Machine Learning-Based Feasibility Checks for Dynamic Time Slot Management

Liana van der Hagen, Niels Agatz
Rotterdam School of Management, Erasmus University

Remy Spliet
Erasmus School of Economics, Erasmus University Rotterdam

Thomas R. Visser, Adrianus L. Kok
ORTEC

Online grocers typically let customers choose a delivery time slot to receive their goods. To ensure a reliable service, the retailer may want to close time slots as capacity fills up. The number of customers that can be served per slot largely depends on the specific order sizes and delivery locations. Conceptually, checking whether it is possible to serve a certain customer in a certain time slot given a set of already accepted customer orders involves solving a vehicle routing problem with time windows. This is challenging in practice as there is little time available and not all relevant information is known in advance. We explore the use of machine learning to support time slot decisions in this context. Our results on realistic instances using a commercial route solver suggest that machine learning can be a promising way to assess the feasibility of customer insertions. On large-scale routing problems it performs better than insertion heuristics.

Key words: time slot management; vehicle routing; supervised machine learning

1. Introduction

Online retail has seen huge growth in recent years (United States Census Bureau 2021), especially in e-grocery (see for instance Ahold Delhaize 2022), and even more so during the COVID-19 pandemic. While convenient for the customer, the home delivery of groceries poses several operational challenges. As grocery products are bulky and temperature sensitive, customers must be home to receive their goods. To prevent missed deliveries, the service provider usually offers a menu of delivery time slots for customers to choose from. Online grocery retailers such as Walmart in the U.S. and Ocado in the U.K. let customers select a one-hour time slot. The use of time slots limits the flexibility of the e-grocer to plan its delivery routes given the available fulfillment capacity. In the short term, the fleet of vehicles and the number of drivers are often fixed.

Online grocery retailers typically determine a set of possible delivery time slot options upfront, e.g. 8:00 - 10:00 on Monday and 10:00 - 12:00 on Tuesday (Agatz et al. 2011). During the booking process, when customers place their orders, the e-grocer can decide to close certain time slot options to manage demand as capacity is reached. That is, the e-grocer aims to only offer those time slots that allow for a timely delivery given the available capacity restrictions. The general booking process for attended home delivery services consists of the following steps.

1. The customer logs in and provides information about the delivery location and order size;
2. The e-grocer determines the set of time slots to offer to this customer;
3. The customer chooses a preferred time slot and places an order, or leaves without placing an order;
4. After the order placement period, at the cut-off time, the e-grocer plans the delivery routes for all customers that have placed an order taking into account their time slot choices.

In this paper, we focus on determining the set of feasible time slots that can be offered to the customer. A time slot is *feasible* for a new customer, if we can find a route schedule after the cut-off time to serve this customer given the already accepted orders and the available fleet capacity. Conceptually, this requires solving a vehicle routing problem with time windows (VRPTW). As such, assessing whether a certain customer-time slot combination is feasible corresponds to determining whether a feasible solution to the associated VRPTW exists. In practice, several complicating elements may need to be taken into account, such as time-dependent travel times, heterogeneous fleets, break-time regulations and multiple depots.

An accurate feasibility check is a core element of many demand management methods in attended home delivery (Agatz et al. 2013). Taking the set of *feasible* delivery slots as input, researchers have developed methods to decide which time slots to offer based on the expected delivery costs, revenues and opportunity costs (Cleophas and Ehmke 2014, Yang and Strauss 2017, Klein et al. 2018). Others have considered incentivizing certain feasible slots by applying dynamic prices or green labels to steer demand (Campbell and Savelsbergh 2006, Yang et al. 2016, Klein et al. 2019, Agatz, Fan, and Stam 2001).

As doing an exact feasibility check for the VRPTW is computationally prohibitive for realistic problem sizes (Savelsbergh 1985), most of the literature in time slot management considers the use of insertion heuristics to perform a rough feasibility check (Köhler, Ehmke, and Campbell 2020, Agatz, Fan, and Stam 2001). The key idea is to evaluate whether it is possible to insert a new customer for a given time slot in a route schedule that serves the already accepted orders. Campbell and Savelsbergh (2005) first propose such an approach in the context of attended home delivery. To increase the likelihood of finding a feasible insertion, they keep track of multiple schedules and try to insert the new customer in each of these schedules in each of the possible time slots.

However, in large-scale attended home delivery systems, there is often not enough time to try to insert a customer in multiple schedules for multiple possible time slots. To ensure a smooth shopping experience, e-grocers strive for almost instant response times (few hundred milliseconds (Strauss, Gülpınar, and Zheng 2021)). It is challenging to perform a fast and high quality feasibility check based on vehicle routing methods for larger route plans. Overall, these methods scale poorly both in terms of problem size and in terms of problem features. Fast insertion heuristics that rely on a single route schedule without intermediate re-optimization typically do not perform well (Visser, Agatz, and Spliet 2019). Additional complicating constraints increase the computation time of a feasibility check, such as time-dependent travel times and work time rules (Visser and Spliet 2020). Moreover, when multiple customers arrive at the booking system over a short time, there is little time between subsequent order arrivals to use precomputation techniques to speed up computations. Note that even geocoding the new customer location and updating the distance matrix can be computationally prohibitive for large instances (Sommer 2014).

While the current academic literature predominantly focuses on VRP methods which rely on constructing a feasible route plan in real-time, we consider an approach that predicts feasibility based on patterns in the instance data, e.g. orders and fleet information. VRP-based methods do not suffer from 'false positives' in a deterministic setting, as a time slot is only offered if a feasible solution to the underlying VRPTW is found. Dropping the assumption that finding a feasible route plan is needed, constitutes a fundamental shift in paradigm. However, as VRP route based methods are inherently too slow to perform detailed real-time checks in practice, we believe it is important to explore alternative approaches. This gives rise to trade-offs between false positives and false negatives. Note that accepting customers without a feasibility guarantee occurs in practice, where it is common to accept customers until a threshold on the total number is reached. Our numerical results show that using prediction based methods can greatly increase the number of customers that are served compared to more traditional feasibility checks, while the number of false positives remain manageably small.

We model the feasibility check as a binary classification problem. This means we can build on a vast stream of literature on classification (Boucheron, Bousquet, and Lugosi 2005) and existing supervised learning methods (Friedman et al. 2009). In this paper, we focus on using machine learning (ML) methods to classify whether a feasible solution exists for the underlying routing problem. As ML methods generally scale well in problem size and are versatile tools for dealing with the large variety of routing problems, they are a valuable contribution to the tool set for time slot management systems in practice. Similar to most academic literature in time slot management, in this paper, we also consider a single-depot deterministic VRPTW as routing context. We also do not include all the uncertainties that create noise on the feasibility check in practice, e.g. stochastic travel and dwell times. This allows a clean comparison with existing time slotting methods.

Our work contributes to the recent stream of literature on the interface between operations research and machine learning (Bengio, Lodi, and Prouvost 2021). The closest to our work are the recent papers by Larsen et al. (2021) and Dumouchelle, Frejinger, and Lodi (2021). They use machine learning to assess the value of accepting a booking request in the context of freight cargo planning. Similar to our work, they use ML methods to approximate the output of an underlying operational optimization problem. However, while these papers primarily focus on forecasting an expected value under data uncertainty, we focus on predicting feasibility in a routing context.

Our contributions can be summarized as follows: (1) we identify and explain the relevant new problem setting of predicting feasibility in the context of time slot management for attended home delivery (2) we model the VRPTW feasibility check as a classification problem and propose a framework to train a ML predictor based on existing state-of-the-art ML methods; and (3) we compare the performance of different ML methods and data features, and benchmark against various alternative methods based on simulation experiments using realistic data. Moreover, we assess the overall performance of a time slot management system that integrates the ML-based feasibility checks.

The remainder of the paper is organized as follows. In Section 2 we describe the time slot management decision problem of the e-grocer and define the feasibility check and associated VRPTW. Then, in Section 3 we present our ML framework. Thereafter, we describe the alternative methods used for benchmarking and the generated data in Sections 4 and 5, respectively. The experiments and corresponding results are discussed in Section 6 and concluding remarks are presented in Section 7.

2. Problem definition

In this section, we describe the operational decision problem of the e-grocer. We first discuss time slot management in the context of the booking process, after which we define the VRPTW related to the feasibility check.

2.1. Operational time slot management

Customers continuously arrive at the website of the e-grocer to shop for groceries and book a delivery time slot. At login, the customer shares a delivery location and grocery basket. A time slot t has an earliest start time a_t and a latest start time b_t. A fulfillment period or shift is associated with a set of time slots $T = \{1, 2, \dots, |T|\}$. Customers can book a delivery time slot for a particular shift until a cut-off time $t_{\text{cut-off}}$. The time between the cut-off and the start of the shift is used to plan the routes and pick and pack the groceries. The delivery routes serve customers across different delivery time slots within the shift. As the planning over the different shifts is independent, we consider a single delivery shift, such that T consists of all time slots of the shift.

After login of the customer, the online grocer needs to decide in real-time which time slots to offer to this specific customer. From that time slot offer, the customer selects a preferred slot or leaves without placing an order. All customers that select a time slot that was offered to them are accepted. The e-grocer, thus, implicitly determines which customers to accept for delivery via the time slot offer. After the cut-off time, the e-grocer determines a set of vehicle routes to serve all accepted customers within their slots, with the available vehicle fleet, while minimizing total route costs. This involves solving a Vehicle Routing Problem with Time Windows (VRPTW). During the booking process, the e-grocer ideally only wants to accept customer requests for which a feasible route schedule can be constructed after the cut-off. In this paper, we consider a setting in which the e-grocer tries to maximize the number of time slots to offer per customer to maximize the service on a first-come-first-serve basis. For that reason, the e-grocer wants to offer the complete set of feasible time slots to each customer.

2.2. Feasibility check

As the e-grocer only wants to accept customers for which a feasible route schedule after cut-off can be found, we must check the feasibility of serving a new arriving customer in each time slot $t \in T$. This feasibility check requires determining if a feasible VRPTW solution can be found for a set of customers $N = \{1, 2, \ldots, n\}$, consisting of the new customer and the already accepted customers at that point in time. Each customer arrival gives rise to $|T|$ VRPTW instances, one for each time slot to check, which we refer to as 'feasibility check instances'. Note that these VRPTW instances differ from the final VRPTW instance after cut-off in the set of customers, which then consists of the final set of accepted customers.

The VRPTW is defined as follows. We need to serve n customer locations from a single depot. The demand of customer $i \in N$ is known and denoted by q_i. Customer i has a time slot $[a_i, b_i]$, specifying the earliest and latest possible start times of service at their location. If the vehicle arrives before a_i at the location, it must wait until a_i to start the service. Additionally, the service time at the customer is denoted by s_i, $\forall i \in N$.

The vehicle fleet available to serve the customers is homogeneous and consists of the vehicles in the set $V = \{1, 2, \ldots, v\}$ with a capacity of Q for each vehicle.

We assume the travel times between all pairs of locations are deterministic and satisfy the triangle inequality. Moreover, we assume that demand of one customer never exceeds the vehicle capacity ($q_i \leq Q$). Each customer location can be reached within their time window if served alone on a vehicle and the opening times of the depot are not a constraining factor.

A route schedule is a set of delivery routes. A feasible solution to the VRPTW consists of a set of feasible delivery routes serving all n customers exactly once with the available vehicles V. A route starts at the depot, visits a subset of the customers in N in a specific sequence, and returns to the depot afterwards. For a feasible route, the total demand of this subset of customers may not exceed Q. Additionally, the service at each customer in the route should start within the specified time slot and the vehicle can only leave the depot between the earliest departure time and the latest arrival time.

Based on the characteristics of the VRPTW, we can make the following observations on feasible and infeasible solutions. These observations provide insights into the characteristics of 'easy' feasibility check instances. These insights can help to speed up the generation of training data and the execution of the feasibility check in real-time during the booking process.

Observation 1: if the total demand of the customers orders N is more than the capacity of the vehicle fleet ($\sum_i^n q_i > Qv$), no feasible solution exists.

This means that for instances where demand sizes are relatively large compared to the vehicle capacity, it is easy to detect capacity infeasible instances.

Observation 2: if there exists a feasible solution for the VRPTW that serves a set customers N given a fixed fleet of v vehicles in which $u \leq v$ vehicles are used, we know there exists a feasible solution for the set of customers N with x vehicles, for $x \geq u$.

This observation implies that the training data can be expanded with several feasible instances by only checking the feasibility for one instance with a VRPTW method.

Observation 3: if the feasible solution that minimizes the number of routes for the VRPTW that serves a set of customers N requires l vehicles, we know that all instances with customer set N and a vehicle fleet of y vehicles, $y \in \{1, \dots, l-1\}$, are infeasible.

This observation implies that the training data can be expanded with several infeasible instances by only checking the feasibility for one instance with a VRPTW method.

Observation 4: if there exists a feasible solution for the VRPTW that serves a set of customers N given a fixed fleet of v vehicles in which $u < v$ vehicles are used, we know there exists a feasible solution when adding a new customer in any of the time slots, i.e., the instance $N \cup \{n+1\}$ is feasible.

This implies that as long as not all available vehicles are used in a solution to the VRPTW serving the already accepted customers, we can always feasibly serve an additional customer in any of the time slots.

3. Machine learning framework

We propose to apply supervised machine learning to 'predict' whether a certain customer can feasibly be served in a certain time slot given the already accepted orders and the total available fleet. Our ML approach consists of several steps. First, we need to obtain observations that can be used to train the model, i.e., VRPTW instances that are labeled as 'feasible' or 'infeasible'. Then, we train several different models and select one. Both of these steps are executed offline. Finally, the selected model can be used to make the time slot decisions in real-time as part of the time slot management system. In the remainder of this section, we explain how to generate observations and we describe the used ML methods.

3.1. Generation of observations

To train the ML model, we generate feasibility check instances and label these as 'feasible' or 'infeasible' by using a VRPTW solver as oracle. The number of possible customer-time slot combinations for feasibility check instances explodes quickly with a reasonable number of customers requesting delivery. To obtain a selection of these combinations, one could randomly sample a set of customers and time slots. However, this would also create instances that would never be encountered in practice. For example 'extremely' infeasible instances, such as instances that would remain infeasible after arbitrarily removing multiple customers. To construct feasibility check instances that are close to those encountered in practice, we run a simulation that emulates the time slot booking process as described in Section 2.1. In this way, the feasibility check instances are constructed sequentially based on the order of arrival of the customers. The general steps of the simulation in this framework are shown in Figure 1. At each customer arrival, T instances are constructed that consist of the accepted orders combined with the new customer-time slot combination. A VRPTW solver labels each of these feasibility check instances which are then stored for training purposes. Only the feasible time slots are offered to the customer, after which the customer selects a time slot and the set of accepted customers is updated with the new customer. In Section 5, we discuss the customer time slot selection and arrival process used in our numerical experiments.

To solve the VRPTW, we can use the same routing solver that is used to obtain the final route schedule after the cut-off to perform the feasibility check in the time slot management simulation and to generate the labels. As the ML model can be trained offline, we have more time than in real-life operations to conduct the feasibility checks and generate the observations. We use the routing solver to determine the labels in order to train the ML model to classify instances as feasible that can also be found by the routing solver after cut-off.

Note that in this way, we train the ML model with training data that was created by a better feasibility check than possible in the short time available in real operations. A simpler insertion heuristic would create more 'false negatives' as it would not always find every solution that can be found by the more sophisticated routing solver. Training on this type of data would limit the potential of the ML model. On the other hand, one would also not want to train on data that was generated by a more sophisticated approach than the routing solver as this would indicate the existence of feasible solutions that could not be found by the routing solver used after cut-off.

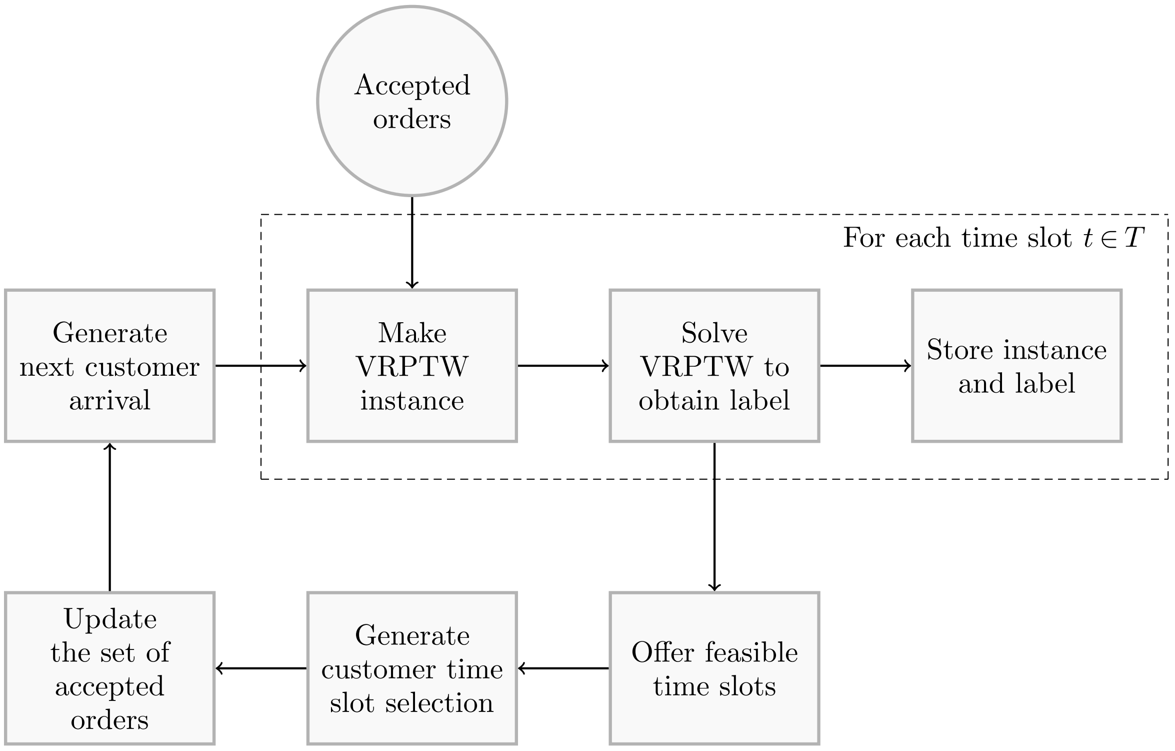

Accepted orders
For each time slot $t \in T$
Generate next customer arrival
Make VRPTW instance
Solve VRPTW to obtain label
Store instance and label
Update the set of accepted orders
Generate customer time slot selection
Offer feasible time slots

Figure 1 Simulation process for instance generation

3.2. ML methods for classification

There is a large stream of literature available on classification and statistical learning (Boucheron, Bousquet, and Lugosi 2005). Caruana and Niculescu-Mizil (2006) present an empirical comparison between seven learning methods for classification: support vector machines, neural networks, decision trees, memory-based learning, bagged trees, boosted trees and boosted stumps. Their results on different binary classification tasks show that random forests, bagged trees and neural networks generally perform well. When calibrating the predictions, the overall best performance is obtained with boosted trees.

A more recent study of Caruana, Karampatziakis, and Yessenalina (2008) compares more learning methods based on high dimensional data. Their work shows that random forests consistently perform well on several binary classification problems, followed by neural networks, boosted trees and support vector machines. This study shows that boosted trees perform very well on lower dimensional problems, whereas neural networks, random forests and support vector machines outperform the other methods for problems with higher dimensions.

We apply random forests, neural networks and gradient boosted trees, as these methods have shown to perform well on a variety of binary classification data sets. The first method that we apply, the random forest algorithm (Breiman 2001), is a general-purpose classification and regression method. The approach combines multiple randomized decision trees and aggregates their predictions by averaging. While random forest combines decision trees in parallel, the trees are combined sequentially with gradient boosted trees (Hastie, Tibshirani, and Friedman 2009). Given the current model, in a forward stage-wise manner, a decision tree is added to minimize a loss function. Finally, neural networks consist of many connected processing nodes organized in different layers (e.g. Bishop et al. 1995). The connections between the nodes pass data through the network to transform certain input to outputs via different weights and threshold operations. These weights and thresholds are calibrated during the learning process as to minimize the number of misclassifications.

These classification methods have also been used in the area of combinatorial optimization (see Bengio, Lodi, and Prouvost 2021, for an extensive overview). Machine learning can, for instance, help to configure existing combinatorial optimization algorithms. Bonami, Lodi, and Zarpellon (2018) apply different classification methods to determine whether or not the quadratic part of a mixed-integer quadratic program should be linearized. The authors show that the trained classifiers are able to improve the runtime compared to the CPLEX default strategy on their generated data.

Bengio, Lodi, and Prouvost (2021) distinguish two ways of using machine learning for combinatorial optimization: imitation learning, and experience learning. Imitation learning is used to approximate decisions, where the expected behavior is shown by an oracle. With experience learning new policies are discovered through experience. In this work, we use machine learning methods for imitation learning where we investigate whether these are able to mimic the routing solver for feasibility classification in the time slot management context.

4. Alternative methods used for benchmarking

We compare the results of our ML approach to some benchmarks based on existing methods for the feasibility check. In particular, we consider the use of an insertion heuristic, order thresholds and continuous approximation.

4.1. Insertion heuristic

Insertion heuristics are commonly used in the time slot management literature, see Waßmuth et al. (2022) for a recent review. The insertion heuristic approach (IH) is to insert a new customer with one of the time slots $t \in T$ into a route schedule that is kept in memory. If this customer can feasibly be inserted in the route schedule, time slot t is offered. This is repeated for all $|T|$ time slots. If the customer selects a preferred time slot from the offer, the route schedule is updated by inserting the new customer with the selected time slot at the cheapest feasible place in the schedule. Note that IH assesses feasibility by incrementally constructing a feasible route schedule in memory. This means that it requires a feasible route schedule visiting the already accepted customers as input. Hence, this approach is different from the other methods discussed in this section and the ML method, which perform the feasibility check based only on the VRPTW input. One could also construct a new feasible route schedule at each customer arrival, but this would increase the computation time.

4.2. Order thresholds

Order thresholds are often used in practice as they are simple and extremely fast, see for instance Campbell and Savelsbergh (2005) and Ehmke and Campbell (2014). We consider two order threshold approaches: an order threshold per shift (OT) and an order threshold per time slot (OTTS). In the OT approach, we assume that accepting a customer in any of the time slots in a shift is feasible as long as the total number of customers (accepted and new) does not exceed the threshold. To handle different vehicle fleet sizes, we set a maximum number of orders γ per vehicle. For a homogeneous fleet of v vehicles this means a new customer for any slot in this shift is feasible as long as the number of customers, n, satisfies: $n \leq \gamma v$.

To better capture the different time slots, OTTS operates with a threshold λ per vehicle and per time slot. This means that accepting a customer in time slot t is feasible if the number of accepted customers in time slot t, n_t, satisfies: $n_t \leq \lambda v$.

The parameters γ and λ are chosen by maximizing the classification accuracy, i.e., the average percentage of correctly classified feasibility check instances on the training data.

4.3. Continuous approximation

Finally, we consider a continuous approximation approach (CA) to benchmark the results. This is based on a stream of literature on cost approximations, first proposed by Daganzo (1987). Similarly as done by Köhler and Haferkamp (2019), we adapt this continuous approximation to our time slot management context. Based on an approximation of the distances between customers, we estimate if there is sufficient time within each time slot t to serve all n_t customers. As in Daganzo (2005), we approximate the distances between customers based on the density of the delivery region. Let A denote the area of the delivery region in km^2. Then the distance between stops in the same time slot t, d_t, is approximated by

$$d_t = \frac{k}{\sqrt{\frac{n_t}{A}}}. \tag{1}$$

Here, k is a constant that is determined by maximizing the average accuracy on the training data. The denominator approximates the demand density.

Let ϕ denote the average driving speed. The total driving time between stops for customers with time slot t is then approximated by

$$\frac{d_t}{\phi} \cdot \frac{n_t}{v}, \tag{2}$$

where we multiply the driving time between stops by the average number of customers per vehicle. Next to the driving time between stops, we also need to account for the service time at the customer locations. Let s denote the average service time at the customer locations. Then the total service time within time slot t is approximated by

$$s \cdot \frac{n_t}{v}. \tag{3}$$

This results in the following feasibility conditions:

$$\frac{d_t}{\phi} \cdot \frac{n_t}{v} + s \cdot \frac{n_t}{v} \leq b_t - a_t \quad \forall t \in T. \tag{4}$$

Note that driving to the first customer location can be done before the start of the time slot. Also, completing the service at the final customer location does not have to be within the time slot, as long as service starts within the time slot. In this approximation we assume consecutive non-overlapping time slots. Then, the driving from the last customer in a time slot to the first customer in the subsequent time slot can be done in one or both of the slots. Similarly, the service of the last customer serviced in a time slot can continue in the following time slot.

So far, the feasibility conditions in Equation 4 only account for time related restrictions. We add an additional feasibility condition to ensure that the total demand does not exceed the total physical vehicle capacity:

$$\sum_{i \in N} q_i \leq Q \cdot v. \tag{5}$$

5. Data generation

We generate feasibility check instances to train and test the ML methods by simulating the booking process as discussed in Section 3.1. In this section, we explain the main characteristics of these instances and the corresponding features.

5.1. Feasibility check instances

We want to generate a diverse set of instances in terms of demand size, clustering, geography and fleet size in order to train the ML method to perform well in a variety of real-world settings. To do so, we generate input for the simulation with different characteristics per shift. In total, we generate 12 different types of instances, where we vary two spatial distributions of the customers, three vehicle fleet sizes and two demand sizes. In general, we consider 400 customers arriving over time on the web-page to place a delivery request for the shift between 16:00 and 22:00. This shift is divided into three non-overlapping time slots of two hours each: 16:00–18:00, 18:00–20:00 and 20:00–22:00. Although we consider a single delivery shift, one could easily consider time slots for multiple shifts at each customer arrival. Note that the capacity is planned per shift and the fleet capacity in one shift is independent on the capacity in other shifts.

For each shift, we draw 400 customer locations within the delivery area. That is, we assume 400 customers request for delivery in a shift. These customer locations are drawn in two ways. In the first type of shift, we draw the customer location coordinates uniform randomly within an ellipse. In the second type of instances we place a 4×4 grid over a circle to generate demand clusters, as illustrated in Figure 2. For a shift with clustered location characteristics, one grid cell is randomly drawn. With probability 0.5, customers are drawn randomly on the intersection of this grid cell and the circle. With probability 0.5, customers are drawn uniform randomly on the entire circle. We consider the radius of this circle and the semi-major axis of the ellipse to be 15 kilometers, and the semi-minor axis of the ellipse to be 9.3 kilometers. We use the center of London (Trafalgar Square) for the depot location at the middle of the delivery area. We choose London as it is a mature e-grocery market with relatively high household and road densities. Figure 3 shows an example of the spread of demand locations for a shift. The driving times are calculated based on a real road network. The geographic coordinates of the customer and depot locations are matched to nodes in the road network.

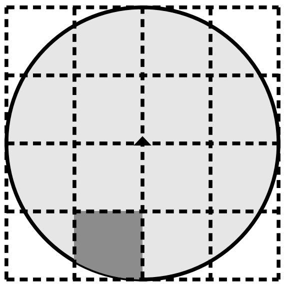
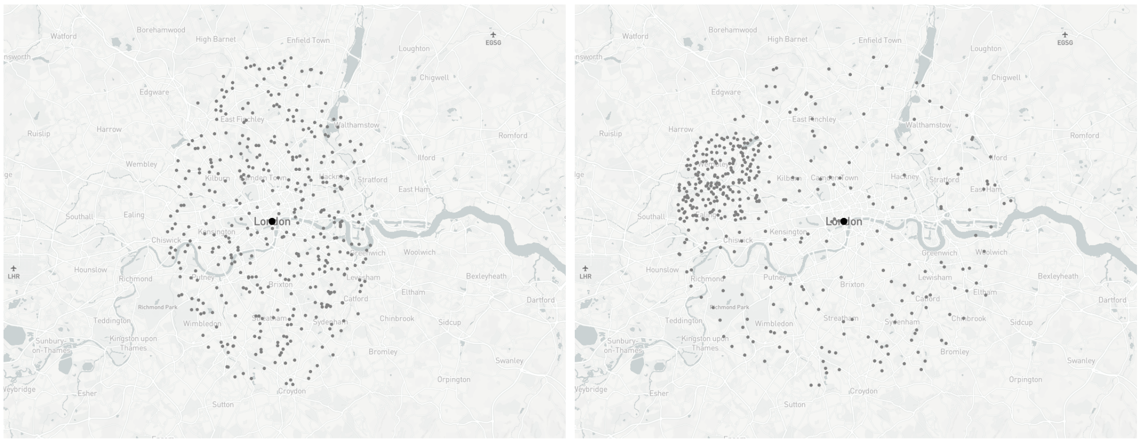


Figure 2 Generation of clustered customer locations

We assume that the retailer has a homogeneous vehicle fleet, with a capacity of 100 units per vehicle. We consider shifts with a vehicle fleet of 4, 10, or 16 vehicles. We generate instances that are more capacity restricted, where each customer has a demand of 6 units, limiting the number of customers per vehicle to 16. We also generate instances that are more time restricted, where demand per customer is 3 units, corresponding to a maximum of 33 customers per route. The service time at a customer location is fixed to 10 minutes.

The time slot preferences of the customers are modeled by an ordered preference list. All customers have all three time slots in their preference list, in uniform random order. If one of the time slots in this list is offered, the customer selects the offered time slot that is highest in the list. Otherwise, the customer leaves and does not place an order. During the simulation, there is



(a) Uniform demand (b) Clustered demand

Figure 3 Uniform and clustered customer locations

sufficient time between the arrival of different customers, such that a customer choice is already processed before the next customer requests a time slot offer.

We denote an instance type by using the following notation: v-locations-q_i. Here v indicates the size of the vehicle fleet, 'locations' indicates 'uniform' or 'clustered' demand and q_i indicates the demand size per customer. The characteristics of these instance types for the training set can be found in Table 1.

An instance set corresponds to one simulation run per instance type, thus, 12 simulation runs. With 400 customers per shift and three possible time slots, this corresponds to 14,400 feasibility check instances per instance set.

v-locations-q_i	% Feasible
All	46.7
4-cluster-3	17.7
10-cluster-3	53.2
16-cluster-3	90.7
4-uniform-3	18.8
10-uniform-3	53.1
16-uniform-3	89.9
4-cluster-6	15.4
10-cluster-6	39.7
16-cluster-6	63.6
4-uniform-6	15.5
10-uniform-6	39.7
16-uniform-6	63.5

Table 1 Instance characteristics on training instances.

5.2. Features

From each feasibility check instance, we extract features as input for the machine learning tool. The first set of features, RAW, corresponds to the raw input data available at the time we need to make a time slot decision. This set is specified as follows. Firstly, it includes five features F: the number of vehicles, capacity per vehicle, depot location latitude, depot location longitude and the fixed service time at the customer in seconds. Furthermore, we include five features C_i per customer i: the customer location latitude, customer location longitude, demand, time slot start time and time slot end time. We define the start and end time as minutes since midnight. To be more precise, to have a fixed-size feature set, we choose an upper bound X on the number of customers and include C_i for all $i \in \{1, \dots, X\}$. For any customer $i \leq n$, C_i is as described above, but for the remaining $n < i \leq X$ we set $C_i = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix}$. If we choose the upper bound on the number of customers as 400, the set includes 2005 features in total.

The number of features in RAW may grow prohibitively large when the number of customers increases, potentially resulting in diminished performance of the ML methods. Therefore, we aggregate the RAW data into two smaller sets of aggregated features: AGR and AGR$^+$. AGR consists of seven general aggregated features of the instances: number of orders, number of orders per time slot, total demand, total vehicle capacity and the number of vehicles. AGR$^+$ expands AGR by including statistics on distance related measures, where we use great-circle distances between the geographic coordinates to speed up calculations. The additional features in AGR$^+$ are the mean, median, minimum, maximum, standard deviation, 1st quartile and 3rd quartile of the distance of customers to the depot and of the distance of customers to the closest customer. AGR$^+$ also contains the mean distance of customers to the closest customer with the same time slot. Note that AGR$^+$ consists of 24 features in total.

6. Numerical results

In this section, we present the results of a set of experiments to evaluate the performance of different ML methods, by comparing these to various common benchmark results. In particular, we try to generate insights into the robustness of the ML methods in different environments.

We implemented our simulation framework in C# (.NET Core 3.1) based on Visser, Agatz, and Spliet (2019). As routing solver, we use the ORTEC Route Optimization service. This is a state-of-the-art commercial vehicle routing solver. The training of the ML models is done in Python 3.9, using scikit-learn 1.0.1 (Pedregosa et al. 2011). Our results are obtained by standardizing the features and using the default parameters from the package for each of the ML methods. For instance, for the Neural Network the default structure consists of one hidden layer with 100 neurons, with the Adam optimizer based on Kingma and Ba (2014) as the default solver and a learning rate of 0.001. We use default parameters to simplify the training process, increase reproducibility and ensure that the methods can be used in a generic way. This means that the ML performance could possibly be improved by tuning the hyperparameters, where good parameter values depend on the type of instances. These trained models are converted to ONNX format with skl2onnx 1.10.2 for them to be used in the simulation, in which the library ML.NET is used to obtain 'online' predictions with the trained model.

We start by comparing the different ML approaches in Section 6.1. In Section 6.2, we then compare the best performing ML method to the insertion heuristic (IH), the order threshold (OT), the order threshold per time slot (OTTS) and the continuous approximation approach (CA). In these sections we focus on comparing the decisions per feasibility check instance and evaluate classification metrics. Subsequently, we perform a time slot management simulation study to compare the overall performance of the booking system using the different feasibility check methods in Section 6.3. This allows us to evaluate the operational consequences associated with the route planning for the delivery shift. Finally, we test the performance on large-scale instances in Section 6.4 and elaborate on how we can inflate the number of training instances in Section 6.5.

6.1. Comparison of ML methods

We start by comparing the performance of the different ML methods; Random Forest (RF), Neural Network (NN) and Gradient Boosted Trees (GB), using the three sets of features. Feasibility check instances are generated by simulating the booking process in which the commercial routing solver performs the feasibility check. We generate 10 instance sets as described in Section 5.1, each set corresponding to feasibility check instances from one simulation run of each type, such that we have in total 120 simulation runs and 144,000 feasibility check instances. We choose to train a model on a diverse set of instances, to mimic the variation encountered in practice.

We apply 5-fold cross-validation to compare the different methods and feature sets. As the feasibility check instances for the same simulation run are related, each fold corresponds to the feasibility check instances from two entire instance sets. That is, in each iteration of the cross-validation, we use 115,200 feasibility check instances, corresponding to 96 simulation runs, for training and 28,800 instances, corresponding to 24 simulation runs, for testing the methods. Table 2 shows the average results on the test sets of the 5-fold cross-validation. We consider the following evaluation metrics: (1) Accuracy (ACC), the average percentage of correctly classified feasibility check instances, (2) True positives (TP), the average percentage of feasibility check instances that are correctly classified as feasible, (3) False positives (FP), the average percentage of feasibility check instances that are incorrectly classified as feasible, (4) False negatives (FN), the average percentage of feasibility check instances that are incorrectly classified as infeasible, (5) True negatives (TN), the average percentage of feasibility check instances that are correctly classified as infeasible.

An e-grocer may want to avoid false positives, because this could correspond to infeasible last-mile operations. Similarly, false negatives correspond to less time slot choices for the customer and potential lost sales. By tuning the probability threshold, we show in Section 6.3 that the ML methods exhibit a trade-off between false positives and negatives. Unless stated otherwise, we use the default probability threshold of 0.5 in our experiments.

Method	Features	ACC	TP	FP	FN	TN
RF	RAW	64.1	46.3	35.5	0.4	17.8
	AGR	93.7	45.8	5.4	0.9	47.9
	AGR$^+$	95.9	45.9	3.3	0.8	49.9
NN	RAW	88.0	45.4	10.7	1.4	42.6
	AGR	**98.0**	45.6	0.9	1.1	52.4
	AGR$^+$	97.3	45.6	1.5	1.2	51.7
GB	RAW	97.5	45.7	1.5	1.0	51.8
	AGR	97.9	45.9	1.3	0.8	52.0
	AGR$^+$	97.9	45.9	1.3	0.8	52.0

Table 2 Cross-validation test performance of the different machine learning methods.

The results show that the ML methods generally perform well. Especially the NN and the GB consistently show high accuracy values and low percentages of false positives. Overall, the NN using the AGR features performs best, correctly classifying 98% of the instances. For this method, the percentage of false positives is less than 1%.

Looking at the different features sets, we generally see that the sparser, aggregated data sets lead to better results. Especially RF performs poorly on the larger dimensional RAW features with an accuracy of only 64.1% and a FP score of 35.5%.

One potential reason for this is that models with more features are prone to overfitting on the training data. Figure 4 shows the average accuracy on the training sets and the test sets for the different ML methods and features. We indeed see that RF provides a high accuracy on the training set compared to the test set. The GB is much less sensitive to overfitting and shows a relatively stable performance for all feature sets across the training and test sets. Although the training accuracy scores for the RAW are comparable with those of the other two feature sets, the RAW models do not generalize well beyond the training data.

We trained these methods on a large number of training instances in each fold, 115,200. We also conducted experiments to assess how much the ML methods benefit from this large amount of training data. In Figure 5 the impact of the amount of training data on the ACC is visible for the different ML methods. We consider a training size of 2, 4, 6 or 8 instance sets, corresponding to 28,800 up to 115,200 feasibility check instances. This shows that especially RAW benefits from

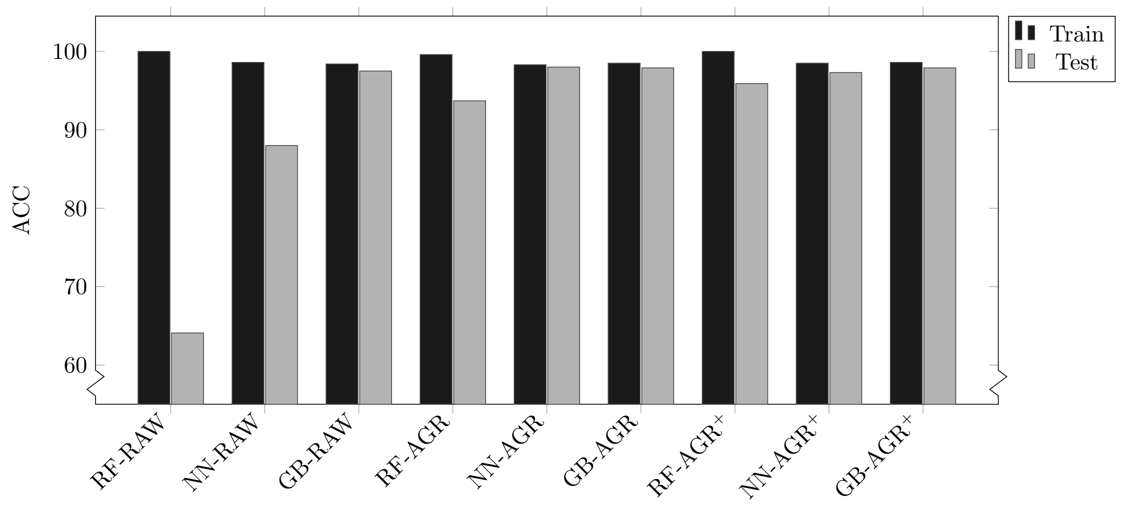


Figure 4 Accuracy on the training data versus the test data

a larger data set. This can be explained by the large number of features in the model. Neural Network with AGR features shows not to improve with a larger training set.

Given that it provides the highest ACC and the lowest FP, we select the Neural Network method with AGR features which we train on the entire training set to continue our experiments. We refer to this approach when we talk about the ML approach in the remainder of this section.

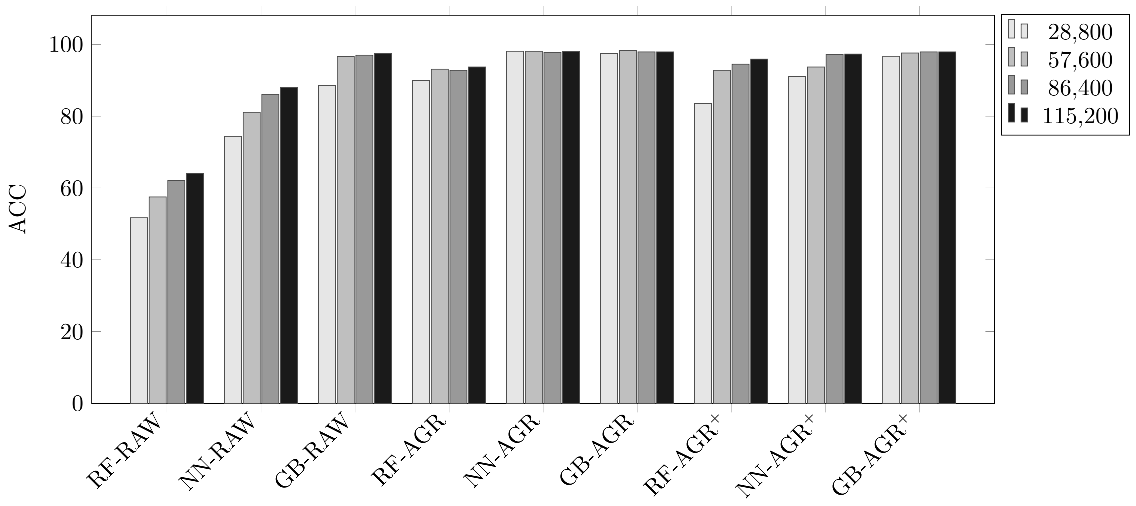


Figure 5 Test accuracy on different numbers of training instances

Table 3 shows the average accuracy per type of instance. This shows that instances with customer demand of 6, for which the number of customers is mostly constrained by the physical vehicle capacity, are more often correctly classified than instances with a demand of 3. For this latter group of instances, the time related restrictions limit the number of accepted customers.

v-locations-q_i	ACC
4-cluster-3	98.2
10-cluster-3	94.4
16-cluster-3	95.7
4-uniform-3	98.3
10-uniform-3	96.9
16-uniform-3	96.4
4-cluster-6	99.4
10-cluster-6	99.5
16-cluster-6	99.3
4-uniform-6	99.4
10-uniform-6	99.5
16-uniform-6	99.4

Table 3 Performance of the machine learning method on each type of instance.

We have trained a single machine learning model on the whole data set consisting of 12 different types of instances. Results indicate that the chosen machine learning method provides good results. We also did an experiment where we trained a specific model on one type of instance, which showed that the ACC of the ML method only slightly improves when using a specifically trained method. This suggests that training a general model provides robust results.

6.2. Comparison with alternative methods

We compare the classification performance of the ML method to the different alternative methods. Similar to how the training instances are generated, as described in Section 3.1, we also sequentially generate feasibility check instances meant for testing, with a simulation of the booking process. We generate these feasibility check instances in two ways. First, we generate instances with a simulation of the booking process in which the routing solver, i.e., the oracle, performs the feasibility check that is used for making the time slot offers during the simulated booking process. This results in one instance set, obtained by running one simulation for each of the 12 shift types. Second, we similarly generate two additional instance sets, but now we use IH instead of the oracle to decide on the time slot offers during the simulated booking process. These two approaches generate different kind of instances. The first approach, using the routing solver during the simulation, yields instances of which 45% are feasible. The second approach, using IH during the simulation, results in instances of which 93% are feasible.

Recall that IH requires a feasible route schedule as input. We explain next what schedule is used when IH is applied to a test instance for our experiments. For the test instances generated with the simulation using IH, we simply store the schedule which is available to IH during the simulation. For the test instances generated with the simulation using the routing solver, we actually also run IH in the background, and store the corresponding schedule. We point out the following important case, in which the route solver offers a time slot to a customer during the simulation, which IH cannot accommodate in its schedule in memory, and the customer accepts this offer. The corresponding instance will be classified as infeasible by IH. More importantly, when the simulation used to generate instances now progresses, IH no longer has a feasible schedule in memory for the set of accepted customers. For these instances, no schedule is stored, and IH consequently classifies them as infeasible.

The parameters γ, λ and ϕ for OT, OTTS and CA are calibrated on the training data, based on the setting that provides the highest accuracy. The values for γ, λ and ϕ are 16, 5 and 2.2, respectively. In our simulation study, we use the real-driving speed on the road network. Based on a sample of planned route schedules from the simulation, we found that the average speed is approximately 32km/hour. This is the average speed we use for our CA approach.

The results for the different methods are shown in Table 4. When the commercial route solver is used to perform the feasibility check in the simulation, ML provides the highest accuracy, 95.6%. The alternative methods show to classify few instances as feasible as compared to the ML method, and we can attribute their lower accuracy to the large percentage of false negative classifications. With OTTS there are even no false positives. Surprisingly, the simplest alternative method, OT, performs reasonably well, outperforming IH, CA and OTTS.

With instances generated by simulating with IH decisions for the time slot offers, IH only correctly classifies 44.3 % of the feasibility check instances, classifying 55.7% of the instances wrongly as infeasible. This also explains the high percentage of feasible instances. IH generates route schedules that can still accommodate more customers, but IH is not able to identify this. OTTS also shows this conservative behavior, performing worse than IH. OT and ML provide the best ACC, and using ML results in lower FP. Note that CA, OT and OTTS perform better if they are calibrated on each type of instance specifically.

Figure 6 shows the ACC dependent on the number of customers in the feasibility check instances for a single instance of type 4-cluster-3 for the simulation with IH time slot offers. With only few customers, all methods are able to correctly classify the instances. Further in the booking process, CA already makes inaccurate decisions with 20-30 customers. IH, ML and OT only make inaccurate decisions with 60-70 customers, where ML provides still the highest accuracy.

6.3. Evaluation of time slot management performance

In the previous sections, we compared the accuracy of different methods with respect to the feasibility check. Now we study how this accuracy impacts the actual performance of the time slot management system when using the different feasibility check methods to make 'online' time slot offers. To do this, we simulate the booking process for each method with two instance sets, with

Simulation		ACC	TP	FP	FN	TN
Routing solver	IH	91.0	36.3	0.0	9.0	54.7
	ML	95.6	44.9	3.9	0.5	50.7
	CA	85.9	33.4	2.1	11.9	52.5
	OT	93.8	39.7	0.5	5.7	54.2
	OTTS	89.7	35.0	0.0	10.3	54.7
IH	IH	44.3	37.3	0.0	55.7	7.0
	ML	87.0	80.2	0.3	12.7	6.8
	CA	46.8	40.0	0.2	53.0	6.9
	OT	82.3	82.1	6.9	10.8	0.2
	OTTS	40.9	33.9	0.0	59.1	7.0

Table 4 Performance of IH, ML, CA, OT and OTTS on feasibility check instances generated by a simulation with time slot decisions by the commercial routing solver or IH.

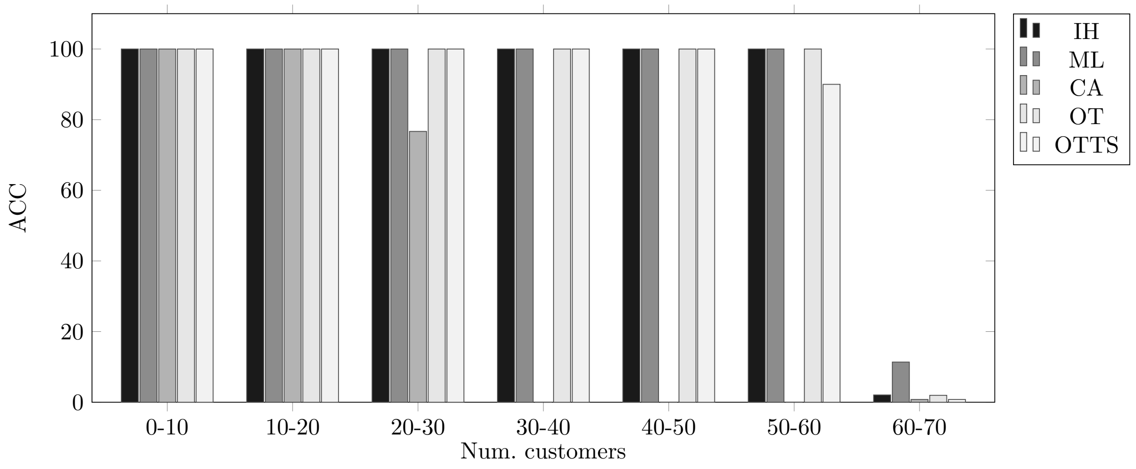


Figure 6 Performance for different number of customers on a single 4-cluster-3 instance with IH time slot decisions in the simulation

a total of 24 simulation runs. After the cut-off time, a final route schedule is constructed with the routing solver.

Table 5 shows the average computation times per customer for the different methods. Next to the overall results, we also provide averages for all instances with less than 50 customers and instances with more than 200 customers. It also shows the 99$^{\text{th}}$ percentile of the computation time to evaluate the larger response times. For IH, the average calculation time is 294 milliseconds, but 1% of the time slot offer calculations take more than 4 seconds. As expected, the computation times for IH increase with the size of the problem instance. For instance, with less than 50 accepted customers the average calculation time is 192 milliseconds, whereas with over 200 accepted customers this is 381 milliseconds. Note that a large part of this time is spend on calculating travel times on the road network. Due to the dynamic nature of our problem, these must be calculated in real-time for every new customer when using the insertion heuristic. The other methods do not need these detailed travel times so they can be much faster. The average calculation time of the time slot offers is < 1 millisecond for CA, OT and OTTS. ML provides stable calculation times of 21 milliseconds.

	Average			99th percentile
	All	$n < 50$	$n > 200$	
IH	294	192	381	4031
ML	20	21	21	83
CA	0	0	0	1
OT	0	0	0	0
OTTS	0	0	0	0

Table 5 Summary on computation time (in milliseconds) of the different methods.

An inaccurate assessment of the feasibility of serving a customer in a specific time slot can have different negatives consequences. False negatives could lead to lost revenue and inefficient use of the vehicle fleet, while false positives could lead to unplanned orders. We consider the following performance metrics: (1) Accepted: the average number of accepted customer orders, (2) Planned: the average number of planned customer orders and (3) Unplanned: the average number of accepted customer orders that could not be served in the final route schedule. If there are any unplanned customer orders in the final route schedule, e-grocers may deal with this in various ways, e.g. making use of emergency capacity, or simply arriving late.

Table 6 shows the results for the IH, ML, CA, OT and OTTS methods stratified for the different fleet sizes. On the one hand, we see that the more accurate assessment of ML leads to more accepted orders but also some unplanned orders. On the other hand, IH and OTTS accept less orders but do not have unplanned orders. Looking at the number of planned orders we see that ML outperforms all other methods. If for some delivery shifts not all orders can be served in the final route schedule, the e-grocer might have to cancel these orders or increase capacity.

To reduce the number of false positives (and thus unplanned orders), we can be more conservative in our predictions. That is, we can increase the required probability threshold to classify an instance as feasible. By default this threshold equals 0.5. Table 7 shows the ML results for different thresholds, δ. Up to the threshold of 0.999 the number of accepted customers is larger with the ML than with IH. However, even though the average number of unplanned customer orders reduces significantly, it does not reach zero. With a threshold of 0.99999 there is no simulation run for which the final route plan contains unplanned orders, at the expense of the number of accepted customers. Note that the number of unplanned orders increases when increasing the threshold from 0.99 to 0.995. This is due to the fact the route solver is a heuristic, where the number of unplanned orders may increase in the final route schedule when having a subset of the accepted customers.

Fleet size	Method	Accepted	Planned	Unplanned
All	IH	156.6	156.6	0.0
	ML	185.7	182.3	3.4
	CA	143.0	142.1	0.9
	OT	160.0	158.7	1.3
	OTTS	150.0	150.0	0.0
4	IH	63.4	63.4	0.0
	ML	65.4	63.9	1.5
	CA	33.0	33.0	0.0
	OT	64.0	63.1	0.9
	OTTS	60.0	60.0	0.0
10	IH	155.0	155.0	0.0
	ML	183.4	178.9	4.5
	CA	135.0	135.0	0.0
	OT	160.0	157.9	2.1
	OTTS	150.0	150.0	0.0
16	IH	251.5	251.5	0.0
	ML	308.3	304.0	4.3
	CA	261.0	258.3	2.8
	OT	256.0	255.0	1.0
	OTTS	240.0	240.0	0.0

Table 6 Results of the final optimization for each of the methods used to make time slot decisions in the simulation.

δ	Accepted	Planned	Unplanned
0.5	185.67	182.25	3.42
0.6	184.38	181.21	3.17
0.7	183.29	180.63	2.67
0.8	181.46	179.50	1.96
0.9	178.88	177.63	1.25
0.95	176.63	175.54	1.08
0.99	171.08	170.83	0.25
0.995	168.58	167.83	0.75
0.999	162.71	162.54	0.17
0.9999	156.00	155.96	0.04
0.99995	140.00	139.96	0.04
0.99999	137.71	137.71	0.00

Table 7 Average results of the final optimization for ML with different feasibility thresholds δ.

Another strategy to reduce the number of unplanned customers is to reserve some vehicles as backup vehicles that are not known to the time slot management system during the booking process. These vehicles are only included for the final optimization after cut-off. To test the impact of this strategy for the ML approach we ran two experiments in which we hold out one or two backup vehicles.

Table 8 shows the results for all instances together and also separated per fleet size. The total fleet size remains the same as before so that a backup vehicle means there is one less vehicle available for the system during the booking period. For comparison, we also show the results for IH and ML without holding out vehicles. As expected, the average number of accepted orders decreases when holding out vehicles, together with the number of unplanned orders. We see that for the largest instances (16 vehicles), we can eliminate the number of unplanned orders while still accepting more orders than IH. However, for the smaller instances, we see a sharp decline in the number of accepted orders. In these cases, IH outperforms the ML model. Note that a-priori, we do not know the minimum number of hold-out vehicles needed in a particular situation.

We see two potential reasons for the poor performance of the backup vehicle strategy for the smaller instances. First, there is less planning flexibility and possibilities for resource pooling in the smaller routing instances. Second, our ML model was not specifically trained on instances with a fleet size smaller than four. For example, the '2 hold-out' case only leaves two vehicles for the ML to work with and this leads to zero accepted orders. The results suggest that the ML model classification accuracy deteriorates for these small fleet sizes. In the next two sections, we will focus on these issues in more detail.

Fleet size	Method	Accepted	Planned	Unplanned
All	IH	156.6	156.6	0.0
	ML	185.7	182.3	3.4
	ML-1 hold-out	162.1	161.8	0.3
	ML-2 hold-out	136.0	136.0	0.0
4	IH	63.4	63.4	0.0
	ML	65.4	63.9	1.5
	ML-1 hold-out	36.6	36.6	0.0
	ML-2 hold-out	0.0	0.0	0.0
10	IH	155.0	155.0	0.0
	ML	183.4	178.9	4.5
	ML-1 hold-out	161.3	161.1	0.1
	ML-2 hold-out	139.6	139.6	0.0
16	IH	251.5	251.5	0.0
	ML	308.3	304.0	4.3
	ML-1 hold-out	288.4	287.5	0.9
	ML-2 hold-out	268.4	268.4	0.0

Table 8 Average results of the final optimization for ML with different number of hold-out vehicles and IH, per type of vehicle fleet.

6.4. Large-scale routing instances

As computation times for vehicle routing methods are prohibitive for larger vehicle routing instances, it is especially relevant to assess the performance of our ML method on large-scale time slot management systems. Therefore, we perform an additional set of experiments with 4000 arriving customers and a vehicle fleet of 130 vehicles.

We have trained our ML model on feasibility check instances with different fleet sizes, i.e., 4, 10 or 16 vehicles. In this experiment, we will not retrain our model but will evaluate how well the ML model performs on fleet sizes beyond the trained sizes. Generating large-scale training instances is very time consuming. We generate five shifts with a uniform spatial distribution and a demand of 3 units as described in Section 5.1. Similar to Section 6.3, we perform a simulation study with each of the feasibility check methods to evaluate the performance on the system.

Table 9 shows the computation times per customer for the different methods. As expected, we see that the IH method does not scale well for larger instances. With 725 milliseconds on average, the method needs more than the 500 milliseconds response time that is often required in practice. The 1% of highest response times for IH even require more than 5 seconds. On the other hand, the ML is still extremely fast and is able to perform the feasibility check in less than 93 milliseconds for 99% of the customers.

	Average			99^{th} percentile
	All	$n < 50$	$n > 200$	
IH	725	243	747	5492
ML	27	37	27	93
CA	2	0	2	6
OT	0	0	0	0
OTTS	0	0	0	0

Table 9 Summary on computation time (in milliseconds) of the different methods on large instances.

	Accepted	Planned	Unplanned
IH	2264.6	2264.6	0.0
ML	3018.6	3018.6	0.0
CA	3534.0	3497.4	36.6
OT	2080.0	2080.0	0.0
OTTS	1950.0	1950.0	0.0

Table 10 Average results of the final optimization for each of the methods used to make time slot decisions in the simulation for large instances.

Table 10 shows the results for these large-scale instances. Surprisingly, we see that the CA method accepts the highest number of customers. However, this is associated with a relatively large number of unplanned orders. Again, we see that the ML performs very well. It accepts 33.4% more customers than IH without any unplanned orders. Moreover, it also performs much better than the simple order threshold methods. The results suggest that the ML model is robust and can successfully be applied to very large instances. It is promising to see that it remains effective when applied to instances that are much larger than those used for training.

6.5. Inflating the number of training instances

Generating training instances to train our ML model can be computationally costly as it requires solving multiple VRPTW instances. However, the observations as discussed in Section 2.2 can help us to inflate the number of training instances without simulating the booking process calling our commercial route solver.

In Section 6.3, we have seen that the trained ML model does not perform well on instances with two or three vehicles. This might indicate that additional training is required to ensure that the ML model can generalize to smaller fleet sizes. Therefore, we focus on enlarging the data set with feasibility check instances with a fleet size smaller than four.

We can use Observation 2 and Observation 3 to generate more labeled feasibility check instances. Let the number of used routes for a VRPTW solution of a feasibility check instance equals l. Then, we can copy the feasibility check instance and modify the vehicle fleet to obtain new labeled instances. We generate additional feasible instances with x vehicles if $l < v$, $\forall x \in \{l, \dots, v-1\}$. Furthermore, we generate additional infeasible instances with y vehicles if $l > 1$, $\forall y \in \{1, \dots, l-1\}$. We apply these generation rules on feasibility check instances with $v = 4$ vehicles. Again, we use the NN with the AGR features on the original set of training instances with $v = 4$, $v = 10$ and $v = 16$ vehicles plus the additional instances with a fleet size of $1, 2$ or 3 vehicles.
We compare the performance of this new trained model, which we refer to as ML*, to that of the original ML model. In Table 11 it can be seen that the accuracy on the test data generated with IH time slot decisions improves significantly for ML*. The percentage of instances wrongly classified as infeasible reduces, but at the cost of an increasing percentage of instances that is wrongly classified as feasible.

Table 12 shows the overall performance of the booking system. We see a large improvement for the smaller instance (fleet size 4). The additional training data with 1, 2 and 3 vehicles results in a significant increase in the number of accepted and planned customer orders for the cases with backup vehicles. Overall, using a single backup vehicle for all instances, on average results in more accepted and planned customers than with IH, while having very few unplanned orders (0.1).

	ACC	TP	FP	FN	TN
ML	87.0	80.2	0.3	12.7	6.8
ML*	93.0	87.1	1.2	5.8	5.9

Table 11 Performance of ML and ML* on feasibility check instances generated by a simulation with IH time slot decisions.

Fleet size	Method	Accepted	Planned	Unplanned
All	IH	156.6	156.6	0.0
	ML*	183.5	180.6	2.9
	ML*-1 hold-out	164.5	164.3	0.1
	ML*-2 hold-out	144.8	144.8	0.0
4	IH	63.4	63.4	0.0
	ML*	65.8	64.6	1.1
	ML*-1 hold-out	48.8	48.8	0.0
	ML*-2 hold-out	30.9	30.9	0.0
10	IH	155.0	155.0	0.0
	ML*	181.0	177.3	3.8
	ML*-1 hold-out	161.0	160.8	0.3
	ML*-2 hold-out	139.4	139.4	0.0
16	IH	251.5	251.5	0.0
	ML*	303.9	300.0	3.9
	ML*-1 hold-out	283.6	283.5	0.1
	ML*-2 hold-out	264.0	264.0	0.0

Table 12 Average results of the final optimization for ML* with different number of hold out vehicles and IH, per type of vehicle fleet.

7. Conclusions and outlook

This paper introduces a new planning setting in the context of time slots management for attended home delivery. In particular, we consider a booking system in which we need to continuously make operational time slotting decisions with little time between consecutive decisions. We model the problem of evaluating whether we can still feasibly serve a certain new customer in a certain delivery time slot given the already placed orders as a classification problem. We propose a framework to train and test state-of-the-art ML methods for this challenging problem.

Our computational experiments on realistic data show that ML is a promising approach for the feasibility check. The results show a higher accuracy than several state-of-the-art methods that are commonly used in practice or proposed in the literature. Our numerical experiments suggest that the number of instances incorrectly classified as infeasible of traditional methods can be very large, while the machine learning methods perform much better. This results in a better utilization of capacity, serving many more customers. Also, the number of instances incorrectly classified as feasible by the machine learning method is not very large. Moreover, our results suggest that we can extrapolate the use of the model to contexts associated with larger routing problems (thousands of customers) than those used to train our methods. With the low computation times, ML is better able to scale in terms of problem size than an insertion heuristic.

With the proposed training framework we present different ways to obtain useful training instances and corresponding labels for the supervised machine learning methods. We also provide insight in how to inflate the number of training instances.

As this is first paper that uses ML in the context of time slot management for attended home delivery, we see many opportunities for future research. To combine the strengths of ML and optimization, it would be interesting to see how to best combine the two methods in a time slot management system. One could for example only use ML if a simple insertion heuristic does not find a feasible solution.

Visser, Agatz, and Spliet (2019) use re-optimizations of the in-memory solution to improve the quality of the feasibility checks done by an insertion heuristic. It might be interesting to see how these improvements measure against the ML approach.

In terms of the ML methods, it may be interesting to see if some of the spatial features of the routing solutions can be included by creating maps with customers and using those images to train the ML classifier.

Also, we have looked at a deterministic setting. However, it may be interesting to see how to incorporate uncertainty in a probabilistic feasibility check. Ehmke and Campbell (2014), Ehmke, Campbell, and Urban (2015), for example, have considered uncertain travel and service times.

Another intuitive extension to go beyond a simple feasibility check and also consider other factors in determining whether or not to open a certain time slot to a customer, such as the costs associated with the different time slot options.

Acknowledgments
This research was funded by the Netherlands Organization for Scientific Research (NWO) under the City Logistics Living Lab (CILOLAB) project (439.18.424).

References

Agatz N, Campbell A, Fleischmann M, Savelsbergh M, 2011 *Time slot management in attended home delivery. Transportation Science* 45(3):435–449.

Agatz N, Campbell AM, Fleischmann M, Van Nunen J, Savelsbergh M, 2013 *Revenue management opportunities for internet retailers. Journal of Revenue and Pricing Management* 12(2):128–138.

Agatz N, Fan Y, Stam D, 2001 *The impact of green labels on time slot choice and operational sustainability. Production and Operations Management* .

Ahold Delhaize, 2022 *Q1 2022 Interim Report.* `https://media.aholddelhaize.com/media/fbtpqr3y/ahold-delhaize-q1-2022-interim-report.pdf`, Accessed June 21, 2022.

Bengio Y, Lodi A, Prouvost A, 2021 *Machine learning for combinatorial optimization: a methodological tour d'horizon. European Journal of Operational Research* 290(2):405–421.

Bishop CM, et al., 1995 *Neural networks for pattern recognition* (Oxford university press).

Bonami P, Lodi A, Zarpellon G, 2018 *Learning a classification of mixed-integer quadratic programming problems. International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, 595–604 (Springer).

Boucheron S, Bousquet O, Lugosi G, 2005 *Theory of classification: A survey of some recent advances. ESAIM: probability and statistics* 9:323–375.

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

Campbell AM, Savelsbergh M, 2006 *Incentive schemes for attended home delivery services. Transportation science* 40(3):327–341.

Campbell AM, Savelsbergh MWP, 2005 *Decision support for consumer direct grocery initiatives. Transportation Science* 39(3):313–327.

Caruana R, Karampatziakis N, Yessenalina A, 2008 *An empirical evaluation of supervised learning in high dimensions. Proceedings of the 25th international conference on Machine learning*, 96–103.

Caruana R, Niculescu-Mizil A, 2006 *An empirical comparison of supervised learning algorithms. Proceedings of the 23rd international conference on Machine learning*, 161–168.

Cleophas C, Ehmke JF, 2014 *When are deliveries profitable? Business & Information Systems Engineering* 6(3):153–163.

Daganzo C, 2005 *Logistics systems analysis* (Springer Science & Business Media).

Daganzo CF, 1987 *Modeling distribution problems with time windows: Part i. Transportation Science* 21(3):171–179.

Dumouchelle J, Frejinger E, Lodi A, 2021 *Can machine learning help in solving cargo capacity management booking control problems? arXiv preprint arXiv:2102.00092* .

Ehmke JF, Campbell AM, 2014 *Customer acceptance mechanisms for home deliveries in metropolitan areas. European Journal of Operational Research* 233(1):193–207.

Ehmke JF, Campbell AM, Urban TL, 2015 *Ensuring service levels in routing problems with time windows and stochastic travel times. European Journal of Operational Research* 240(2):539–550.

Friedman J, Hastie T, Tibshirani R, et al., 2009 *The elements of statistical learning*, volume 2 (Springer series in statistics New York).

Hastie T, Tibshirani R, Friedman J, 2009 *Elements of statistical learning, ed. 2, springer. New York* .

Kingma DP, Ba J, 2014 *Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980* .

Klein R, Mackert J, Neugebauer M, Steinhardt C, 2018 *A model-based approximation of opportunity cost for dynamic pricing in attended home delivery. OR Spectrum* 40(4):969–996.

Klein R, Neugebauer M, Ratkovitch D, Steinhardt C, 2019 *Differentiated time slot pricing under routing considerations in attended home delivery. Transportation Science* 53(1):236–255.

Köhler C, Ehmke JF, Campbell AM, 2020 *Flexible time window management for attended home deliveries. Omega* 91:102023.

Köhler C, Haferkamp J, 2019 *Evaluation of delivery cost approximation for attended home deliveries. Transportation Research Procedia* 37:67–74.

Larsen E, Lachapelle S, Bengio Y, Frejinger E, Lacoste-Julien S, Lodi A, 2021 *Predicting tactical solutions to operational planning problems under imperfect information. INFORMS Journal on Computing* .

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E, 2011 *Scikit-learn: Machine learning in Python. Journal of Machine Learning Research* 12:2825–2830.

Savelsbergh MW, 1985 *Local search in routing problems with time windows. Annals of Operations research* 4(1):285–305.

Sommer C, 2014 *Shortest-path queries in static networks. ACM Computing Surveys (CSUR)* 46(4):1–31.

Strauss A, Gülpınar N, Zheng Y, 2021 *Dynamic pricing of flexible time slots for attended home delivery. European Journal of Operational Research* 294(3):1022–1041.

United States Census Bureau, 2021 *QUARTERLY RETAIL E-COMMERCE SALES 3rd QUARTER 2021.* `https://www.census.gov/retail/index.html`, Accessed January 4, 2022.

Visser T, Agatz N, Spliet R, 2019 *Simultaneous customer interaction in online booking systems for attended home delivery. ERIM Report Series Reference Forthcoming* .

Visser TR, Spliet R, 2020 *Efficient move evaluations for time-dependent vehicle routing problems. Transportation science* 54(4):1091–1112.

Waßmuth K, Köhler C, Agatz N, Fleischmann M, 2022 *Demand management for attended home delivery–a literature review. ERIM Report Series Reference Forthcoming* .

Yang X, Strauss AK, 2017 *An approximate dynamic programming approach to attended home delivery management. European Journal of Operational Research* 263(3):935–945.

Yang X, Strauss AK, Currie CS, Eglese R, 2016 *Choice-based demand management and vehicle routing in e-fulfillment. Transportation science* 50(2):473–488.