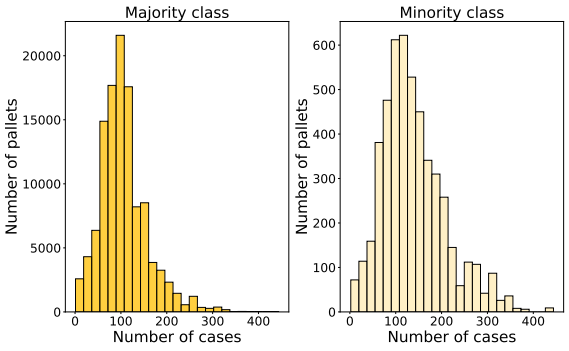
(5) Height Difference

(6) Height Tolerance

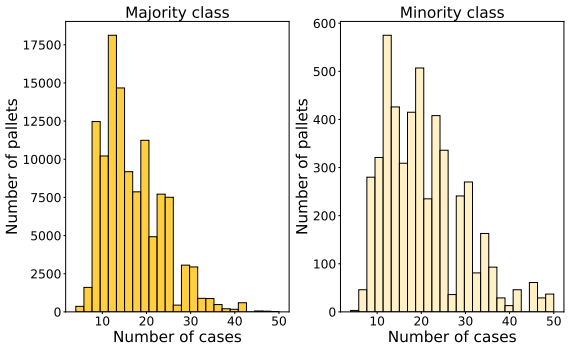
Distribution of features for pallets with and without cases missing



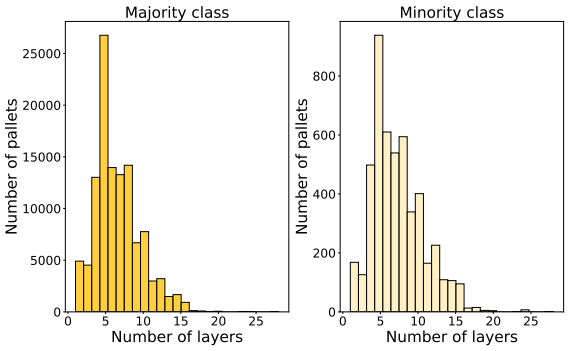
(7) Cases Tray merge



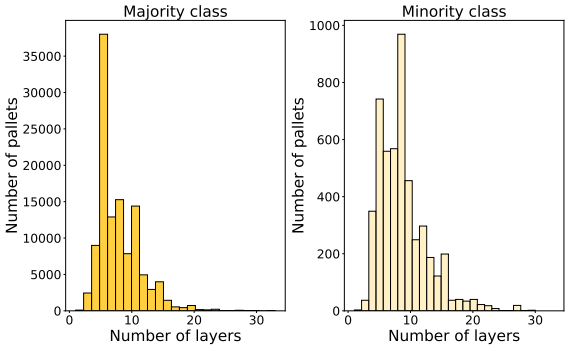
(8) Cases pallet



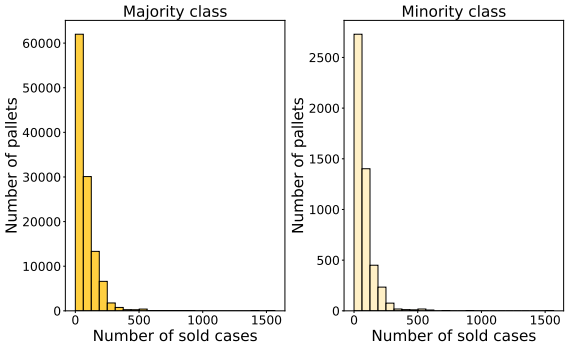
(9) Cases layer



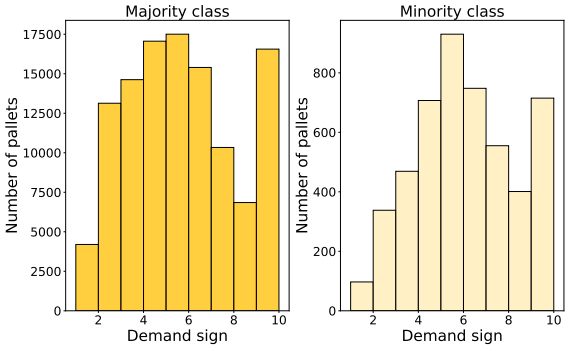
(10) Layers



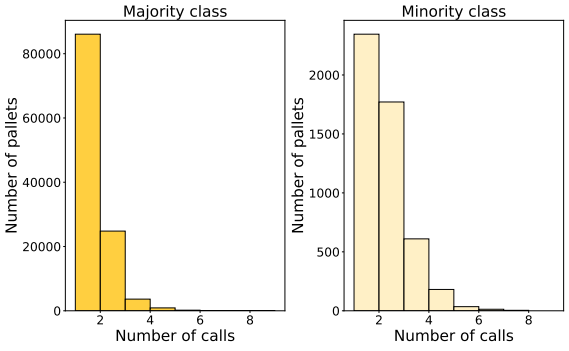
(11) Maximum layers



(12) Demand rate

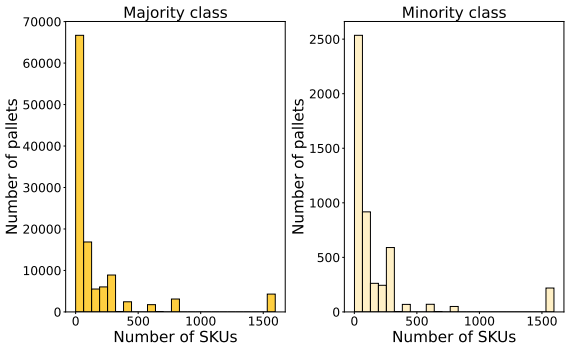


(13) Demand rate Sign

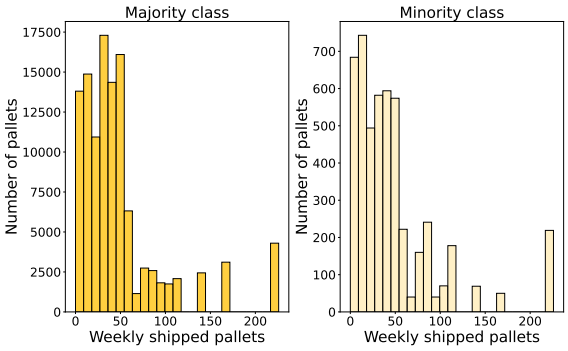


(14) Number of Calls DEPAL

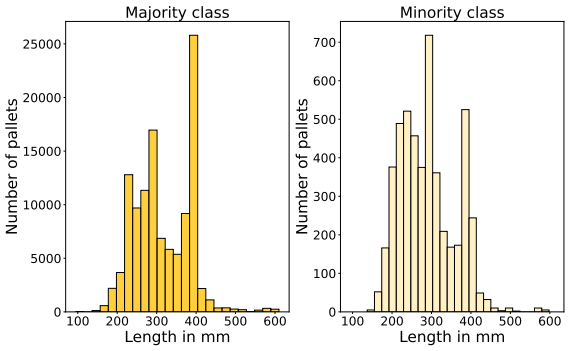
Distribution of features for pallets with and without cases missing



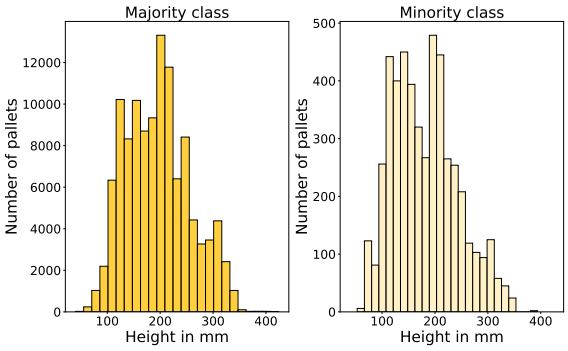
(15) SKUs Supplier



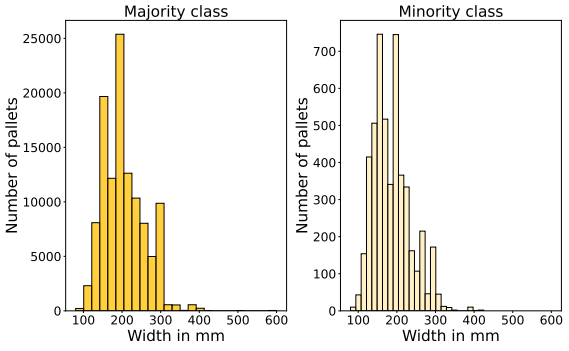
(16) Weekly Pallets



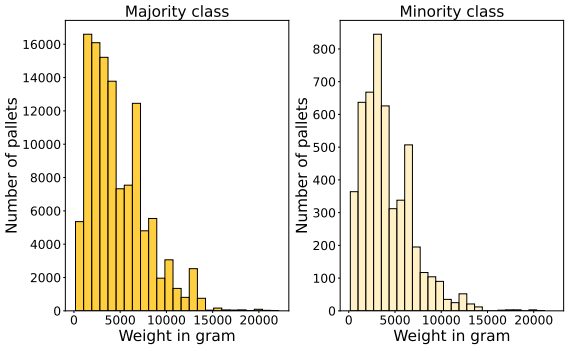
(17) Length Case



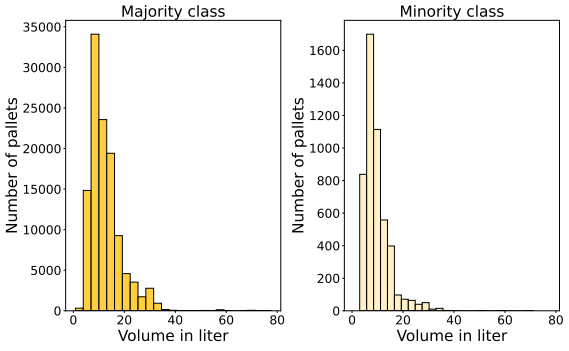
(18) Height Case



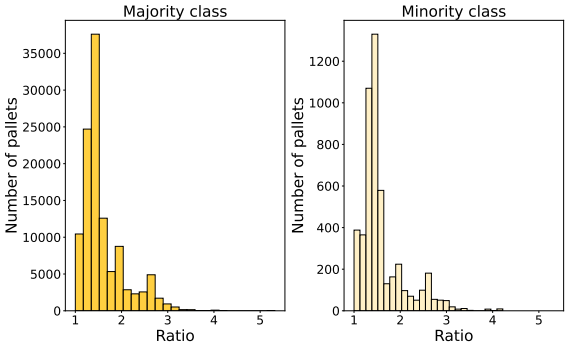
(19) Width Case



(20) Weight Case

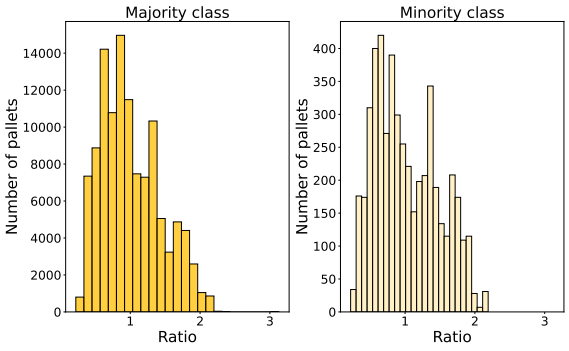


(21) Volume Case

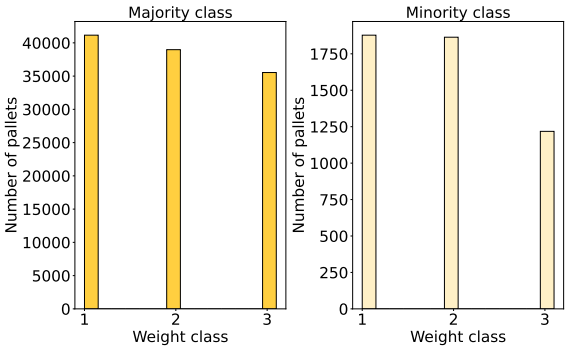


(22) Length Width Ratio

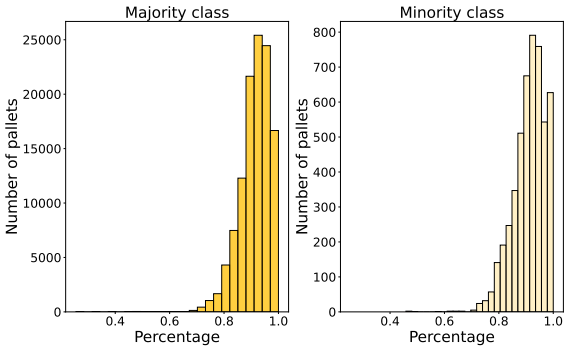
Distribution of features for pallets with and without cases missing



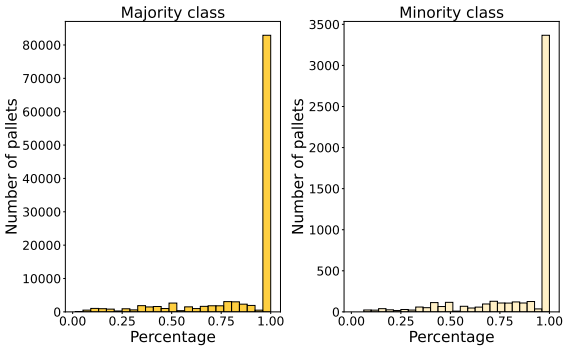
(23) Height Width Ratio



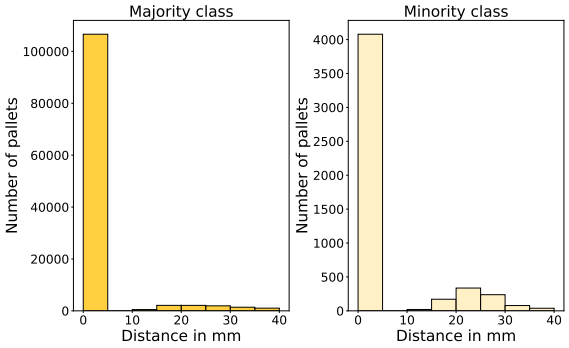
(24) Weight Class



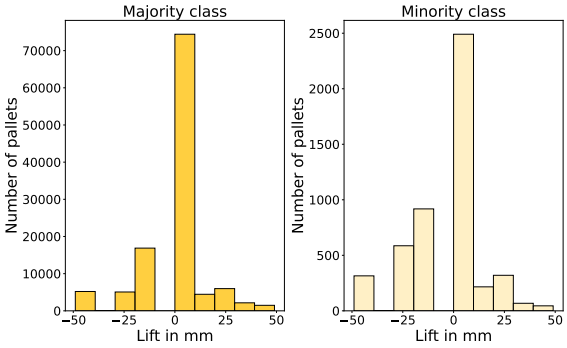
(25) Pallet Area Fill rate



(26) Pallet Height Fill rate

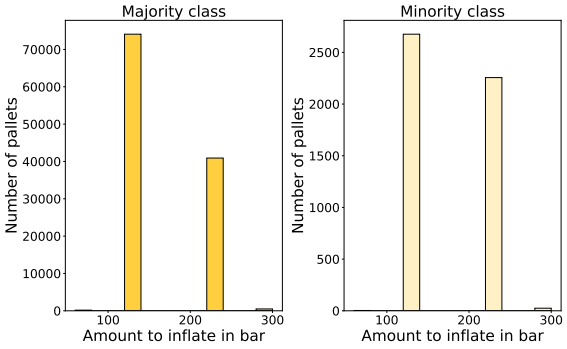


(27) Distance between products



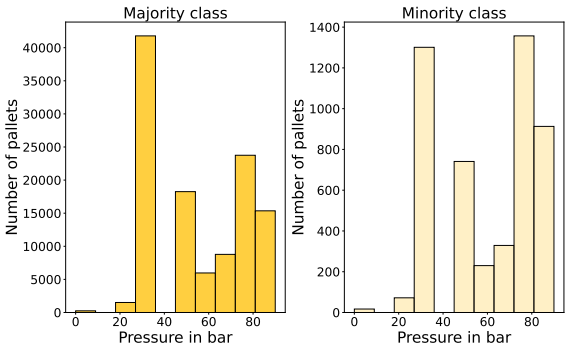
(28) Skirt Touch Height

(29) Suction Plate Offset



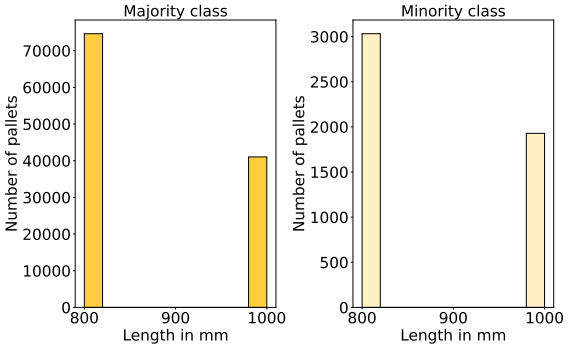
(30) Skirt Pressure

Distribution of features for pallets with and without cases missing

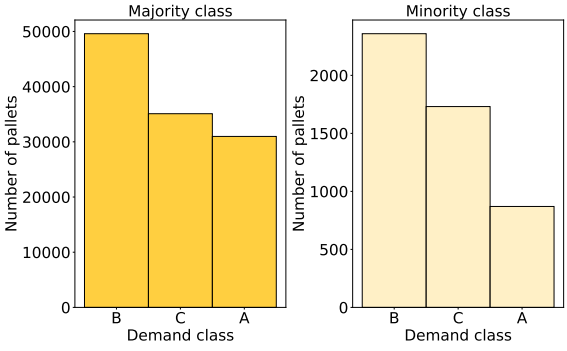


(31) Vacuum Pressure

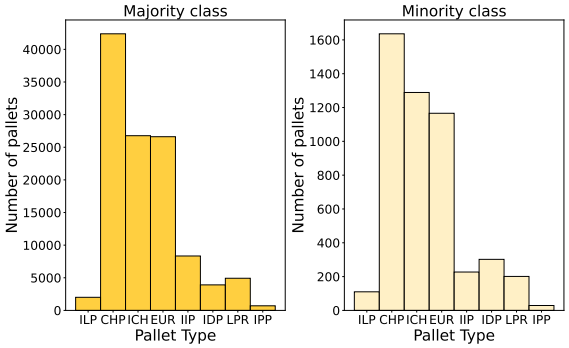
(32) Fix next layer



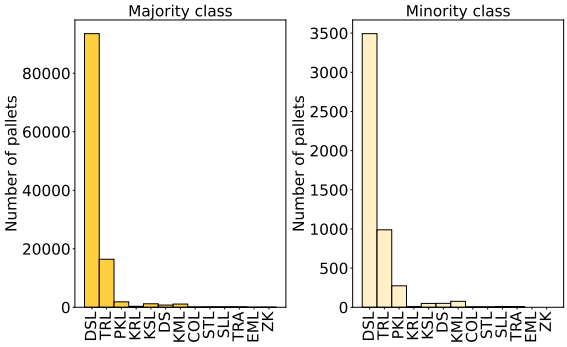
(33) Pallet Length



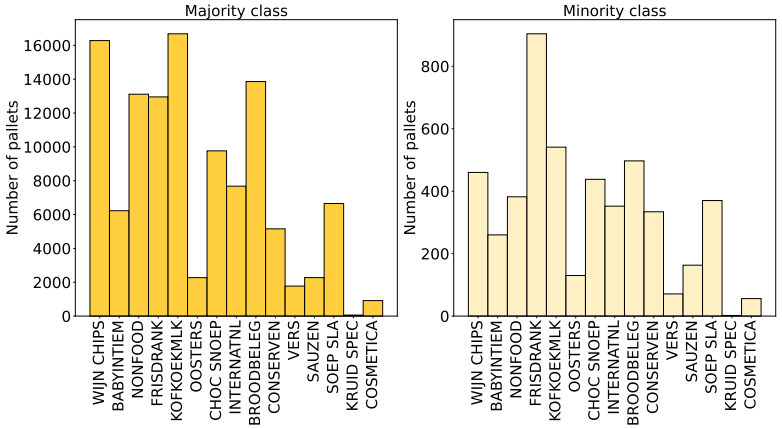
(34) Demand rate Class



(35) Supplier Pallet Type

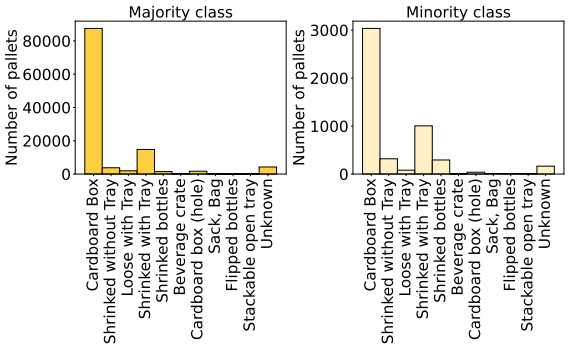


(36) UM

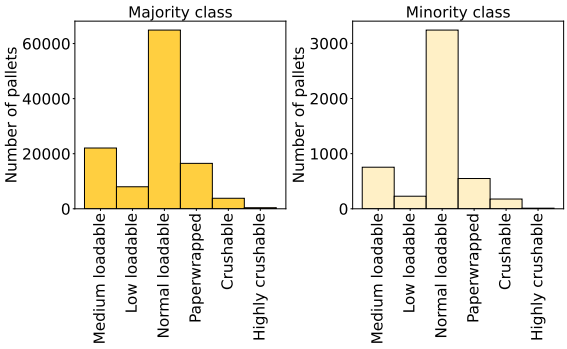


(37) Group

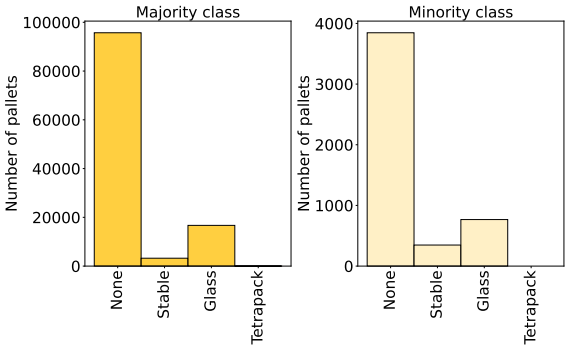
Distribution of features for pallets with and without cases missing



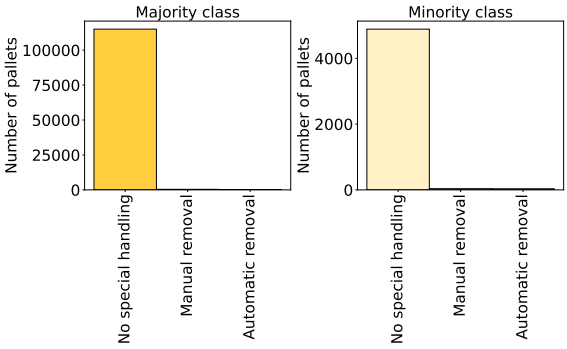
(38) Package Shape



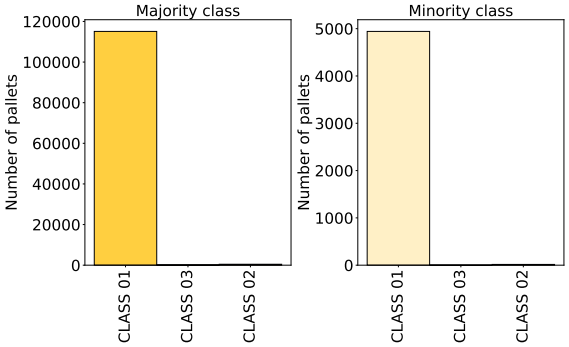
(39) Crushability



(40) Content Type



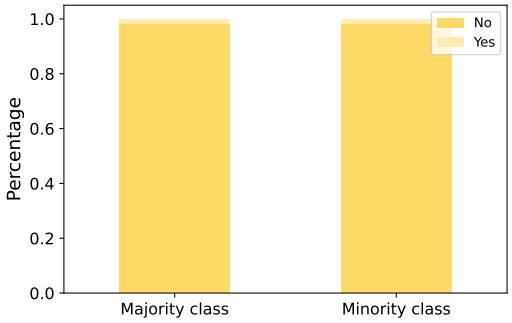
(41) SlipSheet



(42) Profile MHE

(43) Sectional Vacuum

(44) Top Layer

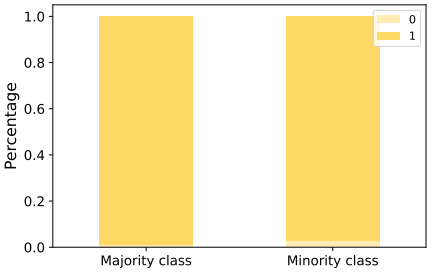


(45) Tilt

Distribution of features for pallets with and without cases missing

(46) Lid

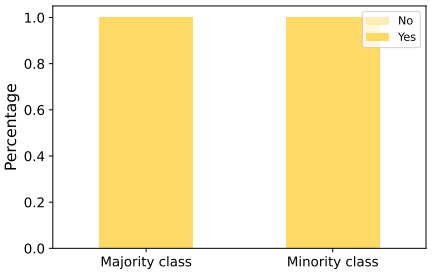
(47) Stripper Plate



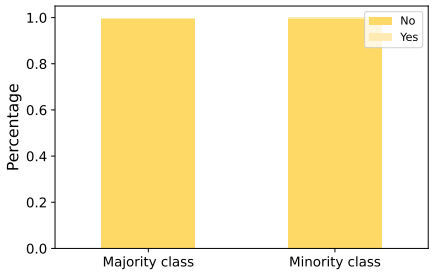
(48) Complete Auto DEPAL

(49) DepalFlag

(50) Pet

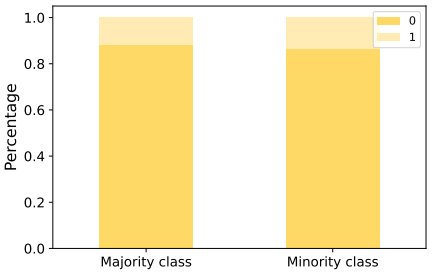


(51) Gaps Cases



(52) Error Height Fill rate

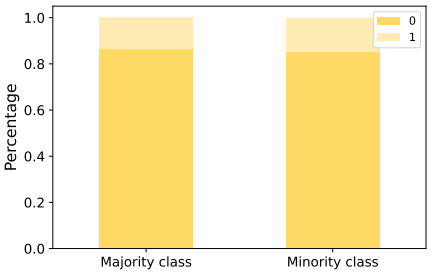
(53) Error Area Fill rate



(54) DEPAL 1

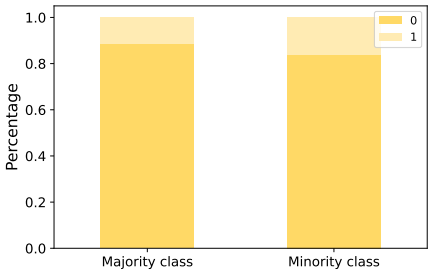
(55) DEPAL 2

(56) DEPAL 3

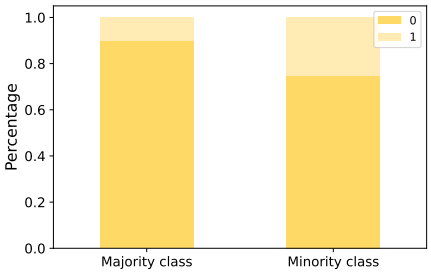


(57) DEPAL 4

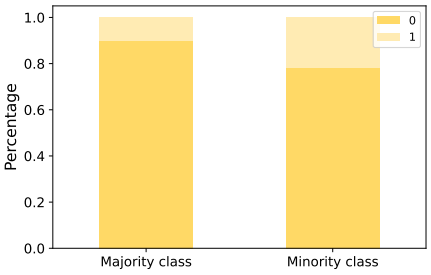
Distribution of features for pallets with and without cases missing



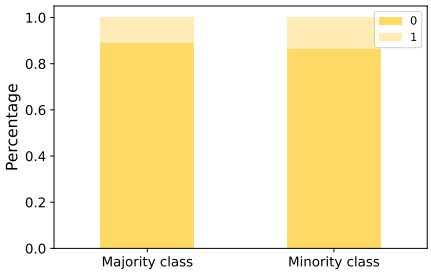
(58) DEPAL 5



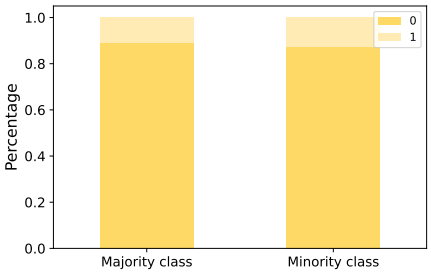
(59) DEPAL 6



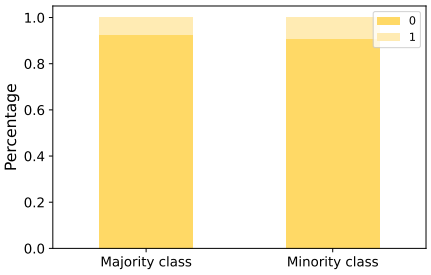
(60) DEPAL 7



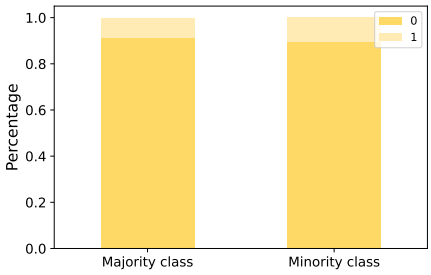
(61) DEPAL 8



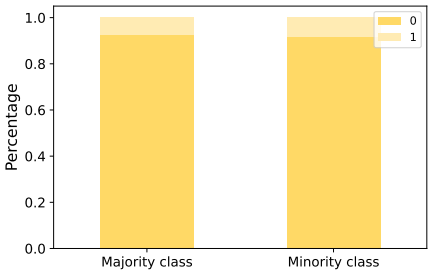
(62) DEPAL 9



(63) DEPAL 10



(64) DEPAL 11

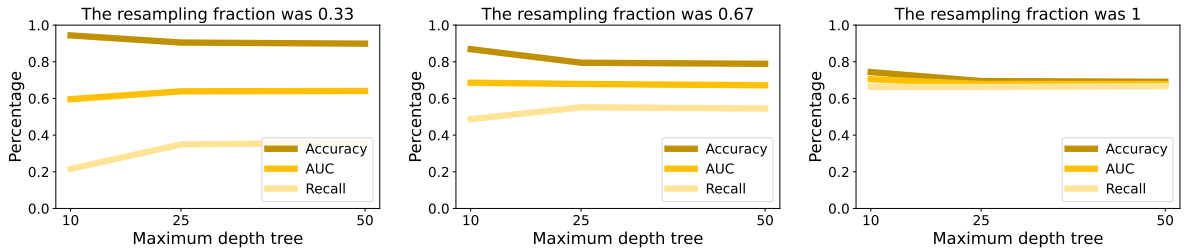


(65) DEPAL 12

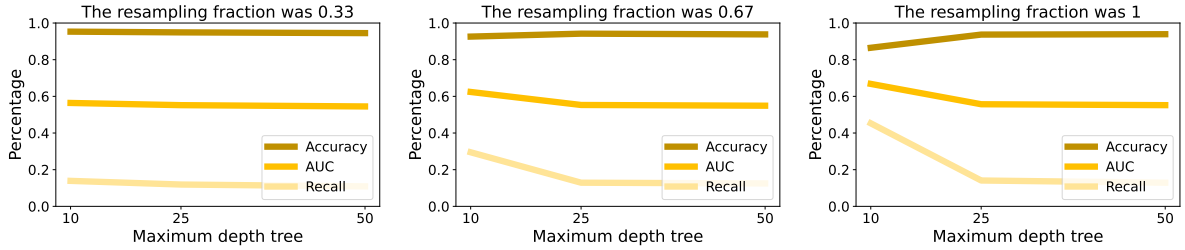
Distribution of features for pallets with and without cases missing

Appendix E

Pairwise plots Hyper parameters RF

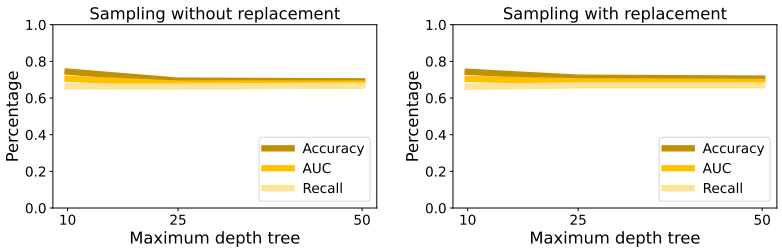


(a) Undersampling

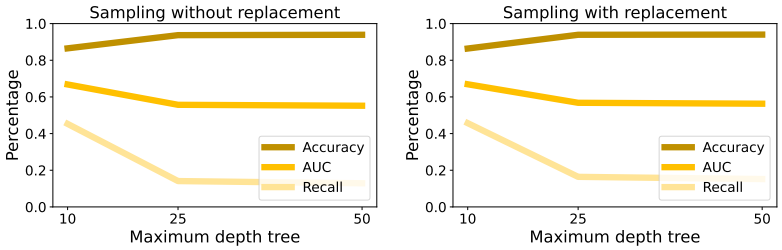


(b) Oversampling

Pairwise plot of Maximum Depth and Sampling fraction for the performance of an RF on the validation set, trained on an under- and oversampled dataset. Numerical features were normalized and samples were drawn without replacement.

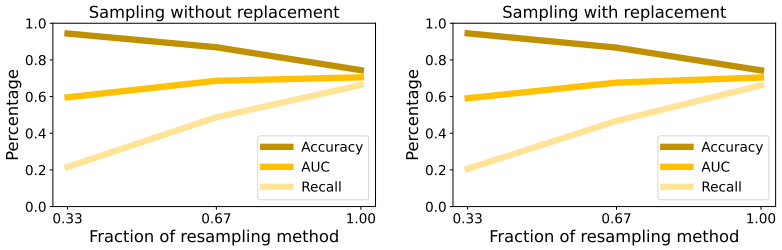


(a) Undersampling

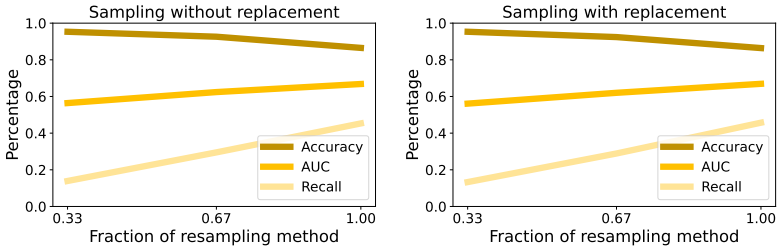


(b) Oversampling

Pairwise plot of Sampling with or without replacement and Maximum Depth for the performance of an RF on the validation set, trained on an under- and oversampled dataset. The datasets were completely rebalanced and numerical features were normalized.

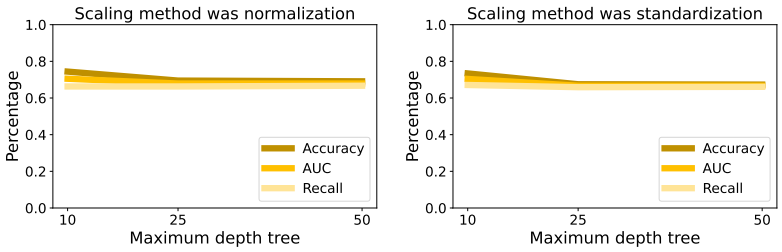


(a) Undersampling

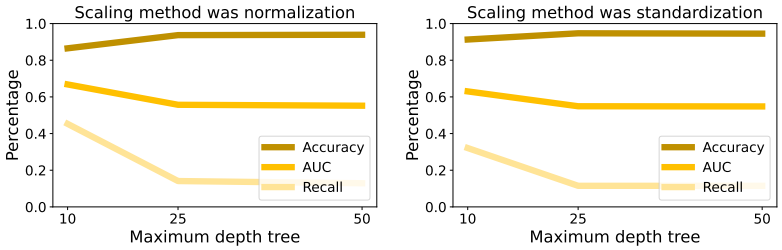


(b) Oversampling

Pairwise plot of Sampling with or without replacement and Sampling fraction for the performance of an RF on the validation set, trained on an under- and oversampled dataset. The trees had a maximum depth of 10 splits and numerical features were normalized.

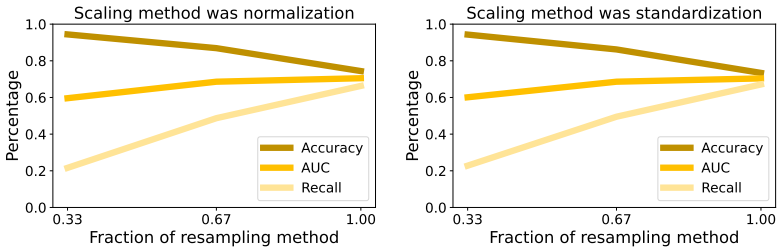


(a) Undersampling

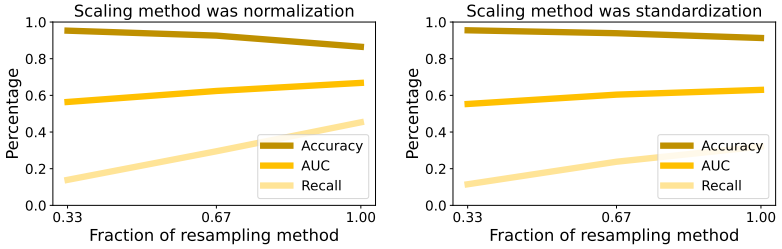


(b) Oversampling

Pairwise plot of Scaling method and Maximum Depth for the performance of an RF on the validation set, trained on an under- and oversampled dataset. The datasets were completely rebalanced and samples were drawn without replacement.

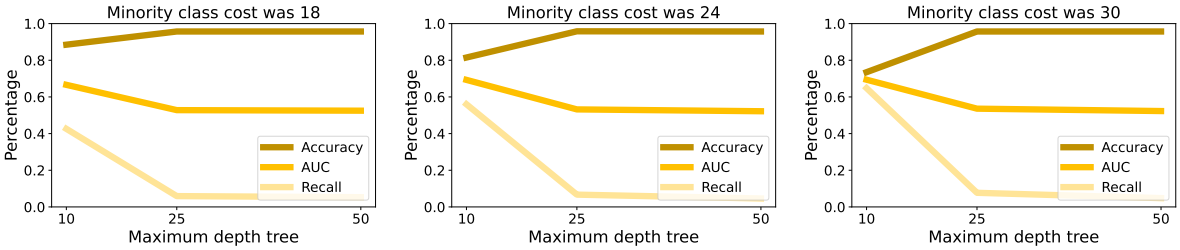


(a) Undersampling

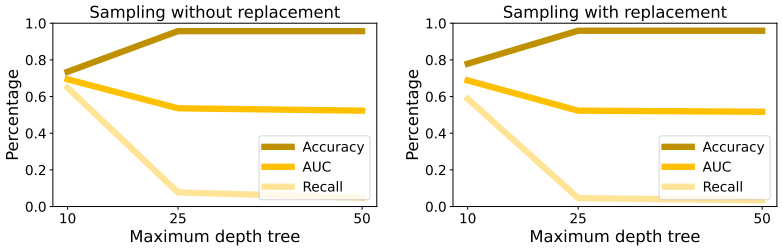


(b) Oversampling

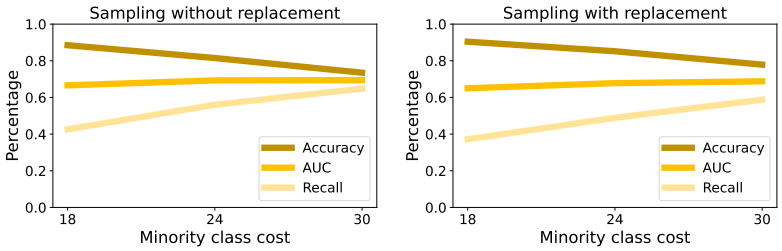
Pairwise plot of Scaling method and Sampling fraction for the performance of an RF on the validation set, trained on an under- and oversampled dataset. The trees had a maximum depth of 10 splits and samples were drawn without replacement.



Pairwise plot of Maximum Depth and Misclassification cost of minority class for the performance of an RF on the validation set. Numerical features were normalized and samples were drawn without replacement.

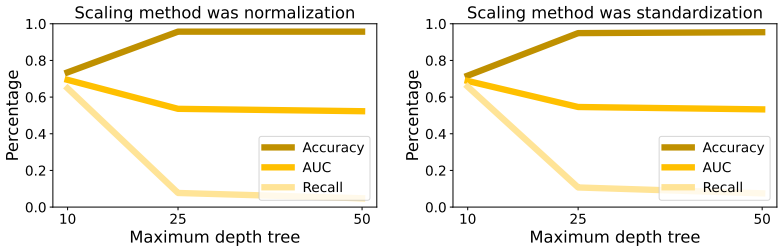


(a) Maximum Depth

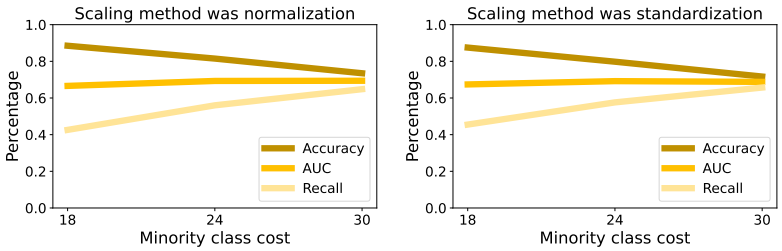


(b) Minority class cost

Pairwise plots of Sampling with or without replacement and Maximum Depth or Minority class cost for the performance of an RF on the validation set. The numerical features were normalized.



(a) Maximum Depth



(b) Minority Class cost

Pairwise plots of Scaling method and Maximum Depth or Minority class cost for the performance of an RF on the validation set. The samples were drawn without replacement.

Appendix F

Stochastic Gradient Descent

As Equation (2.6) indicates, the total cost function is calculated by summing the costs over all training observations $i = 1, \ldots, n$. In each training iteration (or epoch), the gradient $\nabla C(\mathbf{W}^l, \mathbf{b}^l)$ is estimated by averaging over the separate gradients for each observation:

$$\nabla C(\mathbf{W}^l, \mathbf{b}^l) = \frac{1}{n} \sum_{i=1}^{n} \nabla C_i(\mathbf{W}^l, \mathbf{b}^l)$$

However, as the number of training observations n becomes large, computing all separate gradients takes a lot of time (Nielsen, 2015). To limit the computational time, the gradient $\nabla C(\mathbf{W}^l, \mathbf{b}^l)$ can be estimated by averaging over only a random sample of size m (the so-called mini-batch size) of the gradients:

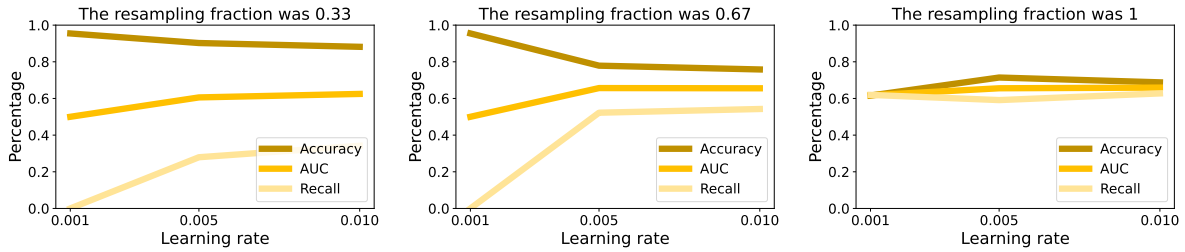
$$\nabla C(\mathbf{W}^l, \mathbf{b}^l) \approx \frac{1}{m} \sum_{i=1}^{m} \nabla C_i(\mathbf{W}^l, \mathbf{b}^l)$$

This process is known as stochastic gradient descent (Bottou et al., 1991). During each training epoch, the model is trained by iteratively picking a randomly selected mini-batch of samples, calculating the adherent gradient and updating the weights and biases. This process is repeated until all mini-batches (or: samples) have been used. This way, the weights and biases can be updated more frequently, speeding up the process of learning.

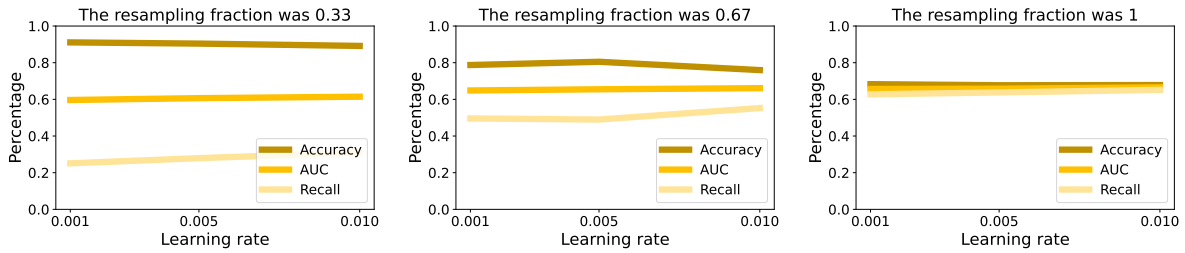
When m is chosen large enough, the mini-batches will provide a good estimate of the average over all separate gradients, whilst being able to train the model more quickly. However, if it is set too large, the weights and biases are not updated often enough (Nielsen, 2015). In all MLPs in this thesis, a mini-batch size of 20 samples was used.

Appendix G

Pairwise plots Hyper parameters MLP

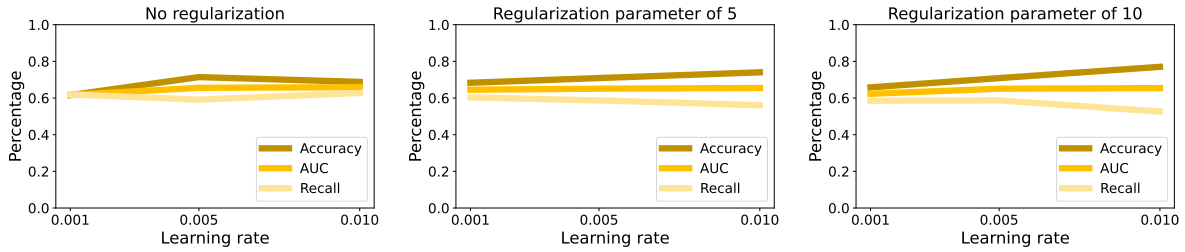


(a) Undersampling

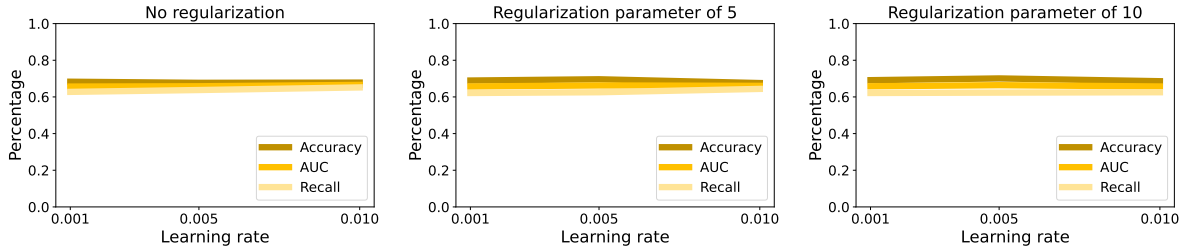


(b) Oversampling

Pairwise plot of Learning rate η and Sampling fraction for the performance of an MLP on the validation set, trained on an under- and oversampled dataset. The models contained 87 neurons in the hidden layer and no regularization was applied.

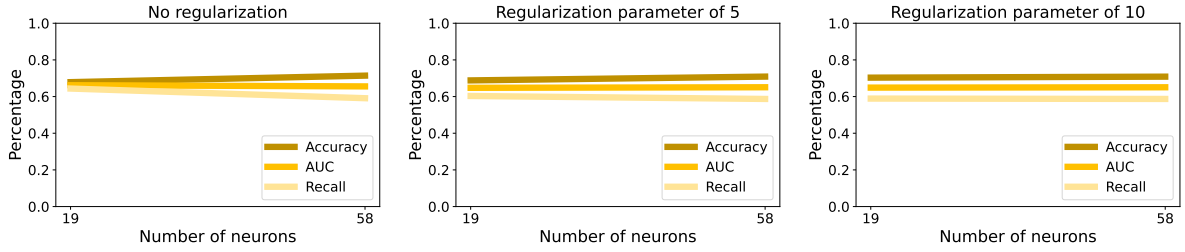


(a) Undersampling

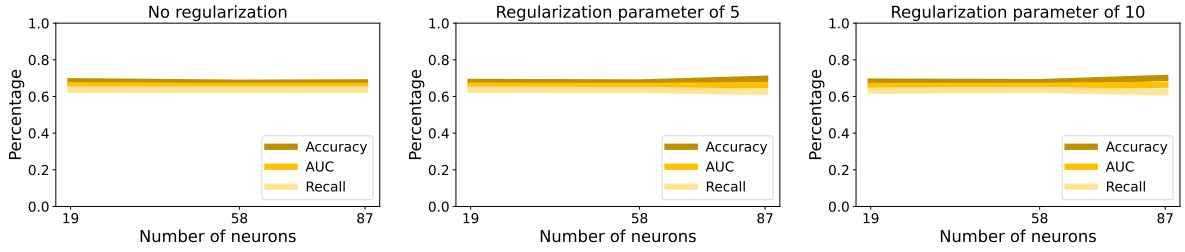


(b) Oversampling

Pairwise plot of Learning rate η and Regularization parameter λ for the performance of an MLP on the validation set, trained on an under- and oversampled dataset. The datasets were completely rebalanced and the models contained 87 neurons in the hidden layer.

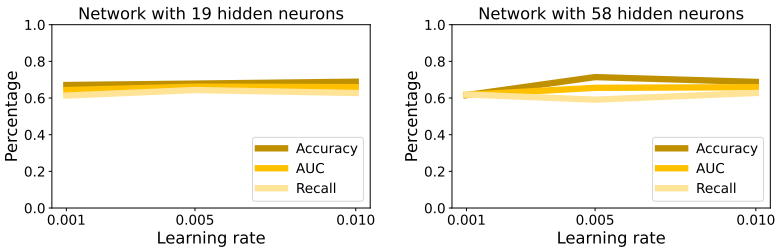


(a) Undersampling

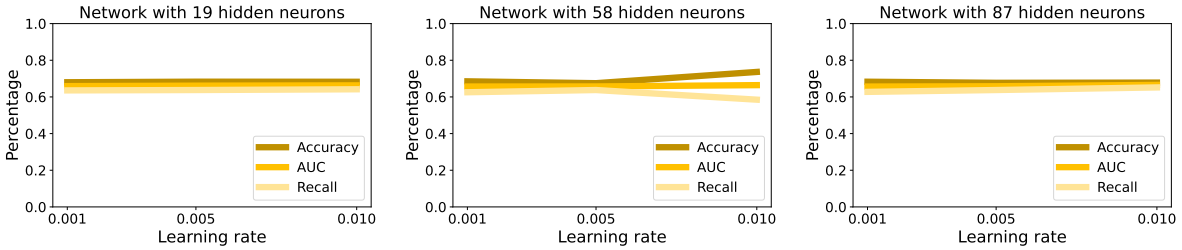


(b) Oversampling

Pairwise plot of number of neurons in hidden layer and Regularization parameter λ for the performance of an MLP on the validation set, trained on an under- and oversampled dataset. The datasets were completely rebalanced and the learning rate η was 0.05.

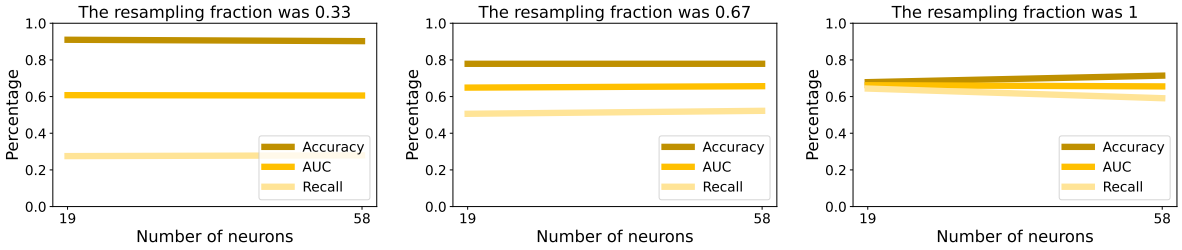


(a) Undersampling

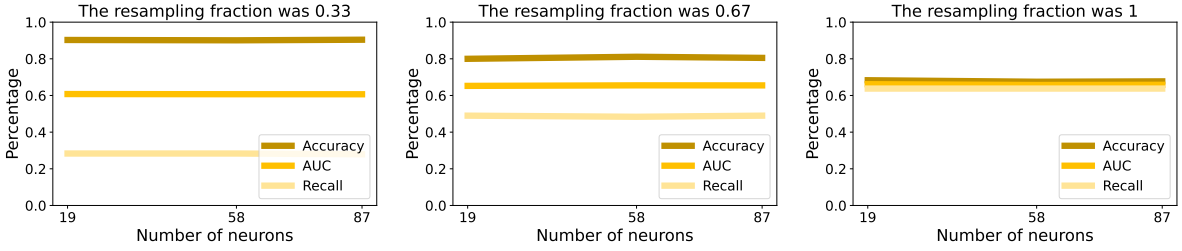


(b) Oversampling

Pairwise plot of Learning rate η and number of neurons in the hidden layer for the performance of an MLP on the validation set, trained on an under- and oversampled dataset. The datasets were completely rebalanced and no regularization was applied.

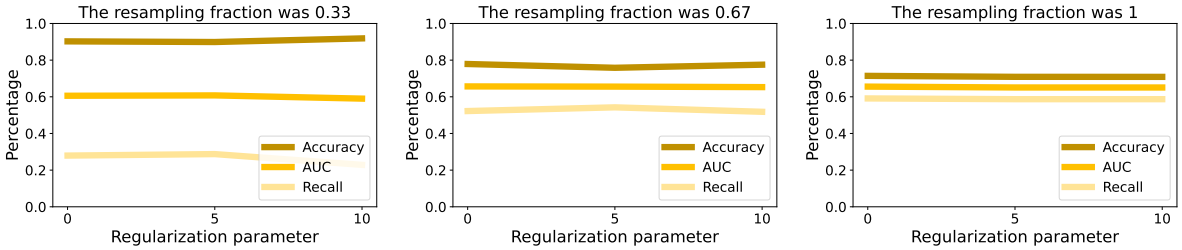


(a) Undersampling

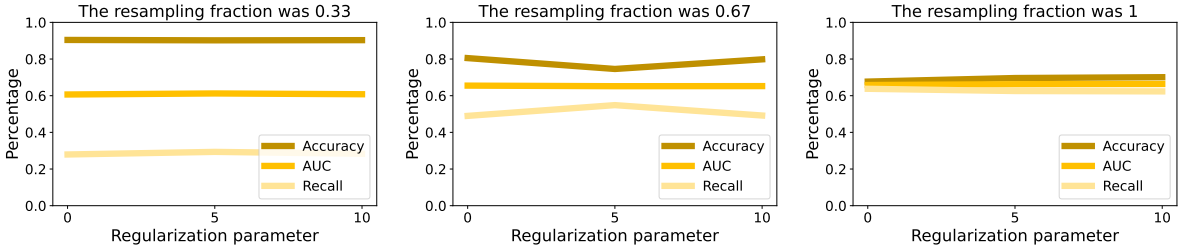


(b) Oversampling

Pairwise plot of number of neurons in hidden layer and Sampling fraction for the performance of an MLP on the validation set, trained on an under- and oversampled dataset. The models contained 87 neurons in the hidden layer and no regularization was applied.



(a) Undersampling



(b) Oversampling

Pairwise plot of Regularization parameter λ and Sampling fraction for the performance of an MLP on the validation set, trained on an under- and oversampled dataset. The models contained 87 neurons in the hidden layer and the learning rate η was 0.05.

Appendix H

Reorder Levels TWH

The reorder levels used in the TWH are based on an average sales quantity, a target quantity and a batch size. To start, the average sales quantity ($ASQ^{ca}_{i,t}$) for each SKU i is defined as the average of six days of historic (real) demand $D^{ca}_{i,t}$ and four days of forecasted demand[1] $F^{ca}_{i,t}$:

$$ASQ^{ca}_{i,t} = \frac{1}{10}\left(\sum_{t=-5}^{0} D^{ca}_{i,t} + \sum_{t=1}^{4} F^{ca}_{i,t}\right),$$

where $t = 1$ corresponds to the current day. The target quantity for each SKU reflects, as the name suggests, the desired stock level in the TWH for each SKU. This parameter is based on the average sales quantity and a so-called replenishment constant RC. This constant reflects the TWH properties and is defined as:

$$RC \approx \frac{K^l}{\sum_i L_i} \cdot \frac{K^l}{\bar{D}^{ca}},$$

where K^l represents the total number of locations in the TWH, L_i represents the number of locations one case of SKU i takes in and $\bar{D}^{ca}$ represents the average daily demand of stores for all SKUs handled by the OPM. Consequently, the first term reflects the average number of cases per SKU which fit in the TWH and the second term reflects the average DOS in the TWH. The replenishment constant was rounded to 160 for all SKUs i and all days t. Then, the target quantity $TQ^{ca}_{i,t}$ is defined as:

$$TQ^{ca}_{i,t} = \sqrt{RC \cdot ASQ^{ca}_{i,t}},$$

The definition of the batch sizes $Q^{ca}_{i,t}$ are presented in Section 5.1. Finally, the reorder levels $s^{ca}_{i,t}$ are defined as:

$$s^{ca}_{i,t} = TQ^{ca}_{i,t} - \frac{Q^{ca}_{i,t}}{2}$$

A replenishment is triggered if the inventory position for an SKU i in the TWH drops below the reorder level. Then, the order quantity corresponds to $n\,Q^{ca}_{i,t}$, where n equals the minimum number of batches $Q^{ca}_{i,t}$ needed to raise the inventory position above the reorder level. An additional rule is implemented that favors the replenishment of half full pallets over other partial pallets. This rule is based on a certain minimum quantity $MQ^{ca}_{i,t}$ for each SKU, which is defined as:

$$MQ^{ca}_{i,t} = \frac{F^{ca}_{i,t}}{3}$$

Then, the rule that potentially rounds replenishment quantities up to half of a pallet is defined as:

[1] In this formulation of the average sales quantity, the forecasted and historic demand all receive the same weight. However, in reality different weights are assigned to these values, where historic demand receives higher weights than forecasted demand and days closer to the actual day receive higher weights than days further in the future or in the past. However, the actually assigned weights were unknown, such that the average sales quantity was approximated as just the average of all values.

Rule Half Full Pallets: If the difference between the target quantity and the minimum quantity is less than half of a pallet (i.e. $TQ_{i,t}^{ca} - MQ_{i,t}^{ca} \leq \frac{1}{2} FP_i^{ca}$) and the inventory on half of a pallet corresponds to less than $RFP = 5$ days of forecasted demand (i.e. $\frac{1}{2} FP_i^{ca} \leq \sum_{t=1}^{RFP} F_{i,t}^{ca}$): $n\, Q_{i,t}^{ca}$ is rounded up to a half of a pallet (or the nearest integer of full layers).

where FP_i^{ca} reflect the number of cases on a full pallet of SKU i and RFP reflects the maximum reach of a full pallet.

Appendix I

Algorithm Rolling Horizon MINLP

Algorithm 1: Implementation Mathematical Model

Data: Λ, set of objective weights;
TWH_λ, replenishment strategy over entire period of interest for certain value of λ;
K^{ca}_{fp}, used TWH capacity of SKUs in $\mathcal{I}_{fp}$;
$C^p_{fp}, C^l_{fp}, C^{ca}_{fp}$, used DEPAL capacity (in pallets, layers, cases) of SKUs in $\mathcal{I}_{fp}$;
Sets $\mathcal{I}$, $\mathcal{I}_{fp}$, $\mathcal{J}$, $\mathcal{T}$, $\mathcal{X}_i$, explanations are provided in Appendix J;
Parameters $SI^l_{i,x}, SI^{ca}_i, K^{ch}, C^p, C^l, C^{ca}, \alpha, \beta$, explanations are provided in Appendix J;
Result: Solution $TWH_\lambda \quad \forall \lambda \in \Lambda$
for $j \in \mathcal{J}$ **do**
  Solution $FP_j \leftarrow$ Solution of standard (R, s, nQ) policy $\forall i \in \mathcal{I}^{fp}, t \in \mathcal{T}$;
  return Solution FP_j, K^{ch}_{fp}, $C^p_{fp}, C^l_{fp}, C^{ca}_{fp}$;
  $K^{ch} \leftarrow (1-\alpha)(1-\beta)K^{ch} - K^{ch}_{fp}$;
  $C^p \leftarrow (1-\alpha)C^p - C^p_{fp}$;
  $C^l \leftarrow (1-\alpha)C^l - C^l_{fp}$;
  $C^{ca} \leftarrow (1-\alpha)C^{ca} - C^{ca}_{fp}$;
  for *all* $\lambda \in \Lambda$ **do**
    if $j = 1$ **then**
      $I^{ca}_{i,1} = SI^{ca}_i \quad \forall i \in \mathcal{I}$;
      $I^l_{i,1,x} = SI^l_{i,x} \quad \forall i \in \mathcal{I}, x \in \mathcal{X}_i$;
      $BO^{ca}_{i,0} = 0 \quad \forall i \in \mathcal{I}$;
    else
      $I^{ca}_{i,1} = I^{ca}_{i,|\mathcal{T}|} \quad \forall i \in \mathcal{I}$, obtained from Solution $MINLP_{\lambda,j-1}$;
      $I^l_{i,1,x} = I^l_{i,x,|\mathcal{T}|} \quad \forall i \in \mathcal{I}, x \in \mathcal{X}_i$, obtained from Solution $MINLP_{\lambda,j-1}$;
      $BO^{ca}_{i,0} = BO^{ca}_{i,|\mathcal{T}|} \quad \forall i \in \mathcal{I}$, obtained from Solution $MINLP_{\lambda,j-1}$
    end
    Solution $MINLP_{\lambda,j} \leftarrow$ Solution of MINLP for λ, $I^{ca}_{i,1}, I^l_{i,1,x}$ and $BO^{ca}_{i,0}$;
    return Solution $MINLP_{\lambda,j}$;
  end
  Solution $TWH_\lambda \leftarrow$ Solution FP_j + Solution $MINLP_{\lambda,j}$;
  return Solution TWH_λ;
end

Appendix J

Nomenclature Mathematical Model

Description of decision variables

Decision variable	Description
$\Theta_{i,t}$	Binary variable indicating whether a replenishment of SKU i on day t included a partially depalletized pallet (1) or not (0).
$\Omega_{i,t}$	Continuous variable indicating the part of customer demand that is lost for SKU i on day t.
$\Phi_{i,t,x}$	Binary variable indicating whether a certain pallet x of SKU i was (fully or partially) depalletized on day t.
$I^{ca}_{i,t}$	Inventory on hand of SKU i on day t in TWH, expressed in cases.
$I^{ch}_{i,t}$	Inventory on hand of SKU i on day t in TWH, expressed in occupied channels.
$I^{l}_{i,t,x}$	Inventory on hand of SKU i on day t on pallet x in HBW, expressed in layers.
$LS^{ca}_{i,t}$	Lost sales (i.e. postponed demand) of customers at TWH, for SKU i on day t, expressed in cases.
$LS^{l}_{i,t}$	Lost sales (i.e. postponed demand) of TWH at HBW, for SKU i on day t, expressed in layers.
$n_{i,t}$	Ordered number of full batches of TWH at HBW for SKU i at day t.
$RE^{l}_{i,t,x}$	Replenishment quantity from HBW to TWH for SKU i on day t, from a certain pallet x, expressed in full layers.
$Z^{l}_{i,t}$	Dummy variable for each SKU i on day t, expressed in layers.

Description of parameters

Parameter	Description
α	Percentage of DEPAL and TWH capacity reserved for high priority replenishments.
β	Percentage of TWH capacity lost due to inefficient usage of channels.
λ	Weight applied to minimization of partial depalletizations.
$ASQ_{i,t}^{ca}$	Average sales quantity for SKU i on day t, expressed in cases.
C^{ca}	Total daily maximum capacity of the DEPALs, expressed in cases.
C^{l}	Total daily maximum capacity of the DEPALs, expressed in layers.
C^{p}	Total daily maximum capacity of the DEPALs, expressed in pallets.
$D_{i,t}^{ca}$	Customer demand for SKU i on day t, expressed in cases.
$\bar{D}^{ca}$	Average daily demand of stores for all SKUs handled by the OPM, expressed in cases.
$F_{i,t}^{ca}$	Forecasted customer demand for SKU i on day t, expressed in cases.
FP_{i}^{ca}	Cases per full pallet for SKU i, expressed in cases.
$IT_{i,t,x}^{l}$	Inbound quantity from external suppliers for SKU i on day t on pallet x, expressed in layers.
K^{ch}	Capacity of TWH, expressed in channels.
K^{l}	Capacity of TWH, expressed in locations.
L_i	Number of locations one case of SKU i fills.
$MQ_{i,t}^{ca}$	Minimum quantity for SKU i on day t, expressed in cases.
P_{i}^{ca}	Cases per layer for SKU i, expressed in cases.
Q_{i}^{ca}	Batch size for SKU i on day t, expressed in cases.
RC	Replenishment constant TWH.
RFP	Maximum days of stock of a full pallet, expressed in days.
RPP	Maximum days of stock of a partial pallet, expressed in days.
$s_{i,t}^{ca}$	Reorder level at TWH for SKU i on day t, expressed in cases.
SI_{i}^{ca}	Start inventory TWH for SKU i, expressed in cases.
$SI_{i,x}^{l}$	Start inventory HBW for SKU i on pallet x, expressed in layers.
$TQ_{i,t}^{ca}$	Target quantity for SKU i on day t, expressed in cases.

Description of sets

Sets	Description
Λ	Set of objective weights that are used to solve the MINLP.
$\mathcal{I}$	Set of SKUs i that have a batch size less than a full pallet and are included in the MINLP.
$\mathcal{I}_{fp}$	Set of SKUs i that have a batch size equal to a full pallet during the entire period of interest and are excluded from the MINLP.
$\mathcal{J}$	Set of periods j for which the MINLP is consecutively solved using the rolling horizon algorithm.
$\mathcal{T}$	Set of days t considered in one period.
$\mathcal{X}_i$	Set of pallets x of SKU i present in the CDC during time period $\lvert\mathcal{T}\rvert$.