Master Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

Optimization of container planning in inland terminals

Bc. Martin Komínek

Supervisor: Ing. Martin Komárek
Field of study: Open informatics
Subfield: Artificial Intelligence
May 2024

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Komínek Martin**	Personal ID number:	**483612**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Computer Science**		
Study program:	**Open Informatics**		
Specialisation:	**Artificial Intelligence**		

II. Master’s thesis details

Master’s thesis title in English:

Terminal Operation Optimization Using Machine Learning

Master’s thesis title in Czech:

Optimalizace běhu kontejnerového terminálu pomocí strojového učení

Guidelines:

METRANS is one of the market leaders in intermodal transport. They transport containers from sea ports to numerous intermodal terminals with their shuttle trains.
The aim of the thesis is to improve the current operations on terminals and use machine learning and planning to reduce the number of operations that are necesary.
1.Familiarize yourself with current processes on terminals
2.Perform a search for current state-of-the-art methods in terminal operation optimization
3.Choose methods that you will use to improve operations on terminals
4.Implement these methods and deploy them into CodeNow infrastructure
5.Evaluate proposed methods

Bibliography / sources:

[1] Galle, Virgile. “Optimization models and methods for storage yard operations in maritime container terminals.” Doctoral thesis, Massachusetts Institute of Technology (2018).
[2] E. Pap, G. Bojanic, N. Ralevic, M. Georgijevic and V. Bojanic, "Crane scheduling method for train reloading at inland intermodal container terminal," 2012 IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics, Subotica, Serbia, 2012.
[3] C. Colombaroni, G. Fusco, N. Isaenko and L. Quadrifoglio, "Optimization of container operations at inland intermodal terminals," 2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS), Naples, Italy, 2017.

Name and workplace of master’s thesis supervisor:

Ing. Martin Komárek Center for Software Training FEE

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **09.02.2024** Deadline for master's thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

Ing. Martin Komárek Supervisor’s signature	Head of department’s signature	prof. Mgr. Petr Páta, Ph.D. Dean’s signature

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Acknowledgements

Chtěl bych poděkovat mojí mamince za její podporu, bez níž by tato práce nemohla nikdy vzniknout. Dále bych chtěl poděkovat svojí nejdražší přítelkyni Pétě za její bedlivý dohled na moje studium, na mojí práci a za podporu, kterou mi každý den dává. Chtěl bych také poděkovat svému vedoucímu za čas a vedení při psaní této práce a příležitosti, která mi tato práce přinesla. Dále bych chtěl poděkovat firmě Metrans za spolupráci při psaní diplomové práce a poskytnutí dat. V neposlední řadě také patří díky Páje, za korektury a kritiku mojí práce.

Declaration

I declare that this work is all my own work, and I have cited all sources I have used in the bibliography.

Prague, May 24, 2024

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje vs souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 24. května 2024

Abstract

The aim of this thesis is to analyze the current state-of-the-art methods for planning on terminals with container shipping. We will look at how terminals operate, what some challenges and bottlenecks are, and what solutions could be used. We will start by defining terms that are used in container transportation. After that, we will look at other papers, present their solutions and discuss if they could be used. The next chapter will describe what bottleneck was identified as the most crucial, what methods could be used to solve it, and what are some advantages and disadvantages of each method. With this knowledge, we will choose one solution and implement it. Finally, we will look at the results and discuss whether our choice was good and what could be some future improvements.

Keywords: Container, Predictions, Optimization, Planning

Supervisor: Ing. Martin Komárek

Abstrakt

Cílem této diplomové práce je analyzovat současné nejmodernější metody používané při plánování na terminálech s kontejnerovou přepravou. Podíváme se na to, jak v současnosti fungují terminály, jaké jsou aktuální problémy, co nejvíce zpomaluje provoz a pokusíme se navrhnout řešení. Začneme zadefinováním potřebných pojmů, které jsou používané v kontejnerové dopravě. Potom se podíváme na jiné publikace, představíme jejich řešení a jestli by bylo možné je použít. Další kapitola popisuje, jaké části procesu jsou nejkritičtější a jaké metody by šlo použít pro jejich vyřešení a jaké jsou jejich výhody a nevýhody. V následující kapitole využijeme těchto znalostí a vybereme jednu z těchto metod a naimplementujeme jí. V poslední kapitole se podíváme na výsledky a prodiskutujeme případná budoucí vylepšení.

Klíčová slova: Kontejner, Optimalizace, Predikce

Překlad názvu: Optimalizace pohybů kontejnerů na terminále

Contents

Figures

Tables

Chapter 1

Introduction

When a company needs a large volume or long-distance transport, intermodal containers are the answer. In 2022, around 96 million containers were handled in ports just for the EU.[1] Thanks to the standardization of containers, we are able to transport a container from a factory on the other side of the planet via trucks, trains and ships and deliver it to another factory that can make a new item. The uninterrupted flow of this logistical chain is crucial. We can see what happened when this logistical chain failed in 2021. When Suez was blocked, transport fees tripled, and some factories had to stop production because of it.

Container transportation is quite new as a concept. The idea of transporting goods in containers started around WW1. The first commercially successful idea was born in 1956, and it significantly reduced the number of people required to load a ship from a truck and decreased time in port. This also led to the ability to stack containers on top of each other on the ship's deck, so ships could transport more cargo per ship using their deck instead of internal cargo bays where all of the cargo was stored previously. The first container ship was able to carry 58 containers[2], and now the current biggest ship is able to carry 24346 containers[3], 420 times more than the first one. This year, depending on the size of the container and distance, it would cost you between 1000 and 6000 dollars to ship a container. [4] That is still a significant number, and if we can find any optimization in these processes, it could have an impact on lowering the cost of delivery even further. When you add the fact that any delay in delivery can cause a factory to shut down, there is a lot of incentive to ensure that containers flow as effectively as they can. Also, decreasing the number of unwanted manipulations decreases emissions that are connected with each such manipulation, which is something that companies have to take into account more and more, especially in the EU.

Inland terminals serve a crucial role in the container transportation network, and they face a unique set of challenges that significantly impact the efficiency and reliability of the supply chain. Unlike their coastal counterparts, inland terminals often have more limited space and infrastructure, making coordinating incoming and outgoing containers more complex. These challenges are further limited by technological adoption and inadequate real-time information systems. Addressing these inefficiencies is crucial, as any disruption in the smooth functioning of inland terminals can lead to delays, which causes plans to fail and can have various effects, such as missed trains, truck waiting, penalties, etc.

The main objective of this thesis is to summarize current knowledge about container transportation, look at all the aspects at once, identify if new advancements in computer science can lead to improvements in existing models, and maybe even redefine how these terminals operate. With new technology and automation, we can track containers more accurately and use this data to make better plans and predict their future behaviour, slowly aiming for fully automatic transport. By analyzing container logistics, this thesis aims to contribute to the more efficient, economical, and environmentally friendly movement of goods.

We will focus on inland terminals as they are usually overshadowed by port terminals and because there are no container terminals in the Czech Republic. We will look at current practices, some of the operational inefficiencies, and the best opportunities for innovation. The main difference from other studies will be its deployment in the real world. We will focus on optimization that can be achieved in the real world as often some prerequisites from other studies cannot be fulfilled.

1.1 Motivation

My own motivation was to be able to design, implement and train a method from machine learning and deploy it to production to learn the challenges of deploying and maintaining such a model. I also really like the planning for all of those containers, which have all sorts of limitations, constraints, and unique challenges that they present. Also, seeing the result of my work being useful in the real world is very rewarding.

This work has been done in collaboration with METRANS, a.s. Their network of terminals and data collected over several years was invaluable and crucial to fully understanding real constraints with container transportation. These data are confidential, but we will try to show as much as possible, and the rest will be shown on artificially created data.

Chapter 2
Introduction to container transportation

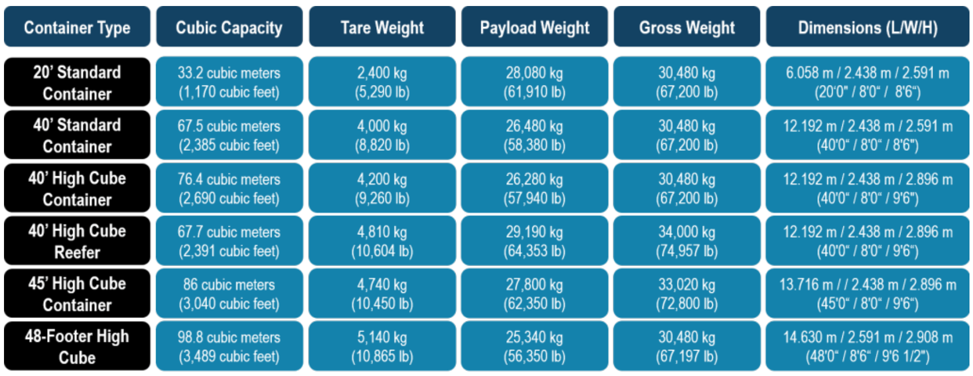
This chapter presents terms used in container transportation and how inland intermodal terminals work. We will discuss different sizes, where containers are stored, what machines are used to manipulate containers, what services containers can use, and what current processes are used when handling them.

2.1 Terms

2.1.1 Container

Most used containers are in 2 sizes. Either 20-foot(6,1 m) containers or 40-foot(12,2 m) containers. This is because container stands are usually for 20-foot containers, so when you need to place a 40-foot container, you place it in two spaces. This has the advantage of using most of the space in the designated area, decreasing wasted space. Other types, such as 10-foot, 24-foot, or even 45-foot, also exist but are much less frequent as they need either space specifically designed for them or they will block space that cannot be used by others. To decrease the space containers take, we can stack them on top of each other. The amount of containers we can stock on top of each other depends on the height, weight, and, most importantly, the position in which it is placed. For loaded containers, the maximum number of containers we can stack on top of each other is usually 3, so it doesn't intervene with crane lines, or the weight of containers above them doesn't crash them. This number can get much higher for empty containers, and we can see stacks of 6 or even 8. The container's width is almost always about 8 feet (2,44 m). There are mainly two heights definitions. Standard 8-foot or High-cube(HC) with 9.6-foot (2,96 m). Each container can carry a maximum of around 28 tuns. These sizes are standardized by ISO/TC 104, and an overview of them can be seen in this figure 2.1.

There are a lot of variants of intermodal containers. A refrigerated variant for goods needs to be stored at low temperatures. Tank containers for transportation of liquids and even flat racks for transportation of oversized cargo. Containers are also handled differently if they contain some dangerous cargo.



Container Type	Cubic Capacity	Tare Weight	Payload Weight	Gross Weight	Dimensions (L/W/H)
20' Standard Container	33.2 cubic meters (1,170 cubic feet)	2,400 kg (5,290 lb)	28,080 kg (61,910 lb)	30,480 kg (67,200 lb)	6.058 m / 2.438 m / 2.591 m (20'0" / 8'0" / 8'6")
40' Standard Container	67.5 cubic meters (2,385 cubic feet)	4,000 kg (8,820 lb)	26,480 kg (58,380 lb)	30,480 kg (67,200 lb)	12.192 m / 2.438 m / 2.591 m (40'0" / 8'0" / 8'6")
40' High Cube Container	76.4 cubic meters (2,690 cubic feet)	4,200 kg (9,260 lb)	26,280 kg (57,940 lb)	30,480 kg (67,200 lb)	12.192 m / 2.438 m / 2.896 m (40'0" / 8'0" / 9'6")
40' High Cube Reefer	67.7 cubic meters (2,391 cubic feet)	4,810 kg (10,604 lb)	29,190 kg (64,353 lb)	34,000 kg (74,957 lb)	12.192 m / 2.438 m / 2.896 m (40'0" / 8'0" / 9'6")
45' High Cube Container	86 cubic meters (3,040 cubic feet)	4,740 kg (10,450 lb)	27,800 kg (62,350 lb)	33,020 kg (72,800 lb)	13.716 m / / 2.438 m / 2.896 m (45'0" / 8'0" / 9'6")
48-Footer High Cube	98.8 cubic meters (3,489 cubic feet)	5,140 kg (10,865 lb)	25,340 kg (56,350 lb)	30,480 kg (67,197 lb)	14.630 m / 2.591 m / 2.908 m (48'0" / 8'6" / 9'6 1/2")

Figure 2.1: Example of container sizes [5]

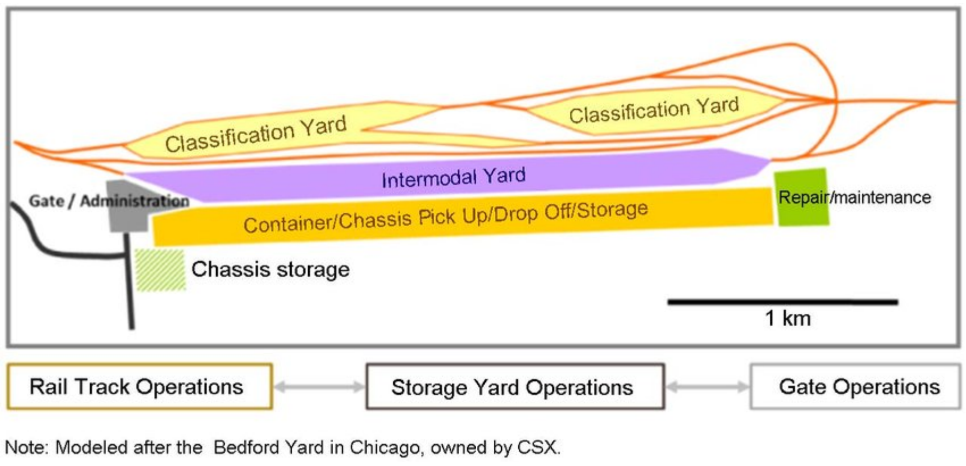


Figure 2.2: Example of terminal layout [6]

2.1.2 Terminal and stands

Terminals represent nodes in a logistical network that are connected together via railway. They serve as hubs, where trucks come to pick up or unload their containers and leave with a new one. Sometimes, containers can arrive at the terminal via train and will leave the terminal on a different train, so they are temporarily stored there. By area, the biggest part of the terminal are the container stands. Stands can be just lines on the ground or a special pedestal that fits just one type of container. Terminal is usually divided into several parts. The figure 2.2 shows an example of terminal layout. There are parts where loaded containers are stored, represented with purple colour in the figure. Other areas can be used to store empty ones. There are almost always rail tracks where trains stop to be unloaded or loaded. There are also small areas where trucks stop and where the loading and unloading of containers happens, represented by dark yellow in the figure. All of this is connected with various manipulation machines.

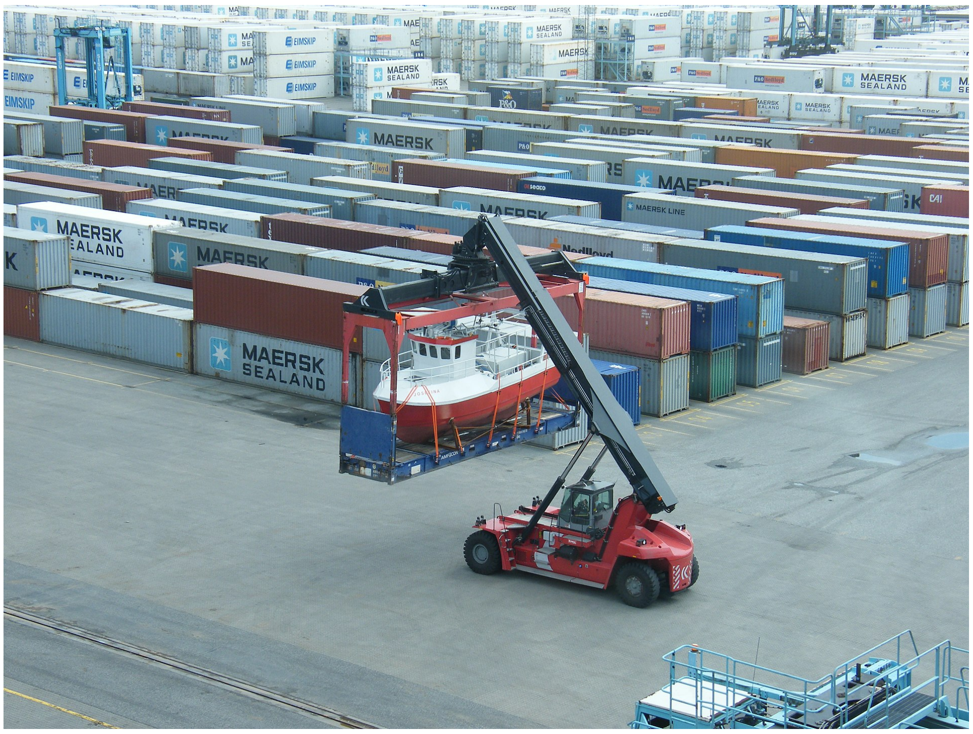


Figure 2.3: Example of terminal layout together with stacker [7]

2.1.3 Manipulation machines

Two types of equipment can manipulate containers. Cranes and stackers. Stackers are cheaper to purchase and are not constrained in their range, giving them a big benefit in mobility around the terminal. Price can vary greatly depending on lift capacity. A stacker for small containers can cost you around 30,000$, but a stacker for fully loaded containers is much more expensive, and you can expect to pay around 150,000$ for a used stacker, and a new one will be even higher.[8] Their biggest disadvantage is the cost of operation compared to the cranes and the inability to take containers from a bigger block. Operating a stacker is order of magnitude more expensive than cranes, as they consume a lot of fuel. If the container is surrounded by other containers, they have to relocate all containers in one direction to get to it, which limits their usage. Cranes are usually the preferred way to manipulate containers as they are cheaper to operate, faster, and can grab containers from the top, so a maximum of 2 containers need to be removed in order to grab the desired one. The limitation of cranes is in their range, which is usually some area around the rail tracks. Also, some terminals don't have cranes, as they are too small, and the cranes, together with their crane lanes, are expensive.

2.1.4 Manipulation types

As you can imagine, grabbing a container that is not easily accessible can be very hard. You have to do a lot of manipulations that are not useful for anything other than accessing some other container. We call each of these manipulations technological manipulation. Useful manipulation, which we call unique manipulations, represents a move from a transport vehicle, such as a train or truck, to a stand or from a stand to a transport vehicle. The ideal type of manipulation is from one transport to another, which requires the minimum number of manipulations and doesn't take space in terminal storage.

2.1.5 Services

Several services are needed during the transport of containers. The most frequent one is weighing, which is done for each container. It is necessary to weigh each container to ensure we don't reach a maximum load of containers beneath them or rail wagons, and it also affects the final price. Other important services are the heating and cooling of a container. Cargo of some containers needs to maintain some range of temperatures, so these containers must be easily accessible so that necessary tools can be attached. The last important service is maintenance, which ensures the good technical state of each container and, in case of some damage, is able to repair them on the terminal.

2.2 Identified operations that could be improved

Looking at current operations on terminals, there are a few potential areas that could be improved without completely changing the modus operandi. In this section, we will look at each of these areas, how significant they are, how things are done now and how they could be done and finally, what challenges the change would create.

2.2.1 Crane Operation

Cranes are very expensive. Machines like this can cost you around 400,000$ [9] plus expenditures on maintenance and operation, not accounting for the price of crane lines. When you consider that there are often around 3 or 4 of these on terminals, the cost really adds up. On the other hand, once the investment is made, they are cheaper than stackers by order of magnitude, so fully utilizing them is a key part of a successful business. They are usually used to load and unload trains or move stored containers. When a loaded train arrives, cranes completely unload the train, and then, if new containers are supposed to be loaded, they will start loading containers from the storage. This means that the crane is travelling empty in half of the trips. By planning the loading and unloading of the train, a new container can be placed in this spot once the container is unloaded from the train. This way, instead of

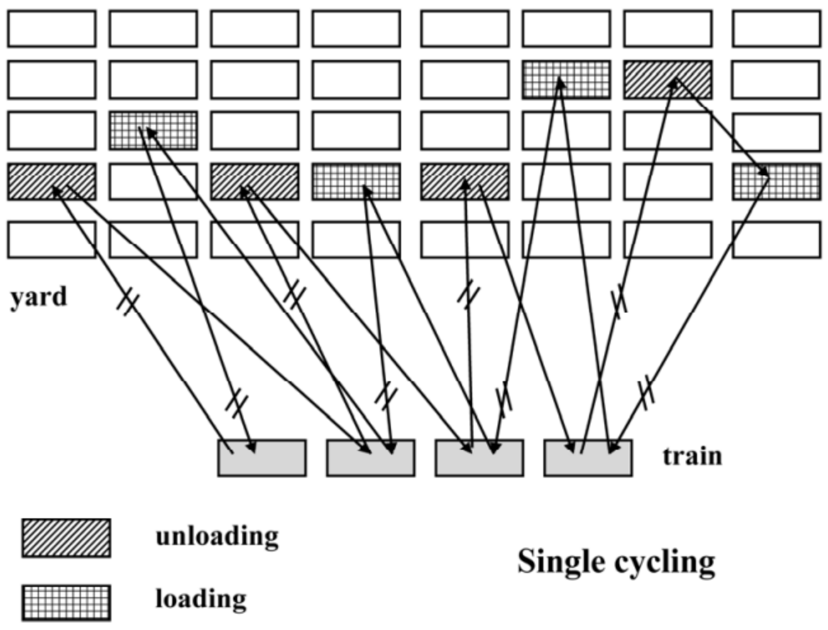


Figure 2.4: Example of separate loading and unloading from train [10]

the crane returning empty to the train, it will travel with a container. The difference between these two methods of loading and unloading can be seen in figure 2.4 and in figure 2.5. According to a study done on intermodal inland terminals [10], the expected reduction of time can be around 35% from previous times, which is a substantial increase in speed. Unfortunately, Metrans did not approve of this approach. It was not approved because, as we mentioned at the beginning of this thesis, there are different sizes of containers. When the containers are loaded onto the train vagon, there are hooks that are required to secure the container. These hooks are adjusted after all containers are unloaded, and only after that can loading begin. There is a possibility that due to mistakes, newly loaded containers could be unloaded or old containers not unloaded at all. Overall, it remains a good option for improving productivity on the terminal if the company could solve these issues and adjust its operations.

2.2.2 Creating a plan for loading and unloading

Another way to improve operations is by better planning container storage. Technological manipulations that we introduced earlier in this chapter represent around 10 to 30 % of all manipulations. With enough data in advance, these manipulations should be avoidable and can again represent a significant improvement in operations. Currently, all containers are placed on their position by a company-fixed policy that tries to fill places based on distance and the number of free places. By creating a planner, we could store the containers in the order they are going to leave. This would ideally remove the

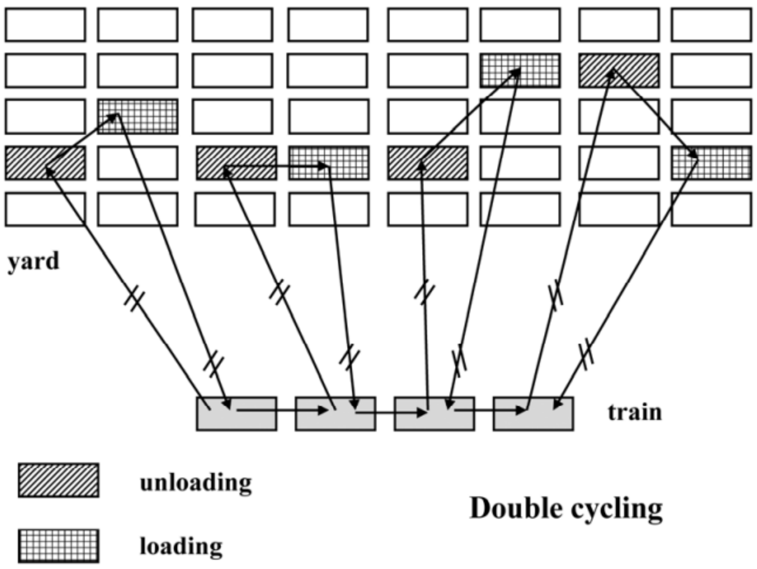


Figure 2.5: Example of simultaneous loading and unloading from train [10]

need for all technological manipulations. Unfortunately, realizing this simple idea is not easy. The number of combinations is extremely high, making the planning computationally demanding. Also, the parameters change over time, so even if we find the perfect plan for the current state of the terminal, a train may arrive and completely change what an ideal plan would look like. Even if all of these problems could be solved, there are many containers that don't have a departure time set, reducing the effect of planning even further, making any planning with them almost impossible. Despite all of these problems, the benefits of better planning are clear, and the company wants to implement such a planner but to do this, we need to solve the problem of containers without a set departure time and other prerequisites such as the accurate state of the terminal in real-time.

2.2.3 Predicting container departure based on historical data

As mentioned in the previous subsection, predicting container departure is a prerequisite for planning operations on the terminal. There are a lot of containers on the terminal that don't have departure times set for various reasons. It can be because the departure of the ship is not yet known, or the company that ordered the container doesn't have a truck ready, and many more. These containers still need to be stored somewhere. To avoid storing them randomly, which would result in a lot of technological manipulations, we could learn a predictor that would learn based on historical data and then, based on available features, would try to predict a departure time. This idea was approved by the company, and we cooperated together to develop and implement a predictor that could be moved to production.

Chapter 3
Data Preparation and Analysis

3.1 Introduction

The main focus of this chapter is to familiarize ourselves with available data, how we can work with them and modify them, and identify which parts of the data are important for learning. The problem we want to solve is, given the time of arrival of a container, how long it will take to leave the terminal, based on historical data, and what parameters affect this departure.

When it comes to improving our predictions, one thing is certain. More data is better. Ideally, we want uniformly distributed data from each category to make the most accurate predictions possible. In reality, those conditions are not usually met. For example, some customers can have more orders than others, or some terminals are more used than others as they have better locations and connections.

3.2 Data Acquisition and Description

In this thesis, we are working with 2 datasets. The first dataset contains the real data Metrans recorded in their operations from the start of 2023 until March 2024. This dataset has two parts, first for import orders and second for export orders. Import orders contain information about around 250000 containers, and export orders around 200000.

Unfortunately, this dataset is private and cannot be shared, but we can discuss some statistical properties and the results derived from it, which will be the topic of this chapter. The second dataset was created for two reasons. Firstly, it contains no sensitive information and can be published with this thesis. Secondly, by creating a completely artificial dataset, we can create a dependency between our data and our results, which is not guaranteed in the real dataset. We can also change the statistical properties of each variable to see the effect on the result. The key difference is that this dataset will have a departure time computed using an equation with some randomness added. Thanks to this dependency, we can ensure our methods work and determine how close they can get to theoretically achievable results or if they overfit the dataset. For part of the data used to train the parameters, we will refer

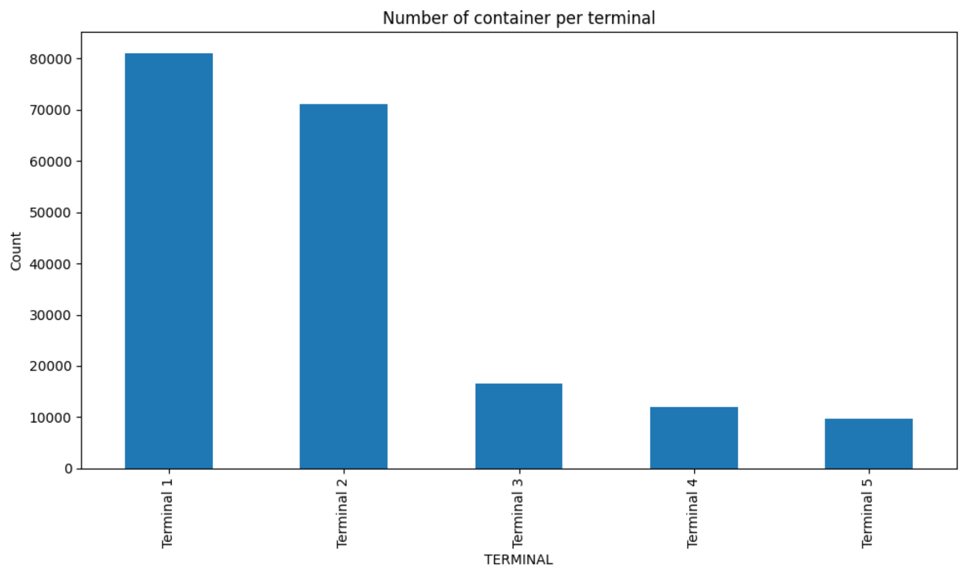


Figure 3.1: Distribution containers per of terminal / Export

to it as X, and for the value we want to predict, we will refer to it as Y. Both datasets are stored as a .xlsx file.

Let's start by analyzing and extracting statistical data from the Metrans dataset. It has 22 columns, of which 9 were identified as features that could be used for learning. All of these features are categorical data and dates. Columns removed from the training data consist of unique identifiers and precise addresses of the target destination, which would only lead to overfitting. Parameters that could affect departure are type of container, who ordered it, who will receive it, which port it is going or coming from, state to where it will be delivered, ZIP code of destination, on which terminal the container currently is, which transport will the container take and terminal where it is supposed to go. Another piece of information we can mine from these data comes from arrival time. Arrival time is quite unique, but from it, we can determine what day of the week it arrived and the week of the year.

Let us look at some features and discuss how they affect learning. The first graph shows the number of containers that flow through each terminal on the export path.

As we can see from this plot 3.1, there are two dominant terminals: Terminal 1 and Terminal 2. They both have close to 75000 containers. On the other hand, the three remaining terminals, Terminals 3,4 and 5, have around 10000 to 15000 containers. That means that there will be a big bias towards the bigger terminals, and the predictor will have better results for these terminals.

We can see a similar story on the histogram 3.2 from the import path. Terminal 1 and Terminal 2 have again around 5 to 10 times more containers than other terminals, shifting the bias towards the bigger terminals. To mitigate this, we could try learning a different predictor for each terminal, but we would considerably decrease the training data size.

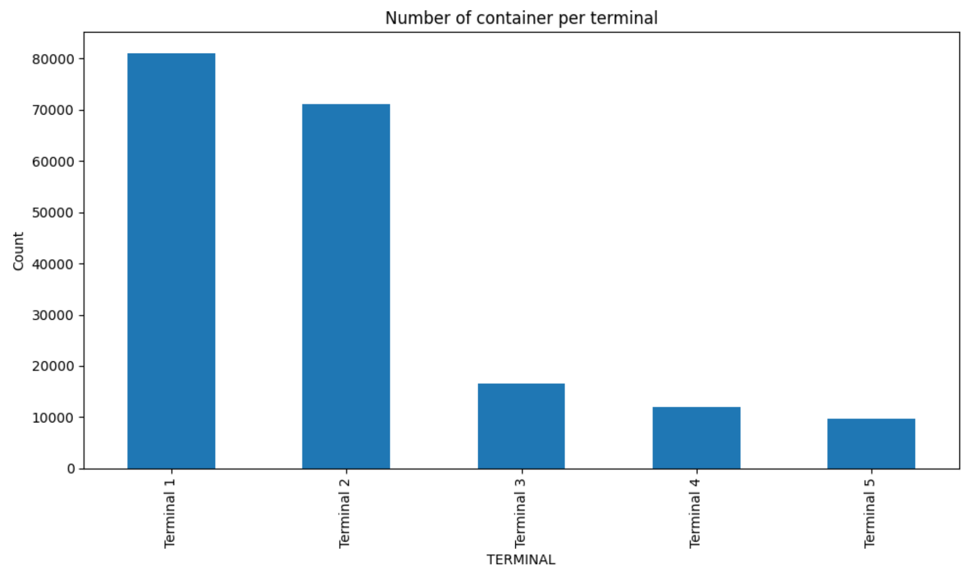


Figure 3.2: Distribution of containers per terminal / Import

Parameters that could have a very important impact on the departure time are OBJEDNATEL(order party) and PRIJEMCE(recipient). Those two values are usually not the same. One company prepares the cargo, it will order the container to be picked up and delivered to the recipient. There are many reasons why this feature could have a high impact on the departure time. Companies using a just-in-time system would emphasise getting the container as fast as possible. Other examples could be food companies, where cargo quality would slowly degrade. On the other end of the spectrum, many companies often cannot know when their container will come if they order something from overseas, so they are not so dependent on fast arrival. Unfortunately, how orders are distributed over customers is not public, so only text description is given.

First, we will examine how these parameters are distributed among import deliveries. There are hundreds of companies in the OBJEDNATEL column. A minimal number of orders is just one, but many companies have orders in thousands. Companies are not uniformly distributed and can differ by several orders of magnitude. The resulting predictor will work best for companies with a higher representation in the dataset, and small companies will represent only a small error in comparison. When we predict a value, there will be a high bias towards the big companies used in training. On the other hand, these small companies will create only a small percentage of requests to make a prediction, so it doesn't necessarily have to have a high impact on the system's accuracy. Exports have a similar distribution of orders.

For PRIJEMCE there are similar numbers. There are thousands of companies in PRIJEMCE category. Minimal number of orders is again 1. The maximum number of orders made by one company is in thousands. A lot of the statistical properties are similar to OBJEDNATEL and will again be biased towards bigger companies.

An example of a parameter that could have an effect on departure is the type of container. Firstly, let's look at their distribution. From figures 3.3 and 3.4, we can see that the two most common types of containers by far are 40hc and basic 20-foot containers. This is practical because a 40-foot high-cube container (40hc) can be placed over two standard 20-foot container stands, which are commonly available at most terminals. Due to its lower cost per volume, the 40hc is often the most popular choice.

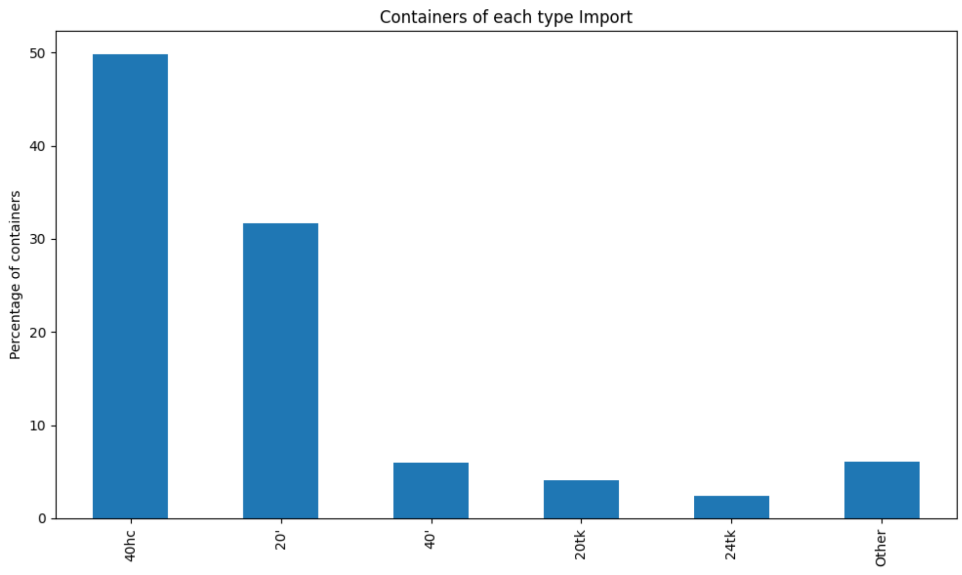


Figure 3.3: Distribution of containers by type for Import

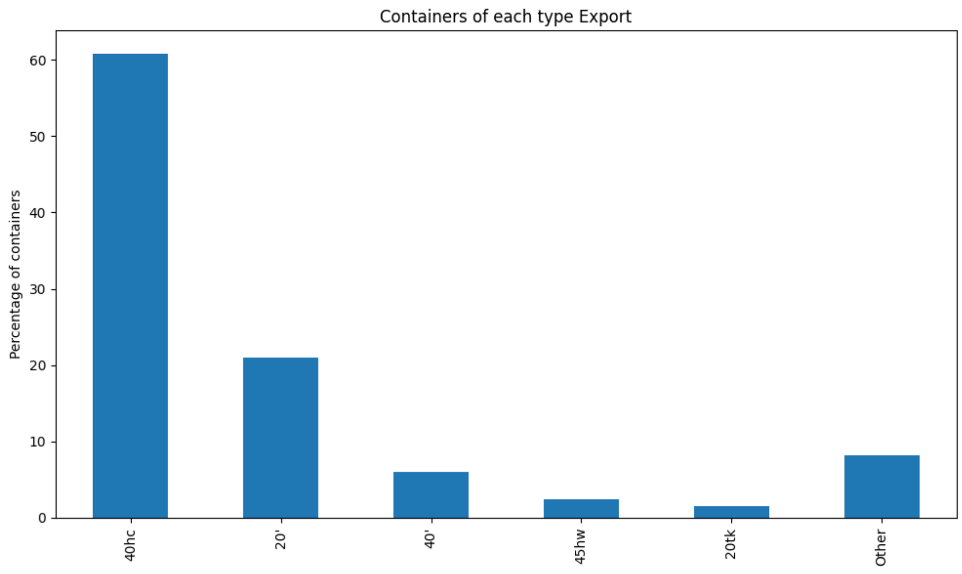


Figure 3.4: Distribution of containers by type for Export

In predicting container departure times, the destination port is a critical parameter. This aspect predominantly influences export routes but also has implications for imports. A high-usage port denotes the frequent arrival of trains, whereas ports with less traffic typically receive train services only a few times per week. An example of this can be seen on Metrans websites[11]. There are 10 trains per week from Hamburg to Prague. From Hamburg to Linz, only one per week. That can create a difference between 12 hours on terminal and 7 days. If the container arrives at Bremerhaven, the train to Prague comes only 5 times a week. The following figures show a distribution of destination ports for both import 3.5 and export 3.6.

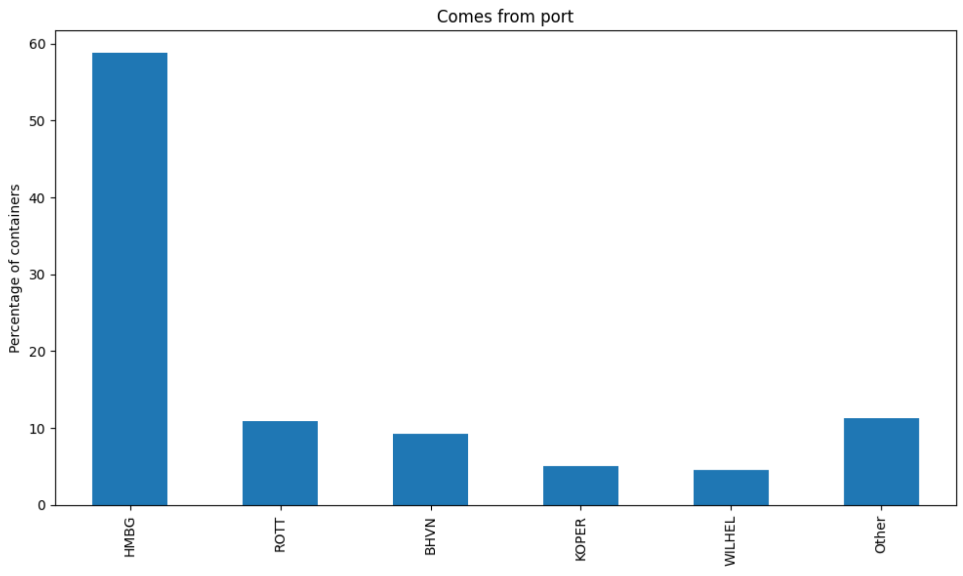


Figure 3.5: Distribution of arrival ports for Import

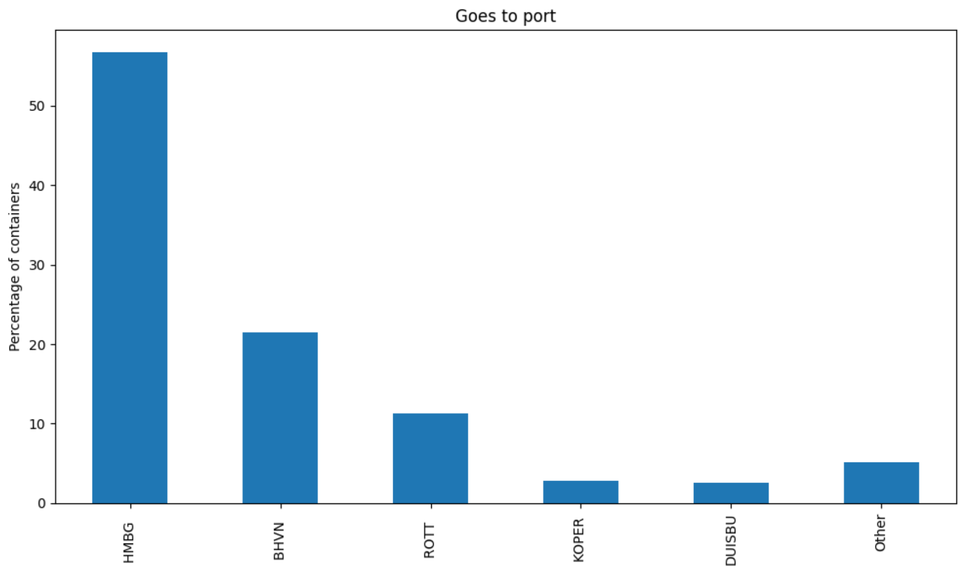


Figure 3.6: Distribution of destination ports for Export

For a similar reason, the country of destination can affect how long the container will last at the terminal. The company headquarters is in the Czech Republic, and the distribution of country destinations shows that for both import 3.7 and export 3.8, it creates a majority of final destinations.

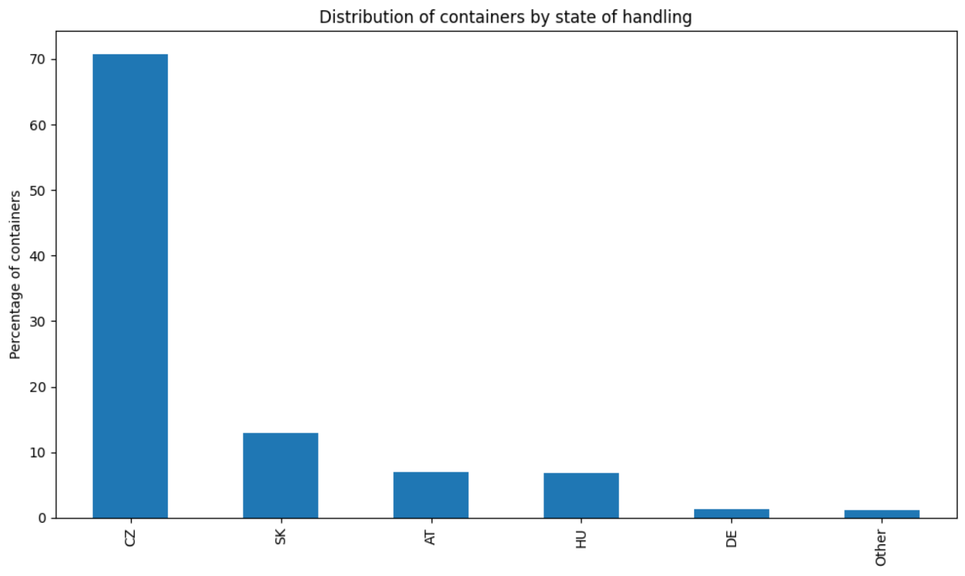


Figure 3.7: Distribution of countries for Import

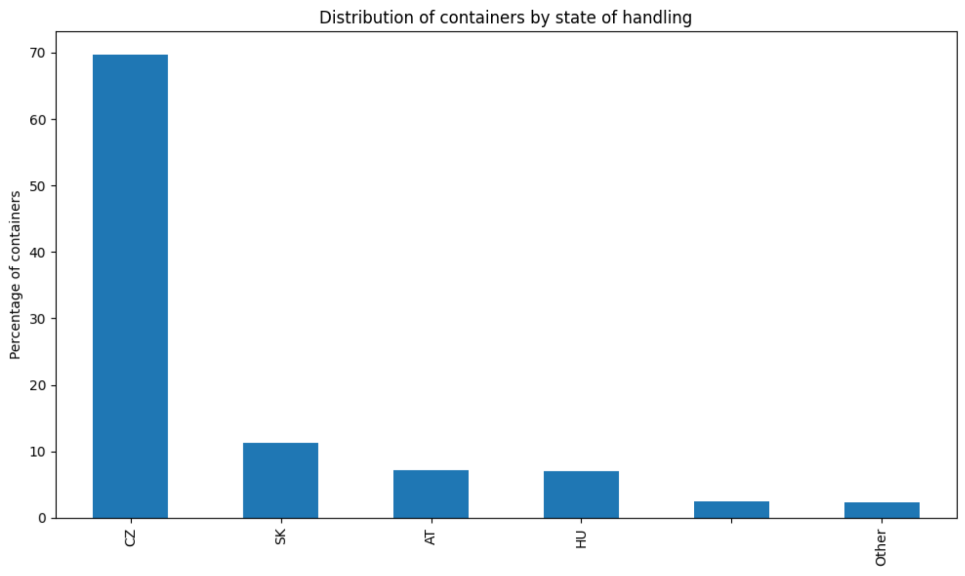


Figure 3.8: Distribution of countries for Export

A big impact on departure time can have on the day of arrival. For example, many companies are closed during the weekend or accept containers only on weekdays. That can mean that a container that arrives on Friday could leave the terminal on Monday instead of the next day. When we look at the import path, we can see this effect very clearly in figure 3.9. On Sunday, there are around 3x fewer containers processed than on weekdays. Saturday has about half of the normal capacity, but what is interesting is that Monday has around 77% of normal capacity. This could be explained by the fact that it takes time before trains arrive from the port, so if they start their journey on Monday, they will arrive at the terminal on Tuesday.

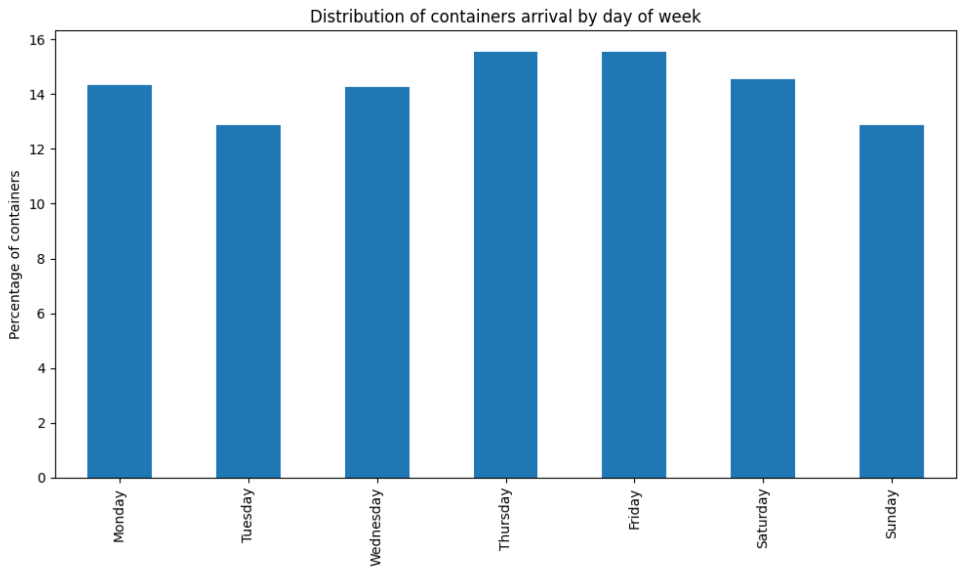


Figure 3.9: Distribution over the days of the week for Import

Even though we would expect to see the same pattern in figure 3.10 for export, this is not the case. One explanation could be that ships don't have this periodical cycle. Crews of ships usually serve for months at once before taking a vacation. Ships are also really valuable assets, and their maximum utilization is desired. Because ships will leave the port ASAP, the containers for that ship must get to the port in time, and trains on that path go even on weekends.

What week of the year the container arrives can also affect the departure of containers. We could expect a decrease in containers per week during major holidays such as Christmas, Easter and summer vacations. We can clearly see that by looking at figure 3.11. We can see that in January, the number of containers increased to around double that of the first week. Then, it remains steady until week 14, the week before Easter, and remains down the next week when Easter Monday is. A similar dip happens in weeks 18 and 19, corresponding with the national day of work and the end of WW2 celebrations. After that operation, they are back on normal levels until week 27, which is a week when another 2 Czech public holidays are. During summer, there are slightly fewer containers, but in fall, this number starts to climb, probably because of higher demands for goods in this period before peaking a week before Christmas and then going way down on Christmas and New Year's Eve. Looking at figure 3.12, there are similar patterns, but for example, the biggest peak of transport is at the end of summer and slowly decreases before

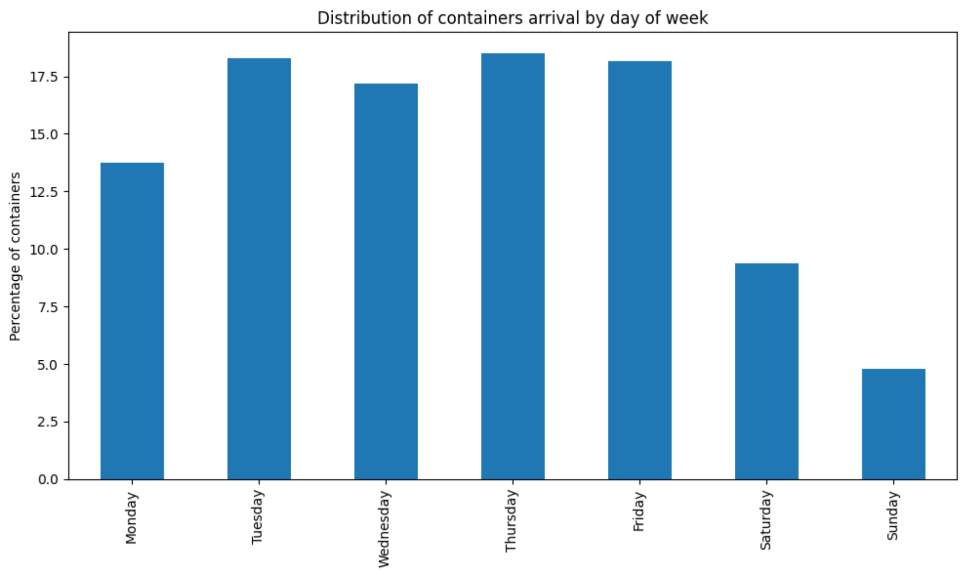


Figure 3.10: Distribution over the days of the week for Export

Christmas. The most probable explanation is that because these goods will be loaded onto ships and shipped elsewhere, it will take some time before they arrive at their final destination. Hence, the Christmas peak arrives earlier, so the goods reach their destination in time.

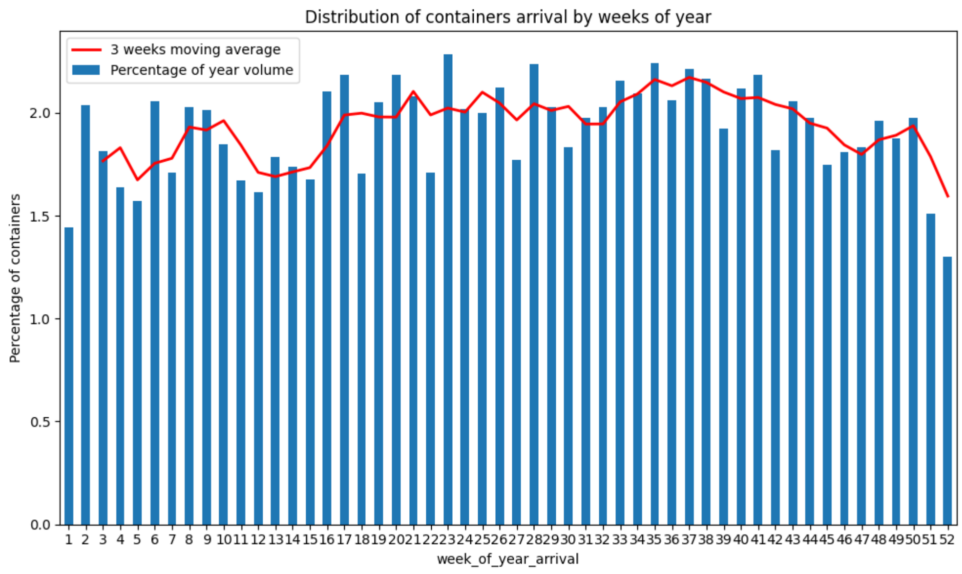


Figure 3.11: Distribution of arrival ports for Import

Lastly, it could be beneficial to look at some statistical properties of the value we want to predict, which is the delta between the arrival of the container at the terminal and the departure from that terminal.

Looking at figure 3.13, departure times are clearly distributed. The majority of containers are there for a short period of time, less than two days. Then,

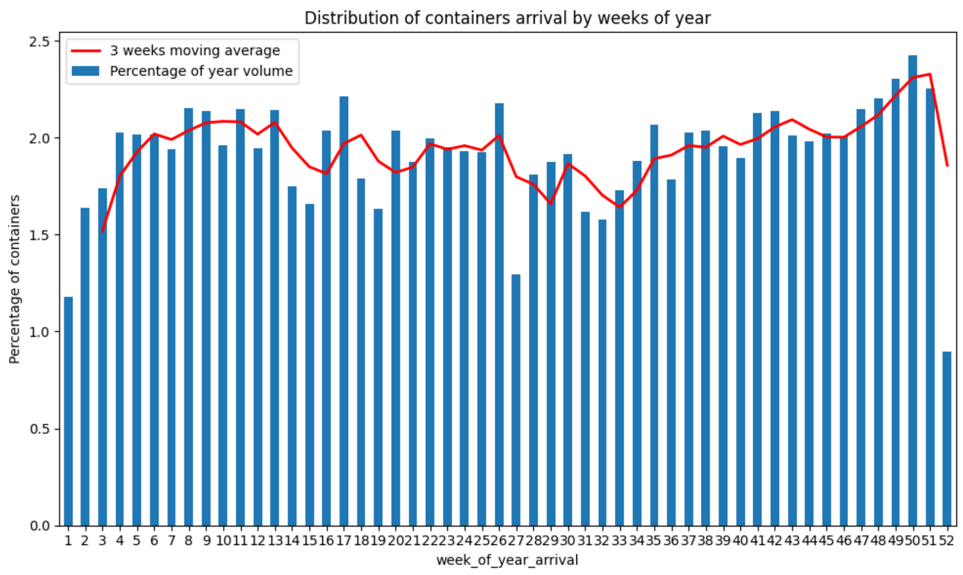


Figure 3.12: Distribution of destination ports for Export

the longer this value is, the fewer containers there are with this delta. Finally, containers with a delta higher than 10 were aggregated into one category, showing us containers that were stored for some time rather than shipped ASAP. A similar story can be seen in the export path. 3.14

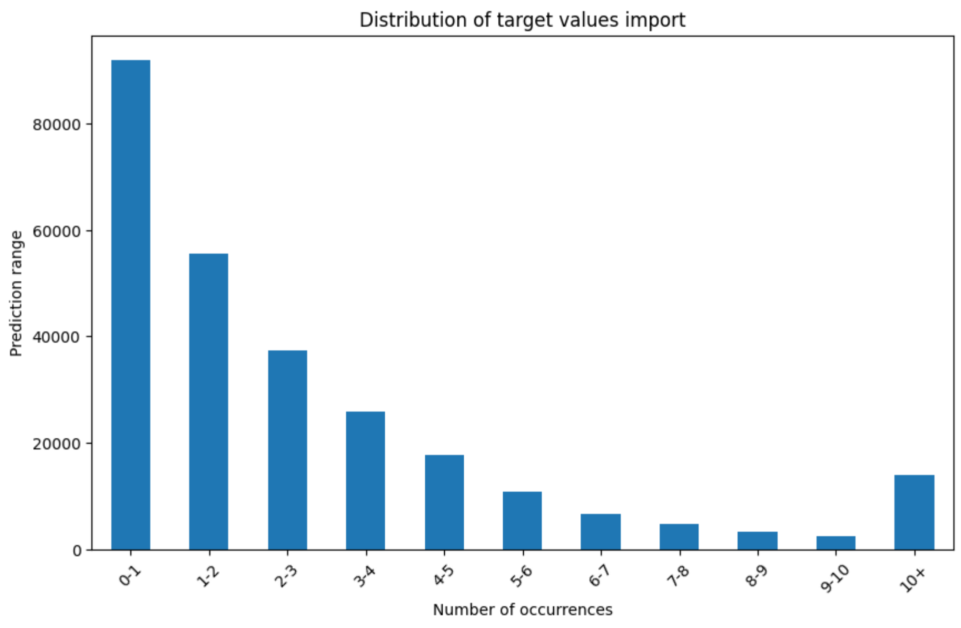


Figure 3.13: Delta between arrival and departure split into categories for Import

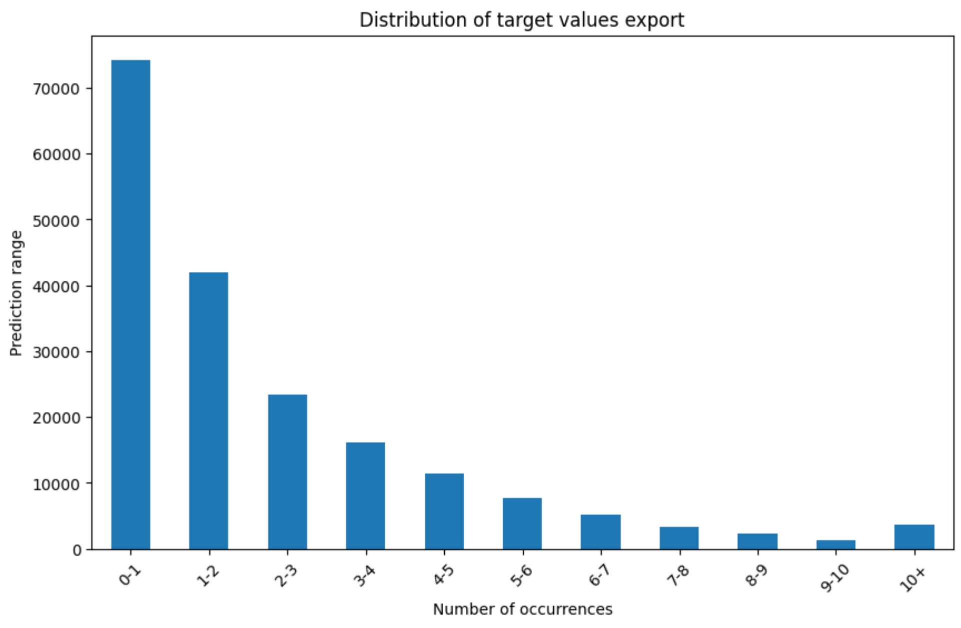


Figure 3.14: Delta between arrival and departure split into categories for Export

3.3 Data Preprocessing and Quality Assurance

There are sometimes missing data within our dataset, so we need to deal with this issue before we do any training. We want to learn how to predict departure time. This is computed as a delta between the time of arrival and the time of departure. If any of those two values are missing, we exclude this row from our dataset, as we need the delta to compare the prediction and the expected value. We also want to remove data with a negative delta, which can happen because of a human error during the dataset's creation. A different way of handling missing data is employed, where there are some missing values from the X part of training. We could exclude those data from the dataset, but for 2 reasons, we will keep and modify them. The first reason is that our system will be deployed in a changing environment and likely encounter data it wasn't trained on, such as new companies. For these new companies, some prediction is better than no prediction. So, converting previously unseen features into default categories will make our predictor more robust. Also, some parameters are not as important as others, so removing a row because of it would result in otherwise valid data being removed.

It is also important to check if our data have some outliers and how to deal with them. There are several ways how to detect and deal with outliers. By definition [12], an outlier is a statistical observation that is markedly different in value from the others of the sample. Dealing with outliers can be hard as there can be a thin line between invalid data created due to some error in the process and valid data with lower or higher values than others. An example of this can be seen in figure 3.15. This point looks like an outlier, but it is possible that something changed, and new data will have values similar to those of this outlier point. In the case of the containers, if a container from company A usually leaves the terminal in 7 days and we encounter another container from company A that took 100 days, it can be because of a typo error when filling the data, or the container may have been damaged so a repair was necessary before leaving the terminal. One way to identify outliers is to plot the values into some graphs and manually search for points that are out of the pattern. Another way to identify them is to compute some statistical values such as mean and standard deviation and use these values to calculate how likely it is that this point is from a given distribution. So, if the difference between some value and mean is greater than three times the standard deviation, then it is highly unlikely that it is valid. [13]

Now that we have identified some outliers, what could we do with them? The first idea is to remove these values. Of course, it can be done, and it's a valid approach for wrong data that would worsen the predictor's performance. Another way to handle these values is to give them a lower weight than other data. That way, these data are still used in learning but have a lower influence on the results. We can compute the weight of each point as a function of how unlikely a given point is from our distribution.

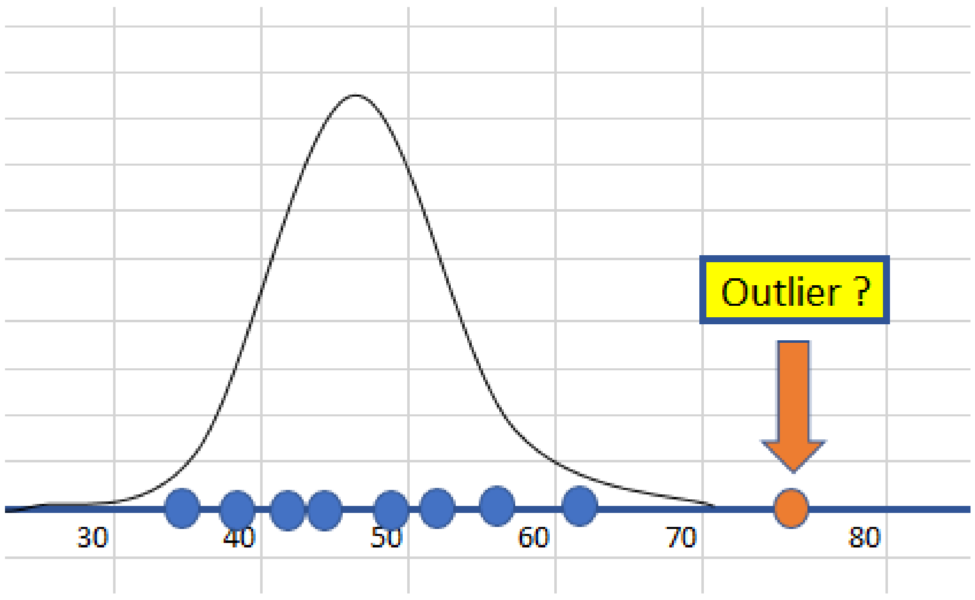


Figure 3.15: Example of an outlier in the data. [14]

3.4 Data Encoding

After all these steps, we need to address the biggest challenge from our dataset. The majority of our data are categorical values. This can be a problem for some methods that only work with numbers as an input. We will talk about these methods in the next chapter. In this section we will look at how we can work with categorical values and what are some benefits and limitation of each approach.[15]

3.4.1 Label Encoding

Label encoding is the first way to convert categorical values into numerical values. Each unique value is assigned a number, usually from 1 to N. It is a simple method and requires only 1*N of space, but it is not very usable for learning. By assigning some value to each category, we are saying that some categories have higher values than others, giving them a higher activation.

3.4.2 One-Hot Encoding

A second way to convert values is One-Hot encoding. Each unique value will create a new column. We then fill the values with 1 in the column associated with some categorical values and 0 everywhere else. That means each row has a sum of 1, and each column has a sum higher or equal to 1. By doing this, we assign the same weight to each of the values, so it will be up to a method to learn how much being from some category influences the final results. On the other hand, its space requirements can grow almost quadratically with the number of samples. Given a column with N rows and M unique category values, the resulting conversion will create M*N values. So, it quickly becomes unsustainable for large datasets with many unique values, which is our case.

3.4.3 Target encoding

Target encoding is a method that assigns a number to each unique categorical value. This can be any artificial number. This way, we directly affect what values we associate with each category values. By assigning a mean of the delta times of a given categorical value, we insert some relationship between a category and its numerical value. Thanks to representing each category by a number, our space requirements are again 1*N, which is great. The downside of this approach is that if the association we are creating is not valid, it will have a negative impact on learning. [15]

3.4.4 Used encoding

I employed a hybrid encoding strategy for optimum results. One-hot encoding was used to manage space requirements for categorical variables with limited unique values. I applied target encoding for other categorical variables using the mean departure time associated with each category. This approach injects meaningful domain knowledge into the encoding while avoiding the arbitrary ordering implied by label encoding

The last change we should make before training any method on this dataset is to standardize all numerical values. All of our data are numbers now, but some can range from 0 to 1 and some from 1 to 10,000. This would again signal to learning that some values have a higher meaning than others. By applying standardization, all values will have the same range of values from 0 to 1 while they keep the same proportional distance between each other from their previous values.

3.5 Conclusion

This chapter has provided an overview of the data preparation and analysis that are crucial for successful learning. By understanding the characteristics and distributions of our datasets, we can modify our data to improve the learning of the predictor. We described how outliers could affect our learning and introduced three ways to encode our data so any method can be used on our dataset. In the next chapter, we will look at different methods that can be used for prediction.

Chapter 4

Methods for Container Departure Time Prediction

4.1 Introduction

With our dataset ready, it is important to analyze the methods we can use to predict the departure time. This chapter introduces a range of predictive model candidates, examining their strengths and weaknesses. We will also consider other important properties, such as dealing with previously unseen data and space and time complexity. We aim to identify the most promising model for the most accurate departure times. We are trying to predict the delta between arrival and departure, so we need regression models. We will look at decision trees, random forests, gradient-boosting, and a neural network.

4.2 Baseline

Before diving into any method, defining how we will measure each method's performance is important. We will use two metrics to measure the performance. The first will be L1Loss. L1Loss is defined as an average of the absolute value of the difference between the predicted value and the correct one. If the loss of our predictor is higher than the mean value of all the deltas, it would mean our training has failed and predicting a simple mean of values would be better. For a better understanding of how accurate the results are, this error is accompanied by the percentage of predictions that have L1 loss less than one. In other words, the predicted value is +-1 day for the correct one.

To understand why we are adding a second metric, it is important to understand the desired usage of the predictor. The problem we are facing during planning on the terminal is that some containers don't have the departure time set. So, we don't know if we should place them lower and stack other containers on top of them or higher because they will be leaving the terminal early. The maximum height of the loaded container is 3 containers. So, for that reason, we will put our prediction into 3 categories. Short term, medium term and long-term. Short-term will be containers stored for less than 2 days, medium-term will be containers stored between 3 and 7 days, and long-term will be stored for 7 days or more. This metric is also very important as the company owning the container will always decide the true value of departure. For us, it is important to store them in a way that will require a minimal amount of manipulations possible during handling.

4.3 Methods

4.3.1 Decision tree

Decision tree is very simple to implement and understand. It creates a series of hierarchical questions(decisions). The tree starts at a root node and branches off into subsequent nodes based on conditions applied to the features in the data. Each split aims to divide the data into increasingly homogeneous groups with respect to the target variable (departure time in this case). Terminal nodes, called leaves, represent predictions. To follow the prediction for a new container, we traverse the tree based on its features until reaching a leaf node, which yields the associated prediction.

Its main advantage comes from its simplicity. By following a series of questions, it is easy to debug and easier to interpret what it is doing and why. They also naturally work with categorical values. It's also computationally and spatially non-demanding. They have, however, many disadvantages. The first problem is how to obtain an optimal decision tree. It can be done either manually or using some algorithm that automatically tries to separate the dataset, but the result is not guaranteed to be globally optimal. Another disadvantage is that by separating data into smaller and smaller groups, we are decreasing the size of our training data for each group, so instead of learning how one value can affect the prediction, we are assuming only the whole combination of values affects the result. [16]

To address some of these problems, methods like a random forest or gradient-boosting tree were created.

4.3.2 Random forest

Random forest offers a powerful ensemble technique to build upon the strengths of decision trees while mitigating their weaknesses. The idea is to create multiple decision trees, each trained on a different random subset of the data and features. When making predictions, the results from these trees are averaged, leading to a more robust and stable prediction compared to a single decision tree. This approach helps to combat overfitting and creates more generalized results. By avoiding overfitting, we can achieve better results than by using a decision tree alone. Another useful property of random forest is that it can handle more data with many features. Because it always uses just a subset of the original data, we can train more smaller trees compared to one big one with all the features. The last small benefit is that thanks to randomly excluding data from learning, we can use the rest to evaluate that tree, thus no need to divide data into training and testing. [17]

A disadvantage of this approach is that by combining multiple decision trees, we lose the easy interpretability of a single decision tree. We can still look at each decision tree individually, but we must look at all of them to understand why the final value was predicted. It is also computationally more demanding than single decision trees, as we need to train multiple trees, but they can be computed in parallel.

Overall, random forest substantially improves decision trees and is more robust for usage in the real world.

4.3.3 Gradient boosting method

The gradient boosting method is the last method that uses decision trees to output a result. It works similarly to random forest, but instead of creating random trees, it iteratively tries to improve prediction by correcting the errors of the previous trees. This approach leads to continuous improvement of prediction until convergence to local optimum or after a predefined number of iterations.

Gradient boosting usually starts with the mean of the value we want to predict. Then, we create a shallow tree and run each sample of data through this tree. When we have our data distributed into separate categories, we take their arithmetic average and calculate residuals(errors). Those are the differences between our guessed number and the real value we want to achieve. We improve our model with a fraction of those values determined by learning rate. In the next iteration, the predicted value will not be just the mean of the values but the mean of the values plus learned values. This will lead to gradually improving predictions. This means that each tree is based upon its previous tree and corrects the mistakes made by the tree before him.

The advantage of this method is that, similarly to neural networks, it tries to decrease the loss function, thus improving the prediction. [18]

The disadvantage of this method is that if we let it run long enough, it can lead to overfitting training data. It's also less interpretable, similar to a random forest.

4.3.4 Neural network

Inspired by nature, a neural network consists of many neurons that are connected together. They learn by observing data that consist of available features(X) and hidden states (Y) we want to predict. There are many types of neural networks, and each has a different purpose. Multilayer Perceptron(MLP) is the general type of neural network used for various tasks such as classification, regression etc. Convolutional Neural networks(CNN) are best used to classify and detect objects in images. Recurrent Neural Networks(RNN) are usually used where some memory is needed to predict the best possible outcome. The most typical uses of RNN are speech recognition, text processing, etc. For our purpose, MLP is the best.

There are many benefits of Neural networks. The first benefit is that they can learn non-linear dependencies in the data. Thanks to gradient learning, they can iteratively minimize errors and achieve high accuracy in prediction.

Even though neural networks are nowadays extremely popular, there are a few disadvantages. The first one is the necessity of a large dataset to learn accurately. The second one is that they are truly black-box solutions, and the results are very hard to interpret. Without extensive examination, we don't know the reasoning behind them. They are also computationally very demanding, and a graphic card or dedicated chip is often required for effective learning.

4.4 Methods comparison for simulated dataset

Before comparing methods on the real dataset, we want to see how well each method can work on a simulated dataset with nice statistical properties that have a dependency between X and Y.

This simulated dataset has 100000 entries with 11 features + the Y value. Tested data were split into 80% for training and 20% for testing. The number of unique values in each category ranges from 10 to 1000. Those values try to reflect the number of unique values in the real dataset. Each unique value got a random value assigned, and the resulting Y value was calculated as the sum of these values multiplied by a multiplier to give some categories a bigger impact on the final value. An exact code can be found in appendix B.

All used trees are from the scikit-learn library. All methods were called with the same random state for consistent results. For all models, an absolute error was set as a loss instead of a squared error. DecisionTreeRegressor was called without any other parameter. Random forest and Gradient Boosted tree had a number of estimators set to 20. Neural network architecture has 5 layers. It takes the input and outputs 1024 values. After each Linear layer, a Relu layer is applied. These 1024 values are then condensed into 256 values, followed by another 256 values, which are then connected to the last hidden layer of 64 values before being connected to the last layer with only one value. This value then represents our result.

4.4.1 Simulated dataset

We will start to measure the performance on the simulated dataset. All results excluding the L1Loss are in %. As we can see from table 4.1, the best results across all the metrics have Random Forest and Neural Network. The random forest has a slightly better mean absolute error but has slightly worse accuracy within 1 day. Gradient Boosted tree has a terrible result compared to any other method. I tried to adjust the parameters, but none of it helped. I think that the initial tree that was created was not deep enough to be able to learn the dependency in the data.

When we look at 4.2, we can see that except for Gradient Boosting, all methods produced respectable results with good accuracy, with the neural network having a slightly better performance in the long-term class.

	Mean absolute error	Accuracy within 1 day	Accuracy within 1 day for days less than 7:
Decision Tree	0.2434	98.41	98.78
Random Forest	0.1893	99.46	99.62
Gradient Boosting	1.1350	54.73	59.44
Neural network	0.1926	99.64	99.64

Table 4.1: Accuracy of methods on simulated data without random error

	Short-term accuracy	Medium-term accuracy	Long-term accuracy
Decision Tree	97.37	91.48	85.53
Random Forest	97.91	93.38	88.58
Gradient Boosted tree	93.15	77.74	0.00
Neural network	97.34	93.26	92.81

Table 4.2: Class distribution of methods on the simulated dataset without random error

We will now try to see what happens when we add some randomness into the data, that the predictor cannot learn. The Y values we want to predict will now be computed the same as before, but then we will add a random value from 0 to 1.

Looking at table 4.3, we can again see that both the Neural network and Random forest have the best results across all metrics, with the neural network slightly ahead. Decision tree performance dropped significantly to the point where its results were beaten by Gradient Boosting.

Table 4.4 confirms results from the previous table. The random forest and neural networks are close to each other, and the neural network again has slightly better performance. The most surprising result comes from the Gradient Boosting, with the highest accuracy for short-term class and 0 % accuracy for long-term.

4.4.2 Real data

Comparing methods on the simulated dataset gave us a hint on what to expect on the real dataset. Neural network had slightly better results compared to Random forest, but nothing major and Random forest had significantly less computational requirements. Other methods, especially Gradient Boosting, were falling behind. We will now test all of these methods on the real dataset on both import and export. As our input data are categorical now, we will use One-Hot encoding for the 8 smaller categories and Target Encoding for the 3 bigger categories. Normalization was then applied to those 3 features.

Looking at results from table 4.5, the biggest change compared to the simulated dataset was in the Neural network performance. Its performance in L1Loss dropped significantly and was beaten even by a simple Decision tree. It is possible that given the scale of the data, the neural network didn't have enough parameters to fully capture the complexity of the data, but we are at the limit of our resources, so there is no space to increase the size of the parameters. Other methods performed similarly to the simulated dataset, with Random forest beating other methods and Gradient Boosting having the worst result.

	Mean absolute value	Accuracy within 1 day	Accuracy within 1 day for days less than 7:
Decision Tree	1.1681	50.57	50.79
Random Forest	0.8770	63.78	64.23
Gradient Boosting	1.135	54.73	59.44
Neural network	0.8261	66.79	67.12

Table 4.3: Method comparison on the simulated dataset with added error

	Short-term accuracy	Medium-term accuracy	Long-term accuracy
Decision Tree	77.26	68.71	48.84
Random Forest	86.95	71.72	48.26
Gradient Boosted tree	93.15	77.74	0.00
Neural network	87.86	73.06	54.36

Table 4.4: Class distribution of methods on the simulated dataset with added error

	Mean absolute error	Accuracy within 1 day	Accuracy within 1 day for days less than 7:
Decision Tree	1.4944	63.26	71.71
Random Forest	1.3871	64.04	64.23
Gradient Boosting	2.083	51.47	64.36
Neural network	1.6532	62.35	74.11

Table 4.5: Method comparison on the real dataset from import

	Short-term accuracy	Medium-term accuracy	Long-term accuracy
Decision Tree	80.71	69.83	69.31
Random Forest	78.88	75.91	66.75
Gradient Boosting	74.52	61.70	27.08
Neural network	81.92	66.75	54.17

Table 4.6: Class distribution of methods on the real dataset from import

Class distribution of real data in table 4.6 confirms the results from the previous table, with random forest having the best overall accuracy. What is interesting is that the decision tree has slightly better short-term and long-term accuracy than the random forest. We can also see that the neural network managed to beat other methods in the short-term category and lost in the long-term category, which could explain the higher L1Loss.

In plot 4.1, we can see the difference between predictions from the random forest and real values for days less than 10 days and for all values in figure 4.2. Looking at these figures, we can see that the predicted values are correlated to real values, but there is quite a high variance, which is something that, if possible, should future predictors decrease.

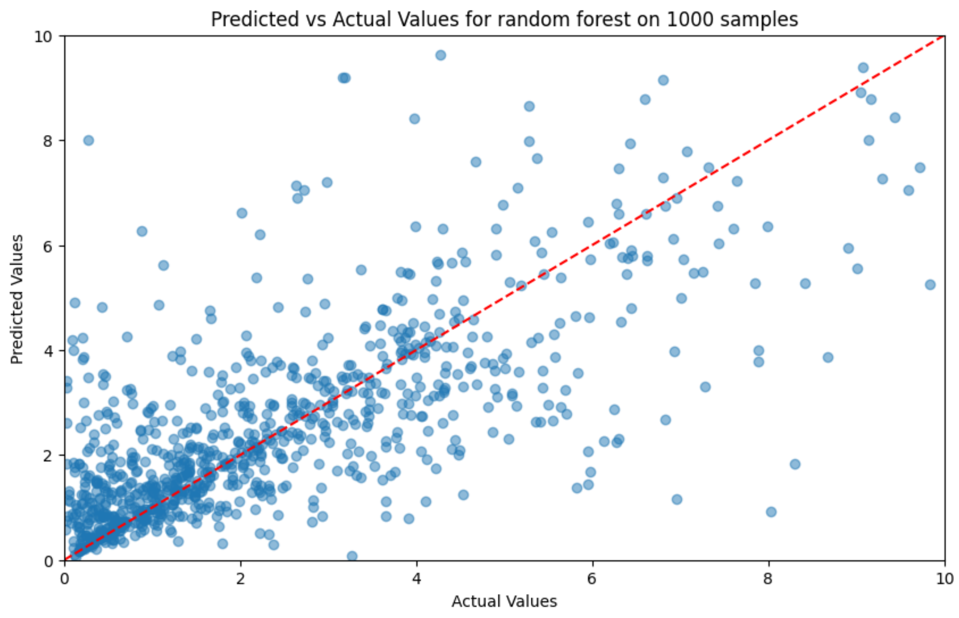


Figure 4.1: Plot of predicted values and real values

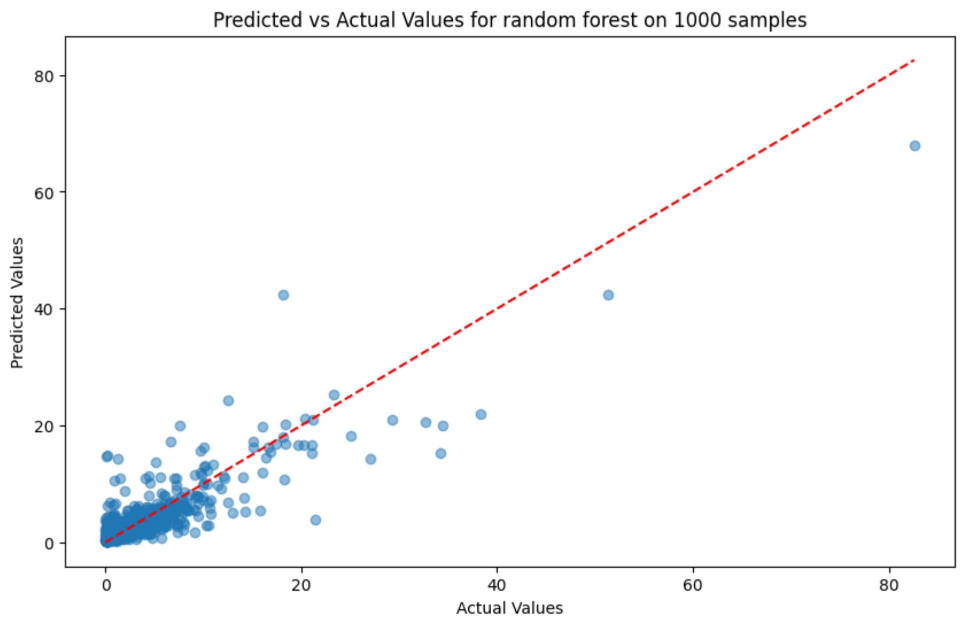


Figure 4.2: Plot of predicted values and real values

4.5 Chapter overview and conclusion

In this chapter, we have introduced 4 methods that can be used for regression. We stated how we would measure these methods' performances and explained why we chose those methods. We introduced all of these methods and explained some advantages and disadvantages that come with them. We then tested them on the simulated dataset that was created with similar properties to the real dataset and had a Y value dependent on the X value. From those tests, it was clear that the random forest and neural networks performed the best, and we moved to the real dataset. By testing on the real dataset, we discovered that the neural network fell significantly behind the random forest and even behind a simple decision tree. The most probable explanation is that the network didn't have enough parameters to capture the complexity of the data, thus decreasing the accuracy of the results. For that reason, Random forest was chosen as a final method for predicting departures.

Chapter 5
Implementation

In this chapter, we will look at what technologies and libraries were used during the development of this predictor, why these libraries were chosen and what problems were encountered during development.

5.1 Business requirements

To make this predictor usable in actual deployment, it was necessary to identify the company's requirements. After discussing this problem with a person from Metrans, several requirements needed to be satisfied. The first requirement was to have only one predictor for all terminals simultaneously on import and export. This was important from an accuracy perspective because import and export have vastly different properties, and by combining them into one dataset, we could lose accuracy. To avoid this, two models are trained, one on import and one on export and an internal switch is used to predict different values.

The second requirement was to make the predictor startup as fast as possible. This is a common property for Microservices to enable scalability and availability. Also, it shouldn't consume a lot of resources on the server to keep the cost down. That means our predictor needs to be pre-trained, and on the server, we only compute the prediction.

5.2 Security

Security is another important requirement that deserves its own section, an often overlooked part of development. Our data are highly sensitive, and it was necessary to prevent leaks during learning or predicting. Security is complicated to achieve because there are a lot of possibilities that can cause data to be leaked. In this section, we will go over some practices that can help to secure our data.

5.2.1 Anonymization

Following a simple rule, what you don't have can't leak, could help as do the security very easily. Instead of directly handling original sensitive data, we could avoid this issue by anonymizing the dataset. By replacing the values with random data, even if an attacker was successful in acquiring the dataset, he would acquire a table of random values that wouldn't make sense on its own. For our predictor, there is no difference between a random string or true value from some column. Metrans would anonymize the data and keep the mapping between true values and random values. Our predictor would learn from anonymized data, and then for each prediction, Metrans would use the mapping to fill the API with random values according to the mapping, and the predictor would return the prediction. Metrans would then return these data to their original values. This means that the only place where the attacker could get everything is on the company side. This benefits both sides, as the company doesn't lose control of its data, and we don't have to secure our predictor so much. Unfortunately, it was considered too complicated, and because there are already other means of security, it was not implemented in the final predictor although the prototype of this was created and could be added in case it would be needed.

5.2.2 Authorization and Authentication

A requirement from the company was to add authorization and authentication. Authentication is used to ensure that the user is someone who has access to the service, in this case, the prediction. This is usually done by some login, where the user fills in credentials that are sent to the backend. There, the credentials are checked or sent to the authentication service in case of 3rd party authentication. Either way, a token is then created and returned to the user. This token is then sent with each request from the user and serves as an identifier. Authorization is then used to determine what parts of the service you can access. For example, having a user who could shut down the whole service is undesirable. That should be something only the admin could do. The same can go the other way, as the user wouldn't like the admin to see his personal data, change his subscription, etc. Authorization is used to grant access to specific parts of the application. As our predictor is quite small in scope, the temporary solution of predefined credentials is stored in a dictionary, and later, before deployment to production, a Keycloak will be added to provide a token.

5.2.3 HTTPS

Because we didn't use anonymization of data and the predictor will be used on over 20 terminals across the continent using RestAPI, it is necessary to secure the communication going over the internet to avoid leaking sensitive data. To do that, we will encrypt the data. Fortunately, we don't have to do that ourselves, as HTTPS was designed to do that for us. It is an extension over the HTTP protocol that adds a layer of encryption. Today, it is usually TLS, but an SSL with some flows can still be found for older services. [20]

5.3 Architecture

Architecture sets guidelines on how to connect parts of the application together, how they should be linked, what parts of the application have what responsibilities, etc.

5.3.1 Microservice architecture

As the name suggests, microservice architecture consists of many very small components. Each microservice has one task. Thanks to that, these components can be very small and easily maintainable. Microservices components are loosely coupled together, which brings two advantages. You can roll out updates more frequently because each component has a predefined API, so as long as you keep it the same, other components don´t need to sync with your development. A bigger advantage is the ability to scale up and down. All microservices are stateless, so each component can handle any request. So, your 1st request can be handled by one microservice, and when you send a follow-up request, it can be processed by another microservice. That means that one component with the most requests can have multiple nodes to distribute the load to achieve maximum throughput with minimal resources. The biggest disadvantage of this system is that the infrastructure behind it is more complicated, and more importantly, fewer people can develop such a system. However, overall, the popularity of microservice architecture grows every year, and with it, a number of people who know how to develop such systems.

5.3.2 CodeNOW

CodeNOW is a platform that enables a platform as a service. It helps you to avoid setting environments and researching all technologies to create an application. Too often, to log some messages, you need to search what technologies are available, their advantages and disadvantages, their price, and so on. By using CodeNOW, the developer only needs to select the technological stack he wants to use, and the platform will configure everything for him and ensure that these components work together. Thanks to that, developers can spend more time focusing on core business instead of taking care of supporting infrastructure.

To deploy the predictor into production, a working Docker Image needs to be created and set environment properties, and the system will deploy it onto a server and will create an endpoint where other computers can consume REST-API.

5.4 Used libraries

5.4.1 Scikit-learn

Scikit-learn is a robust machine-learning library in Python that is used for classification, regression, preprocessing and many more. To ensure our methods won't overfit the data, we can split them into training data where our models learn and then evaluate their performance on the rest. Scikit-learn has many encoders for the conversion of categorical values into numerical ones, such as LabelEncoding, OneHotEncoding, and TargetEncoder. Scikit also has implementation of Decision tree, Random forest and Gradient boosted trees.

5.4.2 Pandas

Pandas is another essential Python library used for data preprocessing and analysis. The library offers various functions for detecting and handling missing data, filtering, and transforming datasets. This was crucial to ensure data consistency.

5.4.3 Pytorch

PyTorch is a library used for machine learning. It provides methods for effective building and training of neural networks. PyTorch simplifies how we can define neural networks and enables us to train tensors on CUDA cards and various gradient optimizers.

5.5 Implemented code

5.5.1 Data preprocessing

Before we train our methods, we need to ensure that historical data has a date of arrival and departure. After that, we can compute the difference between them, filter invalid data that cannot happen, and extract 2 parameters for learning. Day of the week and week of the year.

```
#drop empty rows
df = df[df['DATUM_PRIJEZDU_NA_TERMINAL'].notna()]
df = df[df['DATUM_ODJEZDU_Z_TERMINALU'].notna()]
#convert the difference of dates into float value in days
df['difference_in_days'] = (df['DATUM_ODJEZDU_Z_TERMINALU']
 ↪  -df['DATUM_PRIJEZDU_NA_TERMINAL']).dt.days +
 ↪   (df['DATUM_ODJEZDU_Z_TERMINALU'] -
df['DATUM_PRIJEZDU_NA_TERMINAL']).dt.seconds / (24 * 60 * 60)
#filter invalid data
df = df[df['difference_in_days'] > 0]
```

```
#extract features
df["day_of_week_arrival"] =
↪   df["DATUM_PRIJEZDU_NA_TERMINAL"].dt.dayofweek
df["week_of_year_arrival"] =
↪   df["DATUM_PRIJEZDU_NA_TERMINAL"].dt.isocalendar().week
```

Example of how categorical values can be converted into Onehot encoding.

```
label_encoder_typ = OneHotEncoder()
typ_df =
↪   pd.DataFrame(label_encoder_typ.fit_transform(df[["TYP"]]).toarray())
```

An example of how categorical values with too many unique values are converted into numerical ones. All values are converted into their respective mean and then normalized in order to ensure better learning.

```
 df["OBJEDNATEL_MEAN"] =
 ↪   df.groupby('OBJEDNATEL')['difference_in_days'].transform('mean')
merged['OBJEDNATEL_MEAN_scaled'] =
↪   scaler.fit_transform(merged[['OBJEDNATEL_MEAN']])[:, 0]
```

5.5.2 Method learning and definition

Defining a neural network architecture is easy, thanks to Pycharm automatic computation of gradients. All we need is to define layers and ensure their dimension will align.

```
class RegressionModel(nn.Module):
    def __init__(self, input_size):
        super(RegressionModel, self).__init__()
        self.fc1 = nn.Linear(input_size, 1024)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(1024, 256)
        self.fc3 = nn.Linear(256, 256)
        self.fc4 = nn.Linear(256, 64)
        self.fc5 = nn.Linear(64, 1)

    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        x = self.relu(x)
        x = self.fc3(x)
        x = self.relu(x)
        x = self.fc4(x)
        x = self.relu(x)
        x = self.fc5(x)
        return x
```

Here is an example of how the training of neural network is done. We can see that the model can be defined on one line, followed by a criterion, which is L1Loss, due to the big range of possible values. What is also good to point out is the usage of lr-scheduler, which will slowly decrease the rate of learning, enabling us to avoid oscillation and improve the results.

```
def
↪   train_model(X,Y,num_epochs=30,learning_rate=0.05,batch_size=64):
    X_tensor = torch.tensor(X, dtype=torch.float32)
    y_tensor = torch.tensor(Y, dtype=torch.float32).view(-1, 1)
    X_train, X_test, y_train, y_test =
    ↪   train_test_split(X_tensor, y_tensor, test_size=0.2,
    ↪   random_state=42)
    model = RegressionModel(X_train.shape[1])
    criterion = nn.L1Loss()
    optimizer = optim.Adam(model.parameters(),
    ↪   lr=learning_rate)
    scheduler = lr_scheduler.LinearLR(optimizer,
    ↪   start_factor=1.0, end_factor=0.03,
    ↪   total_iters=num_epochs)
```

In this example, we can see how we can easily use the method from scikit-learn to train a Random forest regressor.

```
rf_model = RandomForestRegressor(n_estimators=20,
↪   random_state=42)
# Train the model
rf_model.fit(X_train, Y_train)
# Make predictions on the test set
predictions = rf_model.predict(X_test)
```

Chapter 6

Conclusion

The aim of this thesis was to analyze processes on the container terminal, propose a method that would reduce the number of necessary operations, and then implement this method into their infrastructure.

At the start of this thesis, a brief introduction to terminal terms and how current operations are conducted is given. After that, we looked at available research on terminal operations and presented a few suggestions for implementation. Together with Metrans, it was decided that in order to make better planning possible, it is necessary to predict the departure of containers that don't yet have a date of departure set. The company gave us a big dataset of historical departures to predict this departure. As this dataset was private, a second dataset with similar features was created. The second dataset used a formula to derive the departure time in order to check if our methods can learn the real dependency and how close they can get to ideal results. An overview of statistical properties was presented about the private database to get a better sense of the real data and how these properties can affect our results. We then discussed how we can mitigate these problems and how to handle previously unseen values.

Both datasets were then used on several methods to test out which method would yield the best results. We compared Decision trees, Random forests, Gradient Boosting and neural networks. After evaluating, visualizing and discussing the results, the best method for the predictor was chosen. From these methods, the random forest had the best results compared to other methods, including neural networks. Random forest was then deployed on the CODENOW infrastructure using REST API as an interface. To handle the sensitivity of this data, a basic security was added to the API to prevent data leaks. As a result, a functioning predictor was deployed to the production and is now used by the company to improve prediction.

6.1 Potential further improvements

6.1.1 Auto re-learning

The nature of this task is that processes and data can change over time, so a model that we learned today will be outdated and needs to be updated in the future. Currently, it is necessary to input a new dataset and retrain the whole method manually. To avoid this, it would be beneficial to create either an automation script that would retrain the model automatically after some period of time or online learning that would update the weights.

6.1.2 Improved Data Preprocessing

To increase the performance of the predictor, several advanced data preprocessing techniques can be implemented to ensure cleaner, more relevant data. They can reduce overfitting and improve the generalization of the model. The following techniques could be applied.

- **Data Augmentation:** We could create new entries for features that were rare in the dataset to improve performance on these less-seen data.
- **Feature Engineering:** Creating new features or modifying existing features can provide additional insights into the model. One example of this technique was to take time of arrival to get a day of the week and week of the year and use these new features to improve learning.

Bibliography

[1] European Commission, "Maritime freight and vessels statistics - Statistics Explained," [Online]. Available: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Maritime_ports_freight_and_passenger_statistics&oldid=218671#:~:text=-%20Goods%20(mar_go)-,In%202022,%2096.0%20million%20TEUs%20of%20containers%20were%20handled%20in,by%204.0%20%20compared%20with%202021. [Accessed: 8-April-2024].

[2] The Geography of Transport Systems, "First Containership, Ideal-X, 1956," 1-Nov-2017. [Online]. Available: https://transportgeography.org/contents/chapter1/the-setting-of-global-transportation-systems/idealx-first-containeriship-1956/. [Accessed: 8-April-2024].

[3] SCF Australia, "World's Largest Shipping Container Ships in 2023," 28-Mar-2024. [Online]. Available: https://scf.com.au/news-articles/largest-shipping-container-ships/. [Accessed: 8-April-2024].

[4] S. S. Kumar, "How Much Does a Shipping Container Cost in 2024?," 16-Sep-2021. [Online]. Available: https://www.container-xchange.com/blog/shipping-container-price/. [Accessed: 8-April-2024].

[5] Port Economics, Management and Policy, "Main Physical Characteristics of ISO Containers," 1-May-2021. [Online]. Available: https://porteconomicsmanagement.org/pemp/contents/part7/ports-policies-and-politics/standardization-advances-containerisation/. [Accessed: 8-April-2024].

[6] R. Bergqvist, "Functional layout of a large-scale intermodal terminal," ResearchGate. [Online]. Available: https://www.researchgate.net/figure/Functional-layout-of-a-large-scale-intermodal-terminal-28_fig4_258865932.

[7] By Marc Cromme, DenmarkUploaded by Chlor at en.wikipedia - File handed over from Marc Cromme to User:Chlor (Hans Schou)Transferred

from en.wikipedia by SreeBot, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=17382578

[8] IndustrialForkliftTruck, “How Much Does a Container Handler Cost?” [Online]. Available: https://www.industrialforklifttruck.org/blog/container-handler-cost/. [Accessed: 6-May-2024].

[9] AICRANE, "How much does a container gantry crane cost?," 2023. [Online]. Available: https://steelmillcranes.com/how-much-does-a-container-gantry-crane-cost/. [Accessed: 6-May-2024].

[10] E. Pap, G. Bojanic, N. Ralevic, M. Georgijevic and V. Bojanic, "Crane scheduling method for train reloading at inland intermodal container terminal," 2012 IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics, Subotica, Serbia, 2012. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6339512&isnumber=6339478 [Accessed: 20-February-2024].

[11] METRANS, "From Hamburg [DE] – METRANS," [Online]. Available: https://metrans.eu/solutions/metrans-intermodal-solutions/import-from-sea-port-hub-terminals/from-hamburg-de/. [Accessed: 15-April-2024].

[12] Merriam-Webster, "Outlier definition & meaning," [Online]. Available: https://www.merriam-webster.com/dictionary/outlier. [Accessed: 17-Mar-2024].

[13] H. Bonthu, "Detecting and treating outliers: Treating the odd one out!," Analytics Vidhya, 28-Sep-2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/05/detecting-and-treating-outliers-treating-the-odd-one-out/. [Accessed: 17-Mar-2024].

[14] “Outlier test statistics in analytical data”, ConsultGLP. [Online]. Available: https://consultglp.com/2019/07/28/outlier-test-statistics-in-analytical-data/. [Accessed: 18-Mar-2024].

[15] S. Saxena, "What are categorical data encoding methods: Binary encoding," Analytics Vidhya, 24-Sep-2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/08/types-of-categorical-data-encoding/.[Accessed: 18-Mar-2024].

[16] A. Prasad, "Regression trees: Decision tree for regression: Machine Learning," Medium, 2024. [Online]. Available: https://medium.com/analytics-vidhya/regression-trees-decision-tree-for-regression-machine-learning-e4d7525d8047. [Accessed: 18-Mar-2024].

[17] S. E. R., "Understand random forest algorithms with examples (updated 2024)," Analytics Vidhya, 3-Jan-2024. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/#Advantages_and_Disadvantages_of_Random_Forest_Algorithm. [Accessed: 18-Mar-2024].

[18] D. S. Wizards, "Understanding the gradient boosting algorithm," Medium, 13-Jul-2023. [Online]. Available: https://medium.com/@datasciencewizards/understanding-the-gradient-boosting-algorithm-9fe698a352ad. [Accessed: 19-Mar-2024].

[19] G. L. Team, "Types of neural networks and definition of neural network," Great Learning Blog: Free Resources what Matters to shape your Career!, 23-Nov-2022. [Online]. Available: https://www.mygreatlearning.com/blog/types-of-neural-networks/. [Accessed: 19-Mar-2024].

[20] Cloudflare, "What is HTTPS?," [Online]. Available: https://www.cloudflare.com/learning/ssl/what-is-https/. [Accessed: 5-May-2024].

Appendix A
Used software

The following software was used in the development of this thesis:

- GitHub Copilot[1] for improved autocompletion of code
- ChatGPT (OpenAI)[2] for text style feedback and rephrasing suggestions
- Grammarly[3] for grammar and spelling checking
- Overleaf[4] to write this thesis using CTU template
- Pycharm [5] Ide for development of necessary codes

[1]https://github.com/features/copilot
[2]https://chat.openai.com/
[3]https://www.grammarly.com/
[4]https://www.overleaf.com
[5]https://www.jetbrains.com/pycharm/

Appendix B

Appendix A

A folder contains the simulated dataset and tested method.