universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Machine Learning-Based Feasibility and Profitability Checks for Time Slot Management in Attended Home Delivery

verfasst von | submitted by

Thi Thuy Trang Le

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2024

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 977
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Business Analytics
Betreut von | Supervisor:	Univ.-Prof. Dr. Jan Fabian Ehmke

Abstract

Ziel dieser Masterarbeit ist es, eine Machbarkeits- und Rentabilitätsprüfung für das Angebot von Zeitfenstern für die Zustellung am nächsten Tag im Hauszustellungssystem vorzustellen. Ähnlich wie die meiste akademische Literatur zum Thema Zeitfenstermanagement betrachten wir auch in diesem Beitrag ein deterministisches VRPTW mit einem Depot als Routing-Kontext. Wir konzentrieren uns auf den Einsatz von Methoden des maschinellen Lernens (ML), d.h. überwachte Ansätze, um festzustellen, ob eine machbare und profitable Lösung existiert, wenn die Nachfrage eines Kunden in einen aktuellen Routingplan aufgenommen wird. Dabei werden Informationen über die Kosten der bestellten Artikel, Lieferkosten, Fahrerkosten und den Nettogewinn des Artikels berücksichtigt. Verschiedene Netzwerkarchitekturen, z. B. mehrschichtige Perzeptronen [1] und rekurrente neuronale Netze [2], oder klassische Klassifizierungsmethoden, z. B. Entscheidungsbaum Klassifizierer [3], Random-Forest-Klassifizierer [4], usw., können verwendet werden, um die Leistung des vorgeschlagenen ML-basierten Ansatzes anhand von Simulationen und realen Daten zu analysieren. Andere Methoden aus der Literatur können ebenfalls in Betracht gezogen werden. Klassische Methoden wie Einfügungsheuristiken, kontinuierliche Approximation und Ordnungsschwellenwertverfahren können für die Durchführung des Benchmarking-Ergebnisses unter Verwendung der vorgeschlagenen Methoden in Betracht gezogen werden.

Contents

1 Introduction

The focus of the thesis is introduced and motivated in this chapter. Subsequently, a literature review is given to clarify the research gap and to introduce the research direction.

1.1 Motivation

E-commerce has shown tremendous growth in recent decades [5, 6]. During the COVID-19 pandemic, this growth has been accelerated more than our expectation [7,8].

E-commerce often involves a process that lets customers choose a delivery time slot to receive their goods. This process is called Attended Home Deliveries with Time Slotting (AHD w. TS). Note that AHD is often required due to security reasons, such as high-value goods, perishable goods, and physically large goods, e.g., household appliances. This implementation provides a high level of customer service and avoids costly delivery failures [9]. If the delivery fails due to either the customers (not available to receive the orders) or the logistic department (not offering the right time slots), goods must be offered for delivery at another time, incurring additional storage, transportation, and scheduling costs. Additionally, for perishable goods, the cost of a delivery failure is even higher, as the goods may be damaged before the next delivery opportunity. Early studies have shown that the cost of AHD is often double the cost of unattended delivery [10]. However, to attract more customers and provide better services, AHD with time slot service has become popular among e-commercial companies, e.g., Amazon [11], PostNord [12].

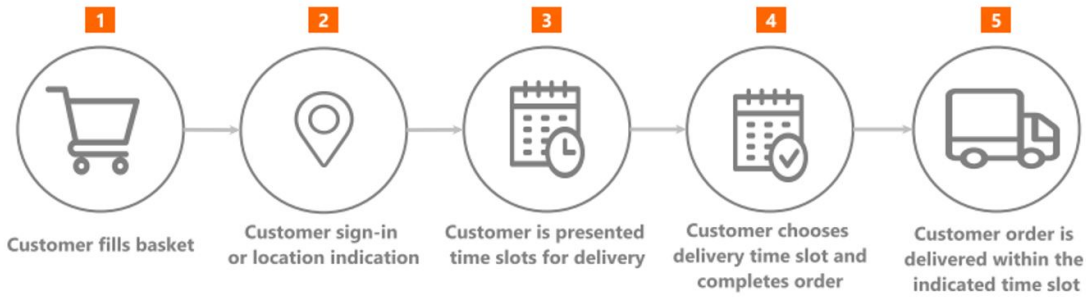

Figure 1: Customer's perspective of the ordering process [13].

The ordering process, from a customer viewpoint, is illustrated in Fig.1, consisting of five stages. In the first stage, a customer fills a basket with desired items. Then this customer's location is received via the sign-in process in the second stage. After signing, the computing engine of the website has to offer feasible time slots to the customer in the third stage. To this end, the time between the second stage and the third stage is called the offering time Δ_O. Note that a time slot for a new customer is feasible if there is a feasible route to serve that customer given the orders already accepted and the available fleet capacity. Generally, this requires solving a vehicle routing problem with time windows (VRPTW) known as NP-hard problem [14], i.e., the optimal solution to this problem cannot be computed in polynomial time. Thus, several heuristic approaches can speed up the computation process to find sub-optimal solutions. Additionally, since

each $100\,\mathrm{ms}$ delay in the load time of websites can decrease sales conversion by 7% [15], the offering time Δ_O, spent checking feasibility, is an important factor.

Note that solving optimally VRPTW with a medium size of customers (over 100 customers) is still a challenge for modern computers. However, the feasibility check, e.g., [16], [17], for several pre-defined time slots, which does not require solving explicitly VRPTW, can be an alternative solution to find a feasible set of time slots within a short offering time. The state-of-the-art will be discussed in the next section, including the research gaps and the importance of investigating the research questions.

1.2 Literature review

Combinatorial optimization problems (COPs) have immense practical applications, e.g., industrial manufacturing [18–20], planning in logistics [21–23] and power grid [24], and systems biology [25]. One of the most famous COPs, the Travelling Salesman Problem (TSP) contains the following question: "Given a list of cities and the distances between each pair of cities, which is the shortest possible route that visits each city and returns to the origin city?". In real-world and practical scenarios, the original TSP can involve challenging constraints such as time windows constraints, i.e., the vehicle routing problem with time windows (VRPTW) [26, 27].

Attended Home Delivery (AHD) [9] is also a variant of the TSP. The AHD, also named last-mile operations supporting, is currently the leading business model in the e-commercial sector. In AHD, the delivery service is typically characterized by time windows and at the selected location of the customer. Additionally, the presence of customers is required during this delivery time. Determining the set of feasible time slots that can be offered to the customer in real time is one of the main challenges of the AHD. Given the accepted orders with corresponding customers and the available fleet capacity, a time slot is determined when there is an available route that can be integrated into the current delivery plan. This process involves solving multiple VRPTW problems. Since a VRPTW problem is NP-Hard which cannot be solved optimally at a large scale, a feasibility check is a core element of providing a feasible set of time slots in AHD, see [28]. Other methods, which steer the demand of customers by applying dynamic prices taking into account incentivizing certainly feasible slots, are proposed in [29], [30], and [31]. Cleophas et al. [32] proposed a method to decide which time slots to offer based on the expected delivery, revenue, and opportunity cost.

Since obtaining the exact *optimal* solution for the VRPTW is impossible for medium size of customers, insertion heuristics is an alternative method to quickly compute *feasible* solutions, see, e.g., [17], [9]. This method evaluates whether it is possible to insert a new customer's demand for a given time slot in an already computed route that serves the existing customers. However, in a large-scale AHD system, the computation time becomes a problem when trying to insert a customer in several plans for multiple potential time slots. For example, fast insertion heuristics [33] do not perform well for a medium-size AHD system since it relies on a single route plan without taking into account intermediate re-optimization. Furthermore, there are several heuristic constraints such as travel times, work time rules [34], geo-coding the new customer location, and updating the distance matrix [35], which increases the complexity of the feasibility check.

Since the hard nature of optimally solving COPs and state-of-the-art algorithms mainly rely on handcrafted heuristics, machine learning naturally becomes the best candidate in a more principled and optimized way [36]. Recently, Van der Hagen et al. [16] propose machine learning-based feasibility checks for dynamic time slot management. This method employs a neural network to predict feasibility based on patterns in the instance data, e.g., orders and fleet information. In their work, the feasibility check is modeled as a binary classification by employing some existing supervised learning methods, Random Forest (RF), Neural Network (NN), and gradient-boosted trees (GB). The proposed method in [16] shows higher accuracy than several state-of-the-art methods, e.g., insertion heuristics [37], Order thresholds [38], and Continuous approximation [17]. Recently, Larsen et al. [39] employs machine learning to assess the value of accepting a booking demand for a shipment of inter-modal containers by rail if the associated expected profit is positive. The performances of the classification and the regression approach utilized in [39] are faster compared to the state-of-the-art integer linear programming (LIP) solver and quite accurate compared to the ground-truth values.

In this thesis work, we will focus on the potential of using a machine learning-based approach to support decision-making for combinatorial optimization problems for time slot management in an AHD system. However, there are noticeable research gaps in the current state of the art. For instance, the proposed machine learning-based feasibility checks in [16] do not consider different time window lengths. This leads to the lack of the generality of the solution since the time window constraints are the important inputs for the model utilized in [16]. Additionally, since the corresponding delivery cost of an offer can outweigh the receiving profit of the overall delivery plan, it is important to consider whether or not to offer this time slot to a customer. Note that this problem is well-considered in the literature. For example, Cleophas and Ehmke [32] considered a given transport capacity and the expected value of the orders to suggest a way of making an AHD system in metropolitan areas more profitable. An important keynote in [32] is to maximize the overall value of orders rather than the number of orders in order to be profitable in the e-commercial sector. More recently, Koehler et al. [40] proposed a customer acceptance mechanism that is based on delivery cost approximation. The proposed method in [40] is designed to help a retailer to accept as many customers as possible to stay profitable. However, since the approach [40] does not utilize time window constraints in the cost approximation function, the method becomes inferior when the demand for time windows is imbalanced. In this thesis work, a machine learning-based approach can be used to investigate both the feasibility and profitability of AHD with time windows.

1.3 Research questions

Through the literature review, the following research questions are taken into account in this thesis work

- "How can the proposed machine learning (ML)-based feasibility tests for AHD be adapted to different time window lengths?" This leads to the following sub-questions:
  - (1) "How the proposed ML-based approach behaves in different time window lengths?"

  - (2) "Is there a way to adapt the proposed approach when considering different time window lengths in a certain scenario, so it will work more effectively and sufficiently?"

- "Can the proposed ML-based approach predict the profitability of a given time slot?". In other words, if the profit of the current plan can be estimated continuously or discretely using the proposed ML-based approach. One possible approach is to utilize binary classifications to predict discrete variables (yes or no) or discrete ranges of the profit value. Another approach is considered to predict profitability on a continuous scale by using regression models. Classical methods, e.g., Logistic Regression, K-Nearest Neighbours, Support Vector Machines, Support Vector Regression, and Random Forest Regression, are taken into account for the classification and regression approach.

1.4 Objective and Methodology

The goal of this work is to present a *feasibility* and *profitability* check for offering time slots for the next-day delivery in the attended home delivery system. Similar to most academic literature on time slot management, in this work we also consider a deterministic VRPTW with a depot as the routing context.

Different characteristics of the time window length, e.g., wider and tighter time windows, will be considered for the data input to verify the effectiveness of the proposed algorithm. Different scenarios will be taken into account such as different distributions of the customers' locations with different time window lengths.

We focus on the use of machine learning (ML)-based methods, i.e., supervised approaches [41], to determine whether a feasible and profitable solution exists when a customer's demand is included in a current routing plan. Information about the cost of the items ordered, the delivery cost, the driver cost, and the net profit of the item are considered. Different network architectures, e.g., multi-layer perceptrons [1] and recurrent neural networks [2], or using classical classification methods, e.g., Decision Tree Classifier [3], Random forest classifier [4], etc, can be employed to analyze the performance of the proposed ML-based approach using simulations and real data. Other methods in literature can also be considered. Classical methods such as insertion heuristics, continuous approximation, and order thresholds, can be considered to perform the benchmarking result with the proposed methods.

2 Problem definition

Recall Figure 1, in the first stage, the customer selects items to add to their basket. Then, in the second stage, the customer signs in, allowing the system to record their location. After sign-in, as shown in Figure 2, the system proceeds to the third stage, where it presents the customer with feasible time slots for delivery or pickup. This selection process relies on a fast computation that factors in the current route plan, optimizing for multiple customers and considering constraints like vehicle capacity and time windows (VRP-TW).

At this point, the Vehicle Routing Problem with Time Windows (VRP-TW) becomes critical. To maintain a positive user experience, it's essential that the VRP-TW solution provides an optimal delivery plan within a very short time frame (e.g., within three seconds), thereby satisfying the customer's preferences and enabling trust in the system. In the subsequent subsections, a detailed formulation of the VRP-TW optimization problem is provided, including constraints and parameters tailored to address the maximum number of capacitated vehicles and the constrained time windows, and several customers.

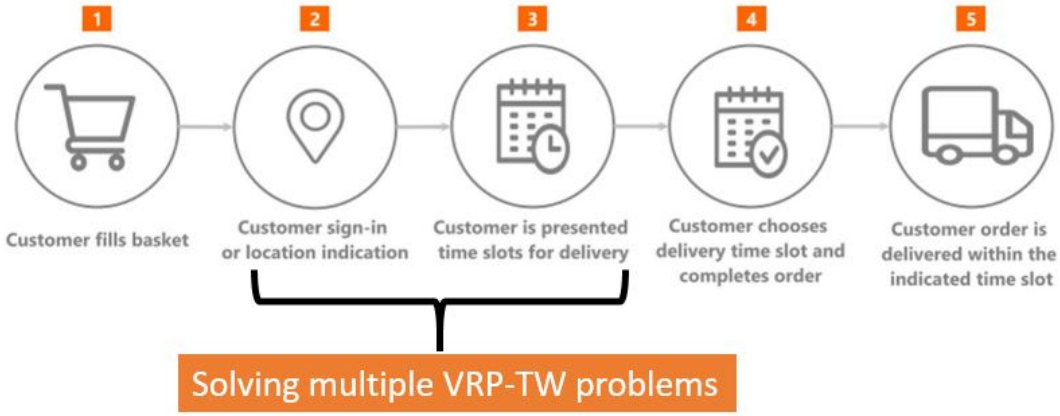

Figure 2: Multiple VRP-TW problems have to be solved within the small time window to give the customer feasible time slots.

It is obvious that solving multiple VRP-TW problems cannot guarantee a solution within a linear time as the number of customers increases. Therefore, the main focus of this thesis is to employ a machine learning-based method to predict whether, given a selected time slot and the current route plan, a feasible delivery plan can be made and whether following that plan will still provide a benefit. To achieve this, after presenting the mathematical formulation of the VRP-TW in Subsection 2.1, we will introduce the data generation pipeline in Subsection 2.2 used to collect the necessary data for the machine learning approach.

2.1 Mathematical formulation of the VRP-TW problem

We consider a VRPTW over a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{A}\}$, where:

- $\mathcal{V}$ is the set of nodes, including customer nodes $\{v_1, v_2, \ldots, v_{n_c}\}$ and a depot node v_d.
- $\mathcal{K} = \{k_1, k_2, \ldots, k_{n_k}\}$ is the set of available vehicles.
- Each customer $i \in \mathcal{V}$ has a demand d_i and a time window $[\underline{t}_i, \overline{t}_i]$.
- Each vehicle has a capacity $\overline{c}$ and starts and ends its route at the depot v_d.

The binary variable x_{ijk} denotes whether vehicle k travels from node i to node j, and a_i is the arrival time at node i.

The VRPTW is formulated as follows:

$$\min_{\mathbf{x},\mathbf{a}} \sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}} x_{ijk} c_{ijk}, \tag{1.1}$$

$$\sum_{k\in\mathcal{K}} \sum_{j\in\mathcal{V}} x_{ijk} = 1, \quad \forall i \in \mathcal{V}, \tag{1.2}$$

$$\sum_{j\in\mathcal{V}} x_{(v_d)jk} = 1, \quad \forall k \in \mathcal{K}, \tag{1.3}$$

$$\sum_{i\in\mathcal{V}} x_{i(v_d)k} = 1, \quad \forall k \in \mathcal{K}, \tag{1.4}$$

$$\sum_{j\in\mathcal{V}} x_{ijk} - \sum_{j\in\mathcal{V}} x_{jik} = 0, \quad \forall i \in \mathcal{V}, \forall k \in \mathcal{K}, \tag{1.5}$$

$$\underline{t}_i \le a_i \le \overline{t}_i, \quad \forall i \in \mathcal{V}, \tag{1.6}$$

$$a_j \ge a_i + s_i + t_{ij} - M(1 - x_{ijk}), \quad \forall i, j \in \mathcal{V}, \forall k \in \mathcal{K}, \tag{1.7}$$

$$\sum_{i\in\mathcal{V}} d_i x_{ijk} \le \overline{c}, \quad \forall k \in \mathcal{K}, \tag{1.8}$$

$$u_i - u_j + \overline{n}_c \cdot x_{ijk} \le \overline{n}_c - 1, \quad \forall i, j \in \mathcal{V}, \forall k \in \mathcal{K}, \tag{1.9}$$

$$x_{ijk} \in \{0, 1\}, \quad \forall i, j \in \mathcal{V}, \forall k \in \mathcal{K}. \tag{1.10}$$

Sub equations in (1) are explained in the following

- The objective function (1.1) minimizes the total transportation cost, consisting of the travel cost between nodes and a fixed cost for each vehicle's usage. c_{ijk} is the cost of using the vehicle k for transporting from customer i to the customer j
- Constraint (1.2) ensures that each customer is visited exactly once by any vehicle.
- Constraints (1.3) and (1.4) specify that each vehicle starts and ends its route at the depot.
- Constraint (1.5) enforces flow conservation, ensuring that if a vehicle arrives at a customer node, it must also leave that node.
- Constraint (1.6) enforces each customer's specified time window.

- Constraint (1.7) establishes the arrival time at each node, considering travel and service times. Here, M is a large constant that deactivates this constraint if the vehicle does not travel from i to j.
- Constraint (1.8) limits the total demand served by each vehicle to its capacity $\bar{c}$.
- Constraint (1.9), the Miller-Tucker-Zemlin (MTZ) subtour elimination constraint [42], prevents subtours within each vehicle's route by using a ranking variable u_i.
- Constraint (1.10) specifies that x_{ijk} is a binary decision variable, indicating whether vehicle k travels from node i to j.

2.2 Data generation

To train a machine learning (ML) model for checking delivery time slot feasibility, instances' labels are generated as either feasible or infeasible using a VRP-TW solver, i.e., CPLEX. This approach is designed to automate the time-consuming task of determining whether a new customer's request can be accommodated within existing delivery constraints. The training process involves creating a dataset of feasibility check instances and labeling each instance according to whether it is feasible or infeasible given current routing and time window constraints. However, simply generating instances through random sampling of customer–time slot combinations often produces cases that are either unrealistic or extremely infeasible. These unrealistic scenarios may include instances where feasibility cannot be achieved without removing a large number of customers, which would rarely occur in real-world operations.

To address this, in this thesis, a simulation that emulates the actual customer time slot booking process is implemented. This simulation generates realistic feasibility check instances that closely mimic practical scenarios encountered in real-world delivery operations. By replicating the sequential nature of customer bookings, our simulation ensures that the generated instances are not only varied but also realistic, avoiding the generation of extreme or impractical cases. The steps of this simulation process are illustrated in Figure 3, which provides a framework for constructing a feasibility check of the generated dataset.

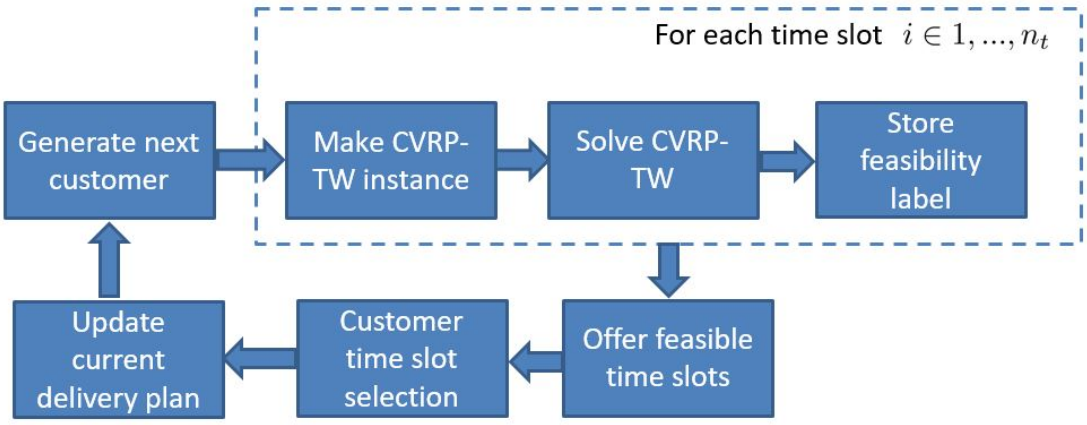

Figure 3: Overview of the data generation process.

The simulation process for generating feasibility check instances involves several key steps, each of which reproduces real-world conditions in delivery scheduling. The process begins with generating a new customer request and then proceeds through a series of steps to determine the feasibility of this request given the current delivery plan. The overall process is iterative, meaning that it repeats for each new customer request, with each iteration involving the creation of multiple instances (one per time slot) that are then solved to determine feasibility. The detailed steps of the simulation process are in the following.

First, the simulation starts by generating a new customer requesting a delivery. *After generating a new customer*, the simulation offers a set of potential delivery time slots for this customer. For instance, if the delivery period spans 8:00 to 20:00 with four evenly distributed time slots, i.e., $n_t = 4$, the available options for this customer would be $8:00-11:00$, $11:00-14:00$, $14:00-17:00$, and $17:00-20:00$. This set of possible time slots is then evaluated to determine which slots can feasibly accommodate the new customer request. *Third*, for each available time slot, a new instance of the Capacitated Vehicle Routing Problem with Time Windows (CVRP-TW) is created. Each instance reflects the current set of accepted orders, combined with the potential new customer–time slot combination. This step ensures that each possible combination of the new customer and a time slot is evaluated independently for feasibility. Finally, once each instance has been solved, the simulation assigns a feasibility label to it, indicating whether the specific customer–time slot combination is feasible or infeasible. These labeled instances serve as the training data.

2.2.1 Data structure of the feasibility check

The data structure constructed from the data generation process is as follows. Each feasibility check instance is defined as an array of input feature vectors $\mathbf{X}_i$ and an output decision Y_i. Each $\mathbf{X}_i$ vector represents the state of the system for a customer's delivery request and is used to determine whether a particular customer–time slot combination is feasible. Note that the Solomon dataset [43] is used to model customer locations and demands, as it is widely used benchmark data for vehicle routing problems.

Each input feature vector $\mathbf{X}_i$ contains relevant information about the delivery problem instance, represented as follows:

$$\mathbf{X}_i = \begin{bmatrix} \overline{v}_h & \overline{c}_h & x_d & y_d & \mathbf{c}_1 & \mathbf{c}_2 & \dots & \mathbf{c}_k & \dots & \mathbf{c}_i \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{4+5\times i}, \tag{2}$$

where $i \leq 100$, as the maximum number of 100 customers is considered in this thesis. Each element in $\mathbf{X}_i$ is normalized. $\overline{v}_h$ represents the maximum number of vehicles available for delivery. Additionally, $\overline{c}_h$ indicates the maximum load capacity that each vehicle can carry. This parameter ensures that any generated route respects the vehicle's capacity limits. x_d and y_d are the depot's location coordinates, providing a reference point for the routing model relative to customer locations. Each customer k is represented by a 5-dimensional vector $\mathbf{c}_k \in \mathbb{R}^5$, which contains essential details about the

customer's delivery requirements

$$\mathbf{c}_k = \begin{bmatrix} x_{c,k} \\ y_{c,k} \\ d_k \\ \bar{t}_{c,k} \\ \underline{t}_{c,k} \end{bmatrix}, \tag{3}$$

where $x_{c,k}$ and $y_{c,k}$ are the location coordinates of customer k, provided by the Solomon dataset. d_k is the demand or order size of customer k, representing the load required for delivery. $\bar{t}_{c,k}$ and $\underline{t}_{c,k}$ are the latest and earliest acceptable for delivery, respectively. For each input vector $\mathbf{X}_i$, the output decision Y_i is a binary variable indicating the feasibility of the customer–time slot combination,

$$Y_i \in \{0, 1\}, \tag{4}$$

where $Y_i = 1$ indicates that the customer–time slot combination is feasible under the current routing constraints. This binary output is derived from the feasibility label obtained through the VRPTW solver (e.g., CPLEX), and it is used as the target variable in training the ML model.

2.2.2 Profitability Calculation

To evaluate the *profitability* of adding a new customer's delivery request to the current route plan, the feasibility analysis is extended by introducing an additional decision component based on cost and revenue. Each decision instance, therefore, now considers the potential profit associated with the delivery. Similar to the feasibility check, the profitability check involves a feature vector $\mathbf{X}_i$ and an output decision Y_i, where Y_i represents whether adding the customer is profitable. This section outlines steps for calculating profitability.

When a customer arrives and requests a delivery, the system assigns a random product value, representing the revenue potential, normalized within the range $[0, 10]$. This product value, denoted c_p, provides the basis for revenue in the profitability calculation:

$$c_p \in [0, 10]. \tag{5}$$

In addition, a time slot is selected for the customer, consistent with the feasibility framework. After the time slot is determined, the route is recalculated to accommodate this customer's delivery within the chosen time slot. The new route, which includes the proposed delivery, is optimized using a VRP-TW solver, ensuring that the addition respects existing route constraints. Based on this updated route, the system computes the distance cost, which is scaled to a range of $[0, 1]$ and represented as C_d:

$$c_d \in [0, 1]. \tag{6}$$

The profitability calculation is then conducted by comparing the product value against the cost impact of this route. Specifically, the *current profit* from adding this delivery is given by:

$$p_c = 10\% v_p - c_d. \tag{7}$$

This calculation adjusts the potential revenue ($10\%V_p$) by subtracting the scaled distance cost of the new route, thereby reflecting the immediate profit or loss associated with adding this customer.

To evaluate the overall profitability, the system considers the cumulative profit prior to adding this customer. Let $\tilde{p}$ represent the cumulative profit from previous deliveries. The estimated profit after including the current customer is given by:

$$\tilde{p} = \sum_{i=1}^{n_c} p_{c,i}, \tag{8}$$

where $n_c \leq 100$ is the latest customer in the route plan and $p_{c,i}$ is the profit obtained from the $i-th$ customer. The output decision Y_i is then defined based on this profitability measure:

$$Y_i = \begin{cases} 1 & \text{if } \tilde{p} > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

If $\tilde{p} > 0$, then adding the customer's delivery is considered profitable, resulting in a positive decision ($Y_i = 1$). Otherwise, the decision outcome is negative ($Y_i = 0$), indicating non-profitability.

2.2.3 Summary

To this point, the decision-making process is considered a classification problem, where the observation for each instance is represented by the feature vector $\mathbf{X}_i$ in (2), and the output decision Y_i in (4) and (9) indicates whether a customer's delivery request is feasible or profitable.

The feature vector $\mathbf{X}_I$ contains relevant information about the system's state, including fleet size, vehicle capacities, customer details (such as location and demand), and time slot information. This vector serves as the input to a classifier, and the decision Y_i is the corresponding output.

The classifier uses these labeled instances of $(\mathbf{X}_i, Y_i)$ pairs to learn the decision, where Y_i can be either 0 or 1, denoting feasibility or profitability, respectively. The machine learning task is to predict the feasibility or profitability of new customer requests based on the observed features in $\mathbf{X}_i$. The source code for generating data is provided in the Appendix.

In the next section, methods for this classification problem are explained in detail. Classification methods, e.g., Naive Bayes, K-Nearest Neighbors (KNN), Binary Decision Trees, and Ensemble methods (such as Random Forest), and ML-based methods, e.g., a fully connected neural network and a long short-term memory network (LSTM), are taken into account.

3 Methods

In this section, methodologies used to determine the decision Y_i based on the input features $\mathbf{X}_i$ for each delivery request are described. The decision Y_i is a binary classification variable, indicating the feasibility check of a customer's delivery request. The goal of this classification task is to train a model that can accurately predict Y_i based on the observed features in $\mathbf{X}_i$, allowing the system to make informed decisions for new customer requests.

3.1 Classification Methods

We employ several classical classification methods to evaluate the decision variable Y_i, including Naive Bayes, K-Nearest Neighbors (KNN), Binary Decision Trees, and Random Forests. Each method offers different ways to analyze the input feature vector $\mathbf{X}_I$ and determine the most probable class (feasible/profitable or infeasible/non-profitable) for the decision Y_i.

3.1.1 Naive Bayes

Naive Bayes is a probabilistic classifier that applies Bayes' theorem with the assumption of independence between the features in $\mathbf{X}_i$. Given an observation $\mathbf{X}_i = (x_1, x_2, \ldots, x_n)$, the probability of Y_i being in class c (where $c \in \{0, 1\}$) is calculated as:

$$P(Y_i = c|\mathbf{X}_i) = \frac{P(Y_i = c) \prod_{j=1}^{n} P(x_j|Y_i = c)}{P(\mathbf{X}_i)}, \tag{10}$$

where $P(Y_i = c)$ is the prior probability of class c, $P(x_j|Y_i = c)$ is the conditional probability of each feature given the class, and $P(\mathbf{X}_i)$ is the probability of the observation. The classification decision for Y_i is determined by choosing the class c that maximizes $P(Y_i = c|\mathbf{X}_i)$:

$$Y_i = \arg\max_{c \in \{0,1\}} P(Y_i = c|\mathbf{X}_i). \tag{11}$$

3.1.2 K-Nearest Neighbors (KNN)

The K-Nearest Neighbors (KNN) method is a non-parametric classifier that determines the class of Y_i based on the classes of the k closest points in the feature space. For an observation $\mathbf{X}_i$, the Euclidean distance between $\mathbf{X}_i$ and each training sample $\mathbf{X}_j$ is computed as:

$$d(\mathbf{X}_i, \mathbf{X}_j) = \sqrt{\sum_{m=1}^{n} (x_{i,m} - x_{j,m})^2}. \tag{12}$$

The k nearest neighbors to $\mathbf{X}_i$ are identified based on this distance metric, and Y_i is assigned to the class most frequently occurring among the k neighbors:

$$Y_i = \arg\max_{c \in \{0,1\}} \sum_{j \in \mathcal{N}_k} \mathbb{I}(Y_j = c), \tag{13}$$

where $\mathcal{N}_k$ is the set of the k nearest neighbors, and $\mathbb{I}(\cdot)$ is the indicator function that returns 1 if $Y_j = c$ and 0 otherwise.

3.1.3 Binary Decision Trees

A Binary Decision Tree is a recursive partitioning method that splits the feature space into regions based on the values of the features in $\mathbf{X}_i$. At each node in the tree, a feature x_m is selected, and a threshold t is set to create a binary split. The objective is to minimize the impurity at each node, which is measured by metrics such as Gini impurity or entropy.

For a Gini impurity G, the impurity at a node s for class c is calculated as:

$$G(s) = 1 - \sum_{c=0}^{1} p_c^2, \tag{14}$$

where p_c is the probability of class c at node s. The splitting process continues until a stopping criterion is met (e.g., maximum depth or minimum samples per leaf), resulting in a final prediction Y_i based on the majority class in the terminal node.

3.1.4 Random Forest

Random Forest is an ensemble method that combines multiple decision trees to improve classification accuracy and reduce overfitting. Each tree in the forest is trained on a random subset of the training data and a random subset of features. Given an observation $\mathbf{X}_i$, each decision tree T_j in the forest outputs a prediction $Y_i^{(j)}$. The final output Y_i is determined by aggregating the predictions from all trees, typically using majority voting:

$$Y_i = \arg\max_{c \in \{0,1\}} \sum_{j=1}^{M} \mathbb{I}(Y_i^{(j)} = c), \tag{15}$$

where M is the number of trees in the forest, and $\mathbb{I}(\cdot)$ is the indicator function. By aggregating predictions across multiple trees, Random Forest reduces variance and improves the generalization ability of the model.

3.2 ML-based Classification Method

In addition to classical classification methods, we utilize machine learning (ML)-based architectures for predicting the output decision Y_i, specifically a Fully Connected Neural Network (FCNN) and a Long Short-Term Memory (LSTM) network. These architectures are designed to capture complex patterns and dependencies within the feature vector $\mathbf{X}_i$ that may be missed by classical models, enabling higher predictive accuracy in determining feasibility or profitability.

3.2.1 Fully Connected Neural Network (FCNN)

The Fully Connected Neural Network (FCNN) is a type of feed-forward neural network where each neuron in a layer is connected to every neuron in the subsequent layer. Given an input feature vector $\mathbf{X}_i \in \mathbb{R}^n$, the FCNN learns a non-linear transformation to predict Y_i. The network consists of multiple layers, each defined by a set of weights and biases. For each layer l, the output $\mathbf{a}^{(l)}$ is computed as:

$$\mathbf{a}^{(l)} = f\left(\mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}\right), \tag{16}$$

where:

- $\mathbf{a}^{(0)} = \mathbf{X}_i$ is the input feature vector in (2),
- $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are the weight matrix and bias vector for layer l,
- $f(\cdot)$ is a non-linear activation function, typically ReLU (Rectified Linear Unit) or sigmoid, applied element-wise.

The final layer of the FCNN outputs a probability score for each class, with the predicted class Y_i determined as:

$$Y_i = \arg\max_{c \in \{0,1\}} \hat{p}_c, \tag{17}$$

where $\hat{p}_c$ represents the predicted probability of class c after applying a softmax function to the output of the final layer:

$$\hat{p}_c = \frac{\exp(z_c)}{\sum_{j=0}^{1} \exp(z_j)}. \tag{18}$$

The network is trained by minimizing a binary cross-entropy loss between the predicted probabilities and the true labels.

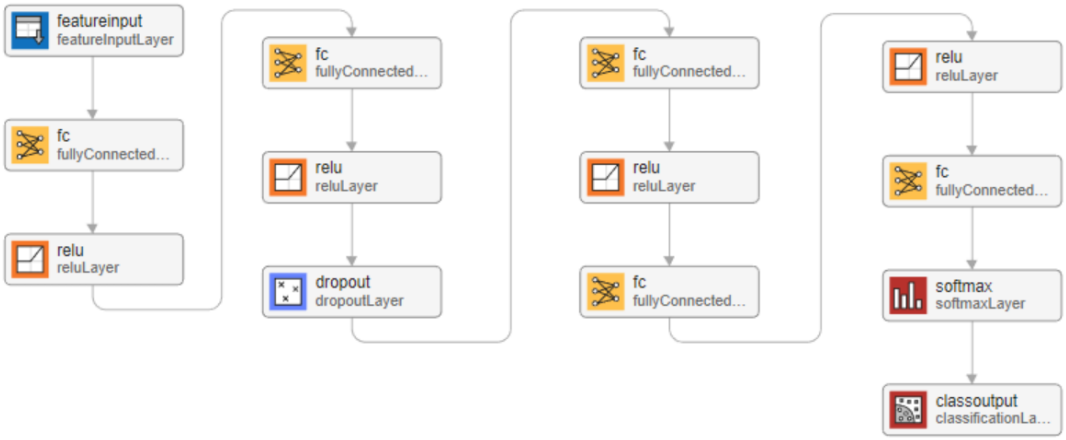

Figure 4: FCNN architecture for feasibility check, consisting of five fully connected layers with ReLU activations, dropout, and softmax classification output.

For the feasibility check, the architecture, as shown in Figure 4, consisting of five fully connected layers, each with 128 neurons, is utilized. The network leverages ReLU activation functions to introduce non-linearity, enabling the model to capture complex patterns in the data. The network includes five fully connected (dense) layers, each with 128 neurons. Mathematically, for each layer l, the output $\mathbf{a}^{(l)}$ is computed as:

$$\mathbf{a}^{(l)} = \text{ReLU}(\mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}), \tag{19}$$

where $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are the weight matrix and bias vector of layer l, respectively. The ReLU activation function, defined as $\text{ReLU}(x) = \max(0, x)$, is applied element-wise to introduce non-linearity. Note that a dropout layer with a rate of 10% is used after the second fully connected layer to reduce overfitting by randomly setting 10% of the neurons to zero during each training iteration.

3.2.2 Long Short-Term Memory (LSTM) Network for Feasibility Check

The feasibility check can also be performed using a sequence-based neural network architecture, specifically a Long Short-Term Memory (LSTM) network. The LSTM is a type of recurrent neural network (RNN) that is particularly well-suited for handling sequential data, thanks to its ability to maintain long-term dependencies through a set of gating mechanisms. To fit the observation $\mathbf{X}_i$ in (2) to a LSTM architecture network, each input sequence $\mathbf{X}_i$ for the LSTM model is modified as:

$$\mathbf{X}_i = [\mathbf{c}_0,\, \mathbf{c}_1,\, \mathbf{c}_2,\, \ldots,\, \mathbf{c}_k,\, \ldots,\, \mathbf{c}_i] \in \mathbb{R}^{4\times(i+1)}, \tag{20}$$

where $i \leq 100$, given that a maximum of 100 customers is considered in this study. Each vector $\mathbf{c}_k$ represents the features for customer k, defined as:

$$\mathbf{c}_k = \left[x_{c,k},\, y_{c,k},\, d_k,\, \bar{t}_{c,k},\, \underline{t}_{c,k}\right]^\top. \tag{21}$$

The initial vector $\mathbf{c}_0$ contains information about the delivery depot:

$$\mathbf{c}_0 = \left[\overline{v}_h,\, \overline{c}_h,\, x_d,\, y_d\right]^\top. \tag{22}$$

The LSTM network processes each feature sequentially while maintaining a hidden state $\mathbf{h}_t$ and a cell state $\mathbf{c}_t$. At each time step t, the LSTM updates its states as follows:

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \cdot [\mathbf{h}_{t-1}, c_t] + \mathbf{b}_f), \tag{23}$$
$$\mathbf{i}_t = \sigma(\mathbf{W}_i \cdot [\mathbf{h}_{t-1}, c_t] + \mathbf{b}_i), \tag{24}$$
$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c \cdot [\mathbf{h}_{t-1}, c_t] + \mathbf{b}_c), \tag{25}$$
$$\mathbf{x}_t = \mathbf{f}_t \odot \mathbf{x}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{x}}_t, \tag{26}$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_o \cdot [\mathbf{h}_{t-1}, c_t] + \mathbf{b}_o), \tag{27}$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{x}_t), \tag{28}$$

where:

- $\mathbf{f}_t$, $\mathbf{i}_t$, and $\mathbf{o}_t$ are the forget, input, and output gates, respectively,
- $\sigma(\cdot)$ denotes the sigmoid function, and $\tanh(\cdot)$ denotes the hyperbolic tangent function,

- $\mathbf{W}_f$, $\mathbf{W}_i$, $\mathbf{W}_c$, and $\mathbf{W}_o$ are the weight matrices, and $\mathbf{b}_f$, $\mathbf{b}_i$, $\mathbf{b}_c$, $\mathbf{b}_o$ are the biases for each gate.

After processing the entire sequence, the final hidden state $\mathbf{h}_n$ is passed through a dense layer with a softmax activation to produce the probability of each class:

$$\hat{p}_c = \frac{\exp(z_c)}{\sum_{j=0}^{1} \exp(z_j)}. \tag{29}$$

The predicted class Y_i is then given by:

$$Y_i = \arg \max_{c \in \{0,1\}} \hat{p}_c. \tag{30}$$

The LSTM network architecture utilized for the feasibility check is illustrated in Figure 5. Note that this architecture can be used as an alternative solution to the FCNN architecture. This LSTM-based architecture was reported to be more effective (more accuracy and faster inference time) compared to the FCNN in the next section, the profitability check will be built upon the LSTM network architecture. Details are in the following.

In Figure 5, a sequence input layer accepts input data in (21) as sequence format. This layer is responsible for passing the input sequences to the subsequent layers in the network. The LSTM layer contains 128 hidden units, which define the dimensionality of the output space. Note that the state activation function for the LSTM cells is the hyperbolic tangent (tanh) function. This function maps the cell state values between -1 and 1, helping to prevent gradients from vanishing or exploding. The gate activation functions (input, forget, and output gates) use the sigmoid function, which maps values between

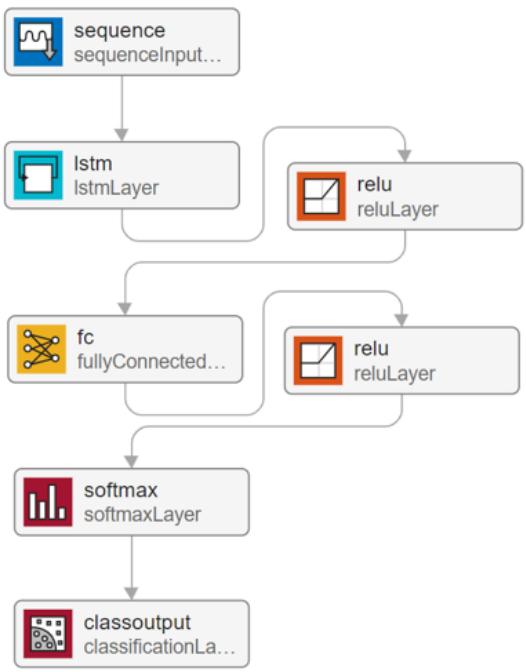

Figure 5: LSTM-based architecture for feasibility check. Compared to the FCNN architecture, we achieve a higher accuracy which will be reported in the next section.

0 and 1. This allows each gate to control the flow of information effectively, deciding which information to keep, update, or discard. Additionally, the fully connected layer consisting of 128 neurons is utilized. This layer helps to combine the features from the LSTM layer and prepares them for the classification stage. This architecture combines the sequential processing capabilities of the LSTM layer with the classification strength of the fully connected and softmax layers, enabling it to effectively learn and classify temporal patterns in the input data.

3.2.3 Long Short-Term Memory (LSTM) Network for Profitability Check

Based on the LSTM network proposed in the previous subsection, the profitability check leverages a multi-branch LSTM-based architecture to effectively determine the profitable utilization of the generated dataset. As shown in Figure **??**, the network is composed of two primary branches: a fixed-weight network and a trainable LSTM branch.

The right-side top corner of Figure 6 illustrates the fixed-weight network, which reuses the learned weights from the feasibility network (Figure 5). This fixed-weight subnetwork serves as a pre-trained feature extractor that provides reliable feasibility assessments based on previously learned constraints and conditions. By freezing the weights in this branch, the feasibility information is preserved and remains stable throughout the profitability check, ensuring consistency in evaluating the feasibility of input sequences.

The left-side LSTM branch, in contrast, contains trainable weights and acts as a complementary feature extractor. This branch is designed to capture additional, contextually rich feature information from the input data. The flexibility of this trainable LSTM branch allows it to adapt to new patterns and nuances in the input, enabling it to learn representations specifically tailored to profitability analysis, beyond what was learned in the feasibility network.

The outputs of both branches are then concatenated and combined via an addition layer, creating an integrated feature set that includes both feasibility and profitability information. This combined representation enables the model to make informed decisions, leveraging both the stability of feasibility features and the adaptability of profitability-specific features.

The resulting architecture, with both fixed and trainable branches, provides a robust framework for profitability checks. It effectively balances the reuse of reliable, pre-learned features (for feasibility) with the flexibility needed to capture profit-specific insights, enhancing the model's ability to make accurate predictions on the profitability of the dataset.

3.2.4 Training and Optimization

Both the FCNN and LSTM architectures are trained using the binary cross-entropy loss, defined as:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\left[Y_i \log(\hat{p}_{Y_i}) + (1 - Y_i)\log(1 - \hat{p}_{Y_i})\right], \tag{31}$$

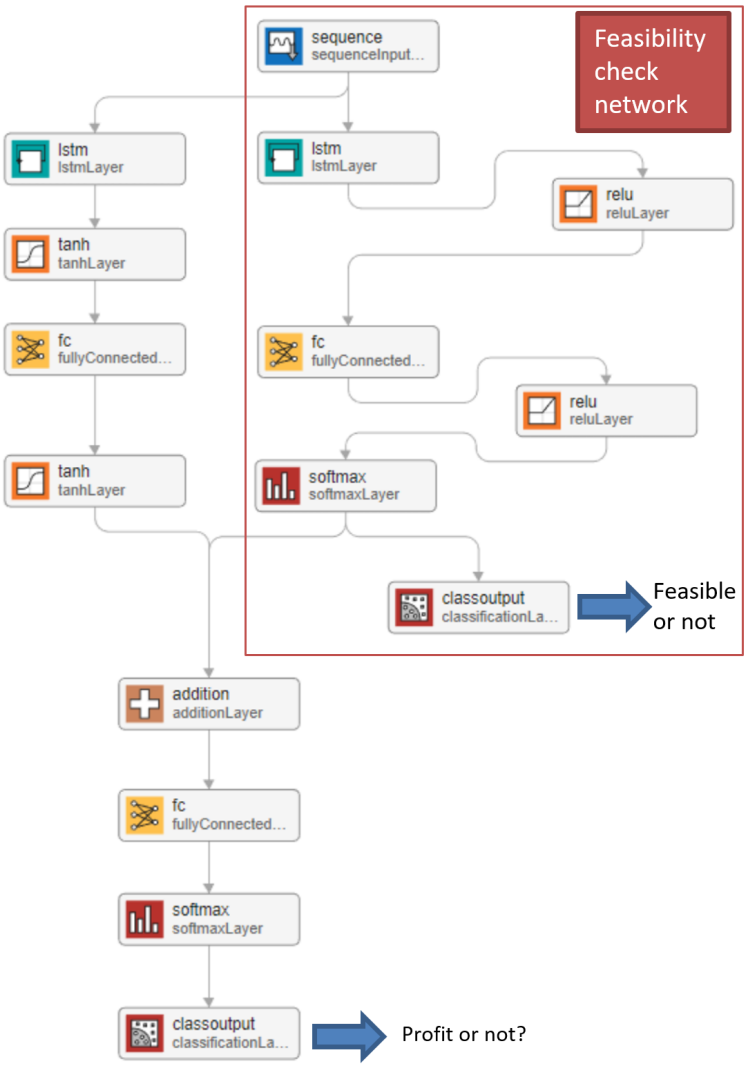

Figure 6: LSTM-based architecture for profitability check. The fixed-weight network of the feasibility check (on the right side of the figure) is concatenated with another LSTM branch to extract rich feature information from the observation input

where N is the total number of training samples, Y_i is the true class label, and $\hat{p}_{Y_i}$ is the predicted probability of the true class.

For training this network, the Adam optimizer [44] is used to minimize the binary cross-entropy loss. Additionally, a batch size of 10^3 is used to stabilize the training process and reduce variance in gradient updates. A small L2 regularization term (10^{-6}) is applied to penalize large weights, which further helps in controlling overfitting. The learning rate is set to 0.01.

3.3 Summary

In this section, we first discussed classical classification methods, including Naive Bayes, K-Nearest Neighbors (KNN), Binary Decision Trees, and Random Forests. We then introduced the proposed fully connected neural network architecture and an LSTM-based architecture for feasibility checks. Furthermore, we extended the LSTM-

based architecture by integrating it with a profitability check, utilizing a combination of fixed-weight and trainable branches to assess both feasibility and profitability within the dataset. In the next section, results from the generated data are presented in detail, providing an in-depth analysis of the performance and effectiveness of the proposed architectures.

4 Results

In this section, the results of the proposed algorithms, i.e., the fully connected neural network architecture and the LSTM-mix fully connected (LSTM-FC) NN architecture, are presented with different numbers of time slots and the maximum number of vehicles. For the generated dataset, the C101 dataset from Solomon [43] is utilized and modified.

The data is generated for 3 different numbers of vehicles, i.e., $n_k \in \{4, 8, 12\}$. The maximum time slots are chosen to be in the set $\{3, 4, 5, ..., 10, 12, 14\}$, and the day shift is from $8:00-18:00$. CPLEX solver is chosen for solving the VRP-TW problem due to its performance. All the simulation, the training process, and the inference process are processed in an Intel core i7-13700K with 32 GB of RAM. For an individual VRP-TW instance with 50 customers, while CPLEX returns a feasible solution within $1\,\mathrm{s}$, it takes approximately $25\,\mathrm{s}$ with the Google OR-tool.

4.1 Quantitative Results on Feasibility Check

For the comparative evaluation of various classification methods on the vehicle routing problem with time windows (VRP-TW), we configured the experimental setup with a maximum of 4 time slots, resulting in a total of $3 \times 4 \times 100 = 1200$ trials. To create a balanced dataset, we conducted feasibility checks on these trials, identifying 643 feasible instances and 557 infeasible instances. This balanced dataset allows for a robust assessment of model performance across feasible and infeasible cases.

As shown in Table 1, we evaluated several classification models, including classical machine learning methods such as Naïve Bayes, k-nearest neighbors (k-NN), binary decision trees, and ensemble methods like random forests, alongside neural network-based approaches: a fully connected neural network and an LSTM-based neural network. The training and testing accuracy of each method are presented to illustrate the relative effectiveness of each model.

The classical methods, including Naïve Bayes, k-NN, binary decision trees, and random forests, achieved moderate accuracy levels. Among these, the random forest performed the best, with a testing accuracy of 60.92%. However, even the ensemble-based random forest model struggled to exceed this threshold. These classical approaches generally suffer from the limited capability to capture the complex, sequential patterns inherent in VRP-TW data, particularly given the temporal and sequential dependencies involved in optimizing vehicle routes and scheduling within time windows. Classical methods typically rely on static feature representations and lack mechanisms for capturing temporal relationships, which are crucial in accurately predicting feasibility in time-dependent tasks like VRP-TW.

In contrast, the neural network models, particularly the LSTM-FC neural network, demonstrated a better performance. While the fully connected neural network (NN) managed to surpass classical methods, reaching a testing accuracy of 63%, it still fell short of achieving high accuracy due to its limited ability to handle temporal dependencies. The fully connected NN lacks recurrent connections that can process sequence data effectively, resulting in suboptimal performance on tasks requiring an understanding of time-based patterns.

The LSTM-based neural network achieves a testing accuracy of 94.57%. This high accuracy demonstrates the advantage of LSTM networks in capturing the temporal dependencies within the VRP-TW problem. The LSTM's recurrent architecture allows it to retain information across time steps, making it particularly well-suited for modeling the sequence of time slots and vehicle scheduling decisions. As a result, the LSTM network was able to learn a more nuanced representation of the data, enabling it to distinguish between feasible and infeasible scenarios with a high degree of accuracy.

While these classical methods might be suitable for simpler, static classification tasks, their performance is limited on tasks like VRP-TW, where temporal dependencies are critical. On the other hand, the LSTM network's ability to model these dependencies allows it to make more accurate predictions, resulting in a significant performance advantage over both classical methods and the fully connected NN.

Classification method	**Training accuracy**	**Testing Accuracy**
Naïve Bayes	56.8%	57.34%
k-nearest neighbor	62.1%	61.5%
Binary decision trees	52.2%	54.5%
Ensembles Random Forest	63.2%	60.92%
Fully connected NN (Ours)	70%	63%
LSTM-FC NN (ours)	95.8%	94.57%

Table 1: Classification methods and their training and testing accuracy

Overall, these results highlight the strengths and limitations of each classification approach in the context of VRP-TW. The fully connected NN provided a modest improvement over classical methods, indicating that even basic neural networks can offer an edge due to their ability to learn from data without manual feature engineering. However, the LSTM-based network's substantial accuracy gain emphasizes the importance of using models capable of handling sequential data in time-sensitive applications. For VRP-TW and similar optimization problems, where the order and timing of events matter, LSTM networks are a betterchoice due to their ability to effectively learn and leverage temporal patterns.

4.2 Analysis of Training and Validation Results for Fully Connected Network

Figure 7 illustrates the training and validation accuracy and loss curves over the training iterations for the fully connected neural network. The figure includes both smoothed and raw accuracy curves for training, as well as validation points recorded at regular intervals. In this test, we use 4 time slots linearly scaled from 8:00 to 18:00, resulting in $3 \times 4 \times 100 = 1200$ instances. The instances are generated by varying the number of vehicles (with three different values) and distributing the feasibility checks across these time slots. This setup provides a balanced dataset to evaluate the network's performance in feasible and infeasible scenarios.

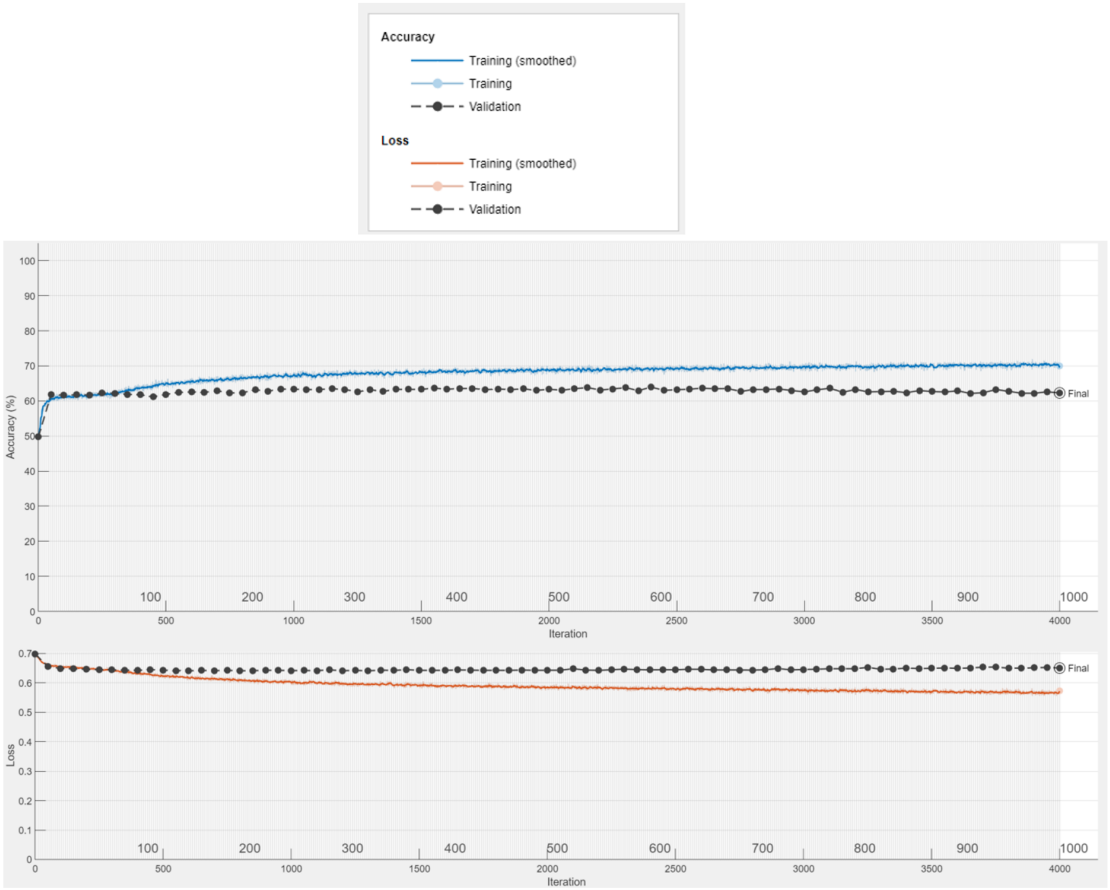

Figure 7: Training and validation accuracy and loss curves for the fully connected neural network with the 4-time slots linearly scaled from 8:00-18:00.

In the upper plot, we observe the progression of accuracy. The training accuracy rapidly improves in the initial training iterations, reaching approximately 60% within the first few hundred iterations. However, as the training continues, the accuracy gain diminishes, stabilizing around the 70% mark. The validation accuracy shows a similar trend, leveling off around 63%, slightly lower than the training accuracy, indicating some degree of overfitting. The limitation in performance, with the validation accuracy plateauing below 70%, suggests that the fully connected network struggles to capture complex patterns in the VRP-TW data that are necessary for achieving higher accuracy.

In the lower plot, the loss curves for both training and validation exhibit a gradual decrease over the training iterations. The training loss decreases relatively smoothly, while the validation loss stabilizes at a higher value than the training loss. This discrepancy between training and validation loss further indicates that the model is learning patterns specific to the training data but has limited generalization ability.

The limitation of the fully connected network to exceed 70% accuracy highlights the limitations of this model architecture in handling sequential and time-dependent patterns. Fully connected neural networks lack mechanisms for modeling temporal dependencies, which are crucial for accurately predicting feasibility within a time-windowed routing context. This limitation underscores the need for more advanced architectures, such as recurrent neural networks, that are designed to capture such dependencies.

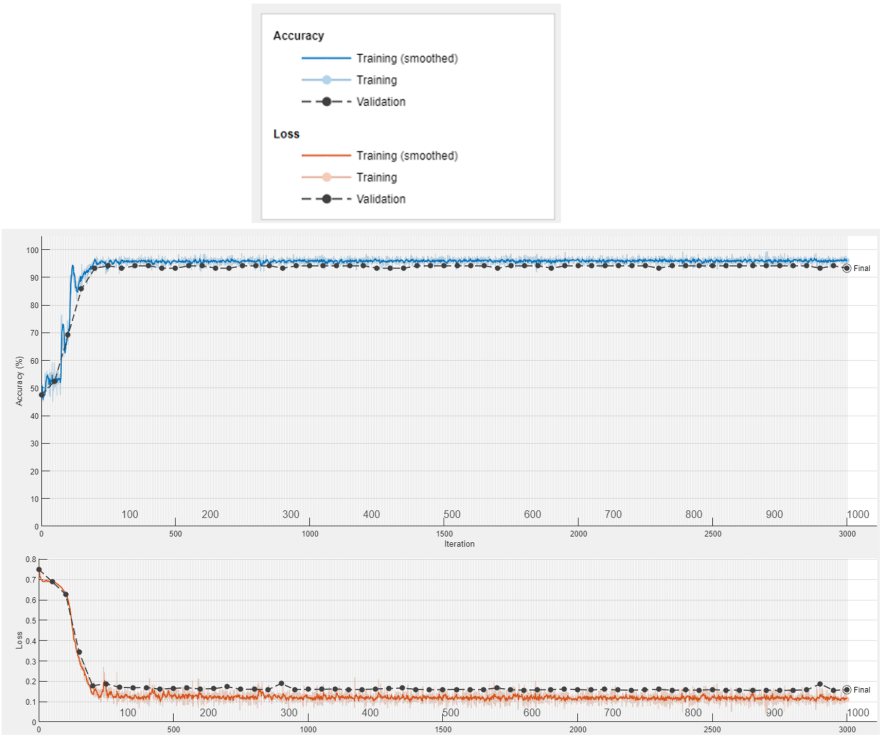

Figure 8: Training and validation accuracy and loss curves for the LSTM-based network with 4 time slots.

4.3 Analysis of Training and Validation Results for LSTM-Based Network Across Multiple Time Slots

This subsection presents the performance of the LSTM-based network trained with various time slot configurations—4, 5, 6, and 8 slots—to predict the feasibility of the VRP-TW problem. For each configuration, the network demonstrates high validation accuracy, significantly exceeding the baseline feasibility rates, thereby illustrating its effectiveness in capturing complex temporal dependencies.

For the configuration with 4 time slots, scaled linearly from 8:00 to 18:00, the dataset consisted of 1200 instances, of which 643 trials were feasible. The LSTM model achieved a validation accuracy of 94.57% after 3000 training iterations (Figure 8). This high accuracy indicates that the LSTM effectively captures the sequential patterns required for accurate feasibility prediction.

With 6 time slots, the dataset consisted of 1800 instances, of which 921 were feasible, corresponding to a baseline feasibility rate of approximately 48.54%. The LSTM model achieved a validation accuracy of around 96.1% after 3000 iterations (Figure 9). Although this accuracy is slightly lower than that of the 4 and 5-slot configurations, it remains significantly higher than the baseline feasibility rate, indicating that the LSTM model continues to generalize well with additional time slots.

For the configuration with 8-time slots, the dataset consisted of 2400 instances, with 1320 feasible trials, giving a baseline feasibility rate of approximately 55%. The LSTM network reached a validation accuracy of approximately 95.72% after 9000 training iterations (Figure 10). The increase in accuracy and the required number of iterations

reflect the model's capability to learn and adapt to the increased complexity in temporal resolution.

The LSTM-based network demonstrates robust performance across all tested time slot configurations, with validation accuracy consistently surpassing the baseline feasibility rates. These results confirm the model's suitability for time-dependent feasibility prediction in VRP-TW scenarios, especially in cases with varying temporal complexities.

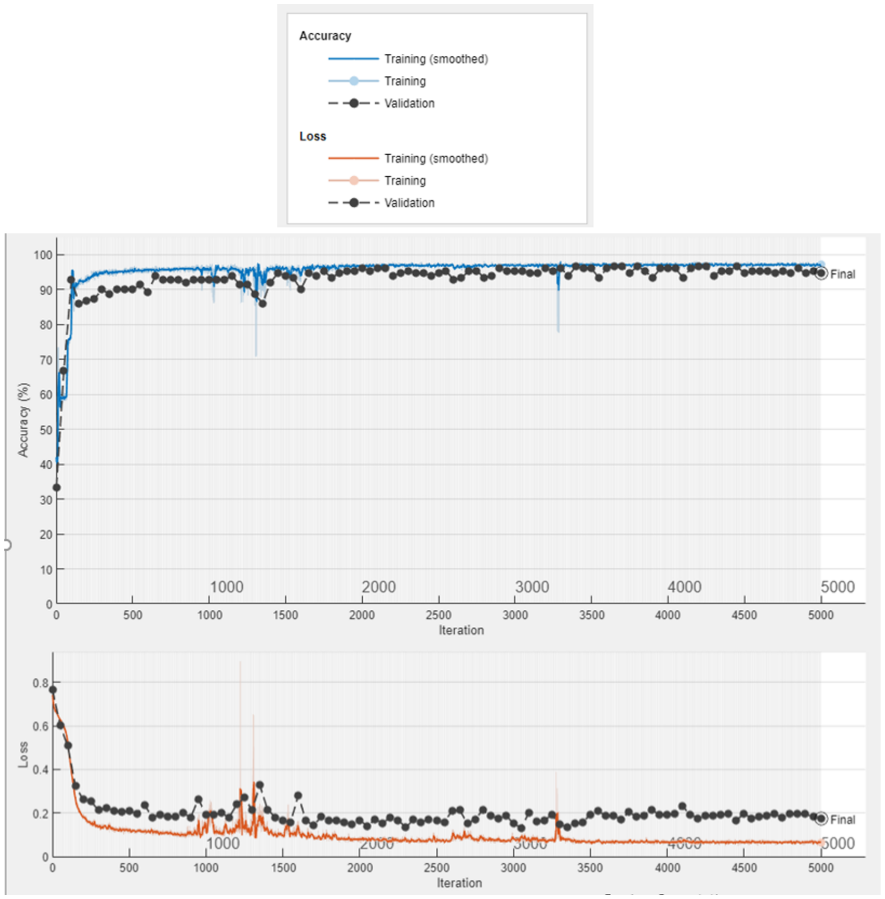

Figure 9: Training and validation accuracy and loss curves for the LSTM-based network with 6 time slots.

4.4 Performance of LSTM Model Across Different Time Windows

In this experiment, we evaluated the performance of our LSTM model for feasibility prediction across a range of delivery time window configurations. Each configuration represented a different number of discrete time slots, defining the potential delivery periods. The goal was to assess the model's adaptability and robustness by analyzing its accuracy across varying levels of temporal granularity. For each configuration, we recorded the baseline accuracy, which represents a preliminary heuristic model's performance, along with the LSTM model's validation accuracy after training. The baseline serves as a benchmark, highlighting the potential improvements offered by the LSTM's sequence-based learning architecture.

The results in Table 2 show that the LSTM model achieves consistently high validation accuracy across all time slot configurations, with values ranging from 92.1% to 96.1%.

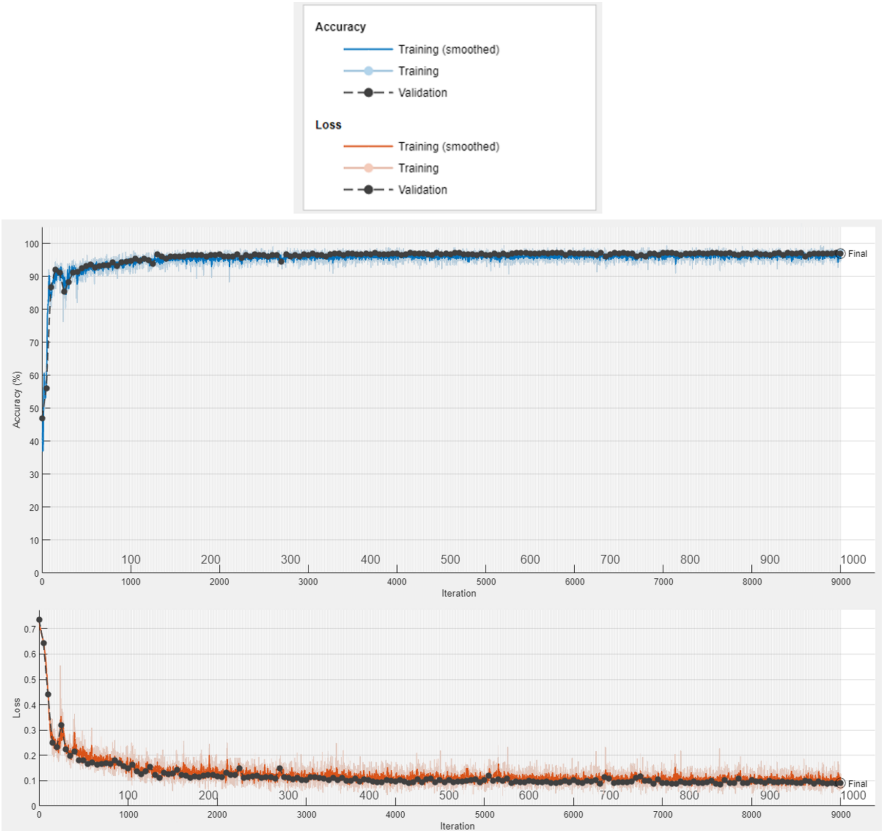

Figure 10: Training and validation accuracy and loss curves for the LSTM-based network with 8 time slots.

No. Time Slots	Baseline (%)	Validation Accuracy (%)
3	53.58	93.00
4	57.85	95.79
5	60.00	94.67
6	48.54	96.10
7	56.24	94.32
8	55.00	95.72
10	49.21	93.75
12	52.17	92.10
14	46.75	95.26

Table 2: Performance Comparison for Different Time Windows

This stability in accuracy across configurations suggests that the model is highly robust and capable of handling the feasibility prediction task regardless of the granularity of the time slots. The LSTM's strong performance across these configurations demonstrates its effectiveness in leveraging sequential information for predictions, as it maintains accuracy even as the time slot configuration changes.

The highest validation accuracy, 96.1%, was observed when using 6 time slots. This peak in performance suggests that an intermediate number of time slots might provide an optimal balance between temporal detail and model complexity. With fewer time slots, the model might lose finer temporal information, while too many time slots could add unnecessary complexity, potentially impacting generalization. Thus, 6 time slots may represent a favorable trade-off, where the model captures relevant patterns without excessive detail that could hinder its predictive power.

The baseline model, by contrast, showed significant variability across the configurations, with accuracy ranging from 46.75% to 60%. This performance was consistently lower than that of the LSTM model, emphasizing the advantage of sequence-based learning for this task. Unlike the LSTM, which is designed to capture dependencies across time steps, the baseline approach likely struggles to capture temporal dynamics effectively, resulting in lower accuracy. The LSTM's structure, which explicitly models sequence information, allows it to achieve a substantial improvement over the baseline, providing strong predictive accuracy in each configuration.

Additionally, while the LSTM model's accuracy remains consistently high across different time slot numbers, there are slight variations that reflect the trade-offs in temporal granularity. As the number of time slots increases, the input sequence length grows, introducing additional temporal dependencies for the model to manage. Despite these fluctuations, the LSTM model demonstrates its adaptability, as it handles these dependencies with only minor changes in accuracy.

Overall, these results underscore the LSTM model's suitability for the feasibility prediction task, as it consistently outperforms the baseline across various time window configurations. The highest accuracy at 6-time slots highlights a potential optimal configuration, while the consistently strong results across other configurations confirm the LSTM's robustness. These findings emphasize the LSTM's ability to generalize effectively across different input structures, underscoring the value of sequence-based architectures for tasks involving temporal dependencies, such as delivery scheduling with flexible time windows.

4.5 Analysis of Training and Validation Results for LSTM-Based Network Across Multiple Time Slots for Profitability Check

The architecture for this profitability network, illustrated in Figure 6, leverages a curriculum-based training approach. In this approach, we reused a pre-trained network from the feasibility check as the initial layer, effectively transferring foundational learning from feasibility assessment to profitability assessment. This base network was then extended by concatenating it with an additional, untrained branch tailored specifically for the profitability classification task. The intent behind this architecture was to capitalize on the LSTM's strengths in sequence learning, allowing it to capture temporal dependencies in the data while benefiting from previously learned features related to feasibility. Here, the model can develop a deeper and more refined understanding of profitability-related patterns across sequential data points.

The architecture of the profitability network largely mirrors the sequence-based design used in the feasibility check. Specifically, it begins with an LSTM layer containing

128 hidden units, optimized for capturing the sequential nature of the input data. The state activation function in the LSTM layer is a Tanh function, which maintains stability in the range of activations, while the gate activation employs a Sigmoid function to regulate information flow. This is followed by a fully connected layer with 128 neurons, which refines the learned features from the LSTM output. ReLU activation functions are applied in this fully connected layer, enabling the model to capture complex, non-linear relationships within the data. The addition of an untrained branch allows the network to introduce profitability-specific learning while maintaining the generalizable patterns learned from feasibility, providing a dual focus on both feasibility and profitability assessment within a single architecture.

In terms of training and validation performance, the model achieved a validation accuracy of approximately 86.1% after 11,200 training iterations. This is a substantial improvement over the baseline accuracy of 48%, which was calculated by assuming all instances across different time slots as feasible for profitability. The baseline provides a reference level, reflecting the model's initial accuracy without any profitability-specific learning. In comparison, the validation accuracy of 86.1% highlights the model's ability to distinguish between profitable and non-profitable instances across various time slots effectively. The use of the pre-trained feasibility network as the starting layer played a crucial role in this improvement, as it allowed the model to build on pre-existing knowledge and focus its learning on profitability-related aspects more efficiently.

The progression of training and validation loss, as well as accuracy over the course of the training iterations, is depicted in Figure 11. The plot shows a steady increase in training accuracy, with validation accuracy following a similar upward trend, both of which indicate successful model learning and generalization. As the number of iterations increased, the accuracy curves converged toward the final performance level, illustrating the stability of the model and its capacity to handle diverse profitability scenarios across different time slots. Meanwhile, the loss plots reflect a consistent decrease in both training and validation loss, suggesting that the network is learning effectively without signs of overfitting. The smoothed accuracy lines also reinforce this interpretation, as they indicate that the model's performance remained stable even in the face of minor fluctuations.

Overall, the results demonstrate that the curriculum-based training approach, which combines feasibility-pretrained layers with profitability-specific learning, yields significant gains in validation accuracy over the baseline. The model's capacity to utilize pre-learned feasibility patterns and adapt them to profitability assessment has proven to be both efficient and effective, providing robust accuracy in profitability classification. This architecture demonstrates potential applicability in practical decision-making contexts, where both feasibility and profitability assessments are essential for informed operational choices across varied customer requirements and time slot configurations. Through this approach, the model is well-suited to evaluate profitability reliably.

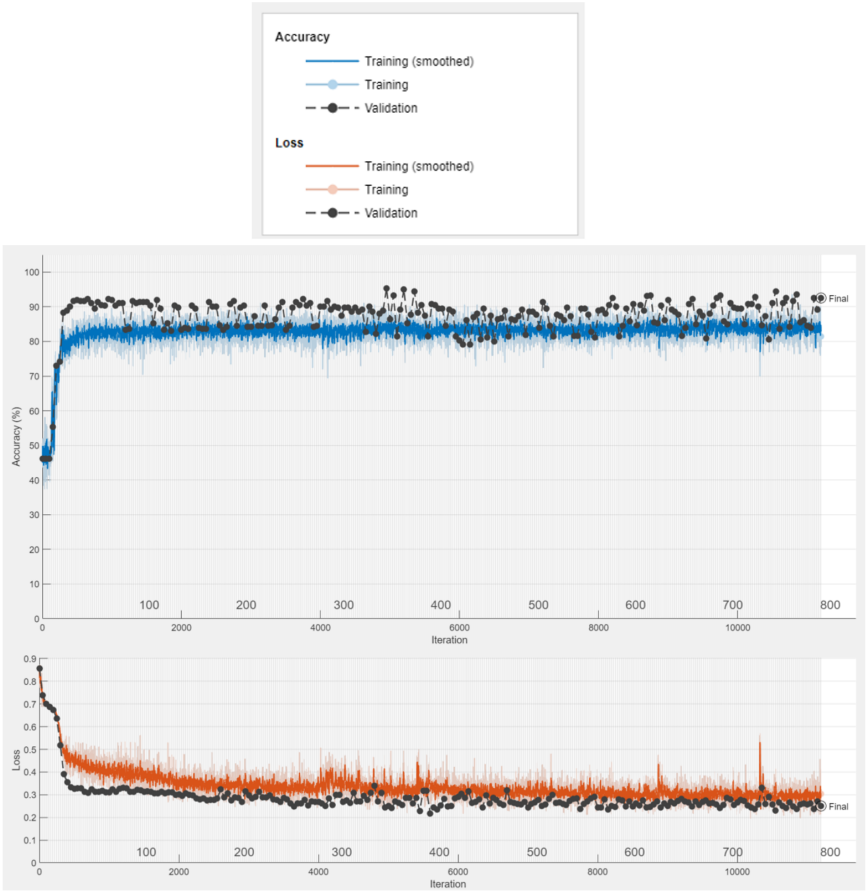

Figure 11: Training and validation accuracy and loss curves for the LSTM-based network for the profitability check with all aggregated data of different time slots

5 Conclusion

In this thesis, machine learning-based methodologies are developed and analyzed to solve the feasibility and profitability checks in delivery scheduling systems. The study focused on vehicle routing problems with time windows (VRPTW), a challenging operational framework where delivery feasibility and profitability must be assessed within strict temporal constraints. By leveraging supervised learning methods, particularly sequence-based models, we aimed to approximate complex constraints and provide real-time decision support for e-commerce and logistics applications.

This research introduced and evaluated a variety of classification methods for predicting the feasibility and profitability of customer delivery requests across multiple time slots. Comparative evaluations between classical classification methods—such as Naïve Bayes, K-Nearest Neighbors (KNN), Binary Decision Trees, and Ensemble Random Forests—and machine learning-based architectures, specifically Fully Connected Neural Networks (FCNN) and Long Short-Term Memory (LSTM) networks, demonstrated the effectiveness of LSTM models in capturing temporal dependencies in the VRPTW context.

The results showed that the LSTM-based network consistently outperformed classical methods and even surpassed the fully connected network, achieving substantial gains

in both feasibility and profitability prediction accuracy. For feasibility checking, the LSTM model achieved a validation accuracy significantly higher than baseline classical methods. The LSTM's recurrent architecture allowed it to effectively learn sequential patterns in delivery constraints, which are essential in VRPTW scenarios where the order and timing of deliveries are critical. In contrast, classical methods and the FCNN, which lack the ability to capture temporal dependencies, were less effective, with accuracy levels remaining considerably below those of the LSTM model.

In the profitability check, the LSTM model was further enhanced through a curriculum-based training approach. This method leveraged the knowledge from a pre-trained feasibility model and combined it with an additional branch specifically designed to assess profitability. By reusing features learned from the feasibility network, the curriculum-based LSTM model provided a strong initial foundation and achieved a validation accuracy of 86.1% after 11,200 training iterations, a significant improvement over the baseline profitability accuracy of 48%. This high accuracy highlights the model's capacity to accurately distinguish profitable delivery configurations across different time slots, effectively utilizing temporal data and learned feasibility patterns to inform profitability decisions.

The comparison with classical methods also highlights the strength of the LSTM model in this domain. Classical classifiers, despite their simplicity and speed, were limited by their inability to capture temporal relationships within the data. For example, K-Nearest Neighbors and Naïve Bayes classifiers achieved moderate performance but did not match the LSTM's accuracy due to their lack of sequence-based learning capabilities. Ensemble methods, such as Random Forests, performed slightly better among the classical approaches but still fell short in handling the complexity of time-dependent delivery constraints. The fully connected neural network (FCNN) also provided some improvements over classical methods, yet without recurrent connections, it was unable to effectively model the temporal dependencies that are essential for this application.

In summary, the LSTM model's sequence-based learning approach was uniquely well-suited for feasibility and profitability predictions within a time-dependent VRPTW framework. The model's superior performance across these metrics indicates that LSTM architectures can deliver substantial improvements in decision-making for dynamic, time-sensitive scheduling environments.

While the results indicate strong model performance, certain limitations remain. The computational requirements of training and inferring LSTM models may pose challenges, especially for large-scale or real-time applications where efficiency is critical. Additionally, this research assumes deterministic demand and delivery times, which may not fully capture the variability present in real-world operations. Future work could explore hybrid models that combine classical optimization techniques with LSTM architectures to address exact constraint satisfaction needs while maintaining fast computation times. Furthermore, integrating real-time traffic data, dynamic customer demand models, or stochastic elements could enhance the robustness of both feasibility and profitability predictions. Testing these methods across various datasets and delivery environments would also provide greater generalizability, making the models more applicable to diverse logistical scenarios.

Code and Data Availability

For reproducibility and further research, relevant code for model training and evaluation is provided in the following placeholders. These snippets illustrate the core implementations for feasibility and profitability checks:

Listing 1: SIMULATION FOR DATA GENERATION

```
import docplex.mp
import matplotlib.pyplot as plt
import seaborn as sns
import math
import numpy as np
import random
import time
import csv

random.seed(1010) # -> reproducible
# pre-define the simulation params
cust_size = 43
veh_max = 4

data_list_single = np.array([veh_max,df_orig['CAPACITY'][0],df_orig['
    XCOORD'][0],df_orig['YCOORD'][0],90])
# the data_list_single is redundant now!!
sel_col = ['XCOORD','YCOORD','DEMAND','READYTIME','DUETIME']
num_feasibility_check = 10

start_clock = time.time()
data_Y_store = []
data_X_store = []
for jj in range(num_feasibility_check):
    # RESET flag
    feasibility_flag = 0
    print(jj+1)
    #
    list_data = random.sample(range(1, len(df_orig)-1), cust_size)
    list_data.insert(0,0)
    ########
    df = df_orig.iloc[list_data]
    #
    df = df.reset_index(drop=True)
    ### save input data
    #sel_col = ['XCOORD','YCOORD','DEMAND','READYTIME','DUETIME']
    df_part = df[sel_col]
    flatten_df = df_part[1:].stack().values
    temp_col = np.concatenate((data_list_single,flatten_df))
    #if (jj == 0):
    #    data_X_all = temp_col[:,None]
    #else:
    #    data_X_all = np.concatenate((data_X_all,temp_col[:,None]),
    axis=1)
    data_X_store.append(flatten_df)
    ### --> define the VRP-TW problem
    n = cust_size
    Q = df['CAPACITY'][0]

    C = [i for i in range(1, n + 1)]
```

```
    Cc = [0] + C + [n + 1]
    V = [i for i in range(1, veh_max+1)]
    df2 = df.iloc[:, 1:3]
    df2.loc[n + 1, :] = df2.loc[0, :]
    dist_matrix = pd.DataFrame(distance_matrix(df2.values, df2.values
    ), index=df2.index, columns=df2.index)
    #
    e = [df['READYTIME'][i] for i in range(n + 1)]
    e.append(df['READYTIME'][0])
    #
    l = [df['DUETIME'][i] for i in range(n + 1)]
    l.append(df['DUETIME'][0])
    #
    ser = [df['SERVICETIME'][i] for i in range(n + 1)]
    ser.append(df['SERVICETIME'][0])
    #
    r = [df['DEMAND'][i] for i in range(n + 1)]
    r.append(0)
    # Variable set
    #X = [(i, j, k) for i in Cc for j in Cc for k in V if (i != j and
     j!=0 and i!=n+1)]
    X = [(i, j, k) for i in Cc for j in Cc for k in V if (i != j)]
    S = [(i, k) for i in Cc for k in V]
      # Calculate distance and time
    c = {(i, j): dist_matrix[i][j] for i in Cc for j in Cc}
    t = {(i, j): dist_matrix[i][j] for i in Cc for j in Cc}
    ## convert to a pure integer problem?
    cost_int = c
    ##
    en_int = 0
    if en_int ==1:
        for k, v in cost_int.items():
            cost_int[k] = int(v)
        ## convert to a pure integer problem?
        t_int = t
        for k, v in t_int.items():
            t_int[k] = int(v)
    ##
    #time_start = time.time()
    mdl = Model('VRPTW')
    # Variables
    x = mdl.binary_var_dict(X, name='x')
    s = mdl.integer_var_dict(S,0,l[0], name='s')

    K = 10000 # is a big number
    # Cc =  [0,1,...,n+1], V = [1,2,...,25]

    # (3.2) each customer is visited at least one time/
    mdl.add_constraints(mdl.sum(x[i, j, k] for j in Cc for k in V if
    j != i) == 1 for i in C)
    # (3.3) maximum cap
    mdl.add_constraints(mdl.sum(r[i] * (x[i, j, k]) for i in C for j
    in Cc if i != j) <= Q for k in V)
    # (3.4) (3.5) (3.6) each vehicle departs from depot 0, leave to
    another node after ariving the current node and end at n+1
    # minimize k
    mdl.add_constraints(mdl.sum(x[0, j, k] for j in Cc if j != 0) <=
    1 for k in V)
```

```
    #
    mdl.add_constraints((mdl.sum(x[i, p, k] for i in Cc if i != p) -
    mdl.sum(x[p, j, k] for j in Cc if p != j)) == 0 for p in C for k
    in  V)
    mdl.add_constraints(mdl.sum(x[i, n + 1, k] for i in Cc if i != n
    + 1) <= 1 for k in V)

    # arriving time
    mdl.add_constraints(s[i, k] + ser[i] + t[i, j] - K * (1 - x[i, j,
     k]) - s[j, k] <= 0 for i, j, k in X if i != j)
    #
    mdl.add_constraints(s[0, k] == 0 for k in V)

    mdl.add_constraints(s[i, k] >= e[i] for i, k in S if i != 0)

    mdl.add_constraints(s[i, k] <= l[i] for i, k in S if i != 0)
    # veh cost of rental = 20
    veh_cost = 20
    en_only_find_feasible_solution = 1
    if (en_only_find_feasible_solution == 1):
        obj_function = 0
    else:
        obj_function = mdl.sum((cost_int[i, j]+veh_cost) * x[i, j, k]
     for i, j, k in X)
    mdl.parameters.timelimit.set(50)
    # Solve
    mdl.minimize(obj_function)
    #
    solution = mdl.solve(log_output = False)
    solve_status = mdl.get_solve_status()
    print(solve_status.name)
    if (solution!=None):
        feasibility_flag = 1
    print('feasibility_flag = ',feasibility_flag)
    data_Y_store.append([feasibility_flag])
    #time_end = time.time()
    # print(solution)

    #running_time = round(time_end - time_solve, 2)
    #elapsed_time = round(time_end - time_start, 2)
##
elapsed_time = time.time() - start_clock
print('Execution time:', elapsed_time, 'seconds')
```

References

[1] M. Riedmiller and A. Lernen, "Multi layer perceptron," *Machine Learning Lab Special Lecture, University of Freiburg*, pp. 7–24, 2014.

[2] I. Banerjee, Y. Ling, M. C. Chen, S. A. Hasan, C. P. Langlotz, N. Moradzadeh, B. Chapman, T. Amrhein, D. Mong, D. L. Rubin *et al.*, "Comparative effectiveness of convolutional neural network (cnn) and recurrent neural network (rnn) architectures for radiology text report classification," *Artificial intelligence in medicine*, vol. 97, pp. 79–88, 2019.

[3] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE transactions on systems, man, and cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.

[4] M. Pal, "Random forest classifier for remote sensing classification," *International journal of remote sensing*, vol. 26, no. 1, pp. 217–222, 2005.

[5] S.-C. Ho, R. J. Kauffman, and T.-P. Liang, "A growth theory perspective on b2c e-commerce growth in europe: An exploratory study," *Electronic Commerce Research and Applications*, vol. 6, no. 3, pp. 237–259, 2007.

[6] K. C. Laudon and C. G. Traver, *E-commerce*. Pearson Boston, MA, 2013.

[7] V. Alfonso, C. Boar, J. Frost, L. Gambacorta, and J. Liu, "E-commerce in the pandemic and beyond," *BIS Bulletin*, vol. 36, no. 9, 2021.

[8] S. Park and K. Lee, "Examining the impact of e-commerce growth on the spatial distribution of fashion and beauty stores in seoul," *Sustainability*, vol. 13, no. 9, p. 5185, 2021.

[9] N. Agatz, A. Campbell, M. Fleischmann, and M. Savelsbergh, "Time slot management in attended home delivery," *Transportation Science*, vol. 45, no. 3, pp. 435–449, 2011.

[10] H. Yrjo *et al.*, "Physical distribution considerations for electronic grocery shopping," *International Journal of Physical Distribution & Logistics Management*, 2001.

[11] Amazon.com E-commerce company, "About delivery preferences," [Accessed on 07 October 2022]. [Online]. Available: https://www.amazon.sg/gp/help/customer/display.html?nodeId=201910450

[12] PostNord AB , "Choose how to receive your delivery," [Accessed on 07 October 2022]. [Online]. Available: https://www.postnord.se/en/receiving/choose-how-to-receive-your-delivery

[13] F. Akkerman, "Delivery cost approximations for dynamic time slot pricing," Master's thesis, University of Twente, 2021.

[14] S. N. Kumar and R. Panneerselvam, "A survey on the vehicle routing problem and its variants," 2012.

[15] Akamai Technologies, "Akamai online retail performance report: Milliseconds are critical," 2017.

[16] L. van der Hagen, N. Agatz, R. Spliet, T. R. Visser, and A. L. Kok, "Machine learning-based feasability checks for dynamic time slot management," *Available at SSRN 4011237*, 2022.

[17] C. Köhler, J. F. Ehmke, and A. M. Campbell, "Flexible time window management for attended home deliveries," *Omega*, vol. 91, p. 102023, 2020.

[18] J. Zhao, Q. Liu, W. Wang, Z. Wei, and P. Shi, "A parallel immune algorithm for traveling salesman problem and its application on cold rolling scheduling," *Information Sciences*, vol. 181, no. 7, pp. 1212–1223, 2011.

[19] J. Li, X. Meng, and X. Dai, "Collision-free scheduling of multi-bridge machining systems: a colored traveling salesman problem-based approach," *IEEE/CAA Journal of Automatica sinica*, vol. 5, no. 1, pp. 139–147, 2017.

[20] X. Meng, J. Li, M. Zhou, X. Dai, and J. Dou, "Population-based incremental learning algorithm for a serial colored traveling salesman problem," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 2, pp. 277–288, 2016.

[21] S. Twaróg, K. Szwarc, M. Wronka-Pośpiech, M. Dobrowolska, and A. Urbanek, "Multiple probabilistic traveling salesman problem in the coordination of drug transportation—in the context of sustainability goals and industry 4.0," *Plos one*, vol. 16, no. 3, p. e0249077, 2021.

[22] P. Baniasadi, M. Foumani, K. Smith-Miles, and V. Ejov, "A transformation technique for the clustered generalized traveling salesman problem with applications to logistics," *European Journal of Operational Research*, vol. 285, no. 2, pp. 444–457, 2020.

[23] A. S. Elgesem, E. S. Skogen, X. Wang, and K. Fagerholt, "A traveling salesman problem with pickups and deliveries and stochastic travel times: An application from chemical shipping," *European Journal of Operational Research*, vol. 269, no. 3, pp. 844–859, 2018.

[24] F. Nekovář, J. Faigl, and M. Saska, "Multi-tour set traveling salesman problem in planning power transmission line inspection," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6196–6203, 2021.

[25] D. Gusfield, "Integer linear programming in computational biology: Overview of ilp, and new results for traveling salesman problems in biology," *Bioinformatics and Phylogenetics*, pp. 373–404, 2019.

[26] M. Desrochers, J. Desrosiers, and M. Solomon, "A new optimization algorithm for the vehicle routing problem with time windows," *Operations research*, vol. 40, no. 2, pp. 342–354, 1992.

[27] M. Keskin, B. Çatay, and G. Laporte, "A simulation-based heuristic for the electric vehicle routing problem with time windows and stochastic waiting times at recharging stations," *Computers & Operations Research*, vol. 125, p. 105060, 2021.

[28] N. Agatz, A. M. Campbell, M. Fleischmann, J. Van Nunen, and M. Savelsbergh, "Revenue management opportunities for internet retailers," *Journal of Revenue and Pricing Management*, vol. 12, no. 2, pp. 128–138, 2013.

[29] A. M. Campbell and M. Savelsbergh, "Incentive schemes for attended home delivery services," *Transportation science*, vol. 40, no. 3, pp. 327–341, 2006.

[30] R. Klein, M. Neugebauer, D. Ratkovitch, and C. Steinhardt, "Differentiated time slot pricing under routing considerations in attended home delivery," *Transportation Science*, vol. 53, no. 1, pp. 236–255, 2019.

[31] X. Yang, A. K. Strauss, C. S. Currie, and R. Eglese, "Choice-based demand management and vehicle routing in e-fulfillment," *Transportation science*, vol. 50, no. 2, pp. 473–488, 2016.

[32] C. Cleophas and J. F. Ehmke, "When are deliveries profitable?" *Business & Information Systems Engineering*, vol. 6, no. 3, pp. 153–163, 2014.

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

[34] T. R. Visser and R. Spliet, "Efficient move evaluations for time-dependent vehicle routing problems," *Transportation science*, vol. 54, no. 4, pp. 1091–1112, 2020.

[35] C. Sommer, "Shortest-path queries in static networks," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, pp. 1–31, 2014.

[36] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: a methodological tour d'horizon," *European Journal of Operational Research*, vol. 290, no. 2, pp. 405–421, 2021.

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

[38] J. F. Ehmke and A. M. Campbell, "Customer acceptance mechanisms for home deliveries in metropolitan areas," *European Journal of Operational Research*, vol. 233, no. 1, pp. 193–207, 2014.

[39] E. Larsen, S. Lachapelle, Y. Bengio, E. Frejinger, S. Lacoste-Julien, and A. Lodi, "Predicting tactical solutions to operational planning problems under imperfect information," *INFORMS Journal on Computing*, vol. 34, no. 1, pp. 227–242, 2022.

[40] C. Köhler and J. Haferkamp, "Evaluation of delivery cost approximation for attended home deliveries," *Transportation Research Procedia*, vol. 37, pp. 67–74, 2019.

[41] D. Bzdok, M. Krzywinski, and N. Altman, "Machine learning: supervised methods," *Nature methods*, vol. 15, no. 1, p. 5, 2018.

[42] M. Desrochers and G. Laporte, "Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints," *Operations Research Letters*, vol. 10, no. 1, pp. 27–36, 1991.

[43] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations research*, vol. 35, no. 2, pp. 254–265, 1987.

[44] D. P. Kingma, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.