Algorithmic Solutions for Improved Carrier-Shipper Matching in a Competitive Transport Marketplace

Master thesis @ UTURN

Tim Huisman

Algorithmic Solutions for Improved Carrier-Shipper Matching in a Competitive Transport Marketplace

Master thesis @ UTURN

by

Tim Huisman

Student number:	4591305	
Project duration:	May 2023 - July 2024	
Faculty:	Faculty Electrical Engineering, Mathematics and Computer Science, Delft	
Thesis committee:	Dr. N. Yorke-Smith,	TU Delft, supervisor
	Dr. A. Giudici,	UTURN, supervisor
	Dr. J. Yang,	TU Delft

TU Delft

Preface

As I conclude this thesis, I am deeply grateful for the support and encouragement I have received throughout this journey. My heartfelt appreciations go to my supervisors, Neil Yorke-Smith and Alberto Giudici, for their invaluable guidance and support. Their expertise and insights were crucial in shaping this work. I am also thankful to my colleagues from UTURN and the Algorithmics group for their collaboration and inspiring ideas. Working together with you all has been a truly enriching experience. To my family and friends, your encouragement and understanding have been my greatest support. I am profoundly grateful for your presence throughout this process. Additionally, I would like to thank Jie Yang for kindly agreeing to be part of my thesis committee. Thank you all for your contributions and support in making this thesis possible.

Tim Huisman
Delft, July 2024

Summary

In this thesis, we aimed to maximize the platform matching rate on the UTURN marketplace by taking conscious and optimal actions throughout the lifetime of shipments. We developed a recommendation system focused on matching shipments with the right carriers by creating detailed carrier profiles based on historical platform data. This system showed promising results, increasing the average matching rate by 2.8%, with improvements up to 3.4% in established regions and a peak increase of 6.2% in the Netherlands. However, our findings indicated that a one-size-fits-all approach is insufficient due to the varying capacities of carriers, highlighting the need for more targeted recommendations.

Additionally, we introduced the Shipment Pulse Monitor (SPM) to send recommendations only when necessary, balancing the maximization of matching probability while avoiding spam. The SPM demonstrated effectiveness in predicting cancellations and adapting to market changes through cost-sensitivity and dynamic parameter selection. Future work should focus on qualitative research to tailor recommendations to the most receptive carriers and combine hard cut-off points with predictive models for better accuracy. By addressing these areas, our solution can be refined to ensure long-term adaptability and improved performance in the dynamic freight transport market.

Contents

1 Introduction

The rapid expansion of e-commerce over the past two decades has greatly influenced the freight transport market. As consumers increasingly prefer to shop online, digital marketplaces have become vital for businesses, while the transportation industry plays a crucial role in connecting suppliers and buyers. In 2022, 1.7 billion tons of goods were transported to, from, and within the Netherlands [36]. Out of these 1.7 billion tons, 45.3% was transported across roads. The transporters of these goods, often referred to as carriers, operate in a highly competitive market [26]. With limited information about pricing across the transport market, transport operators are tasked with acquiring routes for their carriers that maximize revenue while minimizing costs. By inquiring about pricing at various companies or brokering with freight forwarders, they build a price profile with limited information and try to land jobs within a certain estimated profit margin.

One company realized that this process could be made more transparent by providing more information and autonomy to both sides of the market. UTURN offers an online marketplace where cargo owners (shippers) can search for transport companies (carriers) to transport containers to and from terminals at ports or other destinations by road. After a shipper posts a shipment on the platform, carriers can bid on the available shipment, and the shipper can accept or deny bids in an auction-like fashion.

With access to more options, carriers can optimize transport by filling empty trips in their schedules, while shippers benefit from competitive prices. Most shipments receive their first views and quotes within ten minutes of being posted. There are no specified bidding periods or decision points; whenever the shipper is satisfied with a bid, the carrier is assigned, and the shipment is no longer listed publicly. The UTURN marketplace is a high-throughput market; more than half of the shipments find their match within twenty minutes of posting. However, some shipments are more challenging to match.

Although UTURN's two-sided marketplace provides benefits for both sides of the market, it lacks the transport guarantee that is given by regular contracted transport or freight forwarders. Given the logistical challenges connected to storing, preparing and handing over cargo, not being able to find a carrier in time can be quite cumbersome for shippers. Therefore, UTURN is looking for algorithmic solutions to support this matching process.

1.1. Problem statement

In the field of transport and logistics, cost-efficiency is the key motivator for any smart solution [12]. Such solutions must address demand uncertainty by optimizing resource use to maximize revenue or achieve other targets [7, 21]. However, since the core concept of the UTURN platform centers around introducing transparency to the transport market, we cannot selectively use our resources or carriers to meet demand uncertainties. UTURN believes that all users should be able to benefit from the complete platform, and therefore, our solution should only be additive to the platform rather than restrictive.

To improve the matching rate, our approach is limited to bringing more attention to shipments. The solution for getting more attention to a certain item or product in a digital space is often fueled by a recommendation system [1]. For example, 80% of movies watched on Netflix came from recommendations [14]. When browsing the web, it is almost inevitable to run into one. Although the field of recommendation is well-defined and has fathered a great amount of research [2], the marketplace of UTURN poses a more nuanced problem.

In any recommendation system, one must define an approach to find suitable items for a given user. For items on Amazon, for example, there are clearly defined categories which individual users use to browse the website. Next to that, items have a certain amount of stock; while this lasts, the item can be bought. The items on the UTURN platform, on the other hand, are unique. Once a carrier has been assigned to a shipment, it disappears from the platform for all other carriers. Each shipment has its unique characteristics, ranging from container number and type to pick-up and drop-off locations. On top of that, shipments are perishable goods. Each shipment must be picked up and dropped off within certain time windows. If the shipper does not manage to find an interested carrier on the platform before the first deadline, the shipment is automatically cancelled. However, in practice, shippers usually cancel the shipment before this absolute deadline to avoid logistical overhead.

Additionally, most carriers are not individuals. Carriers are transport companies that have multiple drivers working for them. Each of them has their own, for us unknown, schedule, for which the transport company uses UTURN to fill up empty gaps to maximize revenue. This means that we have far from perfect information; we only know which shipments suited their presumably regular schedule in the past by their quoting behaviour. Unlike more clearly defined categorical shopping behaviour in the e-commerce case, the suitability of a shipment for a carrier is much more dependent on its location.

The aforementioned time pressure changes how recommendations should be handled. In e-commerce, users typically see recommended items whenever they visit the platform. However, the UTURN platform does not have a 'recommended shipments' section or personalized suggestions. This is less necessary for UTURN, as there are fewer shipments available at any given time. Carriers can quickly find all potentially interesting shipments once they are on the platform.

Currently, the only method for large-scale recommendations is through email. Although the transport industry primarily operates via email, sending too many emails can be counterproductive and reduce the effectiveness of our recommendations [11]. Additionally, many shipments would likely find a match before the recommendations are seen, making timing crucial.

Therefore, we must be conscious of the recommendations that we make. Rather than finding the right item for a given user, we must find the right user for a given item. More specifically, we must link a user on the carrier side of the platform to a user on the shipper side of the platform through an item (a shipment), such that the user on the shipper side of the platform may accept the linked user. In addition, we need to find a balance between sending out as many recommendations as possible to maximize our matching rate and avoiding overloading the carriers, while keeping in mind the perishability of the shipments. Finally, we must find a way to represent the interests of carriers, which may consist of multiple drivers, by utilizing a variety of shipment attributes, including non-categorical or text-based attributes.

1.2. Research Questions

The goal of this research is to create a recommendation for a two-sided transport marketplace, an application for which there is, to our knowledge, no known documented approach. In this research, we aimed to answer one main question: *How can we take action in a conscious and optimal manner throughout the lifetime of shipments to maximize the overall platform matching rate?*

Taking into account the UTURN platform dynamics, we constructed the following three research questions to explore a suitable solution:

1. How can we maximize the probability of finding a match for the shipments on the platform by sending out recommendations?
    (a) Which factors are important for a carrier when searching for shipments on the UTURN platform?

   (b) How can we model the interests of a single carrier, most likely consisting of a variable number of actors?

2. How can we balance maximizing the matching probability and avoiding unnecessary recommendations or spam?
3. How do we ensure that this solution is long-lived, adapting to changes in the transport market?

1.3. Outline

This report continues with chapter 2, which contains background information on UTURN itself, the freight transport market, recommendation systems and decision-making systems. Then, we introduce the system implemented to tackle the challenge of UTURN in chapter 3. After this, we show the evaluation of our developed system, consisting of both an offline and an online evaluation in chapter 4. Finally, we end the report with a discussion in chapter 5 and a conclusion together with future directions in chapter 6.

2 Background and related work

In chapter 1, we introduced the problem of UTURN and stated the research questions for this project. In this chapter, we provide background information on both UTURN and related work that motivates our approach described in chapter 3. We start with information on the dynamics of the UTURN marketplace in section 2.1. Then, we discuss various recommendation systems approaches and how they relate to our problem in section 2.2. Finally, we discuss background and related work to our second part of the solution regarding decision making in section 2.3.

2.1. UTURN and the freight transport market

The UTURN platform operates as a two-sided market, connecting two parties: shippers and carriers. The platform operates in an auction-like fashion. On one side, shippers post shipments with detailed descriptions, specifying pick-up windows, drop-off windows, container types, and more. On the other side, carriers, or transport companies, can view and browse all posted shipments to find those that match their desired characteristics. Once they identify a suitable shipment, they can quote on the shipment, indicating how much they would like to earn for completing this shipment. The shipper is then informed of this quote and can choose to accept or deny it. If the quote is accepted, the carrier is assigned to the shipment and receives confidential information necessary to perform the delivery.

A shipment remains on the platform until the shipper assigns a carrier or until the pick-up window closes, making it no longer available for pick-up. Unlike web-shops, where distribution is typically handled by third parties guaranteeing delivery services, the transport market requires significant logistics and manpower to move containers from point A to point B. This implies that instead of estimated delivery times, pick-up and delivery are restricted to particular time windows. Each shipment has a pick-up window, which also serves as the definitive deadline for assigning a carrier on the platform. Due to the logistical challenge of ensuring that a 20 to 40-foot container is ready for pick-up, shippers usually do not wait until the last minute. Instead, they cancel the shipment, removing it from the platform.

This creates a crucial difference from other online marketplaces: an inherent time pressure is associated with each shipment. Therefore, to increase the probability of finding a match for a shipment, this must be achieved before the shipment leaves the platform.

Freight transport: how can we maximize profit?

Like many transport-related works, the ultimate goal of our research is to maximize profit by using our resources optimally. In the field of freight transport, such resource allocation-related works are called fleet management. In fleet management, the shipments are viewed as the task to be performed to gain profit, while the transporters are seen as the resource. The problem of fleet management is solved in [32, 33] using Approximate Dynamic Programming (ADP). Similar kinds of problems have been very well studied [32, 33, 13, 27]. These works solve the resource allocation problem of assigning transporters to freight requests that arrive on a regular basis. At any point in time, the challenge is to decide which transporter to assign to which shipment to maximize profit, while ensuring that each driver can return to their home location within a specified time window.

Hypothetically, if carriers used UTURN exclusively to fill their transport schedules, we could create perfect schedules for all carriers, given a sufficient supply of shipments. However, in practice, this is far from feasible. Unlike the traditional fleet management problem, carriers on the UTURN platform are not necessarily singular entities. Most carrier accounts are linked to transport companies with multiple employed transporters. These transport companies primarily use UTURN to fill empty spots in their schedules. They typically have multiple drivers on their payroll, who all require a daily schedule for transporting goods. The majority of these driver schedules are filled with regularly contracted transport, formed through cooperation with distributors or importers, or by brokering with freight forwarders. When planners find gaps in a transporter's schedule, they need to fill these gaps to avoid wasting the paid working hours of their transporters. This is where UTURN comes in—a fast-moving marketplace where shipments are posted and assigned within a few hours.

This difference between the traditional fleet management problem and the UTURN platform highlights that we have far from perfect information. Carrier accounts are not linked to singular entities, so we lack information about individual transporters, their schedules, home bases, and truck types. We only know what interests the transport company through the behaviour of the overarching carrier account on the UTURN platform. Despite this variability in our user base and platform usage, efforts to create an optimized schedule could be made. This, however, is not in line with the vision of UTURN. The core concept of the UTURN platform is centred around introducing transparency to the transport market. UTURN views the current method of brokering for contracts with imperfect information as unfair and believes that all users of their platform should have equal opportunity. Therefore, our solution should only be additive to the platform, rather than restrictive. This means we cannot optimize our fleet in the traditional sense; each carrier in our 'fleet' must be able to view, quote, and take on any shipment posted on the platform.

A solution potentially lies in the field of recommendation systems. Enhancing the visibility of certain items or products in a digital space is often achieved through recommendation systems [1]. For example, 80 percent of movies watched on Netflix came from recommendations [14]. However, unlike platforms such as Netflix, which present recommendations on the screen as users browse, the UTURN platform lacks a 'recommended shipments' box or personalized suggestions. This feature is less critical for the UTURN platform compared to other online marketplaces. While e-commerce recommendations help users sift through vast quantities of available items to find the right product, the UTURN platform typically has a smaller number of shipments available at any given time. Consequently, once a carrier is on the platform, they can find all potentially interesting shipments within a reasonable time.

Currently, the only available method available to make recommendations on a large scale is through sending emails. Even though, according to UTURN experts, the transport world mainly operates through their email inboxes, this changes the nature of the objective of the recommendations. While traditional e-commerce recommendation sections should be used whenever possible, emails should be used sparingly. People generally do not appreciate an overflow of emails, especially when managing day-to-day tasks through their inboxes. Additionally, this may risk a reduction in the effectiveness of our recommendations [11]. Moreover, if we were to recommend every posted shipment, many would already have found a match by the time the recommendation is seen, potentially diminishing the appeal of opening them over time. Therefore, we must balance maximizing the probability of finding a match for each shipment and avoiding excessive recommendations. Instead of viewing our carriers as a resource like in fleet management, we should consider recommendations as our resource.

2.1.1. What would our solution require?

Although, ultimately, our goal is to maximize the matching rate of the platform, our approach should be conscious of how we aim to achieve this. Considering that the vision of UTURN prohibits us from imposing any kind of restrictions on the platform, a potential solution lies in the field of recommendation systems. With recommendation systems, we can bring more attention to shipments through recommendations without being restrictive to the user experience. In the upcoming section 2.2, we will discuss related recommendation system work and how a recommendation system could be used to increase the probability of finding a match for certain shipments.

In addition to this recommendation system, we require a mechanism to decide which shipments require recommendations and which shipments do not. As reasoned in the previous section, we need to be conscious of the number of recommendations we send out. This involves striking a balance between waiting for the shipment to find a match over time and ensuring that we send recommendations before the shipment leaves the platform and revenue is lost. In the third and final section of this chapter, section 2.3, we will discuss what a good intervention moment could be and how this could be achieved with a predictive model.

2.2. Creating a shipment recommendation system

With our understanding of the dynamics of the UTURN marketplace, we can now discuss a potential recommendation system-based solution. First, we need to form a solid understanding of what a recommendation system is. We will begin by providing a general definition and comparing different types of recommendation systems in subsection 2.2.1. Next, we will discuss how to create recommendations specifically for the UTURN marketplace in subsection 2.2.2. Finally, we will explain how to evaluate the effectiveness of our recommendation system in subsection 2.2.4.

2.2.1. What are recommendation systems and why do we need them?

Recommendation systems essentially try to model the interests of each user. By providing suggestions or recommendations, these systems help users find what they are looking for more quickly and easily. To achieve this, it is crucial that a recommendation system can sift through all available items to extract only the most relevant ones for a given user. For giant online e-commerce stores such as Amazon, recommendation systems play a vital role in enhancing the online shopping experience.

There are many types of recommendation systems, each categorized by their method of determining the relevance of a given item. Although there are numerous types, there are three main categories well-known throughout the field: collaborative filtering, content-based filtering and hybrid filtering [1].

1. **Collaborative filtering** revolves around utilising similarity in user behaviour. The Amazon [34] recommendation system has been heavily focused on collaborative filtering since 1998. By comparing your purchase pattern with the purchase patterns of other people, the recommendation system can suggest items that you are likely to purchase based on the similarity of your purchasing pattern to other people. The same can be done for items specifically; for every item i, we find every item j that is purchased with unusually high frequency by people who also bought i. A drawback of this method of filtering is that each item requires a history of purchases or ratings for it to work (also known as the *cold-start problem*).
2. **Content-based filtering** revolves around suggesting purely on item content. Content-based filtering is often used in contexts where comparing user behaviour is not possible or to remedy the *cold-start problem*. This is typically done by extracting keywords out of descriptions of items to allow for comparison between items [1].
3. **Hybrid** filtering uses multiple types of filtering either simultaneously or complementary. For example, Spotify uses collaborative filtering to exploit previous listening behaviour, while exploring new music with content-based filtering [22].

In general, most recommendation systems make use of various forms of filtering as each of them has its own limitations [2]. In the upcoming subsection, we will discuss how a recommendation system can be used for the UTURN marketplace and show a comparison to other recommendation systems for two-sided platforms.

2.2.2. How can we make recommendations in UTURN's two-sided marketplace?

Although there is much to learn from the vast amount of literature on recommendation systems, the UTURN marketplace differs from previous work in several ways. First, the items offered on this marketplace are not permanent and do not have specified stock; once a carrier's quote is accepted, the shipment becomes unavailable to other carriers and ultimately disappears from the public platform upon completion.

This makes the use of collaborative filtering challenging. Whereas Amazon can compare purchasing patterns across long-lasting items and item ratings between various users, each shipment on the UTURN platform is unique and disappears after it is matched or canceled. Although a shipment can receive multiple quotes, allowing us to compare the quoting carriers, the low frequency of this occurrence does not provide sufficient data for collaborative filtering.

Additionally, platforms like Amazon are one-sided: once items are posted, only one party plays a role in generating revenue. There is also literature on recommendation systems in two-sided platforms, but the tackled problems differ from ours. In dating apps, for example, both parties need to express mutual interest. However, dating app recommendation systems rely heavily on the persistence of user profiles to enable collaborative filtering [29, 6]. On top of that, research on dating app recommendations focuses on how matching platforms should decide on assortments to show users, which is not aligned with UTURN's policy; all shipments must be available to all users.

Goswami et al. [15] pose the two-sided matching problem as a bipartite ranking problem instead. Although their problem definition is already more general, it differs in one crucial aspect. They assume that both sides of the market can express interest in each other. Although there is a mechanism in the UTURN platform to only show shipments to a selection of carriers, this is not widely used and should not be considered as the ability to express interest from both sides of the platform. On the UTURN platform, the only information available to the shipper upon receiving the quote is the price, name and rating of the quoting carrier. Given this limited availability of 'mutual interest', we opted not to focus on mutual interest other than the similarity in pricing in the rest of this research. Instead, we opted to mainly focus on the other aforementioned filtering method: content-based filtering.

2.2.3. How can we use content-based filtering for the UTURN platform?

Although collaborative filtering is not possible for our application, we do have many informative details on each shipment that can be compared to other shipments. Discussed in more detail in subsection 3.1.2, each shipment has its own pick-up and drop-off locations, requirements, specified price, and more. Content-based filtering analyzes such details of products, or in our case, shipments, that a user has previously evaluated. In particular, based on the content of its features, similarity functions can detect the most related items, following the assumption that items that are similar in content will be rated similarly [2]. This method of filtering is ideal for the UTURN platform: by basing recommendations purely on the content of each shipment, we can generate recommendations for any shipment.

The challenge lies in representing the content of each item, or in our case, each shipment, as a set of features. Whereas a book merchant may have a description of each book containing keywords, the author name, and title, shipments have attributes of various modalities. In such cases, a (structured) multidimensional feature representation can be used. The various fields need to be weighted appropriately to facilitate their use in the classification process [1].

Next, we need to learn carrier profiles to represent the interests of any given carrier based on the shipments they have interacted with in the past. By analyzing each carrier's previous platform usage, we can estimate how suitable a new, unseen shipment may be. To make such estimations, a classifier is needed. Numerous classifiers are used for content-based recommendations, each with its strengths, weaknesses, and requirements [1, 2]. These range from Bayes classifiers, which consider prior class probabilities, to rule-based classifiers, which use domain knowledge to form a set of "if-then" statements for decision-making. Additionally, modern deep learning approaches have been shown to outperform traditional methods [43]. However, such approaches tend to be based on platforms and/or datasets that have a long history of recommendations, which is not available for UTURN.

Before describing our chosen classifier, we must discuss a crucial difference in our system compared to other work. Traditional recommendation systems can be seen as classifiers: for any user, we need to find the set of relevant items. Once we find potentially relevant items, their relevance can be verified by whether the user ultimately engages with them. In our case, however, we are not striving to make recommendations for all carriers. Since our recommendations can only be sent by email, we must be conscious of the number of recommendations made. Moreover, since most shipments quickly find matches on the platform, we must only send out recommendations for shipments that require extra attention. Therefore, rather than trying to recommend shipments to each carrier, we aim to find a list of carriers ranked by relevance for a given shipment.

This means that, instead of learning carrier profiles to find the right items for each carrier, we must learn carrier profiles to find the carriers with the highest probability of picking up a problematic shipment.

Given that UTURN is only used for a small part of a carrier's schedule, our best bet at finding a carrier for a problematic shipment is to identify one that has shown interest in similar shipments in the past. To do this, we compare the similarity of the platform history of carriers to any problematic shipment to create a ranked list of suitable carriers. This shifts the focus from creating a classifier to creating a sophisticated similarity measure.

Although our problem is no longer a traditional classification task, we can still utilize some classifiers as they essentially calculate the similarity of a data point to those of a given class. Out of the popularly used methods, one stood out in particular: the k-nearest neighbour algorithm (k-NN). The k-NN algorithm is one of the most widely used classification algorithms since it is simple and easy to implement. It is a type of lazy learning algorithm that does not require offline training [17].

How can we use a k-NN algorithm to make content-based recommendations?

When using a k-NN classifier, the first step is to define a similarity function. This function computes the similarity between two data points based on their features (content). For numerical values, the similarity function that is most widely used in the literature is the Euclidean distance [1]. For comparing textual documents, cosine similarity is often used to compare the occurrence of certain words or phrases between documents [17]. For comparing shipments, however, we may require a more tailored distance function.

As mentioned before, the pick-up and drop-off locations will play a large role in whether or not a shipment is suitable for a carrier as these shipments need to fit their schedule. Therefore, instead of calculating the distance between two points using Euclidean distance, we must instead calculate the distance between coordinates on a globe. Additionally, we want to incorporate other shipment content to estimate the similarity between the two shipments. Consequently, we will experiment with creating a custom, tailored distance function to be used in the k-NN algorithm. This process will be described in section 3.1.

This similarity function is useful in making predictions for items in which the user preference is unknown [1]. To represent the interests of each carrier, a carrier-specific k-NN can be created, supplied with their platform history as a training data set used to estimate similarity to unseen data points. For any unseen data point, its k-nearest neighbours, or k-most similar data points in the training data set are determined using the similarity function. Then, the values of those k neighbours can be used to estimate the query data point's label, rating, or similarity. This process is visualised in Figure 2.1.

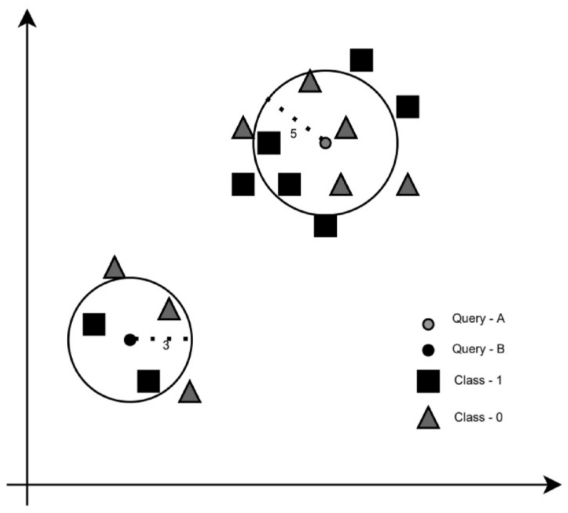


Figure 2.1: An example of classification using a k-NN as illustrated by Uddin et. al [39]

The k-NN classifier specifically suits our case because of the nature of our carriers. As mentioned in section 2.1, carriers may have multiple operational areas. Additionally, most carrier accounts have multiple transporters available, further reinforcing the possibility of multiple operational areas for a single carrier account. Therefore, when determining how similar a new shipment is to the history of a given carrier, this similarity should not be based on the carrier's complete history but rather on whether the shipment falls within one of the carrier's operational areas. By fitting a k-NN classifier to the history of each carrier, we aim to use the efficient SKLearn implementation [25] to calculate the similarity of any given shipment to various operational areas, or 'neighbourhoods', of the carrier.

In using a k-NN, the choice of the value k, which represents the number of shipment history points considered for one operational area, is crucial. The estimated similarity of any shipment to any carrier profile will be heavily influenced by this value. Therefore, multiple strategies for selecting k must be evaluated. Additionally, the aforementioned cold-start problem remains a concern. If a carrier has a small recent platform history, the k-NN will have insufficient data points to make accurate similarity calculations. However, since we are trying to find carriers most likely to take on problematic shipments, we assume that small or low-throughput carriers are much less likely to take such shipments, making the cold-start problem less relevant.

2.2.4. How can we evaluate recommendations on the UTURN platform?

To evaluate such a recommendation system, we can perform both offline and online evaluations. Offline evaluation involves analyzing the past usage of the platform and creating an experimental setup that reflects the platform's dynamics. For many recommendation systems, well-known metrics such as precision and recall are used [1]. These metrics represent the fraction of relevant recommendations among the total recommendations and the fraction of relevant recommendations out of all possible relevant items, respectively. However, due to the nature of our application, these metrics may not be suitable. Whereas traditional recommendation systems determine relevancy from the user's viewpoint, we will reason from the viewpoint of the item. Instead of finding items relevant to the user, we aim to find carriers most likely to take on the shipment. Since only one interested carrier is required for a match, our goal is to identify this singular carrier.

To reflect this goal, we can utilize rank-based metrics used in search engines [3]. Rank-based metrics are especially suitable for applications where the ranking of results is important. Although there are many different rank-based metrics, they often assume there to be multiple relevant results to a query. In our case, however, we more often than not have only a singular quoting carrier for a given shipment. Therefore, a metric that focuses on a singular relevant result would be more suitable. A popular metric for this is the reciprocal rank: the position at which the relevant item is ranked in the returned ranked list. The reciprocal rank can subsequently be averaged to result in the mean reciprocal rank (MRR), and is defined as follows:

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\mathrm{rank}_i} \tag{2.1}$$

Using the MRR, we can estimate how well our recommendation system performs in finding the right carrier for any given shipment. However, evaluation in an offline setting is prone to inaccurate assumptions and merely shows the theoretical performance of the recommendation system. Although such offline experiments aim to represent the dynamics of the real world, the only sure way to measure practical performance is to deploy the recommendation system in the real world. To measure the impact of such systems, a controlled experiment can be set up. In controlled experiments, the effect of either a change within a recommendation system or the introduction of a recommendation system can be measured by splitting the target group into two groups: the control group and the test group. The test group is treated with the system being evaluated, while the control group is left untreated. This division allows us to compare the practical results of both groups, showing the difference made by the newly introduced system.

In the upcoming methodology chapter, section 3.1 will describe our process of developing a recommendation system for the UTURN platform with k-NNs as described above. It will contain a description of our custom similarity measure, along with a discussion on how its effectiveness will be evaluated in section 4.1. Before we can continue with the methodology chapter, we will first discuss the background for the second part of our solution.

2.3. Deciding when to send recommendations

As discussed in section 2.1, UTURN's marketplace is a fast-moving, high-throughput marketplace in which shipments typically do not stay on the platform for long. This poses a significant challenge. The main goal of our product is simple: we aim to increase the matching rate as much as possible through the recommendation of shipments to carriers. However, given the high throughput of the UTURN marketplace, simply sending out recommendations for every shipment would cause us to overload our carrier inboxes, risking a reduction of the effectiveness of our recommendations [11]. Additionally, many recommended shipments will have found their match by the time the carrier views the recommendation. Therefore, we need to be selective about which shipments we recommend.

Since sending recommendations for every shipment is undesirable, a solution should focus on determining which shipments require additional attention. One potential solution could be to define a cut-off point: if a shipment remains on the platform for more than t minutes, it should receive extra attention. However, given the variability in shipment characteristics and deadlines, a better solution based on shipment-specific data is needed. In addition to historical data on matching rates for certain routes, we also have live data on the current number of views or quotes. By combining historical and live data, we may be able to predict whether a shipment will find a match in time or whether an intervention is necessary.

Since the matching process happens in an auction-like fashion, we must allow it to proceed naturally on the platform to a certain extent. Once confidence is lost in the probability of the shipment finding a match on time, we can utilize our resource: recommendations. This problem resembles the *optimal stopping problem*.

The optimal stopping problem is a decision-making problem in which a decision-maker seeks the best time to stop a stochastic process to maximize the expected payoff or minimize the expected cost [20]. In such a process, the decision maker can either wait for a better situation or take action. Optimal stopping can be found in many real-world problems, ranging from bus maintenance problems [30] to online auctions [19]. However, in these works, in which the optimal stopping problem is modelled as a Markov Decision Process (MDP), they either assume the transition dynamics or the immediate value gained by stopping. In our case, the value gained by stopping, or recommending, is unknown. When development was started on the second part of the solution, the recommendation system did not generate enough recommendations to know their potential contribution to the probability of finding a match.

The transition dynamics, on the other hand, could be estimated to a certain degree. To do this, a state space would need to be defined, containing all factors that describe the current state of the shipment in the market. However, many factors make up this current state. A shipment state should contain the current time spent on the platform, the time left until the deadline, the number of views and quotes, and the added risk of early cancellation. Considering that the state space should account for any possible value for each of these factors, the state space would quickly blow up in size. Additionally, this would vary for every route, especially since UTURN is most established in the Netherlands, Belgium, and Germany. Some works use neural networks or deep learning approaches to be able to work with such high dimensional state spaces [20, 4]. Neural networks and deep learning models, while powerful, require extensive amounts of data and computational resources to train effectively. They are known for their complexity and often function as “black boxes”, making it challenging to interpret how decisions are made. Given that there is no prior data or work on estimating the outcome of shipments for the UTURN platform, introducing such complexity may not be a smart approach. Instead, to explore the potential space of solutions, a simpler and more interpretable approach may be more suitable.

In contrast, simpler methods such as decision trees provide several advantages. Decision trees are easy to understand and visualize, making the decision-making process transparent and interpretable [16]. They can handle both numerical and categorical data and require less computational power compared to neural networks. Additionally, decision trees offer clear insights into which factors are most important in predicting outcomes, which can be valuable for refining the model and strategy.

In the following section, we will first define decision trees and explain how they can be used to determine the outcome of a shipment. Then, we will discuss how they can be applied to our specific case: determining when to intervene.

2.3.1. Deciding when to intervene with decision trees

Decision trees are widely used in several disciplines because of their ease of use, robustness and easily interpretable structure [16, 8]. Decision trees are a graphical representation of a decision-making process. An example of such a tree fit for our application can be found in Figure 2.2.

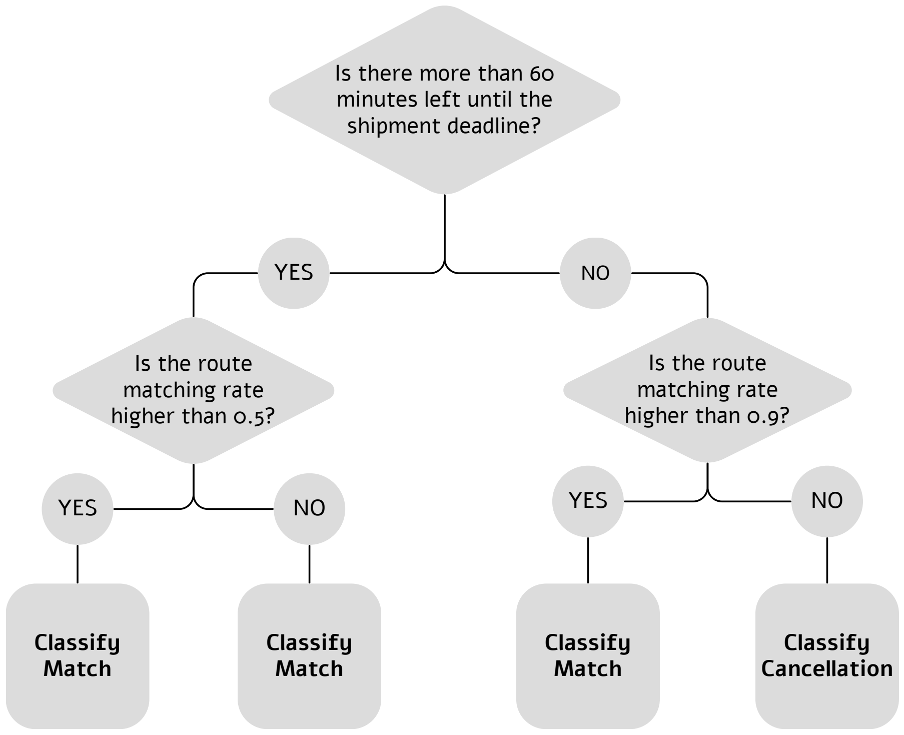

Is there more than 60 minutes left until the shipment deadline?
YES
NO
Is the route matching rate higher than 0.5?
Is the route matching rate higher than 0.9?
YES
NO
YES
NO
Classify Match
Classify Match
Classify Match
Classify Cancellation

Figure 2.2: An example of a Decision Tree for classifying the outcome of a shipment

Each node in the tree represents a decision point based on one of the features of each data point, represented in the diamond blocks. Each feature represents a characteristic of a data point, which we can use to incrementally lead to a prediction of the data point's label. After traversing each decision point in the tree, the leaf nodes at the bottom of the tree are reached, representing the label we ultimately assign to the data point. In our application, such a decision tree could be used to predict whether a shipment will result in a cancellation or a match.

How does a decision tree form decision points?

To build a tree-like model of decisions based on the features of the dataset, the decision tree splits the data at each node according to certain criteria to maximize the separation of the two classes: cancellations and matches. The SKLearn library [25], which will be used for this research, facilitates two splitting criteria: Gini Impurity and Entropy. In this research, the Gini Impurity was chosen as the splitting criteria because it outperformed Entropy in empirical results. Gini Impurity is defined as follows:

$$\text{Gini} = 1 - \sum_{i=1}^{C} p_i^2 \tag{2.2}$$

where p_i is the probability of an element being classified for a particular class. The Gini Impurity measures the frequency at which any element of the dataset would be misclassified when randomly labelled according to the distribution of labels in the dataset. A lower Gini Impurity indicates a better split at the node, helping to maximize the separation between cancellations and matches.

The process of creating decision points according to the Gini Impurity is done as follows.

1. **Calculating the Gini Impurity**: for each feature in the data set, the algorithm considers each possible split of the data (sub)set according to every possible threshold value. For example,

a split could be: any shipment with less than an hour left until the deadline is classified as a cancellation and as a match otherwise.

2. **Choosing the best split:** for any split considered in step 1, the Gini impurity of a split is calculated as defined in Equation 2.3. Subsequently, the split with the lowest Gini impurity is chosen, reflecting the split that separated both classes in the data (sub)set best.

$$\frac{N_{\text{parent}}}{N_{\text{total}}} * \left(\text{impurity}_{\text{ parent}} - \frac{N_{\text{right}}}{N_{\text{parent}}} * \text{impurity}_{\text{ right-child}} - \frac{N_{\text{left}}}{N_{\text{parent}}} * \text{impurity}_{\text{ left-child}} \right) \tag{2.3}$$

3. **Splitting the data:** the (sub)set of data contained in the node is then split according to the splitting criteria with the lowest Gini impurity. This results in two new child nodes, for which a new split is determined to split the data (sub)sets further.
4. **Stopping criteria:** this process repeats recursively for each child until a stopping criterion is met. Stopping criteria can include:
   - A maximum depth of the tree.
   - A minimum number of samples required to split a node
   - A minimum Gini impurity decrease required to split a node.
   - If a node becomes pure (all samples belong to the same class)

Once one of the stopping criteria is met, we end up with our final decision tree. The bottom of the decision tree is populated with the leaf nodes. These nodes can either be homogeneous in data points or contain a mix of both positive and negative data points. When the decision tree is subsequently tasked with classifying a new, unseen data point, its features are used to traverse the tree until the leaf nodes at the bottom are reached.

When opting for a probabilistic classifier, the fraction of the leaf node that is populated by the positive class is returned as an estimation of the probability that the data point belongs to the positive class. When opting for a binary classifier, which will be done for this research, the label of the majority class will be returned.

When evaluating binary classification tasks, the standard representation of classification results is derived from the confusion matrix, which consists of the following components:

- **True Positives (TP)**: The number of instances correctly predicted as positive.
- **True Negatives (TN)**: The number of instances correctly predicted as negative.
- **False Positives (FP)**: The number of instances incorrectly predicted as positive.
- **False Negatives (FN)**: The number of instances incorrectly predicted as negative.

Using the confusion matrix, we can derive several important metrics to evaluate the performance of our binary classifier:

- **Accuracy**: The proportion of correctly classified shipments (both positive and negative) out of the total instances.
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$
- **Precision**: The proportion of correctly predicted positive instances out of all predicted positive instances.
$$\text{Precision} = \frac{TP}{TP + FP}$$
- **Recall**: The proportion of correctly predicted positive instances out of all actual positive instances.
$$\text{Recall} = \frac{TP}{TP + FN}$$

Throughout the rest of this research, we will be referring to cancellations as negative instances (0) and matches as positive instances (1).

How can we use a decision tree to decide when to intervene?

With decision trees clearly defined, we can move on to how they can be used to predict the outcome of a shipment. Since we aim to find the right moment to intervene, we must make predictions throughout the lifetime of a shipment. This means that the decision tree must be able to predict the outcome of the shipment at any time during its lifetime. However, as we reasoned before, the predictor should allow the shipment to find its match naturally to some degree. Therefore, even for a shipment that ends up as a cancellation, we should allow the decision tree to misclassify the shipment as a match at the beginning of its lifetime.

This is a key difference from other decision tree applications. In medical research [8], a diagnosis is made after the deciding party has gathered all necessary data on conditions (e.g., features), and it is made only once, with a focus on being as accurate as possible. In our case, we aim to continuously monitor the platform and make predictions as time passes. This implies that, instead of having a singular data point per classification case, we have multiple.

This raises the question of how to sample our data and reflect this in our metrics. Since the continuation of this discussion ties in with our methodology, we will continue the discussion on using decision trees to determine an intervention moment in section 3.2 of the methodology chapter.

3 Methodology

In this chapter, we describe our method of developing our system and how it will be evaluated. Since our solution comprises two parts, this chapter, like the background and related work, is separated into the methodology for the recommendation system, or as we call it, the Matching Algorithm, and the methodology for the intervention strategy, or as we call it, the Shipment Pulse Monitor. The methodologies can be found in section 3.1 and section 3.2 respectively.

3.1. Finding the right carrier for a shipment: The Matching Algorithm

In the first part of the methodology chapter, we will describe the process of creating a recommendation system that will be deployed on the UTURN live servers to send shipment recommendations to carriers. As described in more detail in the previous background and related work chapter, the goal of the Matching Algorithm is to find the right carrier for a given shipment based on the platform usage history of the carriers. To incrementally build up to the full system, we start by discussing how we can represent the interests of a carrier in a carrier profile by analyzing the platform usage history of a carrier in section 3.1.1. After this, we continue with a description of the various factors that can play a role when representing the interests of a carrier in section 3.1.2. With these factors explained, we continue with an explanation of how we plan to use k-NNs to determine the suitability of a given shipment to a carrier profile in section 3.1.3. Then, we discuss our experimental setup in section 3.1.4. Finally, we close the Matching Algorithm methodology section with a description of the system that was developed for the deployment of the Matching Algorithm in section 3.1.5.

3.1.1. How do we represent the history of a carrier?

To find the right carrier for a given shipment, we must find a way to represent the interests of each carrier on the platform so that we can find a suitable match. As argued in section 2.1, we have no information on the regular schedules or interests of our carriers, and can therefore only predict their future interests based on their past platform usage. In doing so, we assume that a carrier is more likely to be interested in a shipment that is similar to those they have shown interest in in the past through engagement with the platform.

On the UTURN platform, there are three forms of engagement with shipments that can be seen as interest. We describe them in the list below.

- **View**: when browsing the 'Available shipments' page, a carrier clicks on the item listing to show its details.
- **Quote**: after viewing the shipment, a carrier posts a bid on the shipment, indicating that they would take on the shipment for the quoted price.
- **Match**: after quoting on the shipment, the shipment owner accepts the quote, after which the shipment is assigned to the quoting carrier. This allows the carrier to view the confidential details

necessary to pick up and deliver the shipment.

Although it would be possible to utilise all three forms of engagement, this would introduce unnecessary complexity because of their varying magnitude. Whereas matches represent the literal shipment history performed by each carrier, views are a much more casual form of interest.

While views would supply us with the most data, they provide less reliable information than quotes or matches. Whereas quotes show explicit in taking on a shipment, views do not. Additionally, UTURN experts pointed out an additional reason to be wary of the significance of views. As discussed in section 2.1, carriers used to have to broker with many potential employers for contracts. In doing so, they gather information on various rates offered by various parties in the market and use this to form their own estimate of what would be a fair rate. Without UTURN, this was done based on the limited information obtained from their contacts. With UTURN, however, they suddenly have access to a large source of pricing information, allowing them to make more well-informed considerations. This means that views are also used to gain market knowledge, on top of showing interest. Therefore, we opted to refrain from using views to represent carrier interest.

This leaves us with quotes and matches. The latter is the absolute form of interest: the carrier has performed this shipment and therefore it must have been suitable for them. However, due to the auction-like nature of shipment matching, these matches occur less frequently for each carrier than quotes. Therefore, we opted to represent the history of a carrier in terms of the shipment they have quoted on in the past.

3.1.2. What do carriers look for in a shipment?

To create a recommendation system that is able to find a suitable carrier for a given shipment by comparing its properties to the platform histories of carriers, we must first know what carriers look for in a shipment. This section describes the main characteristics of each shipment that, according to discussions with UTURN experts and carriers, matter most.

Delivery route

Naturally, the delivery route plays a large role when transporting shipments as a carrier. Carriers start their day by leaving from either their truck parking spot near their home or the garage of their associated transport company. Then, after finishing their shipments, they return to their home base to start again the next morning. Depending on their base location, they may only operate in a specific area that allows them to return home during their working hours.

As mentioned before, we do not know the complete schedules of carriers. We merely know which shipments, and thus which routes, were of interest to a carrier in the past. We only know which routes complimented their unknown schedule in the past. Given this limited information, pick-up and drop-off locations are especially important for finding suitable shipments for a given carrier. If each carrier on the platform was a singular trucker, we could be able to narrow down their operational area and restrict our suggested shipments to that area. However, this is not the case; less than a quarter of the platform's carriers have only one truck at their disposition. This means that, when deciding on which delivery routes would be suitable for a given carrier account, we need to allow for multiple operational areas.

Price

Another factor that plays a large role in the shipment selection process of a carrier is the price. When posting a shipment, shippers can enter a suggested price that is shown to the browsing carriers as a baseline. Carriers can then proceed to quote higher if they deem this price too low, or lower if the shipment is of specific interest to them. Depending on the state of the market, the prices of shipments tend to go up and down according to the supply and demand on the market. To protect the intellectual property of UTURN, exact details of this will be omitted throughout this report. Furthermore, carriers may have a required price per kilometre to make their trips worthwhile.

Equipment group

Shipments come in many shapes and sizes, suiting the specific goods they carry. This means that each shipment may have additional requirements for the transporter's vehicle. The length of a container ranges between 20 and 45 feet, which should match the maximum carryable length on the carriers' vehicle. Most containers are of standardized height, with some exceptions. These containers are roughly 30 centimetres taller than the typical height and may require additional security on a truck. Containers may have an open top, be extra wide or instead be a flatbed with the content strapped to them. Next to that, a shipment may not transport goods but rather a liquid, requiring a tank instead of a container. Finally, a shipment may require cooling when transporting certain goods.

Requirements
Next to various equipment groups, shipments can also have specific requirements for a carrier to be able to transport the shipment. These are linked to the contents of the shipment. For example, shipments that require cooling require the presence of a generator on the vehicle to power the cooling system. Other requirements can be in the form of a certification, such as the ADR certification [28], which permits a carrier to transport dangerous goods such as explosives, toxins, or carcinogens.

3.1.3. How can we determine the suitability of a carrier?

With this understanding of the various important factors that may play a role for a carrier when finding a suitable shipment, we move on to discussing how we can determine the similarity between shipments based on these factors. In a first effort to understand how variable the selection of shipments performed by carriers is, we investigated the variability in each of the mentioned factors across the carrier base. This unfortunately did not leave us with useful conclusions; the variation in types of carriers, especially in terms of their size, caused them to be difficult to compare. This posed us with two options: either we first cluster the carriers such that we can treat each group separately, or we attempt to create a model suitable for all carriers. Considering that there was no information on which types of carriers were a good target group for recommendations, we decided to go for the latter, which simultaneously reduces the potential complexity of our solution.

With no prior data on the effectiveness of previous recommendations or what factors are most important for finding the right carrier, we opted to build a first version of a recommendation system that could give us more insights into the importance of each factor. As discussed in section 2.2.3.1, we aim to build carrier profiles by fitting a k-NN on the platform history of each carrier. To subsequently calculate the similarity of a given shipment to the profile of a carrier, we require a similarity function. This similarity function calculates the similarity between two items, or shipments, based on the values of its features.

Typically, similarity functions only calculate similarity for features of a singular modality [1, 17]. The aforementioned shipment characteristics, however, vary in modality, ranging from location data to numerical and categorical values. If we instead opt for a multi-dimensional, multi-modal feature representation, we must find the similarity for each feature and carefully weigh them in a weighted sum [1]. To clarify, we define distance as the opposite of similarity; if two there is no distance between shipments, they are completely similar. We define the similarity between two shipments s_1 and s_2 each represented with N features in Equation 3.1 below.

$$\begin{aligned} \text{Similarity}(s_1, s_2) &= \sum_{i=0}^{N} w_i \cdot sim_i(s_1, s_2) \\ \text{with} \quad \sum_{i=0}^{N} w_i &= 1 \end{aligned} \tag{3.1}$$

In Equation 3.1, the similarity function of each feature i is represented with sim_i and is subsequently weighted with feature weight w_i. In general, we opted to keep the individual feature similarity measures functions simple as a means to avoid introducing additional complexity into the system. We do realise that, given the variability between the various transport markets of various countries, these similarity functions may not be universally suitable. However, with no prior data on recommendations, this custom similarity function is aimed at getting more information on the effectiveness of recommendations, leaving more sophisticated individual feature similarity functions for future work. We now continue with discussing the individual feature similarity functions.

Pick-up and drop-off location similarity

First and foremost, there are the pick-up and drop-off locations. These are arguably the most important factors for comparing shipments. Naturally, these locations need to fit the schedule of the carrier for the carrier to be able to take on the respective shipment. If a carrier has taken on many shipments starting in the Maasvlakte, the Rotterdam harbour, then they are assumed to be interested in taking on shipments starting there in the future.

To describe the pick-up and drop-off locations, we have complete addresses with corresponding latitude and longitude coordinates available. A simple method of comparison would be to compare the names of the associated provinces, municipalities or cities with each other. However, this disregards all information regarding its location on the globe, implying that a small city outside of Rotterdam would be a completely different location than the Rotterdam harbour.

Instead, we should utilise the coordinates associated with each address to calculate the actual distance between two locations. In comparing such coordinate pairs, we can not use distance metrics most commonly used for numerical values in two-dimensional spaces such as the Euclidean distance [1]. Due to the curvature of the globe, using Euclidean distance will cause us to deviate heavily from actual distances. Instead, we can use the haversine formula[1]popularly used in navigation to calculate the great-circle distance between two points on a sphere. Using the haversine formula, we calculate the distance in kilometres between two latitude and longitude coordinate pairs. This is then used to calculate the pick-up and drop-off location similarity factors as follows:

$$\text{Location Similarity}((lat_1, lon_1), (lat_2, lon_2)) = 1 - (\text{Haversine}((lat_1, lon_1), (lat_2, lon_2)) * 0.01) \tag{3.2}$$

This location similarity formula is used to calculate the distance between both the pick-up and drop-off locations of two shipments s_1 and s_2, with their respective latitude and longitude coordinates represented by (lat_1, lon_1) and (lat_2, lon_2).

Equipment group similarity

The equipment specifications mentioned in section 3.1.2 come in various combinations, concatenating their length, type of container and possible extra restrictions. Examples of these are: '40FT HC Dry', and '20FT Reefer'. In total, the platform knows 30-40 different equipment groups. Although not all equipment group terms provide extra restrictions on truck specifications, some of them do and therefore are taken into account by the browsing carrier.

Each specified equipment group is denoted as a series of terms as shown in the examples above. Although domain knowledge could be used to develop a sophisticated similarity function that weighs each specific term based on its implications for the carrier, this was suspected to not be worth the complexity. The implications may vary for each carrier; for instance, some transport companies may have the necessary tools to additionally secure the load, while others may not.

As can be seen by the aforementioned examples, equipment groups may not have an equal amount of terms. Therefore, when deciding on a similarity function, we cannot make use of index-wise similarity functions such as the cosine-similarity. Instead, we opted to use a similarity function that is well-known in the field of information retrieval: the Jaccard Index [23]. The Jaccard Index is particularly useful in contexts where the presence or absence of elements (rather than their frequency or order) is important. The Jaccard Index allows us to compare equipment groups of varying amounts of terms and returns a similarity based on the overlap between the two equipment groups. We define the equipment group similarity as equal to the Jaccard Index in Equation 3.3.

$$\text{Equipment similarity}(x, y) = \frac{|\text{equip}_x \cap \text{equip}_y|}{|\text{equip}_x \cup \text{equip}_y|} \tag{3.3}$$

There are, however, many ways in which this similarity could be represented. In a follow-up experiment described in section 3.1.4.3, we evaluate several other methods of determining the similarity of the equipment group and the upcoming feature, the requirements.

[1]The Haversine formula: https://en.wikipedia.org/wiki/Haversine_formula

Requirements similarity

Like the equipment group, requirements are represented in a series of terms. What we require from our similarity function is that it returns a high similarity if we suspect that, based on the history of a carrier, they can take on the target shipment based on the overlap between the terms in the requirements. Therefore, when comparing the terms of the requirements of the target shipment s_1 to a shipment s_2 in the history of a carrier, we must reason from the requirements of the target shipment s_1. Therefore, we base the similarity on how many of the requirements terms of s_1 are also present in the requirements terms of s_2. Given that requirements terms can be seen as 'restrictions' to a certain degree, we argue that when there are more requirements on a shipment in the history of a carrier, this is not a problem. We define the requirements feature similarity function in Equation 3.4.

$$\text{Requirement similarity}(x, y) = \begin{cases} 1 - \frac{\left|\text{set}(\text{req}_x) - \text{set}(\text{req}_y)\right|}{|\text{req}_x|} & \text{if } \left|\text{set}(\text{req}_x) - \text{set}(\text{req}_y)\right| > 0 \\ 1 & \text{otherwise} \end{cases} \tag{3.4}$$

Price per kilometre similarity

To compare prices of various shipments we convert the price to price per kilometre. In doing so, we aim to find carriers whose minimum price per kilometre value is close to that of the shipment. Through small validations with platform history data of a small group of carriers, we define the price per kilometre similarity function in Equation 3.5.

$$\text{Price per km similarity}(x, y) = 1.0 - \min\left(1.0, \frac{|x - y|}{0.5 \cdot \max(x, y)}\right) \tag{3.5}$$

3.1.4. Evaluation setup and experiments

With the custom similarity function defined, we move on to our evaluation setup. In chapter 2, we argued that, instead of creating recommendations for a carrier visiting the platform, we need to find the right carrier for a given shipment. Therefore, in our evaluation, we evaluate each shipment rather than each carrier. In finding the right carrier for every shipment, we compare the target shipment to each carrier profile formed by fitting a k-NN on the history of the carrier. As discussed in section 3.1.1, we represent the history of each carrier by the shipments that they have quoted on. With these carrier profiles, we can rank each carrier by their similarity to a given shipment. By calculating the similarity between any target shipment and the k-nearest neighbours in the carrier profiles of all carriers, we can create a ranked list of similarity values. When this recommendation system would be deployed, the carriers most similar to the target shipment would be recommended to.

To evaluate the capability of our recommendation system to find the right carrier for the right shipment, we task the system to find the carriers that quoted on each shipment. If our recommendation system ranks carriers that quote on a given shipment high, then we have successfully managed to represent their interests. Therefore, in our evaluation, we sample from the history of quotes on the platform and calculate the similarity to all carriers on the platform for each quoted shipment. The performance of our model is then measured according to the position of the quoting carrier in the ranked similarity list. The ability of our recommendation system to rank quoting carriers highly is summarised in the mean reciprocal rank (MRR) metric as described in subsection 2.2.4, of which the definition can be found in Equation 3.6 below.

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i} \tag{3.6}$$

In Equation 3.6, rank_i denotes the rank at which the quoting carrier's profile was placed in the ranked similarity list returned by our system. By taking the average over all evaluated quotes, the MRR metric reflects the average position at which quoting carriers are placed in the ranked similarity list.

Choosing the value of k

When using a k-NN algorithm for any task, the choice of the value k is crucial. The value k denotes the number of most similar data points that are averaged to determine the estimation of the value of a query shipment. As described in more detail in section 2.2.3, we opted for the use of k-NNs because they can allow us to represent multiple operational areas. Therefore, the value k determines the number of data points, or shipments, that we consider to be an operational area.

Since some carriers may have more than one transporter working for them, and thus, may have a larger platform history than others, determining a fixed value of k to denote an operational area will not be suitable. Instead, we opted for k functions to scale the value of k according to the size of a carrier's history, N. As it was difficult to estimate the average number of operational areas per carrier, we evaluated a wide range of k functions, ranging from various fractions such as $k = \frac{N}{10}$, to $\sqrt{N}$ and $\log N$. In the evaluation results presented in section 4.1.1, we only present the results of the best-performing k functions.

Weight configuration search

As there was no prior information about the importance of each of the selected features, we must search for an optimal weight configuration to be used in the custom similarity function as defined in Equation 3.1. By changing the weights of the similarity functions of each of the features, we can assign more or less importance to a singular feature in the overall similarity function. With five different weights, there is a large space of potential weight configurations to explore. Unfortunately, our evaluation method requires a substantial amount of calculations to rank every carrier for every shipment that we consider. This meant that it was too computationally expensive to exhaustively search the weight configuration space for an optimum. To still approach this optimum, we evaluated both weight configurations based on domain knowledge and randomised weight configurations. Additionally, we perform ablation studies to determine the relative importance of features.

Follow-up experiments

The previous section on weight configuration search briefly mentioned that our evaluation method is computationally expensive. After performing runtime analysis, we concluded that the bulk of the runtime was caused by our custom similarity measure.

To remedy this, we considered a different method. Instead of taking into account every feature for every shipment in the history of each carrier, we find the carriers operating in the area of the shipment purely based on the pick-up and drop-off locations. Then, once we find a collection of 20 suitable carriers, we additionally adjust their returned location-based similarity value according to the other features mentioned above. This greatly reduced our runtime and allowed us to evaluate two other similarity functions for the equipment group and requirements features. We provide a list of evaluated similarity functions for both features and the motivation behind them below.

- **Overlap similarity**. The overlap similarity denotes the similarity measures as described and motivated in section 3.1.3. The overlap similarity is added in this list to reference in the corresponding evaluation section, section 4.1.1.2.
- **Capability similarity**. Instead of determining the similarity between the equipment group and requirements features of the target shipment to each carrier history shipment, we view the shipment history of the carrier as an indication of their capabilities. For example, if one of the k carrier history shipments has the ADR requirement, then we assume the carrier to be capable of performing ADR shipments.
- **Inverse Occurrence Frequency (IOF) similarity**: In the previous two similarity measures, each term is treated as equally important. However, we argue that equipment group and requirements terms that occur less frequently throughout the platform may be less likely to be easily accounted for. The IOF measures how unique a term is across all shipments in the dataset. Then, when comparing the terms of the features of the two shipments, mismatches on less frequent terms weigh more heavily towards dissimilarity than mismatches on frequent terms [5].

Online evaluation

In addition to our offline evaluation method described earlier in this chapter, we perform an online evaluation to measure the actual effect on the overall matching rate of the platform. We conducted a controlled experiment over three months, splitting all triggers of the Matching Algorithm evenly into two groups: the control group and the test group. All shipments associated with triggers from the test group were recommended, while control group shipments were not. The trigger mechanism of the Matching Algorithm will be explained in the next section on deployment, **??**.

By conducting such a controlled experiment, we can directly measure the impact of the algorithm. First of all, we can measure the overarching goal of this research, increasing the platform's overall matching rate. Second of all, we can evaluate how carriers of various sizes respond to our recommendations.

3.1.5. Deployment

After determining the best-performing version of our recommendation system in the offline evaluation, we deployed the Matching Algorithm on the UTURN live servers for three months. Before being able to deploy the matching algorithm on the live servers of UTURN, we needed to ensure that the matching algorithm would not clash with other running algorithms. To ensure its safe deployment, both in terms of working seamlessly with other algorithms and in terms of regulating the number of recommendations sent out, a substantial amount of time was spent on developing an overarching system that manages all algorithms: the Router. Since the exact architecture of the Router is not relevant to the research, only an outline of the system and its implications for the research are given in this section.

The Router architecture has three main implications for the Matching Algorithm. First of all, UTURN requires the number of emails sent out by algorithms to be regulated, regardless of their originating algorithm. Any individual carrier is allowed to receive a maximum of five emails per day. For shipments, only three emails per shipment are allowed in total. An occasional exception was made for shipments posted by new shippers, for which the maximum was increased to five recommendations.

Second of all, we performed additional filtering of recommendation recipients. Although a carrier may be ranked very high by the Matching Algorithm, we assume them to not be interested in receiving a recommendation when they have already interacted with the shipment. Therefore, if a carrier has already viewed or quoted the shipment, they are not recommended to, leaving the recommendation for the next carrier in the ranked list.

Finally, recommendations are not sent for every shipment. As described in detail in chapter 2, it is undesirable to send out recommendations for all shipments, and thus we require some trigger mechanism to decide on which shipments require recommendations. As a baseline trigger, it was decided to only recommend shipments if they had spent more than 90 minutes on the platform. In the upcoming second part of the methodology, we describe our second part of the solution which will replace this baseline trigger with a more sophisticated trigger mechanism.

3.2. Finding the right moment to intervene: the Shipment Pulse Monitor

In the previous section, we described the approach taken to find suitable carriers for any given shipment. In an attempt to maximize the overall matching rate of the platform, we could simply recommend all shipments as soon as they are posted. However, given the platform throughput and the use of emails as our recommendation channel, this would cause us to fill the inboxes of each of our carriers.

For lack of a better trigger mechanism, the Matching Algorithm described in the previous section was deployed with a simple trigger. If a shipment is on the platform for more than 90 minutes without finding a match, it was recommended to suitable carriers found by the Matching Algorithm. However, given the fast pace of the UTURN marketplace and the variety between shipments' characteristics and deadlines, we suspected there to be a better approach. Ideally, we want to be able to estimate the outcome of the shipment based on its characteristics and current platform engagement, and subsequently intervene once we suspect that the shipment will not find a match anymore.

This section describes our process of developing a model that takes into account both historical data and live platform data to predict the outcome of a shipment. First, we go over how such a model would be deployed and what would be a good intervention moment in section 3.2.1 to define our goals. Then, we discuss our choice of model, the decision tree, and how a decision tree can be applied to this use case in section 3.2.3. This is followed by a description of our data pre-processing and what features were engineered to represent the state of a current shipment in subsubsection 3.2.3.3. Finally, we discuss how the model is deployed within the UTURN live servers in section 3.2.4.

3.2.1. What is a good intervention moment?

By developing a predictive model for the outcome of shipments, we aim to be able to determine whether a shipment will be cancelled. This, in turn, allows us to intervene by sending out recommendations to carriers, potentially increasing the shipment's probability of finding a match. With the inherent time pressure exerted by the deadlines specified for each shipment and the possibility of a shipment being cancelled earlier than these deadlines, we ideally want our model to predict a cancellation before the cancellation occurs. Given that this cancellation time is irregular, we should aim to predict cancellations as early as possible.

In making these predictions, we should keep into account the aforementioned tradeoff: we need to avoid sending out too many unnecessary recommendations. Considering these points, we can concisely articulate our objectives in three key points.

1. At any point in time, we want to predict whether a shipment will be cancelled as accurately as possible.
2. When we predict a cancellation, we want this prediction to be 'on time' such that the Matching Algorithm can be triggered and the recipient carriers have enough time to respond.
3. We want to avoid sending out recommendations for shipments that end up being matched, or 'raising a false alarm'.

Furthermore, the choice was made to only intervene once for every shipment. As shipments generally have short lifetimes, we opted to focus on finding a singular intervention moment rather than multiple. On top of that, as discussed in section 3.1.5, UTURN enforced a limit of three to five recommendations per shipment. Spreading these few recommendations out over time would only introduce unnecessary complexity.

3.2.2. Where can we make the largest difference?

The UTURN platform has a high throughput of shipments, with the majority finding their match shortly after being posted. Unfortunately, not all shipments are matched so quickly and some may take longer to find a suitable carrier, if they are matched at all. Before we can determine the focus of our model, we first take a look at how matches and cancellations are distributed over time.

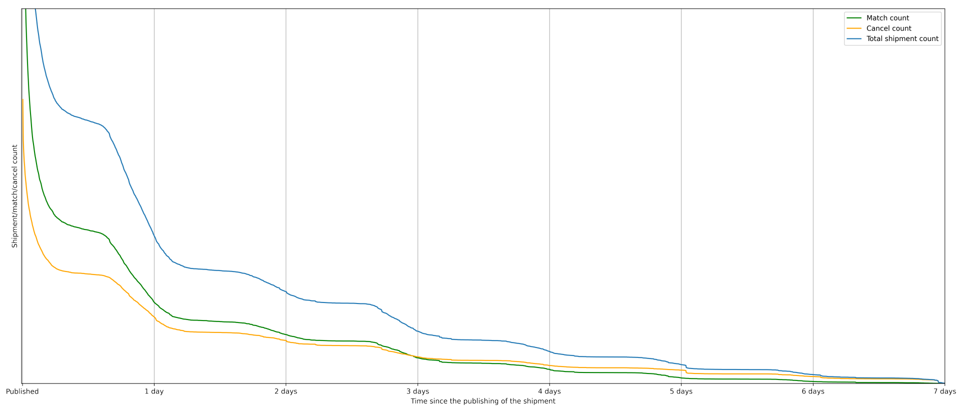


Figure 3.1: Total shipments, matches and cancellations on platform after a certain time on the platform. Shipment counts are omitted to comply with company guidelines.

Figure 3.1 shows the number of shipments that were on the platform for longer than a certain amount of time shown on the x-axis. The graph shows an interesting shape: every decline in the number of shipments is followed by stagnation. This periodicity can be attributed to the hours of the day. Unlike a platform like Amazon, where users may have more time to browse the platform in the evening, the users of the UTURN platform use the platform for their day-to-day jobs, and therefore generally follow working hours. After investigating the engagement throughout every hour of the week, we concluded that roughly 90% of all traffic on the platform happens between 7:00 and 19:00, and a negligible amount of traffic happens over the weekend. After correcting for these 'out-of-office' hours, we end up with Figure 3.2. In this figure, one day corresponds to one day of working hours, which corresponds to 720 minutes.

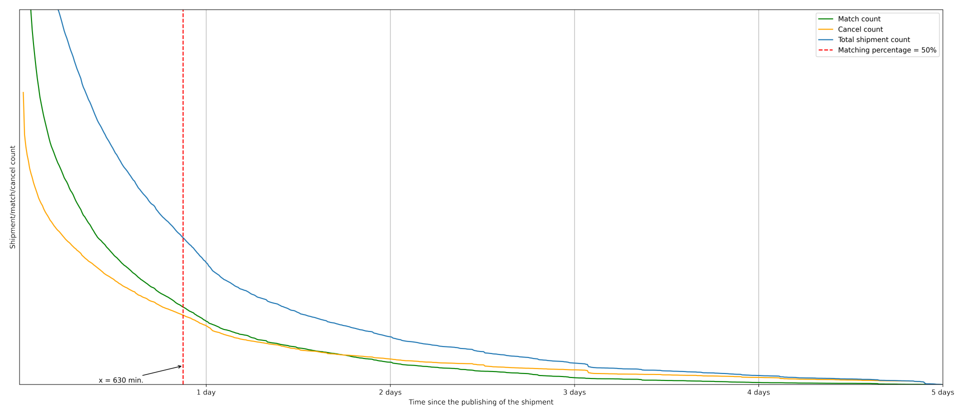


Figure 3.2: Total shipments, matches and cancellations on the platform after a certain time on the platform. Time spent on the platform adjusted to working hours. Exact values are omitted to comply with company guidelines.

Figure 3.2 shows a much smoother graph when compared to Figure 3.1. This indicates that we have successfully adjusted to the working hours of the users of the platform. Next to that, we see that around the 630-minute mark, the amount of shipments that end up as a match is equal to the amount of shipments that end up as a cancellation. As from that point onward, it is more likely that a shipment ends up being cancelled than being matched, a naïve classifier could be created that classifies each shipment after this 630-minute mark as a cancellation. However, this classifier would miss out on 75% of all cancellations. Instead, the classifier should focus on the earlier 'make or break' stages of any shipment's lifetime.

Before we define our area of focus precisely, there is one more adjustment we need to make to the data. As is reflected by the almost vertical decline in the left-most part of the graph, a substantial amount of cancellations occur right after the shipment is posted. This can be attributed to human mistakes; shippers sometimes make mistakes in the details of the shipments. Once this is noticed, the shipment is immediately taken down from the platform to prevent any issues. After inspecting the rate at which shipments are cancelled per minute after posting, we noticed that there was a large peak in the first ten minutes, after which the cancellations per minute steadily declined. Therefore, we opted to refrain from focusing on the first ten minutes.

Coming back to the area of focus, an endpoint to the 'make or break' stage of a shipment must be defined. Although in theory, a model could be created that still predicts the outcome of shipments after being on the platform for three days, it would be better just to send out the recommendations given the historical match/cancel ratio shown in the graph. As a final cutoff point to reduce our potential data set on which to make predictions on, we decided to focus on the area in which 50% of cancellations have not occurred yet: 315 minutes after being posted on the platform.

3.2.3. How can we use a decision tree to predict the outcome of a shipment?

With the goal and area of focus of the model clearly defined, we move on to the model development. In finding the right moment to intervene, we are essentially trying to mimic the decision process that a UTURN employee would take in determining whether a shipment needs additional attention. As motivated and described in section 2.3, we aim to model this decision process with a decision tree. Since our problem differs from other decision tree work, we start by discussing how we can quantify the performance of our model in section 3.2.3.1. After this, we define our custom metrics in section 3.2.3.2. Finally, we close off the description of the decision tree-based method by discussing the selected features in section 3.2.3.3 and our evaluation method in section 3.2.3.4.

How can we quantify performance in metrics?

When assessing the performance of binary classifiers such as the decision tree, accuracy is the most commonly used metric [18, 35]. However, problems arise when utilising the accuracy metric for imbalanced classification problems because its value is mainly determined by the samples from the majority class [40]. Much like the challenge we are addressing, where there are significantly more matched shipments than cancelled ones, medical diagnosis serves as a good example of an imbalanced classification problem. Suppose only 10% of a medical data set belongs to the negative class, then a model that always predicts the positive class would have an accuracy of 90% while having learned nothing about the negative class. Instead, metrics such as precision and recall are more suitable for such applications [24].

There is one key difference between medical diagnosis and our objective when evaluating performance. While a medical diagnosis is made only once per case, our goal involves continuous monitoring of the platform and making predictions over time. This means we deal with multiple data points per classification case, raising questions about data sampling and how we reflect this in our metrics.

If we were to use metrics such as accuracy, precision, or recall (defined in section 2.3.1.1), each shipment must be represented in the dataset equally to ensure that these metrics accurately reflect our ability to predict the outcome of every shipment. This, however, is rather difficult given the variation in shipment lifetimes. Even though we have reduced our area of focus substantially, there remains a considerable number of shipments that do not reach the endpoint we defined, specifically matched shipments. One way this could be remedied is to sample each shipment x amount of times throughout its lifetime on the platform, with an enforced ‘ending’ at the 315-minute mark. Unfortunately, this introduces another problem. With many shipments either finding their match or being cancelled within 30 minutes of the platform, and others reaching the 315-minute mark (or even far beyond that), we would get an imbalanced representation of the shipment distribution as shown in Figure 3.2. Whereas short-lifetime shipments would result in a great amount of data points in the first 30 minutes, longer-lifetime shipments would have more sparsely distributed data points across the area of focus.

Given that we will monitor shipments throughout their lifetimes on the platform, continuously making predictions to intervene when needed, we aim to have as much information available as possible at any possible intervention point. Additionally, since our goal is to utilize live platform data such as views and quotes to predict shipment outcomes, frequent sampling is needed to capture all new engagement on shipments. Therefore, we chose to sample each shipment every minute. This implies that the aforementioned metrics are still not suitable, as shipments with longer lifetimes will have a much larger impact on the metrics compared to those with shorter lifetimes.

Defining custom metrics

Fawcett and Provost [9] propose alternative metrics to a similar problem: activity monitoring to detect phone fraud. In a continuous stream of data gathered throughout calls, there may be a point t in time after which 'suspicious activity' starts. Their goal is not to identify all suspicious activity, nor to classify each data point as positive or negative. Rather, their goal is to identify suspicious activity in a timely fashion after it has begun, whilst avoiding raising too many false alarms. They argue that the ROC-curve[2], although it is seen as the appropriate metric for analyzing classifiers under imprecision, is not suited for a continuous case like theirs. Instead, they introduce the notion of 'False Alarm Rate', the expected number of false alarms over time, as a substitute for the False Positive Rate. Next to that, they adjust their notion of True Positive Rate to a definition that scales a score by the time it took for them to identify a fraudulent call after the fraudulent activity started at time t. With these definitions, they refactored the True Positive Rate and False Positive Rate to suit their continuous classification problem.

[2]ROC-curve: https://en.wikipedia.org/wiki/Receiver_operating_characteristic

There are numerous parallels that can be drawn between the problem tackled by Fawcett and Provost and the problem at hand. First of all, our goal is also not to predict every cancellation data point correctly, but to predict the cancellation in a timely fashion. Next to that, classifying a shipment that ends up as a match as a cancellation is identical to raising a 'false alarm' as we would send out a recommendation when it is not needed. Finally, their resulting adjusted ROC curve highlights the trade-off mentioned before; while we aim to identify cancellations on time as much as possible, this must be done whilst avoiding raising too many false alarms.

Using this alternative method to interpret continuous classification results, we redefine the meaning of True Positives, False Positives, True Negatives and False Negatives to more accurately reflect our use case as follows.

True Positive (TP): a match that, throughout the entirety of its lifetime, is predicted as a match.

False Positive (FP): a cancellation that, throughout the entirety of its lifetime, is predicted as a match.

True Negative (TN): a cancellation that, at any point in its lifetime, is predicted as a cancellation and therefore triggers recommendations to be sent.

False Negative(FN) $\rightarrow$ **False Alarm (FA):** a match that, at any point in its lifetime, is incorrectly predicted as a cancellation and therefore triggers unnecessary recommendations to be sent.

With these adjusted definitions, we define the following metrics:

$$\textbf{TNR (True Negative Rate)} = \frac{TN}{TN + FP} \tag{3.7}$$

$$\textbf{FAR (False Alarm Rate)} = \frac{FA}{FA + TP} \tag{3.8}$$

Finally, we introduce an additional specification on the True Negatives based on point two of our objectives stated in section 3.2.1. Even though we predict a cancellation correctly at some point in its lifetime, it may be the case that the cancellation occurs shortly after the cancellation is predicted, leaving little time for the carriers to respond to the recommendation. To account for this, we introduce the concept of 'on-time True Negatives', or OTTN. Upon inspection of the time between sending a recommendation and engagement of a carrier on the recommended shipment, we found that, on average, it takes carriers 120 minutes to respond. To reflect this in a metric, we discount any correct cancellation prediction at time t_{pred} if it is predicted less than 120 minutes before the moment of cancellation (t_c).

$$\textbf{OTTN}(t_{pred}, t_c) \begin{cases} 1 & \text{if } t_c - t_{pred} \geq 120 \\ \frac{t_{pred}}{120} & \text{if } t_c - t_{pred} < 120 \text{ for all } (t_{pred}, t_c) \\ 0 & \text{if } t_{pred} = \texttt{None} \end{cases} \tag{3.9}$$

With t_{pred} representing the first cancellation prediction (in minutes) in the lifetime of a cancelled shipment and t_c representing the moment of cancellation (in minutes). **What is NaT?**

With OTTN defined, we can define the OTTN rate (OTTNR) as follows:

$$\textbf{OTTNR (On-Time True Negative Rate)} = \frac{OTTN}{OTTN_{possible}} \tag{3.10}$$

With **OTTN**$_{possible}$ defined as:

$$\textbf{OTTN}_{possible} = \textbf{OTTN}(0, t_c) \texttt{ for all } t_c \tag{3.11}$$

With these metrics, we can represent our objectives and the trade-off described in section 3.2.1 directly. This tradeoff can be explained well with our baseline trigger employed for the Matching Algorithm. If we classify each shipment as a cancellation, we would achieve the maximum OTTNR (1.0), but simultaneously the worst FAR value (1.0). If we instead classify each shipment that is online for more than x minutes as a cancellation, we are essentially avoiding sending out too many False Alarms, with the cost of potentially losing out on some cancellations.

This brings up the question of how we should go about optimizing our model w.r.t. both of these metrics. One option would be to optimize for our main performance indicator, OTTNR, and filter out model configurations that leave us with an undesirable FAR. Another option would be to combine these two metrics into a singular metric. However, this would require us to quantify how much a correctly predicted cancellation would weigh up against a false alarm.

After discussing this problem with UTURN colleagues and proposing both options, the decision was made to drop the idea of a combined metric. The lack of information regarding the willingness of carriers to receive recommendations made it challenging to establish a meaningful relationship between both metrics that would support a robust model. Instead, we reasoned that a higher OTTNR is always good to have, while we can allow for False Alarms to a certain degree. Keeping this in mind, we ultimately chose to optimize for the OTTNR, while enforcing a limit of 0.4 on the FAR. This boils down to optimizing to predict as many cancellations on time as possible, whilst ensuring that we do not send out recommendations for 60% of the shipments that end up as a match.

What can we use to predict the outcome of a shipment?

To effectively employ a decision tree for predicting the outcome of a shipment, it is essential to have features that comprehensively capture the state of the shipment at any point in time. Decision trees leverage these features to construct a tree-like structure based on the distribution of feature values across both classification groups: cancellations and matches. In this section, we describe each feature generated to represent the state of each shipment. A complete overview of all generated features can be found in Table **??**, for which we will discuss each feature group separately. Some feature groups contain multiple representations of the same concept which will be evaluated and selected in section 4.2.1.

First and foremost, there is a factor that directly influences whether we can predict cancellations on time or not: the pickup deadline. If the shipment is not cancelled by the owner before reaching the pickup deadline, it is automatically removed from the platform. Time is therefore an important factor that we can use as a feature to represent the state of the shipment. We represented the lifetime of each shipment in three ways. First, `min_until_deadline` represents the time left until the absolute deadline. Naturally, this will be important in estimating whether a shipment will be able to find its match in time. Second, `min_since_publish` represents the time passed since the shipment was published. As discussed before, the bulk of carriers will see the shipment for the first time within the first hour after posting. By using this feature, the model may be able to take this into account. Finally, we have a more general representation of a shipment's lifetime, `lifetime_percentage`, which represents the percentage of the shipment's total possible lifetime that has passed. To clarify: the total possible lifetime of a shipment is the span between the moment of publishing and the pickup deadline.

Feature group	Feature name	Description
Time	`min_since_publish`	Minutes since publishing of the shipment
	`min_until_deadline`	Minutes until the absolute deadline of the shipment
	`lifetime_percentage`	Percentage of total possible lifetime on platform passed
Engagement	`views`	Total number of views on the shipment
	`quotes`	Total number of quotes on the shipment
	`min_since_view`	Minutes passed since last view
	`min_since_quote`	Minutes passed since last quote
Shipment lane	`nuts_lane_match_rate`	Matching rate of the shipment's NUTS lane in the past month
	`nuts_lane_capacity`	Estimated capacity of the shipment's NUTS lane

Table 3.1: Description of all features generated for each datapoint of each shipment

The second group of features represent the engagement on the shipment. We represented engagement with the shipment in two ways: absolute engagement (`views` and `quotes`) and relative engagement (`min_since_view` and `min_since_quote`).

The first type of engagement is the straightforward representation of engagement: the absolute number of views and quotes. Since a shipment needs to be viewed and quoted to find a match, there is a positive relationship between the engagement and the probability of finding a match for a shipment.

The second type of engagement was created to reflect the notion of 'traction' on a shipment. Suppose a shipment has not received any engagement for a long period. This might imply that all potentially interested carriers have already considered the shipment but did not end up quoting and/or matching it.

Finally, we have the shipment lane features. These features correspond to the shipment's lane, or general route. After the Matching Algorithm described in section 3.1 had been developed, UTURN added a new data field to each location field of each shipment: the NUTS region. NUTS[3], or 'Nomenclature of Territorial Units for Statistics' is a geographical nomenclature subdividing the economic territory of the European Union into regions at three different levels. Out of these levels, we opted to choose the smallest level possible, of which the region size is comparable to the size of municipalities. Having these regions defined for the start and endpoints of each shipment, we are able to generate features based on historical data on these routes, or as we call them, lanes.

The first feature that was generated with this, and arguably the most informative feature out of all features, is the `nuts_lane_match_rate`, representing the matching rate of the lane in the past month. Additionally, we created an estimation for the remaining capacity on a specific lane represented by the `nuts_lane_capacity` feature as follows. We take the total number of quotes on this lane in the past five weeks, with the exclusion of the previous week. Then, we take the average to create an estimation of the past weekly capacity. Then, we subtract the number of quotes in the previous week from the estimation of the weekly capacity, to ultimately end up with an estimation of the remaining capacity for this shipment.

How do we evaluate a decision tree on continuous data?

In any decision tree application or machine learning scenario, a training dataset is necessary to train the model, and a separate test dataset is used to evaluate the performance of our model. In creating such data sets, these two data sets must be completely separate [10]. If this is not ensured, the model may be trained with information that it should not have yet, leading to a biased estimation of the model's true performance. With the high throughput of the platform, we have plenty of shipment data that can be used to train and evaluate our model. However, we must not forget that the training and test data sets should be separate.

One approach could be to create this separation purely on the shipment level. By ensuring that all sampled data points belonging to a shipment in the test set are not in the training set, we ensure that our model makes predictions on a shipment it has not seen the data of yet. However, after inspecting empirical results, this showed not to be a suitable approach. This is caused by the fact that we are using the historical matching rate of shipment lanes as one of our features (`nuts_lane_match_rate`, in Table **??**). By using the shipment lane matching rate in the past month as a feature, information about the outcome of shipments within that lane is encapsulated in each data point. When attempting to evaluate in this manner, we noticed that the tree tended to grow extremely large, with an almost ever-increasing OTTNR.

With this use of historical data as features, we must instead ensure that any point in the training data set precedes any point in the test set. This means that we could for example train on data points corresponding to shipments that were on the platform in the summer months, and subsequently evaluate on shipment data points that were on the platform in the autumn. One might think that if we just give the model as much data as possible historical data as possible, it will be able to learn the dynamics of the platform and therefore make predictions more accurately. Unfortunately, this is far from being true. First of all, the transport market varies heavily throughout the year. Supply- and demand-driven seasons alternate, making platform data from over six months ago less relevant to the current market conditions. Furthermore, it is challenging to base our current-day model on older data due to two reasons. First, the platform's user base is rapidly expanding as UTURN continues to expand throughout

[3]NUTS (Nomenclature of Territorial Units for Statistics):
https://en.wikipedia.org/wiki/Nomenclature_of_Territorial_Units_for_Statistics

Europe. Second, the transport market was significantly impacted by the COVID-19 pandemic and has been in recovery throughout 2022.

This brings us to our final approach. Instead of collecting a large test set and an even larger training set, we opted for periodic retraining and evaluation to allow the model to adjust to the changes in the market. Unfortunately, this meant that we could no longer use the built-in SKLearn methods such as `train_test_split` and `GridSearchCV`. Therefore, we created a version tailored to our use case to retrain and evaluate iteratively. As a first evaluation, we performed weekly retraining and evaluation throughout the entirety of 2023. To account for potential seasonal changes in the market, the training data set length was introduced as a parameter to the model. Next to that, we conducted a search in the feature set space to determine the best-performing set of features. The results of this search can be found in the beginning of section 4.2.1. Finally, to avoid overfitting the tree on the training data as discussed in section 2.3, various maximum depths for the decision tree were evaluated. Other stopping criteria were not explored due to the decline in model retraining and evaluation speed caused by our inability to utilize GridSearchCV.

Introducing cost-sensitivity

Although the custom metrics defined in subsection 3.2.3.2 accurately reflect the goal of the model, the current method of creating the data sets poses some problems. Currently, each shipment, regardless of its outcome, is sampled every minute and introduced with equal importance into the data set. The decision tree subsequently learns decision points based on its splitting criterion. For a detailed description of this process, please refer to section 2.3.

In creating these splits, a decision tree treats every data point equally. With our data, a problem arises because of the class imbalance visualised in Figure 3.1. With imbalanced data sets, decision trees tend to form a bias towards the dominant class [10]. Machine Learning algorithms assume that misclassification errors (false negatives and false positives) are equal [37]. Such an assumption can be dangerous in the aforementioned applications of fraud detection and medical diagnosis [42], where some misclassifications may weigh substantially heavier than others. Class imbalance can be remedied with resampling [10], but, this comes with the risk of omitting valuable data points from the data set. Since we are aiming to predict the outcome of any shipment at any time, regardless of their characteristics, we should avoid changing the distribution of both classes [41].

A different solution to this problem is called cost-sensitive learning [24]. Rather than treating each sample equally, cost-sensitive learning considers the cost associated with the misclassification of each individual sample. We see a possible application when we relate this to our objectives. The objective is to classify as many cancellations 'on time' as possible, while allowing for the misclassification of matches to a certain degree. Additionally, we should consider allowing for the misclassification of cancellations in the early stages of their lifetime, as it is still possible to find a match naturally. However, as time progresses, it becomes increasingly critical to avoid misclassifying cancellations, as we need time to intervene with recommendations

There are numerous methods to incorporate cost-sensitivity into decision trees. Sahin et al. [31] performed a thorough evaluation of both cost-insensitive and cost-sensitive decision trees applied to the field of fraud detection. Their methods include building a 'direct-cost' decision tree that splits each node purely based on the estimated misclassification cost, assigning more cost to transactions that have a higher risk of financial loss. Next to that, they evaluated various adaptations of splitting criteria based on the underlying class distribution of the data. They concluded that cost-sensitive models generally outperformed cost-insensitive models, with the exception of the 'direct-cost' model. This exception was attributed to the fact that the direct-cost method did not take into account the underlying class distribution. Out of the methods that did take this into account, the tree built with a cost-sensitive adaptation of the Gini impurity splitting criteria was shown to perform the best out of all models.

Given the conclusion made by Sahin et al., we opted to use the adaptation of the Gini impurity splitting criteria to introduce cost-sensitivity into our model. Although evaluating various implementations of cost-sensitivity could lead to interesting conclusions, this would require us to implement a decision tree ourselves. This, however, was outside of the scope of this project.

A cost-sensitive version of the Gini impurity is realised with a small adaptation to the Gini impurity splitting criterion shown in Equation 3.12 below. For a more detailed explanation on the Gini impurity splitting criterion, please refer to section 2.3.

$$\frac{N_{\text{parent}}}{N_{\text{total}}} * \left(\text{impurity}_{\text{ parent}} - \frac{N_{\text{right}}}{N_{\text{parent}}} * \text{impurity}_{\text{ right-child}} - \frac{N_{\text{left}}}{N_{\text{parent}}} * \text{impurity}_{\text{ left-child}} \right) \tag{3.12}$$

In Equation 3.12, the impurity of the left and right nodes are weighted purely based on the fraction of samples that split off from the parent node into the child nodes ($\frac{N_{\text{right}}}{N_{\text{parent}}}$ and $\frac{N_{\text{left}}}{N_{\text{parent}}}$). Following a method described by Ming Ting et al. [38], we can introduce cost-sensitivity by assigning weights to each individual sample. For instance, we could assign a weight of two towards cancellations while maintaining a weight of one for matches. This doubles the importance of creating a homogenous cancellation node, while still incorporating the overall distribution of the data. In this way, we emphasize classifying cancellation data points correctly depending on their characteristics.

This leads us to the question of which data points to put emphasis on to align with our objectives. To reiterate, we essentially want to allow for the misclassification of cancellations early on, whilst simultaneously ensuring that the cancellation is predicted 'on time' before the cancellation occurs. As the probability of a cancellation occurring only increases over time as visualised by Figure 3.1, more emphasis should be put on cancellations over time. This could be realised by scaling the weight of a cancellation data point based on the number of minutes it still had left on the platform. However, this approach was deemed invalid since this information would not be available for new, unseen shipments. Instead, we could scale the weight by its time left until the first deadline, as represented by our `min_until_deadline` feature. Upon reviewing the distribution of cancellation occurrences relative to their first deadline, we found the timing to be too irregular to establish a reliable weighting scheme.

Ultimately, we chose to scale the cancellation data point weights based on the time currently spent on the platform, while leaving match data points at a weight of one. This was realised by introducing the cost-sensitivity factor R, which is used to determine a cancellation data point weight as shown in Equation 3.13.

$$w_c(t) = \frac{1}{R} * t \tag{3.13}$$

with t representing the minutes after the publishing of the shipment, and R essentially representing the duration (in minutes) it would take for cancellation data points to surpass the weight of match data points.

By introducing the cancellation data point weighting scheme with the cost-sensitivity factor R as defined in Equation 3.13, we aim to permit early misclassification of cancellation data points while gradually increasing the importance of correctly predicting cancellations. This approach allows the model to identify critical cases early in a shipment's lifetime while minimizing false alarms for shipments likely to find a match. Additionally, we hypothesize that this cost-sensitivity factor R can play an important role in adjusting to the state of the market. In periods where the platform is supply-driven and shipments generally tend to have a short lifespan, the optimal R value may be low. In a demand-driven market, on the other hand, the optimal R may lie a lot higher, allowing shipments more time to find their match.

To evaluate this potential beneficial influence of the cost-sensitivity factor R, we additionally performed evaluations with varying values of R throughout the entirety of 2023, of which the results can be found in section 4.2.1.4 of the evaluation chapter.

3.2.4. Deployment

To close off the methodology section of the Shipment Pulse Monitor, we describe how the model was deployed on the UTURN live servers.

In the deployment section of the Matching Algorithm, section 3.1.5, we described how the Matching Algorithm is triggered. Any shipment that was on the platform longer than 90 minutes without finding a match triggered the Matching Algorithm to send recommendations. With the introduction of the Shipment Pulse Monitor, we are completely replacing this trigger mechanism.

The Shipment Pulse Monitor is a separate entity on the UTURN system that communicates with the Router. The Router continuously tasks the Shipment Pulse Monitor to predict the outcome of the available shipments based on their current features through an API. Once a shipment is estimated to end as a cancellation, the API call returns the corresponding shipment IDs, after which the Router triggers the Matching Algorithm for those specific shipments. After the Shipment Pulse Monitor returns a shipment ID to trigger recommendations, it is excluded from further predictions as we only recommend once for a single shipment. Finally, at the end of each day, multiple decision trees are generated with various parameters and evaluated on the past weeks of platform data. We evaluate various model selection methods in section 4.2.1.5. The best-performing decision tree is then stored in memory to monitor the platform on the next day.

4 Evaluation

The approach described in chapter 3 was researched and evaluated successively. As there was little to no prior data on recommendations on the UTURN platform, the Matching Algorithm was deployed with a simple trigger mechanism as described in section 3.1.5. After that, we developed the Shipment Pulse Monitor, which was later added to the overall system to provide a more sophisticated and informed trigger mechanism.

To illustrate the development and evaluation of the system, we follow the chronological order of research and deployment. We begin with the Matching Algorithm in section 4.1, followed by the integration of the Shipment Pulse Monitor in section 4.2. Each section starts with an offline evaluation performed to develop the system and concludes with an assessment of its online performance.

4.1. Matching Algorithm

In this section, we describe the evaluation of the Matching Algorithm. As described in detail in section 3.1, the Matching Algorithm is tasked with finding the right carrier for any given shipment such that we can send shipment recommendations to them. We start this section by describing our offline results gathered during the development of the Matching Algorithm in section 4.1.1, and then continue with the results of the controlled experiment on the UTURN live servers in section 4.1.2.

4.1.1. Offline evaluation

Given our goal of creating both a recommendation system and a shipment outcome predictor, development was done in an agile manner. Unfortunately, building a recommendation system for UTURN with the UTURN data meant that we were not able to use TU Delft resources like DelftBlue[1]. Although this restricted the size of our experiments, it also encouraged a more iterative and adaptive process. This section continues with the results of each of these iterations, showcasing the progress and improvements made along the way.

First iteration: custom distance measure, various k values

In the first iteration of the Matching Algorithm, we follow the approach described in section 3.1.3. With a sizeable parameter space comprised of various carrier quote history sizes, weight configurations and various methods of determining k, small experiments were executed to limit our parameter space. Empirical findings showed that, when considering the length of our training data for each carrier model, collecting more than three months of carrier platform history did not show any benefits compared to three months of history. To ensure that the value of k would be suitable for each carrier, regardless of their size, we evaluated k values scaled to the carriers' quote history size N as described in section 3.1.4.1

This iteration of the Matching Algorithm turned out to be very inefficient, not allowing for large, representative evaluation. Therefore, the results of these experiments were deemed inconclusive. However,

[1]DelftBlue: the TU Delft supercomputer; https://www.tudelft.nl/dhpc/system

we established a baseline for future experiments: using all five features, we achieved an MRR of 0.29. In contrast, location-only KNNs achieved an MRR of 0.26, indicating that the additional non-locational features contributed to the system's performance.

Second iteration: post-processing based on categorical features

After analysis of the runtime of a complete evaluation, the bulk of the runtime was spent in the custom similarity function. This was anticipated, as it calculates and combines five different similarity measures into one. This combined similarity function is then used to calculate the similarity between each shipment we evaluate and each shipment in the history of each carrier. Consequently, we sought ways to reduce this runtime to facilitate agile development, enabling us to run the controlled experiment on the live servers and continue with the development of the Shipment Pulse Monitor (4.2).

As reasoned about in section 2.2.3.1, our primary motivation for using KNNs is the possibility of carrier accounts consisting of multiple transporters, and thus having various operational areas and requirements. Our features comprise of both locational data indicating this operational area and categorical data representing vehicle and driver requirements for given shipments. We hypothesized that by identifying carriers operating near the target shipment area and then applying post-processing based on categorical features, we could enhance efficiency.

For each shipment, we identified the 20 carriers whose carrier profile models returned the highest similarity to the shipment, and re-ranked those carriers according to their categorical feature similarity. For this re-ranking, we evaluated three categorical similarity functions listed in subsubsection 3.1.4.3. After re-ranking according to categorical similarity, we additionally re-ranked according to the price per kilometre similarity. An evaluation was run across three months of platform data, sampling 10% of the quotes and refitting the carrier models every week. The results of this experiment are found in Figure 4.1.

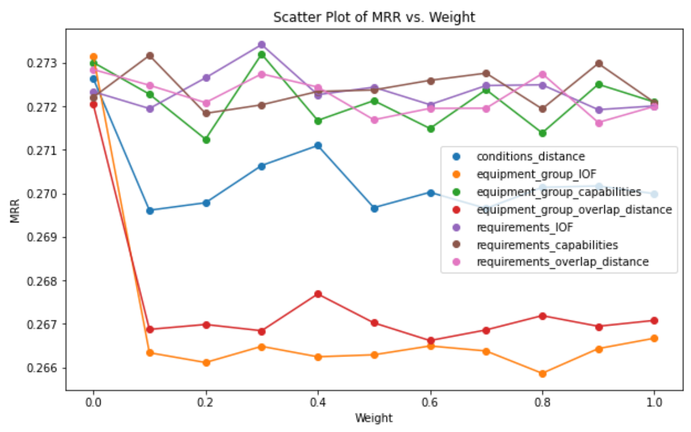


Figure 4.1: Results for post-processing with categorical features

This experiment yielded surprising results. Using locational data alone, we achieved an MRR of 0.272. Contrary to the belief reinforced by UTURN professionals that these categorical features were crucial in a carrier's shipment selection process, post-processing by these features seemed to have little to no effect, if not a negative effect. Furthermore, this contradicted the results described in the previous section, where we observed that these categorical features, when included in a custom distance measure, indeed contributed to better performance.

The only plausible explanation at the time was that these categorical features are crucial when identifying the 20 most suitable carriers for a given shipment; otherwise, the right carriers might not make the cut. Consequently, we decided to abandon this approach and revert to the method used in the first iteration.

Third iteration: larger evaluations, larger weight configuration search

With the contrasting results from the post-processing experiment, we reverted to the first approach, which initially showed better performance. Although the first approach showed better performance, this approach did show varying results. One possible explanation for these variations could be that the test set was not large enough, introducing a bias due to a too short evaluation time span. To address this, we optimized our code by removing unnecessary calculations and caching intermediate results. This optimization allowed us to double our experiment size and expand our weight configuration with randomization. The results of this expanded experiment can be found in Table 4.1. Generally, the `log2` function used to determine the value of k for each carrier model outperformed the other k functions in nearly all cases, so these are omitted from the table.

Table 4.1: Matching Algorithm: large experiment experiment results with randomized weight configurations

w: loc1	w: loc2	w: equip	w: req	w: price	MRR
0.0	0.0	0.33	0.33	0.33	0.176
0.4	0.4	0.0	0.0	0.0	0.273
0.4	0.4	0.2	0.0	0.0	0.279
0.4	0.4	0.0	0.2	0.0	0.290
0.4	0.4	0.0	0.0	0.2	0.280
0.35	0.35	0.15	0.15	0.0	0.2977
0.35	0.35	0.0	0.15	0.15	0.292
0.25	0.25	0.166	0.166	0.166	0.301
0.039	0.23	0.277	0.20	0.247	0.306
0.316	0.227	0.223	0.06	0.17	0.305
0.316	0.227	0.211	0.309	0.108	0.294
0.208	0.03	0.07	0.29	0.39	0.294

The top half of the table shows weight configuration experiments to deduce the importance of our features. As expected, a configuration without location features performed substantially worse than those with location features. Additionally, when comparing the third and fourth rows in the table, we observed that among the two categorical features, the requirements feature contributed the most to an improved MRR. Combining all three additional features, we achieved an even higher MRR of 0.301.

Continuing with the randomization of the weight distribution space, we found remarkable results. Interestingly, the best-performing weight configuration assigned only a weight of 0.039 to the start location feature. Although this seemed like a lucky find, the row below showed almost identical performance with a start location weight of 0.316. Moreover, the top randomized result had a high weight for the requirements feature, which was previously identified as highly informative. However, the second-best randomized result achieved similar performance with a substantially lower requirements feature weight. We then considered that the price feature might play a larger role than expected, but this was contradicted by the final row in the table, which showed lower performance with a high price weight.

In summary, there was no clear relationship between adjusting certain weights and an improvement or deterioration in MRR. While we hoped to find a semi-optimal weight configuration, we encountered highly variable results.

Why do we not see any changes in performance when changing parameters?

With our results contradicting hypotheses and weight configurations showing inexplicable variations, we revisited the entire process. We rechecked various k values/functions, sampling methods, and feature generation, finding no faults. Finally, after taking every bit of the evaluation method under the loop, we identified the issue: the presence of giant carriers. Even though we were sampling test quotes according to transport type, start- and end location to ensure a representative distribution of the shipments of the platform, we did not account for the presence of giant carriers in our data set. These large transport companies quote on most shipments, causing them to be heavily represented in our test set. As a result, any parameter or weight configuration favouring these giant carriers would automatically perform better according to our MRR metric. Our attempts to create a recommendation system for every carrier regardless of their size by using k functions instead of fixed values were ineffective due to this inherent bias in the test set.

This realization led to a crucial decision on how to proceed. If our goal was to create a model that accurately finds the assigned carrier for each quote, we could avoid this bias by ensuring that carriers quoting more frequently are not over-represented in the test set. However, while we are trying to find the right carrier for the right shipment, we are ultimately still trying to maximize the overall matching rate with our recommendations. Ignoring the large carriers, which make up the majority of quotes and matches, would neglect their significant role in the platform. As they have the largest capacity of all carriers, they are simultaneously most likely to be able to pick up problematic shipments. Considering this, we decided to filter out test quotes from giant carriers while leaving the rest unfiltered. This approach removed the largest bias from our test dataset while acknowledging that carriers with larger capacity have a higher probability of picking up problematic shipments.

Final weight configuration search

Due to the various challenges and computationally intensive evaluations, we were faced with a tight schedule leading up to the company's release deadline. Nevertheless, we effectively focused our remaining weight configuration research on two targeted evaluations: one filtering out carriers with more than 2000 quotes and the other with more than 1000 quotes. This strategic approach allowed us to maximize our use of the available time and resources.

In these results, we observed more predictable changes in performance when adjusting weight configurations and making additions or removals. Table 4.2 shows the results averaged between the two evaluations. Notably, the requirement feature consistently proved to be the most informative addition. Ultimately, the weight configuration in the bottom row demonstrated the best performance across both evaluations. It is particularly interesting to note that the second location feature decreased in weight, suggesting that carriers prioritize the starting location of a shipment. However, we acknowledge that this evaluation method has its limitations, and therefore, any conclusions drawn about the importance of individual features may not be entirely conclusive.

Table 4.2: Matching Algorithm: final weight configuration search results

w: loc1	w: loc2	w: equip	w: req	w: price	MRR
0.5	0.5	0	0	0	0.186
0.33	0.33	0.33	0.0	0.0	0.1862
0.33	0.33	0.0	0.33	0.0	0.201
0.33	0.33	0.0	0.0	0.33	0.189
0.25	0.083	0.083	0.25	0.33	0.232

To conclude, the process of developing and evaluating the Matching Algorithm went far from how was planned. The main oversight discussed in subsubsection 4.1.1.4 caused all of our experiments to show biased results, with inexplicable variance between various experiments, weights and parameter configurations. As there were no real informative or interesting conclusions about our method due to the shortcomings of our evaluation method, we will discuss how things could have been done differently in detail in section 5.1. The shortage of development time for this part caused by this oversight ultimately led us to not be able to properly answer our research questions, and produce a far from optimal model. Still, with an MRR of 0.232, and the final carrier filtering choice discussed in subsection 3.1.5, we suspect the Matching Algorithm to produce some useful recommendations. On top of that, we are sending 3-5 recommendations per trigger of the Matching algorithm to carriers that are guaranteed to not have interacted with the recommended shipment, which increases our odds of finding a good match. The next section will show the results of a controlled experiment that will show whether the Matching Algorithm turned out to have a positive effect on the overall matching rate of the platform.

To conclude, the process of developing and evaluating the Matching Algorithm, while challenging, provided valuable insights and opportunities for improvement. Despite the main oversight discussed in subsubsection 4.1.1.4 leading to biased results and variance in weights and parameter configurations, we learned significant lessons that will guide future work. The limitations of our evaluation method prevented us from drawing definitive conclusions about our approach. We continue the discussion on the lessons we learned and potential improvements in section 5.1.

The shortage of development time due to this oversight presented challenges, but we still achieved an MRR of 0.232. With the final carrier filtering choice discussed in section 3.1.5, we are confident that the Matching Algorithm can produce useful recommendations. On top of this, the engagement filtering choice described in section 3.1.5 ensures that the recipients of our recommendations have not yet interacted with the recommended shipment, thus further increasing the shipment's possibility of finding a suitable match. Despite the hurdles, these results lay a strong foundation for future advancements.

4.1.2. Online evaluation

After completing the offline evaluation of the Matching Algorithm, we proceed to discuss the results of the controlled experiment conducted on UTURN's live servers.

To assess the impact of the algorithm's recommendations on the overall matching rate, each trigger of the Matching Algorithm was evenly assigned to either the test or control group. Shipments in the test group received recommendations to carriers suggested by the Matching Algorithm, while shipments in the control group did not receive any recommendations.

We begin this section by drawing general conclusions about the differences between the test and control groups in section 4.1.2.1. This is followed by a discussion on the engagement with the recommended shipments, an analysis of differences across various countries and carrier sizes, and finally, an evaluation of engagement with our email recommendations to suggest future improvements for UTURN.

General experiment results

Over the course of three months, the Matching Algorithm was triggered for a total of 2995 shipments. Table 4.3 shows the total amount of shipments assigned to both the test and control groups. When we look at the overall outcome of all shipments assigned to the two groups, the matching rate of the test group is 2.8% higher than the control group.

Table 4.3: Matching Algorithm Experiment recommendation count and match rate

	Test group	Control group
Unique shipments recommended	1506	1489
Recommendations	4971	-
Matching rate	38.3%	35.5%

To evaluate the effect of the Matching Algorithm in detail, we continue by evaluating the engagement of the recommended carriers with the recommendations. In Table 4.4, we show engagement rates for both the total number of recommendations and the number of unique shipments that were recommended. For the unique shipments column, we count any number of views or quotes made by a recommended carrier as a 'viewed' or 'quoted' unique shipment.

Table 4.4: Matching Algorithm Experiment engagement count and rates

	Total recs	Unique shipment recs
Total recommendations	4971	1506
Quotes	304	247
Quote rate	0.061	0.164
Matches	98	98
Match rate	-	0.065
Views	715	563
View rate	0.144	0.374

Considering the quote and view rates in Table 4.4, the Matching Algorithm has demonstrated a substantial positive impact. Of the 1506 unique shipments for which we sent recommendations, 6.5% were successfully matched to a carrier we recommended. Since each shipment can only have one matched carrier, the overall match rate for total recommendations is not provided.

When examining the number of shipments that received quotes, we find that 16.4% of the recommended shipments received a quote from at least one of the chosen carriers. Although the total recommendation quote rate is lower, this is expected due to the competitive nature of the UTURN marketplace. Once a quote on a specific shipment is accepted, other carriers that received a recommendation for that shipment can no longer quote. Views are more readily performed by carriers, so we anticipated higher view rates compared to quote rates. This expectation is reflected in the data, with 37.4% of all unique shipments receiving at least one view. However, only 14.4% of recommendation emails resulted in a view.

A possible explanation for this is that the Matching Algorithm's effectiveness varies by shipment type or carrier. As UTURN expands to other European countries, some regions may be underrepresented in terms of active carriers, resulting in lower matching rates. This aspect will be evaluated in section 4.1.2.2. Additionally, some carriers may benefit less from recommendation emails due to having assigned platform scanners, as discussed in section 2.1. We will the Matching Algorithm's effectiveness for various carrier sizes in section 4.1.2.3.

Performance in various countries

UTURN is expanding rapidly throughout the EU, but development is still ongoing in many of the EU markets. With a lower base of carriers in certain regions, the overall matching rate possibly decreases. Therefore, it is important to consider the specifics of the recommended shipments to better understand the strengths and weaknesses of the Matching Algorithm and whether the 2.8% increase in the matching rate holds across multiple countries.

Table 4.5: Matching percentage (M%) of the Matching Algorithm for different pick-up and drop-off locations of recommended shipments.

Start loc.	End loc.	avg. grp size	Test M% δ
Germany	Germany	391	1.5%
The Netherlands	Germany	375	3.4%
The Netherlands	The Netherlands	308	6.2%
The Netherlands	Belgium	101	4.4%
Belgium	Belgium	98	2.9%
Belgium	Germany	89	4.2%

Table 4.5 shows the performance of the algorithm grouped by the pick-up and drop-off locations of each recommended shipment. The pick-up and drop-off locations shown in the table were selected based on their group size; any other groups had fewer than 50 recommended shipments.

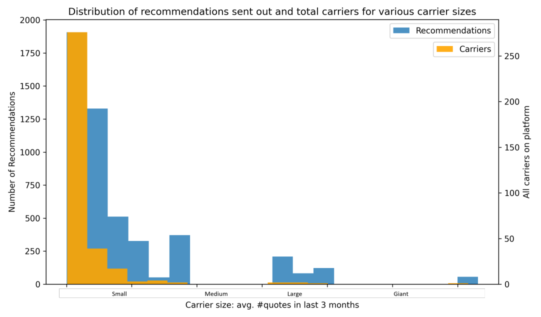
We can see that performance varies depending on where the shipment starts and ends. Among all country pairs, the algorithm performed best in the Netherlands with a 6.2% increase in the matching rate, despite the Netherlands already having the highest overall matching rate. Interestingly, although Germany triggered the most recommendations, it saw the smallest increase in matching percentage. This is likely due to the smaller carrier base in Germany; the number of active carriers operating within Germany is roughly 50% lower than those operating only in the Netherlands. Another factor could be the suitability of the location similarity function, as the larger distances within Germany may require a different approach than in the Netherlands.

Despite the positive percentages across the table, it is challenging to determine whether the Matching Algorithm can universally find suitable carriers for any shipment on the platform, regardless of its pick-up and drop-off locations. Numerous factors influence the matching process, such as varying carrier bases, country-specific costs, and market characteristics. These factors are beyond the scope of this project, as our primary focus is on evaluating the application of a general recommendation algorithm to the two-sided transport market.

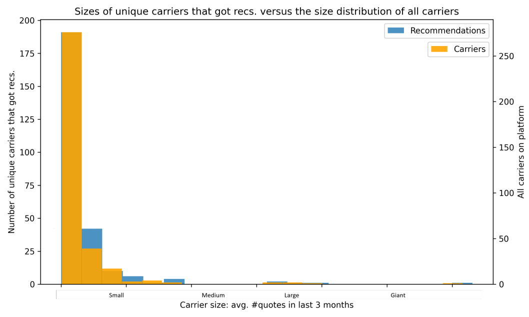
The data resulting from the controlled experiment showed that the test group has consistently outperformed the control group for any pick-up and drop-off location pair and that the algorithm therefore appears to have had an impact. Limiting our scope on the calculation of the matching percentage to the three main countries above, the overall gained matching percentage increases to 3.4%.

Recommendations for various sizes of carriers

One of the goals of the Matching Algorithm was to create an algorithm that is able to find suitable carriers regardless of their size. As described in section 4.1.1.4, we faced a difficult decision regarding filtering based on carrier sizes. Ultimately, we opted to refrain from filtering based on size and instead filtered on engagement. This approach ensures that the carrier to whom a shipment was recommended had no prior knowledge of that shipment. To evaluate whether the Matching Algorithm can find suitable carriers of all sizes and to assess our filtering method, we analyzed the Matching Algorithm's performance for various carrier sizes.



(a) Distribution of carrier sizes for recommendations sent out (blue) and all carriers on the platform (orange)



(b) Distribution of carrier sizes for unique carriers we recommended to (blue) and all carriers on the platform (orange)

Figure 4.2: Distribution of total number of recommendations **(a)** and unique carriers recommended to **(b)** plotted against the general distribution of the size of carriers.

Figure 4.2 illustrates the distribution of our recommendations with respect to the recipient carrier size. Actual carrier sizes are omitted due to privacy reasons and instead, represented in general terms. Figure 4.2a shows the number of recommendations each carrier size bin received, while Figure 4.2b shows the number of unique carriers that received recommendations for each size bin.

In Figure 4.2a, we observe the impact of our filtering choice. While the offline evaluation of the Matching Algorithm in section 4.1.1 showed a large bias towards large carriers, this bias is less evident in the number of recommendations sent to large carriers. This can be attributed to our filtering method; larger carriers usually have already engaged with the shipment before the Matching Algorithm is triggered and are therefore not recommended. However, we see a noticeable difference in the size of the first carrier size bin compared to the rest of the histogram. This difference can be attributed to a change in the UTURN platform. Halfway through the experiment, UTURN introduced a payment plan, requiring carriers to subscribe to the UTURN platform. This instigated a large change in the carrier base, which mainly reduced the amount of smaller carriers. Given that the smallest bin of carriers was reduced in size, the distribution of both figures 4.2a and 4.2b will be more similar to the actual carrier size distribution, confirming that our algorithm generates recommendations for all carrier sizes. Additionally, when we examine the number of unique carriers that were recommended to for each size bin in Figure 4.2b, we see that it roughly follows the overall carrier distribution.

Engagement for various carrier sizes

Even though the histograms above show that our deployed algorithm that filters on engagement did not exhibit a large bias towards larger carriers, the question of whether recommendations are suitable for all types of carriers remains. Therefore, we also examine the engagement with recommendations across each carrier size bin.

In Figure 4.3, we observe that the percentage of recommendations leading to a quote is lowest for the smallest carriers. This is expected, as smaller carrier companies have less capacity to pick up jobs on short notice compared to larger ones. Engagement rates, in terms of views and quotes, increase as carrier size grows, with the highest engagement rates found among the largest carriers. However, as discussed in section 3.1, by inquiring large carriers about their experience with the Matching Algorithm, we found that the largest carriers typically have an assigned 'platform scanner' and are therefore not in need of recommendation emails. Although the carriers we recommended to ended up viewing and/or quoting on the recommended shipment, this does not necessarily mean that the view or quote was prompted by the recommendation.

To determine whether these engagement rates are a direct result of our email recommendations, we proceed with an evaluation of open and click-through rates in the following section.

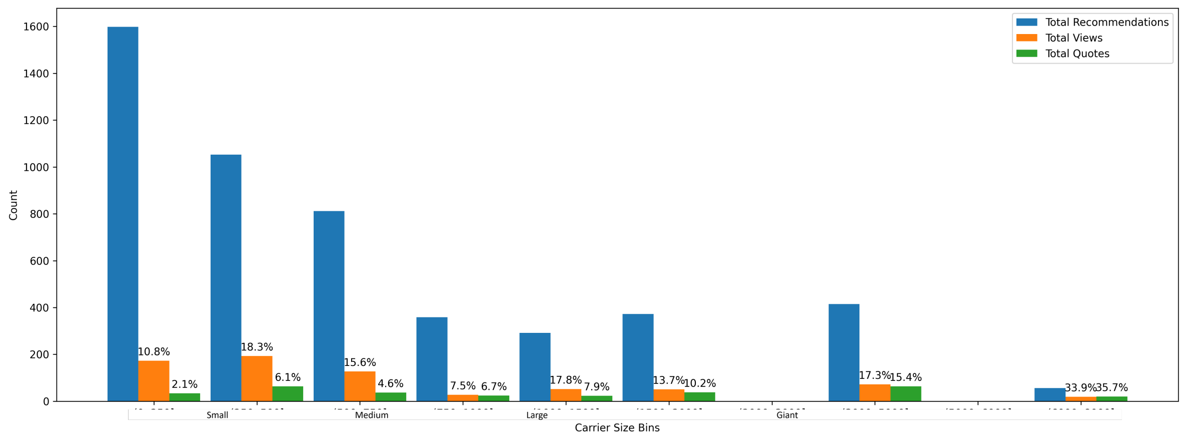


Figure 4.3: Recommendation engagement for various carrier sizes

Relating shipment engagement to recommendation email engagement

To finalise the online evaluation of the Matching Algorithm, we investigate the open and click-through rates of our recommendation emails. An overview of the rates can be found in Table 4.6.

Table 4.6: Email open and click rates. The second column shows the engagement per unique recommended shipment.

	Total	At least one per rec.
Total recommendations	4971	1506
Emails delivered	4870	1503
Emails opened	2154	1192
Open rate	0.44	0.79
Emails clicked	652	524
Click rate	0.13	0.35
Views from rec. carriers	715	563
Quotes from rec. carriers	304	247

There we see that far from all recommendation emails are opened, let alone clicked. Next to that, we see that our click rate is comparable to our estimated view rate in Table 4.4 (0.144, 0.374). Although they are comparable, we see evidence of views not originating from recommendations as the total measured engagement view counts (bottom of the table) are higher than the actual clicks.

Table 4.7 indicates that not all views came from our recommendation emails. A noteworthy finding is that while 652 recommendation emails were clicked, only 40% led to a view. This is surprising because a click should directly lead to the shipment page, seemingly translating directly to a view. One possible explanation is that the shipment was no longer available at the time of the click. However, after checking, we found that 96% of these clicks occurred before the shipment became unavailable. This suggests that the shipment might have been temporarily unavailable at the time of the click, or there may have been an issue with the email data.

Nevertheless, these email engagement rates can help identify which carriers may be a suitable target group for recommendations. Figure 4.4 shows open and click rates for the various carrier sizes.

Table 4.7: Matching Algorithm: Recommendation email opens and clicks related to carriers that have also viewed or quoted

	Count	Total	Rate
Emails delivered	4718	-	-
Opened	2154	-	-
Opened and viewed	382	715	0.53
Clicked	652	-	-
Clicked and viewed	260	715	0.4
Clicked and quoted	64	304	0.21
Clicked and matched	23	98	0.23

The rates indicate that larger carriers rarely engage with the recommendation emails, whereas smaller carriers show more interest, with an engagement peak among small to medium-sized carriers. This finding supports our decision to filter out larger carriers in the offline evaluation, as discussed in subsection 4.1.1. This graph suggests that for future recommendation practices on the UTURN platform, focusing on small to medium-sized carriers could be more effective than targeting larger carriers.

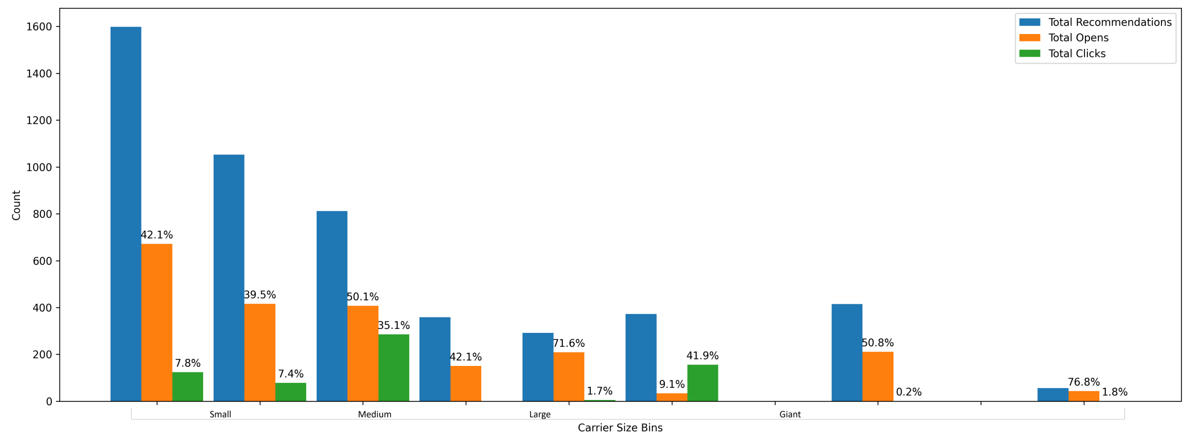


Figure 4.4: Email engagement for various carrier sizes

4.2. Shipment Pulse Monitor

After the release of the Matching Algorithm on the UTURN live servers, development began on the Shipment Pulse Monitor (SPM). As described in both the introduction and methodology, there is a trade-off in recommending shipments on the UTURN platform. While the goal is to maximize the matching percentage with recommendations, we must be mindful of avoiding spam.

The SPM is designed to actively monitor the platform, continuously predicting the outcome of each shipment. Its primary goal is to predict cancellations as early as possible while simultaneously avoiding raising too many false alarms (e.g., wrongful cancellation predictions). This section continues with the offline evaluation of the SPM in subsection 4.2.1, followed by a comparison of this offline evaluation with its performance on the UTURN live servers insubsection 4.2.2.

4.2.1. Offline evaluation

During the development of the SPM, there were many factors to be taken into account. As discussed in section 3.2, we focused on optimizing the tree depth, the size of the training data and the set of features used for generating the decision tree that would ultimately predict the outcome of shipments on the platform.

Feature selection

To reduce this large parameter space, the evaluation was first run with weekly retraining and evaluation. This was done across the entirety of 2023. As complete evaluations cost a substantial amount of time due to computational limitations, we employed the following strategy.

First, a weekly evaluation of 2023 was conducted using all possible features. During the evaluation, SKLearn's feature importance feature was used to identify which features were more informative than others. Out of the complete set of features, listed in Table **??**, three features showed to be most informative, namely the NUTS lane match rate, the NUTS lane estimated capacity and the minutes left until the first deadline. Among these, the NUTS lane matching rate proved to be the most informative. These three top features were then used as a base for feature selection, after which additional features were tested to improve the value of the OTTNR metric.

One of the goals for the Shipment Pulse Monitor was to create a model that utilises live platform data to make predictions throughout the lifetime of a shipment. To achieve this, we proposed two methods of representing user engagement: absolute and relative. Both representations of engagement were hypothesized to have their own merits; while relative engagement represents a sense of traction on the shipment, absolute engagement represents the overall popularity of the shipment. Table 4.8 shows that out of the two representations of user engagement, absolute engagement clearly outperforms relative engagement.

One possible explanation for the difference in performance is that, combined with the `min_until_first_deadline` feature, absolute engagement effectively captures the traction on a shipment as well. Although there is typically a spike in engagement when a shipment is first posted, we can assume a somewhat spread-out distribution over time. With this in mind, a decision boundary might be: "if there are fewer than 20 views with less than 10 hours left until the deadline, predict a cancellation." Relative engagement, on the other hand, provides little to no information about the shipment's overall popularity.

Table 4.8: SPM: Decision tree evaluation of various feature sets. Values show results of Weekly retraining and evaluation across 2023.

Feature set	OTTNR mean	FAR mean	OTTNR std.	FAR std.
Base	0.570	0.397	0.011	0.005
Base + absolute engagement	0.631	0.399	0.008	0.005
Base + relative engagement	0.591	0.397	0.014	0.006

We then continued the search for the optimal feature selection by adding singular features. However, during this search, we did not manage to find any substantial improvements. Minor improvements (less than 0.01) in the OTTNR mean were often paired with a higher variance in both OTTNR and FAR. Given that there was no significant improvement found, the details of this experiment are omitted from the report. Given that no significant improvement was found, the details of this experiment are omitted from the report. As a final sanity check, we attempted to substitute the time representation `minutes until the first deadline` with the two other time representations, `lifetime percentage` and `minutes since the shipment was published`. Again, no significant improvements were found, and therefore, the 'base + absolute engagement' feature set was retained for future experiments.

Definition of baselines

Before evaluating whether the metric scores are good, we need to establish a baseline. Throughout this section, we compare the model performance to two types of baselines.

The first baseline is a baseline that predicts based on a threshold, or as we call it, an x-minute baseline. This baseline determines the outcome purely based on the number of minutes a shipment has spent on the platform. Any shipment online for less than x minutes is classified as a match, while any shipment online for longer than x minutes is classified as a cancellation. The Matching Algorithm is deployed with a trigger mechanism that enforces a threshold of 90 minutes. This 90-minute baseline has proven to work surprisingly well, and thus, we will compare our model to this baseline throughout this section.

The second baseline is based solely on a singular feature: the NUTS lane matching rate. Given that we have access to the lane matching rate, a simple and reliable predictor could use a threshold on the matching rate to make predictions. The baseline predicts as follows: only if a shipment has a NUTS lane matching rate of below x, we classify it as a cancellation. The threshold x is chosen such that the mean FAR falls below the 0.4 threshold. This will be referred to as the Matching Rate (MR) baseline.

Comparison to baselines

Table 4.9 shows the performance of both baselines. As can be seen, both baselines perform surprisingly well, even better than our model. Whereas the matching rate threshold was chosen such that the mean FAR value was below 0.4, this could not be done for the 90-minute threshold baseline. Naturally, if a predictor is allowed to exceed this limit of 0.4, it will be able to achieve a higher OTTNR. As a point of reference, we searched for an X-minute threshold that had a mean FAR of below 0.4, of which the results can be found in the '150 min. threshold' row. Even though the OTTNR mean values of the baselines are higher than those of our model, we see some difference in the variance. The model appears to perform slightly more consistently.

Table 4.9: SPM: Baselines compared to best model for weekly retraining and evaluation across 2023

Model / Baseline	OTTNR mean	FAR mean	OTTNR std.	FAR std.
Base + relative engagement	0.63	0.399	0.008	0.006
90 min. threshold	0.74	0.495	0.078	0.080
150 min. threshold	0.65	0.397	0.093	0.081
Matching Rate (threshold: 0.86)	0.670	0.392	0.093	0.091

Introducing cost sensitivity

Our next experiment was aimed at strengthening our model by introducing cost-sensitivity, discussed in more detail in section 3.2.3.5. In essence, the introduced cost-sensitivity tries to model what our 90-minute baseline modelled in a very clear-cut way. While, ideally, we want to estimate that a shipment is going to be cancelled as early as possible, we want to allow a shipment to find a match on its own until a certain point in time. The 90-minute baseline enforced this with a hard cutoff; however, this came with the trade-off of introducing many False Alarms. With cost sensitivity, we want to model this in a more nuanced way; in an attempt to learn from the strengths of the 90-minute baseline, we want our model to have a decreasing trust in a positive outcome for the shipment as time progresses.

In the following experiments, we introduced the cost-sensitivity factor R as described in 3.2.3.5. Using the same methodology as the previous section, we conducted a year-long evaluation with weekly retraining and assessment. The results are presented in Table 4.10. The model configuration that performed best on average throughout the year, shown in the 'R - Best on avg.' row, does not show a significant improvement over the non-cost-sensitive model. However, by selecting the best-performing model each week, we observe a substantial improvement in OTTNR. This result approaches the OTTNR of the 90-minute baseline while maintaining a mean FAR below 0.4. This highlights that the cost-sensitivity factor R enables the model to adapt effectively to supply- or demand-driven market conditions.

Table 4.10: SPM: results of introduction of cost sensitivity factor R for weekly retraining and evaluation across 2023

Model config	OTTNR mean	FAR mean	OTTNR std.	FAR std.
No R	0.63	0.39	0.093	0.091
R - Best on avg.	0.64	0.396	0.008	0.006
R - Best per week	0.72	0.356	0.062	0.055
90m baseline	0.74	0.495	0.078	0.08
MR baseline	0.67	0.39	0.093	0.091

The introduction of the R value caused the 'best performing' model configuration to vary between weeks and allowed us to better suit the changes in the market. Consequently, we wondered if we could achieve better performance by optimizing for an even smaller time span, such as daily. Table 4.11 shows the results of an evaluation conducted daily throughout 2023. We can already see an improvement in the 'best on avg.' row compared to the weekly retraining and evaluation. This is surprising because retraining daily only changes the training set by one day at a time.

This suggests a potential limitation of this method of retraining and evaluation. By taking the mean of daily evaluations, we lose normalization according to evaluation size. While weekly evaluation somewhat guaranteed a consistent evaluation set size, this can vary more from day to day. We will further discuss this in the discussion chapter in section 5.2.

When we look at the best model per day, we see the OTTNR improve even further. With a mean OTTNR of 0.795 and a mean FAR of 0.366, it significantly outperforms the 90-minute baseline, which exceeds the FAR limit. A comparison of the two can be found in Figure 4.5. Here, we observe that while the 90-minute baseline often comes close to the best-performing daily model, it violates the FAR limit nearly every day. Additionally, the 90-minute baseline performs substantially worse during validations 135-165, which corresponds to August, a month when many shipments were cancelled early in the day, averaging around the 90-minute mark. With the cost-sensitivity factor R, the model was able to adapt and classify cancellations earlier.

Finally, a comparison to the Matching Rate baseline is given in Figure 4.6. This comparison demonstrates that the model outperforms the MR baseline on almost all days of the year.

Table 4.11: Performance of model against the baselines. Rows indicated with 'Best avg.' contain the performance of the model or baseline that performed best across the entire evaluation period.

Model / baseline	OTTNR mean	FAR mean	OTTNR std.	FAR std.
R - Best on avg.	0.665	0.387	0.178	0.147
R - Best per day	0.795	0.366	0.088	0.034
Best MR threshold on avg.(0.84)	0.652	0.374	0.15	0.11
Best MR threshold per day	0.648	0.328	0.146	0.067
90m baseline	0.723	0.478	0.117	0.11

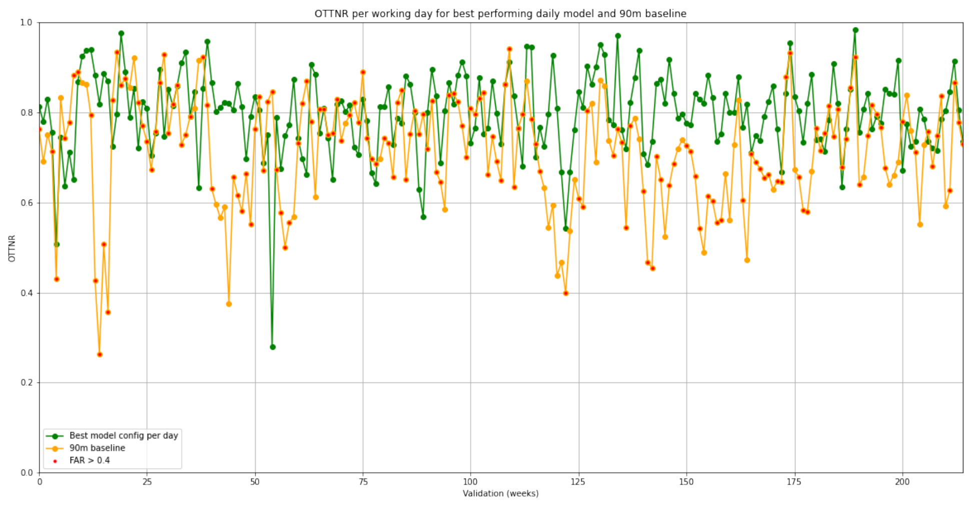


Figure 4.5: Comparison between the 90-minute baseline and the best performing daily models. Days where the 90-minute baseline exceeded the FAR limit are marked with red dots.

How would this work on live servers?

The cost-sensitivity factor proved to be a promising addition to the model due to its ability to adapt to market changes. However, the 'best per day' results, while very promising, are somewhat misleading. These results show the metric values when considering the performance of the model in hindsight, indicating the model's potential rather than its actual performance. If we were to deploy the model on UTURN's live servers, we would not have the luxury of cherry-picking the best-performing model each day. Instead, we need to base our model on the best previous performance.

This introduces decisions to be made. What do we view as the 'best previous parameter configuration'? A simple solution would be to just take the model that performed best on the previous day. However, in

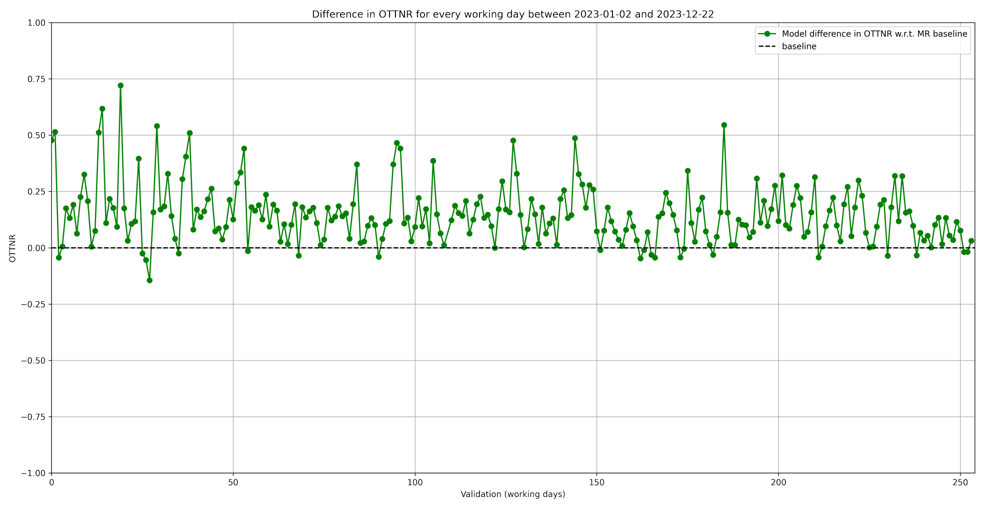


Figure 4.6: Comparison between the best performing daily Matching Rate threshold and the best performing daily models. Days where the 90-minute baseline exceeded the FAR limit are marked with red dots.

empirical findings, we noticed that the day-by-day model performance can vary significantly. Therefore, we evaluated three methods, which are discussed below.

Not having the luxury of cherry-picking the best-performing model, unfortunately, deteriorates the performance of the model substantially. Whereas our cherry-picked configurations reached an OTTNR mean of 0.795, selecting the best previous configuration drops the OTTNR mean down to 0.680, with the mean FAR increasing to 0.403. This highlights a drawback of daily retraining and evaluation; one day of irregular behaviour on the platform can suggest a parameter configuration that is unrealistic for the next day. This volatility makes it challenging to enforce a FAR below the 0.4 threshold.

To counteract this volatility in the UTURN marketplace, we can use a rolling average. This method works as follows: for each validation, we average the performance of all configurations over the past x days and select the configuration with the highest mean OTTNR that also satisfies the FAR constraint on average. Figure 4.7 shows the mean OTTNR and FAR for models selected by a rolling average of varying x values. When we compare rolling averages of $x > 5$ to those with $x <= 5$, we can see the difference caused by daily retraining and evaluation. Any $x > 5$ generally results in a drop in OTTNR. FAR, however, remains more steadily below the 0.4 threshold. Rolling averages with $x <= 5$, on the other hand, show a significant improvement in OTTNR, which is unfortunately accompanied by an increase in FAR.

Considering the similarity in performance for various values of x and the volatility of the market, we suspected that a single rolling average would not be optimal for every validation. To search for an optimum, we employed the following strategy: for every validation, derive the best-performing previous models based on their rolling average performance for each value of x. Then, pick the rolling average with the highest mean OTTNR that still has a mean FAR below 0.4. This proved to be the best-performing model selection method.

Table 4.12 shows the results of this final experiment. By selecting the best rolling average for each validation, OTTNR improved by 0.02 when compared to selecting only the best previous configuration, while only being accompanied by an increase in FAR mean of 0.008. Unfortunately, this method still causes the model to exceed the FAR limit of 0.4. To counteract the high volatility of the UTURN marketplace, we additionally tried setting a minimum for x. However, this turned out not to be beneficial for both the FAR and OTTNR. In a final attempt to reach the 0.4 limit, we restricted the selection method to only select rolling averages with a mean FAR < 0.35. This achieved a mean FAR of 0.401, whilst still keeping a mean OTTNR that is higher than that of the Matching Rate baseline.

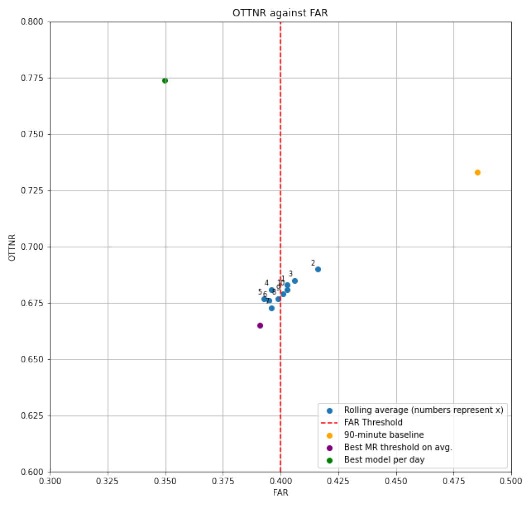


Figure 4.7: OTTNR plotted against FAR for baselines and rolling average results

Table 4.12: SPM: Final model selection results and baseline comparison

Model selection method	OTTNR mean	FAR mean	OTTNR std.	FAR std.
Best previous config	0.680	0.402	0.163	0.13
Select best x	0.699	0.411	0.155	0.129
Select best $x >= 2$	0.697	0.419	0.148	0.131
Select best $x >= 3$	0.695	0.414	0.168	0.147
Only mean FAR < 0.35	0.677	0.401	0.15	0.121
90m baseline	0.723	0.478	0.117	0.11
MR baseline (best previous)	0.640	0.336	0.148	0.107

SPM: Offline evaluation summary

To summarize, we developed a model that utilizes both historical data about shipment lanes and real-time engagement on a shipment to predict shipment outcomes. By introducing the cost-sensitivity factor R, the model adapted to market changes throughout the year, showing improvement over general baselines. The model achieved a mean OTTNR value close to that of the robust 90-minute baseline while maintaining a lower mean FAR value. This indicates that we can predict cancellations 'on time' almost as effectively as the strong 90-minute baseline while classifying fewer matches as cancellations. Pushing the model to achieve a mean FAR as close to 0.4 as possible still results in performance surpassing the best Matching Rate baseline. After discussing the results with UTURN colleagues, the 'Select best x' model selection method was chosen for release on the live servers, introducing a relaxation of the FAR limit.

4.2.2. Online evaluation

Section 4.2.1 concluded that our best-performing predictor for the outcome of a shipment was a model using, among other features, absolute engagement (views and quotes), combined with the cost sensitivity factor R. After deriving the best-performing model selection method, we showed performance across the entirety of 2023, resulting in a mean OTTNR of 0.699 and a mean FAR of 0.411.

Over a period of two months, the SPM has been integrated into the Router architecture as described in section 3.2.4. The SPM continuously monitors each shipment on the platform and predicts the outcome of each shipment with the stored decision tree. The model predicted the outcome of a total of 12.000 shipments. It achieved a mean OTTNR of 0.739, a substantial increase compared to the mean offline OTTNR of 0.699. As expected, this was accompanied by an increase in mean FAR of 0.033, resulting in a mean FAR of 0.443.

As the performance of our model differed from our mean offline results of 2023 quite a bit, we investigated the period leading up to the deployment more closely. We observed a shift in the market from the second half of December to late February. Previously, the market was demand-driven, with shipments typically having shorter lifespans and the most optimal models having an optimal R in the range of 40-60. However, the first months of 2024 were supply-driven, a state that has continued since. Shipments had longer lifespans on average, reflected by higher optimal R values in the range of 100-120. This highlighted the model's ability to adapt to the current state of the market.

Looking at our offline performance for the months leading up to deployment (Jan-Feb) in Table 4.13, we see very comparable performance to our online results. we see performance very comparable to our online results.

Table 4.13: SPM: Online results compared to offline results

Model	OTTNR mean	FAR mean	OTTNR variance	FAR variance
Online results	0.739	0.443	-	-
Best previous config	0.702	0.425	0.19	0.14
Select best x	0.732	0.432	0.16	0.137
Select best $x >= 2$	0.702	0.404	0.158	0.123
Select best $x >= 3$	0.691	0.396	0.168	0.147
90m baseline	0.74	0.495	0.078	0.08
MR baseline	0.67	0.391	0.08	0.025

Furthermore, an interesting finding here is that selecting the best $x >= 2$ actually could have helped counteract volatility in the market. While increasing the x drops the OTTNR, it does allow us to reach a mean FAR of below 0.4. After checking the results, we saw that in these months, there were occasional days where the chosen R value would suddenly drop to 50-60. This can happen when there is a day on which all matches happen very quickly after the shipment is posted. Then, when the model evaluates configurations on previous days, it notices that on the previous day, it was able to get a high OTTNR and a FAR below 0.4 by setting a very low R. This means that our model will tend to predict cancellations early on, which results in a very high OTTNR. In turn, this high OTTNR causes the configuration to be selected for the next day. This again shows the drawback of our daily optimization; if there is a day of low- or irregular traffic on the platform, the model tends to be influenced a lot. This is also reflected in the higher variance in both metric scores.

The reason why this method of selecting an $x > 1$ did not have the same effect in the year-long evaluation is that the R that performed best on average throughout the year was already quite low (50). This meant that if there was a day in which there were many early matches, it had a lower impact than in the demand-driven market evaluated by the model. This indicates that, depending on the state of the market, it might be worth looking into setting a minimum for the value x if one would want to more strongly enforce the 0.4 limit.

Still, the live version of the SPM produced slightly higher OTTNR and FAR values than estimated with our offline evaluation of January and February. Now, this may just be caused by a change in the market, but we do see that the FAR has deviated even further from the 0.4 limit. We suspect that this may be caused by the database connection on the live servers. The database from which our shipment data is retrieved only updates every 15 minutes. This means that we will only be able to make predictions every 15 minutes, and therefore are more likely to miss a decision point that we were able to capture in the offline case. Unfortunately, not much can be done to counteract this. We considered including this limitation in the offline evaluation as well but ended up refraining from doing so because of the following reasons. As shipments are posted at different times, their data is updated at different deltas from the moment of publishing. This means that the updates that come through to the SPM will also represent different deltas from the moment of publishing, and we figured that the SPM would be able to perform better if it were to be equipped to make predictions at any of these moments, rather than at 15-minute intervals.

How did the model compare to the baselines?

To allow us to gather enough data on the performance of the model, the SPM was not released with any kind of comparative experiment. However, we can still compare the performance of the model to the baselines by calculating the performance that our baselines would have had. An overview of this can be found in Table 4.14.

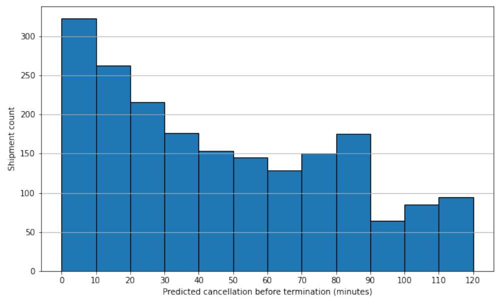
Table 4.14: Overview of model performance and baseline performance over the period in which the model was deployed.

Model / baseline	OTTNR	FAR	TNR
SPM	0.741	0.443	0.737
90-minute baseline	0.84	0.59	0.784
180-minute baseline	0.751	0.431	0.688
Matching Rate baseline	0.667	0.39	0.667

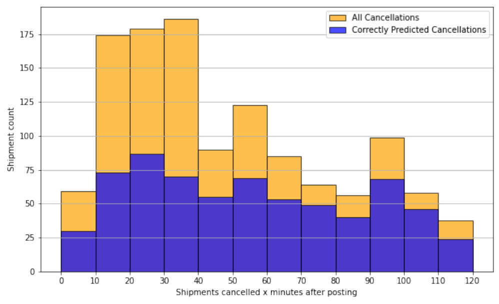
The table shows that, if the 90-minute baseline were still deployed, it would have sent many more false alarms than our model. However, as we reasoned before, the market changes throughout the year and therefore our 90-minute baseline may not always be suitable. Thus, we included an x-minute baseline that had a comparable FAR to our model in the table.

This adjusted x-minute baseline slightly outperforms our model in terms of the established metric, OTTNR. Does this mean that our model is worse than the baseline? Well, not completely. The third column shows the True Negative Rate (TNR), or the fraction of cancellations we predict as cancellations at any point in time. The TNR indicates that we predict more cancellations than the 180-minute baseline. Since the 180-minute baseline classifies every shipment online for more than 180 minutes as a cancellation, the difference must lie in short-lifetime shipments. To show this in more detail, Figure 4.8 shows the distribution of cancellation predictions for cases in which the shipment was cancelled less than 120 minutes after posting.

Figure 4.8a shows that many cancellations are correctly predicted less than 120 minutes before the actual cancellation occurs. Since our OTTNR metric discounts correctly predicted cancellations based on the time between the prediction and the cancellation, these contribute less to the overall OTTNR. Figure 4.8b shows the distribution of cancellations that happened within 120 minutes of posting and the fraction that was correctly predicted by the model before they occurred. Although the resulting



(a) Distribution of predictions of cancellations with less that were predicted as cancellation less than 120 minutes before the cancellations



(b) Distribution of cancellations that happened less than 120 minutes after posting and the fraction that the model predicted correctly

Figure 4.8: Distribution of cancellations that happened 120 minute

recommendations may not have had the full 120 minutes of possible response time, they may still be timely for some carriers. Moreover, starting from the 40-minute mark, we see a very high success rate. The lower success rate before the 40-minute mark can partly be attributed to 'accidental' cancellations, as discussed in section 3.2.2.

To summarise, our model performs slightly almost identical to an optimized t-minute threshold baseline in terms of our OTTNR metric. However, there are numerous cancellations that were predicted less than 120 minutes before the cancellation occurred and are therefore discounted by the OTTNR metric. On top of that, the model also predicted numerous cancellations correctly that had a total lifespan of less than 120 minutes, which would be completely missed by this t-minute threshold baseline. Even though the recommendations resulting from these predictions may not have been predicted 120 minutes before the cancellation occurred, these recommendations may have still been on time for some of the recipients. We think that, if UTURN were to continue employing a Shipment Pulse Monitor, a better solution could lie in the combination of hard cutoff points such as Matching Rates and t-minute baselines and the use of a predictive model.

5 Discussion

In the previous chapter, we gave an in-depth evaluation of both parts of our solution: the Matching Algorithm and the Shipment Pulse Monitor. In this chapter, we will continue with a discussion of the evaluation, highlighting both the lessons we learned based on the evaluation and possible shortcomings or future improvements. In a similar fashion to the previous chapters, we discuss the Matching Algorithm in section 5.1 first, and the Shipment Pulse Monitor second in section 5.2.

5.1. The Matching Algorithm

In deciding on an approach for the generation of recommendations for the UTURN platform, we found that, instead of finding the right shipments for every user like a regular recommendation system, we should rather find the right carriers for every shipment. With no prior information on how such recommendations would be received, or which carriers would be suitable to send recommendations to, we took on a general approach aimed at creating a system suitable for all carriers on the platform.

Following this goal, we developed a recommendation system that is aimed at representing the interests of each carrier in a carrier profile based on their platform history. In building these carrier profiles, we aimed to create a system that, for every shipment that received a quote, would rank the quoting carrier highly in terms of suitability. In creating a recommendation system focused on this goal, we hypothesized that this system would be able to find carriers that are likely to quote on any shipment that would require additional attention in the future.

With this approach, we created a custom similarity function as defined in subsection 3.1.3. In the first experiment described in subsubsection 4.1.1.1, we achieved an MRR of 0.29 using all five features. This appeared to be a great result as this value implies that, on average, for each shipment, we ranked the carrier that ended up quoting on the shipment on the third or fourth rank. However, while this custom similarity function allowed us to take into account all features while determining the most similar neighbourhood of a carrier, it introduced many dependent variables through the introduction of feature weights and individual feature similarity functions. This complex similarity function significantly increased the computational complexity of an offline evaluation, which hindered us from optimizing each feature similarity function separately.

In search of a more scalable and optimizable solution, we reduced the k-NN to only take into account the pick-up and drop-off location similarity in the second iteration described in subsubsection 4.1.1.2. This reduced the number of optimizable parameters significantly and allowed us to experiment with post-processing the result returned by the location-only system. However, even though we were determining the overall similarity in an almost identical manner, the sequential nature of it appeared to cause a significant drop in MRR. This led us to go back to the original approach and perform a more thorough evaluation.

The third iteration, described in subsubsection 4.1.1.3, resulted in us finding a configuration with a better MRR than our first iteration. However, in this iteration, we found there to be inexplicable variations in

MRR when comparing various weight configurations. Ultimately, we discovered that our MRR metric was dominated by the behaviour of the 'giant' carriers. Any version of our system that gave prevalence to the giant carriers resulted in a higher MMR because of the evaluation data bias towards carriers that quote more frequently. After accounting for this by filtering out these giant carriers, we finally saw explainable variations in the results of varying weight configurations.

Although this oversight hindered us in creating an optimized system, it did highlight what was not encapsulated in our goal. We assumed that to maximize the probability of finding a match for any problematic shipment, we should find the carrier who can most likely take on the shipment. However, our online evaluation showed that this task is more nuanced than originally thought.

Unlike online marketplaces in which items are simply bought, shipments require actual delivery time and therefore must fit the carrier's schedule and capacity. As the 'giant' carriers on the platform have a substantially larger capacity than smaller carriers, they are much more likely to be able to take on these shipments (and often do). This was reflected in the bias of our recommendation system shown in the offline evaluation - given their capacity and thus, large platform history, many of the shipments for which we generated recommendations found similarity to their carrier profile, resulting in them being highly ranked and thus receiving recommendations. Although, according to our goal, the recommendation system was exactly doing what it was tasked to do, the online evaluation showed that this is not what we should try to achieve.

If we look at the engagement on recommended shipments for various carrier sizes in Figure 4.3, we see that, out of all sizes, the giant carriers viewed and quoted on recommended shipments the most. However, when we look at the recommendation email engagement in Figure 4.4, we see that this engagement does not originate from the recommendation emails. This suggests that these giant carriers may not be the right target group for recommendations. After contacting two of these giant carriers, we found out that they are indeed not the right target group because they have employees assigned to scan the UTURN platform continuously. This is reinforced by the fact that many of the highly ranked giant carriers were filtered out by our engagement filtering choice described in subsection 3.1.5, as probably, their platform scanner had already viewed the shipment.

Overall, our online evaluation showed that out of all recommended shipments, 37.4% of shipments that were recommended were viewed by at least one of the recommended carriers, and 16.4% was quoted. However, email recommendation interaction data showed that only 40% of these views and 20% of these quotes originated from our email recommendations. This observation highlights the need for additional research into the carriers themselves. Even though we might be finding the right carriers for the problematic shipments, these carriers may not be the right recipients for recommendations.

This shows that employing a 'one-size fits all' solution may not be suitable for this problem given the high variation between carriers. A future solution should perform qualitative research on which carriers are open to recommendations. Additionally, follow-ups on the recommendations can be done to determine why they were or were not suitable for them. This would allow for the narrowing down of the target group or even creating multiple target groups, allowing for the definition of various carrier-size-specific recommendation systems.

5.2. The Shipment Pulse Monitor

In developing the Shipment Pulse Monitor (SPM), we were faced with a multitude of challenges. First of all, we were dealing with continuous data. In our first attempts at creating a decision tree, we followed the traditional way of training and evaluating a decision tree. We generated a large data set and split it into training and test sets whilst ensuring that all data points belonging to a singular shipment were not spread out across both sets. In doing so, however, we noticed that our tree grew endlessly while achieving almost ever-increasing performance. This was caused by the fact that we were using dependent data. By using the historical NUTS lane features as described in Table 4.8, the data points contained in the training data set contained matching rate information about the data points in the test set; the two sets were not completely independent. This could be remedied by deciding on a cutoff point in time between the training data set and the test set (e.g. training on the first half of the year, evaluating on the second). However, this was suspected to be sub-optimal given the highly volatile transport market.

Instead, we opted for periodic retraining and evaluation, averaging the metric scores across the whole evaluation. In the first experiments described in subsection 4.2.1, we performed weekly evaluations. By predicting the outcomes of all shipments for a week, we ensured that each evaluation set was approximately equal in size to a certain degree. The experiments described from Table 4.11 and onwards, on the other hand, performed daily evaluations. Since we are averaging the metric scores over the complete evaluation period, some days may have had significantly less data than others, while having the same amount of impact on the overall metric score. This is a limitation of our evaluation method, which could have been remedied by weighing each daily evaluation according to its test set size. However, when we compare the online results to the offline results in Table 4.13, we see that the offline results are very comparable to the online results, indicating that this limitation did not significantly impact our results.

One of our goals for the SPM was to incorporate live shipment data in addition to historical shipment data to make predictions about the outcomes of shipments. In subsubsection 4.2.1.1, we discuss our feature selection. In Table 4.8 we see that, compared to the base feature set, the addition of the absolute engagement was accompanied by a significant increase in OTTNR. After determining the best-performing feature set for the weekly configuration, this feature set was used throughout the rest of the experiments. After changing the evaluation method to daily evaluation and introducing cost-sensitivity, we did not re-evaluate the impact of the engagement anymore. But, we argue that, since the cost-sensitive model only extends the base model with data point weighting, the engagement data will have had a comparable impact in the final model.

The introduced cost-sensitivity was shown to significantly improve the overall OTTNR metric, both for the weekly evaluation showed in subsubsection 4.2.1.4 and the daily evaluation showed in Table 4.11. By selecting the best-performing cost-sensitivity factor R throughout the year, the model was able to adapt to the general trends in the market. Examples of these changes are given in the first section of the online evaluation subsection 4.2.2. An exact evolution of the R factor over time is omitted from the report as it is sensitive information.

In selecting this best-performing cost-sensitivity factor R along with the other model parameters, we select the model configuration that showed the highest OTTNR value whilst maintaining a FAR of below 0.4. In our first offline evaluations in subsection 4.2.1, this model selection was done after the performance of each configuration was known. However, as we argue in subsubsection 4.2.1.5, this is not possible when we deploy the model live as we must base it on the previous performance. As shown in both the offline model selection results in Table 4.12 and the online results shown in Table 4.13, this, unfortunately, does not guarantee that the model will maintain a FAR below 0.4 for future predictions. In general, this threshold will be difficult to enforce given the volatility of the platform. In discussion with the UTURN supervisor, we decided to preserve our current model selection. It would be possible to counteract this by selecting models that achieved past FAR values of less than 0.35, for example. Still, this does not guarantee a FAR of less than 0.4, but it would ensure a FAR further towards the 0.4 limit.

Another challenge in using a decision tree to continuously make predictions throughout the lifetime of shipments was that this required us to fit the decision tree on multiple data points per shipment in the shipment history. This meant that traditional metrics could not be used, as this would cause some shipments to overrepresent the model's performance. To remedy this, we introduced problem-specific alterations of the True Negative Rate and the False Negative Rate. We do realise that, given that there is a variable number of data points per shipment, this causes the decision tree to be more fitted towards shipments that have longer lifetimes. However, we argue that this is not necessarily detrimental to the performance. In fact, we have a higher probability of making a difference for shipments that are on the platform longer, as carriers will have more time to respond to the recommendations.

To reflect this idea in our metrics we defined the OTTNR metric as an extension of the TNR metric as described in subsubsection 3.2.3.1. The TNR metric weighs any cancellation that was predicted as a cancellation throughout its lifetime equally. The OTTNR metric additionally discounts the contribution of a correctly predicted cancellation based on the minutes left until the cancellation occurred. In our online results of the SPM shown in Table 4.14, we see that our model performs comparably to the adjusted 180-minute baseline in terms of OTTNR and FAR. This behaviour was expected; as the R factor represents the minutes-after-posting mark at which cancellations are weighed more heavily than matches in the training data, we are essentially softly enforcing an x-minute baseline.

When we consider the TNR, on the other hand, we see that our model correctly classifies 5% of the cancellations more than the strongest baseline. This is further shown in Figure 4.8, in which we see that there were also a substantial number of shipments that were cancelled before they spent 120 minutes on the platform, for which our model correctly predicted most. It is unfortunate that these shipments are cancelled this early, as this likely means that the carriers did not have enough time to respond. We suspect that, if UTURN would somehow let the shipper know that the shipment was recommended, shippers might leave their shipments on the platform longer to allow carriers to respond to the recommendations.

As a final closing remark to our discussion, we note that the SPM does not take into account the added matching probability induced by the recommendations made by the Matching Algorithm. At the time development began, estimating this probability was not possible due to insufficient recommendation data. However, this could be included in our model in the future. Currently, the SPM decision tree returns binary labels when queried with a shipment data point. With the SKLearn library, it is also possible to return a probabilistic value based on the data distribution in the leaf node associated with the data point. By adding the estimated increased matching probability of a recommendation to this probabilistic value, we can subsequently derive its binary label by checking whether this new probabilistic value is larger than 0.5. In hindsight, we argue that, given our estimated effect of the Matching Algorithm on matching probability, this would not have made a significant difference to our overall SPM results.

6 Conclusion and future directions

This chapter will conclude this thesis report. In this research, we aimed to answer one main question: *How can we take action in a conscious and optimal manner throughout the lifetime of shipments to maximize the overall platform matching rate?*

After familiarizing ourselves with the UTURN platform and the transport market, we narrowed this question down to the following three research questions.

1. How can we maximize the probability of finding a match for the shipments on the platform by sending out recommendations?
    (a) Which factors are important for a carrier when searching for shipments on the UTURN platform?
    (b) How can we model the interests of a single carrier, most likely consisting of a variable number of actors?
2. How can we balance maximizing the matching probability and avoiding unnecessary recommendations or spam?
3. How do we ensure that this solution is long-lived, adapting to changes in the transport market?

In this chapter, we evaluate each of these research questions according to our research and suggest future directions for each part of our research. We end our conclusion with a general conclusion of our work in section 6.4

6.1. How can we maximize the probability of finding a match for the shipments on the platform by sending out recommendations?

When determining the best method for generating recommendations for the UTURN platform, we discovered that, unlike traditional recommendation systems that match users (carriers) with items (shipments), we should focus on matching shipments with the right carriers. Given the lack of prior data on how these recommendations would be received or which carriers would be appropriate to target, we adopted a broad approach designed to accommodate all carriers on the platform.

To achieve this, we developed a recommendation system that ranks carriers based on the similarity of their platform history to a shipment, following the assumption that carriers will likely be interested in taking on shipments similar to those in the past. This was realised by creating personalised carrier profiles for each carrier to represent their interests. Given that some carrier accounts may have multiple transporters working for them, the system needed to accommodate this by allowing for multiple operational areas. To address this, we created personalized k-nearest neighbour (k-NN) models for each carrier, fitted on their platform history (**RQ 1b**).

To determine how similar a shipment was to the history of a carrier, we developed a custom similarity function. This custom similarity function compares the multimodal characteristics of shipments through a weighted function of individual feature similarity functions. Through weight optimization, we demonstrated the relative importance of the various characteristics (**RQ 1a**). By building these carrier profiles, we aimed to create a system that ranks carriers highly in terms of suitability for any shipment in their personal history. We hypothesized that, if we can rank the carrier that quoted on a shipment highly for any shipment in the past, then we will be able to rank the carriers that will quote on any shipment in the future as well.

Our recommendation system was deployed on the UTURN live servers over three months in a controlled experiment. In this controlled experiment, roughly 3000 shipments that were online on the platform for longer than 90 minutes were split into either the control or test group. The results of the experiment showed that the test group that received recommendations showed an increase in the matching percentage of 2.8%, increasing to 3.4% when focusing on the countries in which UTURN is most established with a peak increase of 6.2% in the Netherlands. Next to that, 37.4% of shipments that were recommended were viewed by at least one of the recommended carriers, and 16.4% was quoted. Although these engagement percentages are promising results, our email recommendation interaction data showed that only 36% of these views and 21% of these quotes originated from our email recommendations.

This showed that our first research question is more complex than originally anticipated. Unlike online marketplaces in which items are simply bought, shipments require actual delivery time and therefore must fit the carrier's schedule and capacity. We found there to be a class of carriers on the UTURN platform that comprises the bulk of the platform's traffic: the giants. As the giant carriers on the platform have a substantially larger capacity than smaller carriers, they are much more likely to be able to take on these shipments (and often do). This was reflected in the bias of our recommendation system shown in the offline evaluation - given their capacity and thus, large platform history, many of the shipments for which we generated recommendations found similarity to their carrier profile, resulting in them being highly ranked and thus receiving recommendations. Although, according to our goal, the recommendation system was exactly doing what it was tasked to do, the online evaluation showed that this is not what we should try to achieve. Our evaluation concluded that even though the giant carriers were receiving recommendations and engaging with the recommended shipment through views and quotes the most, this engagement did not originate from the recommendations. This shows the difficulty of our recommendation objective; even though we might be finding the right carriers for the problematic shipments, these carriers may not be the right recipients for recommendations.

We conclude that employing a 'one-size fits all' solution may not be suitable for this problem given the high variation between carriers. A future solution should perform qualitative research on which carriers are open to recommendations and which carriers can benefit from recommendations the most. Additionally, follow-ups on the recommendations can be done to determine why they were or were not suitable for them. This would allow for the narrowing down of the target group or even creating multiple target groups, allowing for the definition of various carrier-size-specific recommendation systems.

Furthermore, if UTURN were to build upon the work described in this report, we suggest continuing with the second iteration of the Matching Algorithm as described in subsubsection 3.1.4.3 and subsubsection 4.1.1.2. Although the evaluation results appeared to have no positive influence on the overall performance of the recommendation system, it poses a more scalable and optimizable solution. While the custom similarity function allowed us to take into account all features while determining the most similar neighbourhood of a carrier, it introduced many dependent variables through the introduction of feature weights and individual feature similarity functions. Additionally, this complex similarity function significantly increased the computational complexity of an offline evaluation, which hindered us from optimizing each feature similarity function separately. If, instead, the carrier k-NNs are only used to find carriers active in the shipment's route, these carriers can then additionally be ranked according to a rule-based recommendation system or by more thoroughly evaluated individual feature similarity functions. Rule-based systems give comprehensible control over the outcome and are suitable for applications where domain knowledge is crucial [1]. This would allow for additional recommendation strategies based on carrier size, estimated carrier schedules and much more.

6.2. How can we balance maximizing the matching probability and avoiding unnecessary recommendations?

Unlike other recommendation systems, where recommendations are typically presented at every opportunity possible, we had to be conscious about which shipments we recommended because of three main reasons. Firstly, the UTURN marketplace is fast-paced; if we simply recommend every shipment, we will be recommending many shipments that have found their match by the time the recipient views the recommendation. Secondly, given that our only method of sending recommendations is through sending emails, we needed to avoid sending out too many emails as this can be considered spam. Finally, each shipment has a deadline; the shipment must find a match before this deadline, otherwise, it disappears from the platform. In reality, shippers cancel their shipments way before this deadline is reached.

This meant that we had to strike a balance between waiting for the shipment to find a match over time and ensuring that we sent recommendations before the shipment left the platform and revenue was lost. To decide which shipments required recommendations, we developed a predictive model that continuously predicts the outcome of a shipment, and triggers the recommendation system once the outcome is predicted as a cancellation. This predictive model, which we call the Shipment Pulse Monitor (SPM) was realised in the form of a decision tree trained on the history of the platform. We represented each shipment with both features based on historical platform data (route matching rate, estimated route capacity) and live features (current views and quotes, time left until deadline) and sampled this data stream throughout the lifetime of each shipment. Since a singular shipment was represented by a multitude of data points depending on its time spent on the platform, regular decision tree metrics could not be used as this would cause some shipments to over-represent the model's performance. Therefore, we defined custom metrics that directly reflect the aforementioned trade-off. First, we defined our objective: the On-Time True Negative Rate (OTTNR), or, the fraction of cancelled shipments that were correctly classified before the cancellation occurred, discounted by how much time there was left for the recommendation to be viewed by the carrier. Second, we defined our trade-off: the False Alarm Rate (FAR), or, the fraction of matches that were, at some point in their lifetime, classified as a cancellation and therefore triggered a 'false alarm'. While our goal was to optimize the OTTNR, we were tasked to adhere to a maximum FAR of 0.4.

Although the metrics reflected our goal, the way the training data was represented was not yet complete. As described before, we need to strike a balance between waiting for the shipment to find its match over time, and ensuring that we sent recommendations before the shipment was cancelled. This was realised by additionally enhancing our decision tree by introducing cost-sensitivity through data point weighting. Each cancellation data point was additionally weighted according to its time spent on the platform, making the prediction of cancellation data points more important over time. This showed a significant boost when compared to the non-cost-sensitive model. By periodically retraining and evaluating, we showed that the model was able to adapt to changes in the market. Additionally, we evaluated multiple parameter selection methods based on the historical performance of the SPM.

The SPM was deployed as a trigger mechanism for the recommendation system over two months. In this period, our model achieved an OTTNR of 0.741, accompanied by a FAR of 0.443. Although we selected model parameters that, in previous evaluations of the SPM, achieved a FAR strictly below 0.4, the volatility in the market caused the model to exceed the 0.4 threshold.

Throughout the evaluation, we compared the SPM to a thresholding-based predictor that classifies each shipment that spent more than x minutes on the platform as a cancellation. This proved to be a surprisingly strong baseline, as we found our model to perform very comparably in terms of our defined metrics. However, when examining the True Negative Rate (undiscounted OTTNR), our model predicts 5% more cancellations than the thresholding-based predictor. Although this may not seem like a large difference, these last few percentages mean a lot as there are also many 'unexplainable' cancellations likely caused by human error. After looking further into this difference, we found that our model works very well in predicting cancellations with a platform lifetime of shorter than 120 minutes, which were completely missed by the thresholding-based predictor. Since we are basing the OTTNR metric on the time between the correct cancellation prediction and the time at which the the cancellation occurs, this metric does not capture the correctly classified predictions on short-lifetime shipments as much. While this reflects the reduced response time for carriers to the recommendations, we believe this could be partially remedied. If UTURN could inform shippers that their shipment has been recommended, shippers might leave their shipments on the platform longer, allowing carriers more time to respond.

Given the surprising strength of our evaluated baselines, we believe that if UTURN continues to employ a Shipment Pulse Monitor, a better solution could involve combining hard cutoff points, such as matching rate thresholds and t-minute thresholds, with the use of a predictive model. This approach would leverage the strengths of the baselines while allowing the predictive model to focus on less clear-cut prediction cases. Finally, our model could be extended by additionally accounting for the estimated moment of cancellation based on a specific shipper's platform history.

6.3. How do we ensure that this solution is long-lived, adapting to changes in the transport market?

In the Matching Algorithm, each carrier profile was based on the past three months of platform history. This accommodated possible changes in both the carrier preferences and carrier schedules. In future developments, periodic weight configuration optimization should be added to the overall Matching Algorithm system. This could further accommodate changes in the market. For example, in supply-driven markets, carriers may become less selective with respect to routes and prices, as they are more eager to find any shipment to take on. In demand-driven markets, on the other hand, carriers have more options and therefore will be more selective, picking shipments that perfectly fit their schedule and price range.

In the evaluation of the SPM, we showed that the model was able to adapt to changes in the market. This was caused by both our periodic retraining and evaluation, and the introduction of cost-sensitivity. With our model parameter configuration selection method, the best-performing configuration in recent history was chosen for each day that maintained an average FAR of below 0.4. By including the cost-sensitivity factor R in the model parameter configuration selection method, we saw evidence of the model adapting to both supply- and demand-driven periods on the platform.

6.4. Overall conclusion

We developed a recommendation system that focuses on matching shipments with the right carriers by creating detailed carrier profiles based on historical platform data. The system showed promising results in increasing the matching rate. In a controlled experiment, the recommendation system increased the average matching percentage by 2.8%. This improvement rose to 3.4% in established regions, with a peak increase of 6.2% in the Netherlands. However, we discovered that a one-size-fits-all approach was insufficient due to the varying capacities of carriers. The Shipment Pulse Monitor (SPM) was introduced to send recommendations only when necessary, effectively balancing the maximization of the matching probability through recommendations while avoiding spam. The SPM demonstrated effectiveness in predicting cancellations and ensuring timely recommendations. Furthermore, the SPM showed adaptability to market changes. The use of cost-sensitivity and dynamic parameter selection allowed the SPM to respond to both supply- and demand-driven market fluctuations. Overall, our findings highlight the importance of tailored, adaptive solutions in achieving optimal platform performance and improving the matching rate.

Future work should focus on qualitative research to tailor recommendations to carriers who would be most receptive to recommendations, rather than solely focusing on finding the carrier that is most likely to be interested in the shipment. As for the SPM, future work should combine hard cut-off points as utilised by our baselines with a predictive model to allow the model to focus on the less clear-cut prediction cases. Next to that, the SPM's accuracy and effectiveness could further be enhanced by integrating shipper behaviour insights. By addressing these areas, the two-part solution can be refined, ensuring long-term adaptability and improved performance in the dynamic freight transport market.

In conclusion, our research provides a comprehensive framework for developing and optimizing a recommendation system for the UTURN platform. By addressing the unique challenges of the freight transport market, our work lays the foundation for future improvements and highlights the importance of tailored, adaptive solutions in maximizing the platform matching rate.

References

[1] Charu C Aggarwal et al. *Recommender systems*. Vol. 1. Springer, 2016.

[2] Fatemeh Alyari and Nima Navimipour. “Recommender systems: A systematic review of the state of the art literature and suggestions for future research”. In: *Kybernetes* 47 (Mar. 2018). DOI: 10.1108/K-06-2017-0196.

[3] Judit Bar-Ilan, Mazlita Mat-Hassan, and Mark Levene. “Methods for comparing rankings of search engine results”. In: *Computer Networks* 50 (July 2006), pp. 1448–1463. DOI: 10.1016/j.comnet.2005.10.020.

[4] Sebastian Becker, Patrick Cheridito, and Arnulf Jentzen. “Deep optimal stopping”. In: *Journal of Machine Learning Research* 20.74 (2019), pp. 1–25.

[5] Shyam Boriah, Varun Chandola, and Vipin Kumar. “Similarity Measures for Categorical Data: A Comparative Evaluation”. In: vol. 30. Apr. 2008, pp. 243–254. DOI: 10.1137/1.9781611972788.22.

[6] Lukas Brozovsky and Vaclav Petricek. “Recommender System for Online Dating Service”. In: *CoRR* abs/cs/0703042 (2007). arXiv: cs/0703042. URL: http://arxiv.org/abs/cs/0703042.

[7] Juliana Castaneda et al. “Optimizing transport logistics under uncertainty with simheuristics: Concepts, review and trends”. In: *Logistics* 6.3 (2022), p. 42.

[8] Barbara Fallon et al. “Opportunities for prevention and intervention with young children: lessons from the Canadian incidence study of reported child abuse and neglect”. In: *Child and adolescent psychiatry and mental health* 7 (2013), pp. 1–13.

[9] Tom Fawcett and Foster Provost. “Activity monitoring: noticing interesting changes in behavior”. In: *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’99. San Diego, California, USA: Association for Computing Machinery, 1999, pp. 53–62. ISBN: 1581131437. DOI: 10.1145/312129.312195. URL: https://doi.org/10.1145/312129.312195.

[10] Alberto Fernández et al. *Learning from imbalanced data sets*. Vol. 10. 2018. Springer, 2018.

[11] Steffen Försch and Evert de Haan. “Targeting online display ads: Choosing their frequency and spacing”. In: *International Journal of Research in Marketing* 35.4 (2018), pp. 661–672. ISSN: 0167-8116. DOI: https://doi.org/10.1016/j.ijresmar.2018.09.002. URL: https://www.sciencedirect.com/science/article/pii/S0167811618300478.

[12] Max Gath. *Optimizing transport logistics processes with multiagent planning and control*. Springer, 2016.

[13] Gregory A Godfrey and Warren B Powell. “An adaptive dynamic programming algorithm for dynamic fleet management, I: Single period travel times”. In: *Transportation Science* 36.1 (2002), pp. 21–39.

[14] Carlos A. Gomez-Uribe and Neil Hunt. “The Netflix Recommender System: Algorithms, Business Value, and Innovation”. In: *ACM Trans. Manage. Inf. Syst.* 6.4 (Dec. 2016). ISSN: 2158-656X. DOI: 10.1145/2843948. URL: https://doi.org/10.1145/2843948.

[15] Anjan Goswami, Fares Hedayati, and Prasant Mohapatra. “Recommendation Systems for Markets with Two Sided Preferences”. In: *2014 13th International Conference on Machine Learning and Applications*. 2014, pp. 282–287. DOI: 10.1109/ICMLA.2014.51.

[16] Trevor Hastie et al. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009.

[17] Li-Yu Hu et al. “The distance function effect on k-nearest neighbor classification for medical datasets”. In: *SpringerPlus* 5 (2016), pp. 1–9.

[18] Bilal Khan et al. “An Empirical Evaluation of Machine Learning Techniques for Chronic Kidney Disease Prophecy”. In: *IEEE Access* 8 (2020), pp. 55012–55022. URL: `https://api.semanticscholar.org/CorpusID:214692651`.

[19] Robert D Kleinberg. “A multiple-choice secretary algorithm with applications to online auctions.” In: *SODA*. Vol. 5. Citeseer. 2005, pp. 630–631.

[20] Xiying Li and Chi-Guhn Lee. “ΔV-learning: An adaptive reinforcement learning algorithm for the optimal stopping problem”. In: *Expert Systems with Applications* 231 (2023), p. 120702. ISSN: 0957-4174. DOI: `https://doi.org/10.1016/j.eswa.2023.120702`. URL: `https://www.sciencedirect.com/science/article/pii/S0957417423012046`.

[21] Yajie Liu et al. “Robust optimization for relief logistics planning under uncertainties in demand and transportation time”. In: *Applied Mathematical Modelling* 55 (2018), pp. 262–280.

[22] Chhavi Maheshwari. “Music Recommendation on Spotify using Deep Learning”. In: *arXiv preprint arXiv:2312.10079* (2023).

[23] Christopher D Manning. *Introduction to information retrieval*. Syngress Publishing, 2008.

[24] Ibomoiye Domor Mienye and Yanxia Sun. “Performance analysis of cost-sensitive learning methods with application to imbalanced medical data”. In: *Informatics in Medicine Unlocked* 25 (2021), p. 100690. ISSN: 2352-9148. DOI: `https://doi.org/10.1016/j.imu.2021.100690`. URL: `https://www.sciencedirect.com/science/article/pii/S235291482100174X`.

[25] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[26] Milos Poliak et al. “Competitiveness of price in international road freight transport”. In: *Journal of Competitiveness* 13.2 (2021), p. 83.

[27] Warren B Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*. Vol. 703. John Wiley & Sons, 2007.

[28] Rijksoverheid. *Gevaarlijke stoffen vervoeren over de weg*. 2023. URL: `https://business.gov.nl/regulation/transport-dangerous-goods-road/`.

[29] Ignacio Rios, Daniela Saban, and Fanyin Zheng. “Improving match rates in dating markets through assortment optimization”. In: *Manufacturing & Service Operations Management* 25.4 (2023), pp. 1304–1323.

[30] John Rust. “Optimal replacement of GMC bus engines: An empirical model of Harold Zurcher”. In: *Econometrica: Journal of the Econometric Society* (1987), pp. 999–1033.

[31] Yusuf Sahin, Serol Bulkan, and Ekrem Duman. “A cost-sensitive decision tree approach for fraud detection”. In: *Expert Systems with Applications* 40.15 (2013), pp. 5916–5923. ISSN: 0957-4174. DOI: `https://doi.org/10.1016/j.eswa.2013.05.021`. URL: `https://www.sciencedirect.com/science/article/pii/S0957417413003072`.

[32] Hugo P Simao et al. “An approximate dynamic programming algorithm for large-scale fleet management: A case application”. In: *Transportation Science* 43.2 (2009), pp. 178–197.

[33] Hugo P Simão et al. “Approximate dynamic programming captures fleet operations for Schneider National”. In: *Interfaces* 40.5 (2010), pp. 342–352.

[34] Brent Smith and Greg Linden. “Two Decades of Recommender Systems at Amazon.com”. In: *IEEE Internet Computing* 21.3 (2017), pp. 12–18. DOI: `10.1109/MIC.2017.72`.

[35] Roghayeh Soleymani, Eric Granger, and Giorgio Fumera. “F-measure curves: A tool to visualize classifier performance under imbalance”. In: *Pattern Recognition* 100 (2020), p. 107146.

[36] Centraal Bureau voor de Statistiek. *Hoeveel Goederen worden er in Nederland vervoerd?* 2023. URL: `https://www.cbs.nl/nl-nl/visualisaties/verkeer-en-vervoer/goederen/transportsector/goederen`.

[37] Nguyen Thai-Nghe, Zeno Gantner, and Lars Schmidt-Thieme. “Cost-sensitive learning methods for imbalanced data”. In: *The 2010 International joint conference on neural networks (IJCNN)*. IEEE. 2010, pp. 1–8.

[38] Kai Ming Ting. “An instance-weighting method to induce cost-sensitive trees”. In: *IEEE Transactions on Knowledge and Data Engineering* 14.3 (2002), pp. 659–665. DOI: 10.1109/TKDE.2002.1000348.

[39] Shahadat Uddin et al. “Comparative performance analysis of K-nearest neighbour (KNN) algorithm and its different variants for disease prediction”. In: *Scientific Reports* 12 (Apr. 2022). DOI: 10.1038/s41598-022-10358-x.

[40] Ni Wayan Surya Wardhani et al. “Cross-validation metrics for evaluating classification performance on imbalanced data”. In: *2019 international conference on computer, control, informatics and its applications (IC3INA)*. IEEE. 2019, pp. 14–18.

[41] Chong Zhang et al. “A cost-sensitive deep belief network for imbalanced classification”. In: *IEEE transactions on neural networks and learning systems* 30.1 (2018), pp. 109–122.

[42] Lei Zhang and David Zhang. “Evolutionary cost-sensitive extreme learning machine”. In: *IEEE transactions on neural networks and learning systems* 28.12 (2016), pp. 3045–3060.

[43] Shuai Zhang, Lina Yao, and Aixin Sun. “Deep Learning based Recommender System: A Survey and New Perspectives”. In: *CoRR* abs/1707.07435 (2017). arXiv: 1707.07435. URL: http://arxiv.org/abs/1707.07435.