FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Solving Dynamic Inventory-Routing Problems Through Supervised Learning

André Filipe Cardoso Barbosa

Mestrado em Engenharia Informática e Computação

Supervisor: Prof. Gonçalo Figueira

Second Supervisor: Prof. Fábio Neves-Moreira

July 28, 2025

Solving Dynamic Inventory-Routing Problems Through Supervised Learning

André Filipe Cardoso Barbosa

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

President: Prof. Henrique Lopes Cardoso
Referee: Prof. João Pedroso
Referee: Prof. Gonçalo Figueira

July 28, 2025

Resumo

No panorama atual da indústria, as organizações são cada vez mais desafiadas pela crescente complexidaded das decisões operacionais. Desde a área de logística operacional até às cadeias de abastecimento globais, a minimização dos custos totais mantendo um serviço eficiente requere políticas de decisão que, para além de exatas, sejam adaptáveis à incerteza do mundo real.

O Problema de Roteamento de Inventário integra a gestão de inventário e o roteamento de veículos em conjunto. Introduzindo incerteza e uma tomada de decisão sequencial, o problema evolui para o Problema de Roteamento de Inventário Dinâmico e Estocástico (DSIRP). Neste cenário, a procura dos clientes é atualizada ao longo do tempo é utilizada para apoiar políticas de decisão dinâmicas informadas. Cada decisão inclui os clientes a visitar, a quantidade a entregar e a forma de os combinar numa rota.

Recentemente, desafios relacionados com a escalabilidade, complexidade computacional e a necessidade de ajustar as políticas actuais às informações obtidas em tempo real têm fomentado um interesse crescente na aplicação de *Machine Learning* a problemas de otimização combinatória dinâmica, incluindo o DSIRP. Neste contexto, esta dissertação tem como objetivo explorar a aplicação de *Supervised Learning*, especificamente para apoiar a decisão de quais os clientes a visitar em cada período de decisão. O objetivo é avaliar se a aprendizagem a partir de soluções passadas de elevada qualidade pode aproximar o comportamento de especialistas em situações de incerteza, reduzindo os custos computacionais.

A solução proposta baseia-se numa política de decisão em três etapas. Primeiro, um modelo de *Supervised Learning* seleciona os clientes que devem ser visitados. Em segundo lugar, a quantidade de entrega é calculada utilizando duas regras clássicas de gestão de inventário. Em terceiro lugar, a rota do veículo é optimizada através da resolução de um Problema do Caixeiro Viajante para os clientes selecionados. Foi seguido um processo de desenvolvimento iterativo, em que vários modelos foram treinados utilizando diferentes conjuntos de caraterísticas, aperfeiçoados ao longo do tempo. O seu desempenho foi avaliado utilizando um ambiente de simulação personalizado, que permitiu comparações baseadas nos custos com soluções de referência calculadas com pleno conhecimento da procura futura.

Por fim, os resultados obtidos sugerem que, embora os modelos tenham melhorado de forma consistente ao longo das iterações e superado uma política de base da literatura, ainda ficaram aquém de políticas de referência mais complexas. No entanto, *Supervised Learning* é um método promissor para a aprendizagem de políticas de decisão utilizando recursos computacionais limitados, embora sejam necessários mais aperfeiçoamentos para reduzir a diferença de desempenho em relação às estratégias com melhor desempenho.

Abstract

In today's industrial landscape, organizations are increasingly challenged by the growing complexity of operational decisions. From operational logistics to global supply chains, minimizing total costs while maintaining efficient service levels requires decision policies that are not only accurate but also adaptive to real-world uncertainty.

The Inventory-Routing Problem integrates inventory management and vehicle routing into a unified framework. When uncertainty and sequential decision-making are introduced, the problem evolves into the Dynamic and Stochastic Inventory Routing Problem (DSIRP). In this scenario, customer demands are progressively revealed over time and are leveraged to support informed and proper decisions. Each decision includes which customers to visit, how much to deliver, and how to combine the orders into a vehicle route.

Recent challenges related to scalability, computational complexity, and the need to adjust to real-time information realizations have fostered growing interest in applying Machine Learning to dynamic combinatorial optimization problems, including the DSIRP. In this context, this dissertation aims to explore the application of Supervised Learning, specifically to support the decision of which customers to visit in each decision point. The goal is to evaluate whether learning from past high-quality solutions can approximate expert behavior under uncertainty while reducing computational costs.

The proposed solution is based on a three-step decision policy. First, a trained Supervised Learning model selects which customers should be visited. Second, the delivery quantity is computed using two classical inventory management rules. Third, the vehicle route is optimized by solving a Traveling Salesman Problem for the selected customers. Models were trained in an iterative development process, using different feature sets refined over time. Their performance was evaluated using a custom simulation environment, which enabled cost-based comparisons against reference solutions computed under full knowledge of future demand.

Finally, the obtained results suggest that while the models consistently improved across iterations and outperformed a baseline from the literature, they still fell short of more complex benchmark policies. Nonetheless, Supervised Learning is a promising method for learning decision policies using limited computational resources, though further refinements are needed to close the performance gap to the best-performing strategies.

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework to achieve a better and more sustainable future for all. It includes 17 goals to address the world's most pressing challenges, including poverty, inequality, climate change, environmental degradation, peace, and justice. The specific Sustainable Development Goals mentioned have the following names:

SDG 8 Promote sustained, inclusive and sustainable economic growth, full and productive employment and decent work for all

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation

SDG 12 Ensure sustainable consumption and production patterns

SDG 17 Strengthen the means of implementation and revitalize the Global Partnership for Sustainable Development

SDG	Target	Contribution	Performance Indicators and Metrics
8	8.2	Increasing economic productivity by introducing machine learning models that enhance the responsiveness and adaptability of industrial scheduling.	Improvements in resource utilization
8	8.4	Promoting resource efficiency in production by reducing delivery redundancies, optimizing inventory levels, and avoiding unnecessary transportation.	Reduction in delivery mileage under dynamic conditions
9	9.4	Upgrading scheduling infrastructure in manufacturing by integrating intelligent, real-time decision-making based on historical data and predictive modeling.	Increase in scheduling accuracy and reduction in reprocessing operations
9	9.5	Enhancing scientific research and innovation capacity by developing supervised learning approaches tailored to dynamic industrial environments.	Number of research outputs, model prototypes, and applied evaluation scenarios
12	12.2	Supporting the efficient use of natural and logistical resources through optimized routing strategies and adaptive replenishment policies.	Simulation results showing lower transport costs and improved inventory turnover
17	17.6	Strengthening academic-industry cooperation by sharing technological knowledge, aligning research goals with real-world constraints, and co-developing scalable solutions.	Participation in knowledge transfer sessions

Acknowledgements

First, I would like to express my appreciation for all the availability, guidance, and support provided by Professor Gonçalo Figueira and Professor Fábio Neves-Moreira throughout the development of my dissertation. I am very grateful to them. I would also like to extend my gratitude to Engineer Francisco Maia for the continuous feedback and patience in helping me with concepts I was initially unfamiliar with, like the fundamentals of the Dynamic and Stochastic Inventory-Routing Problem, among others. Their contributions were essential to the completion of this work.

I would like to thank INESC-TEC for allowing me to carry out my dissertation at their facilities. I am also thankful to the Faculty of Engineering of the University of Porto and its professors for giving me the opportunity to expand my knowledge and challenge myself across diverse areas, always being accessible and willing to support.

I want to express my appreciation to all my friends and fellow classmates, who supported and helped me whenever needed, namely Pedro Macedo and Ricardo Ribeiro, who must have heard countless descriptions of my dissertation topic, along with many others who have left a lasting mark on my life.

I would also like to thank all my friends outside of my course for their endless support and encouragement throughout these five years. Their friendship offered balance and perspective to my life, which allowed me to relax during more intense moments of this journey.

At last, and most importantly, I thank my family, my parents, Cristina and António Barbosa, my brother, Bruno Barbosa, and so many others too numerous to mention. The immense belief they had in me has been immeasurable, and it has continuously inspired and motivated me to keep pushing forward.

André Barbosa

"Success doesn't mean the absence of failures; it means the attainment of ultimate objectives. It means winning the war, not every battle."

Edwin Bliss

Contents

List of Figures

List of Tables

List of Acronyms

ADP	Approximate Dynamic Programming
ANN	Artificial Neural Network
BP	Back Propagation
CO	Combinatorial Optimization
CRISP-DM	Cross Industry Standard Process for Data Mining
DNN	Deep Neural Network
DP	Dynamic Programming
DSIRP	Dynamic and Stochastic Inventory-Routing Problem
DVRP	Dynamic Vehicle Routing Problem
EOQ	Economic Order Quantity
FF	Feed Forward
GBDT	Gradient Boosted Decision Trees
IRP	Inventory-Routing Problem
LGBM	Light Gradient Boosting Machine
LSTM	Long Short-Term Memory
MaxL	Max Level
MLE	Maximum Likelihood Estimation
ML	Machine Learning
MLR	Multiple Linear Regression
MIP	Mixed-Integer Programming
MP	Mathematical Programming
NN	Neural Network
OR	Operations Research
OU	Order-Up-To
PH	Perfect Hindsight
QR	Quantile Regression
RBF	Radial Basis Function
RPR	Regularized Poisson Regression
RL	Reinforcement Learning
RTVRP	Real Time Vehicle Routing Problem
SKU	Stock Keeping Unit
SL	Supervised Learning
SVM	Support Vector Machines
TSP	Travelling Salesman Problem
VRP	Vehicle Routing Problem

Chapter 1

Introduction

1.1 Contextualization

The day-to-day operations of industries involve complex sequential decision-making, where choices made at one point in time directly influence future decisions and outcomes. Managing these dependencies remains a central challenge across various application domains, such as supply chain logistics, manufacturing systems, transportation networks, and service operations. These settings often require identifying interconnected decisions that minimize long-term costs or maximize overall efficiency over time.

Within the Operations Research (OR) field, many of these challenges have been modeled as optimization problems. Examples include the Inventory Management Problem [26], which aims to determine optimal replenishment policies to meet demand; the Vehicle Routing Problem (VRP) [18], which focuses on finding the most efficient routes to serve multiple locations with limited resources; and the Delivery Scheduling Problem [29], which aims to determine the best timing for dispatching goods. These three problems, traditionally treated separately, are combined in the Inventory-Routing Problem (IRP) [8], which addresses (1) what customers to visit, (2) how much to deliver to each one, and (3) how to combine them into a vehicle route. The IRP thus integrates inventory control and vehicle routing into a unified decision-making framework.

Traditionally, many of these problems have been modeled assuming all relevant information is known beforehand, a setting commonly referred to as *static*. Such models can be effectively addressed using Mathematical Programming (MP) or Constraint Programming, which are capable of computing optimal solutions globally under deterministic conditions. In contrast, when information is gradually revealed and decisions must adapt over time, problems are better framed as *dynamic*. For instance, Dynamic Programming (DP) addresses this by decomposing the problem into smaller subproblems, solving them recursively [9].

This dynamic perspective becomes essential in real-world applications like the IRP, which faces uncertainty in demand, supply availability, or travel conditions. These factors are typically modeled using probabilistic distributions, introducing *stochasticity* into the formulation. Therefore, when both uncertainty and sequential decision-making are present, the problem is formalized

as a Dynamic and Stochastic Inventory-Routing Problem (DSIRP) [16], a more complex but realistic setting in which planners must adapt decisions over time as new information becomes available, i.e., customer demands.

In recent years, the increasing need for scalability, real-time performance, and adaptability has shifted attention away from exact methods such as MP and DP, especially for large-scale or time-sensitive problems. Heuristic and meta-heuristic methods, such as Adaptive Large Neighborhood Search [46], Tabu Search [21], and Genetic Algorithms [42], have emerged to provide reasonable quality solutions efficiently. In parallel, Approximate Dynamic Programming (ADP), as developed by Powell [41], has emerged as a powerful framework for solving high-dimensional, sequential decision problems under uncertainty, by approximating value functions rather than solving them exactly. More recently, Machine Learning (ML) approaches have gained increasing attention in this domain. Reinforcement Learning (RL), in particular, has emerged as a promising approach in OR, offering theoretical support for solving sequential decision-making problems under uncertainty, typically modeled as Markov Decision Processes (MDP) [50].

1.2 Motivation

Solving the DSIRP is critical for industries that rely on efficient and scalable decision-making under uncertainty, such as retail, logistics, and healthcare. While RL methods and other approximate strategies can solve these problems, they still face significant limitations regarding computational complexity, convergence times, and operational simplicity. An appealing alternative is to leverage Supervised Learning (SL) to learn from high-quality decisions computed *a priori* by an expert or solver, such as an optimizer with Perfect Hindsight (PH). This learning-by-imitation [47], commonly associated with SL, provides a pathway to develop decision models that are fast, scalable, and easily deployable, as they avoid the need for costly real-time optimization or training loops. Additionally, using SL allows for explicitly incorporating structured knowledge from deterministic solvers into a learning pipeline, enabling the model to generalize to unseen states in a dynamic and stochastic context. Therefore, SL is a promising approach worth exploring in this problem. It also provides an opportunity to evaluate whether such models can approximate optimal behaviors under uncertainty, while meeting the practical requirements of industries that require quick and reliable decision support tools. This dissertation explores the application of SL to solve the DSIRP, comparing the performance obtained with current literature policies. The learned models are evaluated against existing approaches in the literature, with the goal of assessing their ability to replicate high-quality decisions while maintaining computational efficiency.

1.3 Research Questions

This dissertation explores the capabilities of SL in addressing the DSIRP. The work goes beyond developing SL solutions by evaluating their performance against classical heuristics and the PH. To guide this investigation, we focus on the following research questions:

- **RQ 1:** How can we design a system that applies Supervised Learning on the DSIRP?
  We intend to analyze how other similar problems approach the use of SL and adapt the design to our current problem. Main decisions will include deciding what to learn and where we will apply it in the complex DSIRP system.
- **RQ 2:** How much do the Supervised Learning application results compare to the literature?
  We aim to compare the performance between SL and the literature approaches. Optimality comparison will allow us to validate the results obtained.

1.4 Dissertation Structure

In addition to the introduction (Chapter 1), which briefly presents the context, motivation, and research questions, this dissertation contains five more chapters. In Chapter 2, the state of the art on subjects such as inventory-routing and connection with ML is described, and related work is presented. Chapter 3 details the characteristics of the problem, covering a general description of it, corresponding notation, and transformation into a sequential decision process. The methodological approach is described in Chapter 4, first with an overview of all steps taken and then the specifics on the decision policies for the DSIRP. Chapter 5 reflects the iterative development of the model and results obtained from its simulation, while detailing the experimental setup and datasets used. At last, Chapter 6 presents the dissertation's conclusions and options for future work.

Chapter 2

Literature Review

This chapter presents a structured review of the current research on applying SL methods to the DSIRP. The study on the topic is recent and remains underexplored. For this reason, the scope is expanded across three complementary research paths. The first path includes the origin of the DSIRP and examination of its problem dimensions. The second path comprises existing solution approaches, including reactive and proactive policies, and establishing comparative baselines for our work. Finally, by analyzing the successful applications of SL in inventory management and vehicle routing problems, we seek to evaluate its potential to tackle the DSIRP. This includes identifying where SL can be applied, which attributes are used for predictions, and investigating additional factors that could inform its application.

2.1 Material collection

To better understand the existing literature on DSIRPs, we refer to a comprehensive review that systematically surveyed the field [35]. That review explores the current research on DSIRPs, divided into problem dimensions and decision policies. It is the primary source of literature for this thesis. In addition, we broaden the scope from the DSIRP to related problems to identify applications of SL. This decision was motivated by the limited literature on using SL for the DSIRP specifically, as well as the potential benefits of decomposing the problem into two well-studied subproblems: Inventory Management and Vehicle Routing. To explore this, we conducted a search in Scopus using the following query: ("dynamic" OR "real-time" OR "online") AND ("routing" OR "inventory") in the paper's title AND ("learning" OR "data-driven" OR "classification" OR "predictive" OR "prediction" OR "neural" AND "networks") in the paper's keywords. That query yielded 297 journal articles. Subsequently, we applied a set of exclusion criteria:

- Absence of any application of SL in any experiment stage described in the paper.
- Applications not relevant in Inventory or Routing, including problems, such as wireless network routing or flood detection, and datasets, such as image datasets.

Applying these criteria reduced the list to 22 papers discussed in the remainder of this section. Out of those, ten were related to inventory problems, summarized in Table 2.1, and twelve with routing problems, covered in Table 2.2. Each paper is characterized in terms of the SL model employed and the domain of learning (**D** for delivery quantities, **R** for routing decision, **P** for problem parameters).

2.2 Problem Dimensions

To establish the foundations of the DSIRP, it is necessary to begin with its simpler predecessor, IRP. The first publication describing the IRP dates back to 1983 with Bell et al. [8]. Since its introduction, multiple variants of the problem have emerged.

A fundamental characteristic of the IRP is its endpoint structure, depending on the number of depots and customers. For instance, Jia, Schrotenboer, and Chen [28] and Greif et al. [22] explore the one-to-many structure, while Markov et al. [36] research many-to-many and Violi et al. [51] approach a many-to-one-to-many network.

Regarding inventory, different features can vary. Stockouts can be managed through approaches like lost sales or backlogging. Recent studies differ in how they model unmet demand, such as Jia, Schrotenboer, and Chen [28] and Greif et al. [22]. The first emphasize backlogging, where unmet demand is postponed to future periods, incurring shortage costs; the second explores applying penalties on the price of the unmet demand, but losing that demand. Furthermore, another important aspect to consider is the variety of products. Incorporating multiple items or products increases computational complexity, with most models focusing on single-item scenarios [17]. In multi-item contexts, substitution emerges as a viable strategy for stockouts [1].

Perishability adds another layer of complexity, with items classified by age and shelf life. Violi et al. [51] explore agri-food supply chains, integrating perishable products with uncertain demands. Inventory replenishment policies further define operational approaches. Coelho, Cordeau, and Laporte [14] apply and describe the two most common policies, Order-Up-To (OU) Level and Max Level (MaxL). The OU policy mandates replenishment to a fixed capacity whenever a customer is visited, ensuring consistency and simplicity in decision-making. The MaxL policy, on the other hand, offers flexibility, allowing delivery quantities up to the maximum inventory capacity.

Regarding routing features, the logistical context spans scenarios involving exclusive pickups or deliveries and simultaneous or non-simultaneous pickups and deliveries (P&D). Traditional IRPs primarily focus on deliveries from a central depot to customers, as described by Federgruen and Zipkin [20]. Conversely, exclusive pickup scenarios, such as waste collection, involve transporting items from multiple locations back to a central depot [40]. Neves-Moreira et al. [39] include simultaneous P&D that allows customers to receive and dispatch products during the same period. Transshipment-enabled models, such as those proposed by Mirzapour Al-e-hashem, Rekik, and Mohammadi Hoseinhajlou [38], integrate routing strategies to manage uncertainties, reduce greenhouse gas emissions, and optimize costs. Emergency transshipments further complement routing features by addressing stockouts through lateral transshipments, subcontracted

carrier interventions, or emergency collections that reset inventory to zero, as explored by Markov et al. [36].

The dynamic component has gained increasing attention recently, due to the resemblance with real-world scenarios and applicability, where information is disclosed over time [14]. Due to the sequential and uncertain nature of the problem, a single precomputed solution is no longer sufficient; instead, a policy must be learned or computed to guide decisions over time. For instance, Roldán, Basagoiti, and Coelho [45] extended Coelho, Cordeau, and Laporte [14]'s work by addressing a robust approach to customer selection and inventory replenishment rules. The authors explored two replenishment policies: fixed quantity and (s,S), and three priority rules. Afterwards, the vehicle route was computed by solving a Traveling Salesman Problem (TSP).

Leveraging the ability to learn decision policies by minimizing long-term discounted costs leads to the increasingly common use of ML algorithms. Policy and value iteration, instances of RL algorithms, are employed offline and online, where simulation optimization helps fine-tune parameters and evaluate policies. For instance, McKenna, Robbins, Lunday, et al. [37] adopt an approximate policy iteration algorithm with least squares temporal differencing for policy evaluation, demonstrating improved decision-making strategies. Other studies, like Brinkmann, Ulmer, and Mattfeld [10], employ dynamic lookahead policies where dynamic horizons are computed using Bellman's equation and ADP to anticipate future demands more effectively.

2.3 Supervised Learning on the DSIRP

Jia, Schrotenboer, and Chen [28] introduce a Scenario Predict-then-Optimize framework that bridges prediction and optimization in solving the DSIRP. This framework employs a Multi-Horizon Quantile Recurrent Neural Network (NN) to predict quantiles of future demand. These predictions form scenarios that feed into a two-stage stochastic programming model, which jointly optimizes inventory replenishment and vehicle routing. Scenario Predict-then-Optimize focuses on generating cost-efficient decisions without relying on assumptions about demand distribution, enhancing flexibility and adaptability. By incorporating quantile forecasts instead of single-point predictions, the approach reduces sensitivity to forecast errors.

Greif et al. [22] research takes a hybrid ML and CO approach by integrating a Physics-Informed Neural Network and the Capacitated Prize Collecting Traveling Salesman Problem (CPCTSP). The first predicts a prize vector that parameterizes the second, aligning solutions with optimal replenishment and routing decisions by encoding inventory and routing problems into the prize vector.

Both approaches represent distinct strategies for integrating prediction with optimization in DSIRPs. While the former emphasizes robust, quantile-based scenario generation, the latter leverages dynamic adjustments through learned prize vectors, particularly in single-vehicle and contextual demand scenarios. Both frameworks effectively balance computational complexity with solution quality, leveraging real-time decision-making under stochastic demand.

2.4 Supervised Learning in Inventory Management

A promising approach to applying SL in inventory management involves forecasting market dynamics, such as fluctuations in sales, prices, and other economic indicators, that influence inventory decisions. Liang [34] employs intelligent prediction models, including Backpropagation neural networks and Support Vector Machines (SVM), to forecast future market indexes like consumer income, interest rates, product prices, and inventory costs, using historical data. These predictions help optimize dynamic pricing and inventory control by aligning sales prices and order quantities with future conditions. Li et al. [32] go further, with an ensemble of multiple models and multimodal inputs such as sales history, inventory levels, and promotional data. These models predict sales while minimizing stock-out and overstock costs, supported by elastic-adjustment mechanisms that adapt predictions to inventory states.

Beyond market-driven forecasting, multiple contributions explore the design of decision policies using SL models. Akkerman, Prak, and Mes [4], who explore the Inventory Record Inaccuracy problem, use inventory positions, inspection intervals, and item-specific attributes to estimate delivery quantities, particularly focusing on reorder decisions. To this end, a neural network is used to predict reorder probabilities in multi-item inventory scenarios, aiming to optimize cost-effectiveness.Asadi and Pinkley [6] also predict a value function, within a monotone ADP framework. Linear Regression (LR) models map system states, such as battery levels, to value function estimates. However, this decision is made as an initial point to an ADP model that further solves the problem.

A central concern in inventory management is the balance between understock and overstock. Ren et al. [43] searches for an optimal inventory point, through an integrated forecasting-optimization approach, leveraging ML models such as Long Short-Term Memory (LSTM), Light Gradient Boosting Machine (LGBM), and Artificial Neural Network (ANN) models, all incorporating Quantile Regression (QR) to directly predict optimal inventory levels for pickup points. These models incorporate features like sales and promotion history. A different approach is used by Reyes-Aldasoro et al. [44], who instead search for a threshold of inventory needed in a week.

Demand forecasting plays a critical role in addressing inventory problems across various domains. Li et al. [33] employ a Regularized Poisson Regression (RPR) model enhanced by Maximum Likelihood Estimation (MLE), incorporating indicators such as seasonality, Stock Keeping Unit (SKU), and store identifiers. It aims to dynamically learn demand distributions from sales data to optimize delivery quantities and maximize expected revenue. Haixiang et al. [24], in contrast, use a Back Propagation(BP) NN for a sewage recycling VRP, using historical data. Fan et al. [19] and Yang et al. [54] both utilize LGBM, with Fan et al. [19] applying it to tensor demand data for spare parts with intermittent time series, enhanced by statistical methods like Syntetos–Boylan Approximation for safety stock adjustments. In contrast, Yang et al. [54] complement LGBM with ANN and Gradient Boosted Decision Trees (GBDT), leveraging ensemble diversity to improve generalization across transactional contexts. This ensemble predicts transaction-driven demand, where features such as promotional markdowns and store characteristics are used to opti-

mize inventory and financing decisions, making predicted demand a critical input for optimization problem parameters.

Table 2.1: Inventory applications of Supervised Learning

Author	Year	Model	Input	Output
[44] Reyes-Aldasoro et al.	1999	NN	Thresholds for every week	Threshold
[34] Liang	2015	BPNN, SVM	Market Data	Inventory Price & Cost
[24] Haixiang et al.	2021	BPNN	Historical Sewage Data	Sewage Production
[33] Li et al.	2021	RPR+MLE	Seasonality, SKU and Store Indicators	Demand Rate Estimates
[6] Asadi and Pinkley	2022	LR	State of the system, Number of batteries	VFA for initial ADP value
[32] Li et al.	2023	Ensemble of SL Models	Sales, Promotional & Product Features	Product-Warehouse Sales
[43] Ren et al.	2024	LSTM-QR, LGBM-QR, ANN-QR	Sales & Promotions History	Optimal Inventory Level
[19] Fan et al.	2024	LGBM	Tensor Demand Data	Demand Values
[54] Yang et al.	2024	GBDT, ANN, LGBM	Transaction Data Features	Demand Values
[4] Akkerman, Prak, and Mes	2024	NN	Stock levels, Inspection intervals, Item attributes	Reorder Policy

D - Delivery Decisions, P - Problem Parameters.
NN - Neural Network, BPNN - Backpropagation Neural Network, SVM - Support Vector Machine, RPR - Random Parameter Regression, MLE - Maximum Likelihood Estimation, LR - Linear Regression, LSTM-QR - Long Short-Term Memory with Quantile Regression, LGBM-QR - Light Gradient Boosting Machine with Quantile Regression, ANN-QR - Artificial Neural Network with Quantile Regression, LGBM - Light Gradient Boosting Machine, GBDT - Gradient Boosted Decision Trees, ANN - Artificial Neural Network

2.5 Supervised Learning in Vehicle Routing

Some papers explore learning which edges should be selected to solve routing problems. One approach by Jemai and Mellouli [27] introduces a feed-forward (FF) NN trained via backpropagation to learn routing decisions based on historical data. This approach leverages minimal input information, focusing solely on binary indicators of requests at customer locations. By learning from previously solved optimization problems, the model provides initial solutions to the Real-Time Vehicle Routing Problem (RTVRP), further refined by a tabu search heuristic when time permits. In contrast, Xiang et al. [53] employ a more detailed methodology, introducing a Pairwise Proximity Learning-based Ant Colony Algorithm for Dynamic Vehicle Routing Problems (DVRP). This approach applies a Radial Basis Function (RBF) network to predict pairwise proximity values between customer nodes, learning the local visiting order of consecutive nodes to optimize route sequencing dynamically. The model adapts in real-time as new nodes are introduced, using updated proximity calculations and trained RBF predictions to maintain efficient

routing. Lastly, Arnau, Juan, and Serra [5] diverge from these methodologies by predicting the dynamic costs associated with edges after a route is partially or fully constructed. Using a Multiple Linear Regression (MLR) model, this approach integrates dynamic factors such as vehicle load and capacity to anticipate cost variations, enabling more informed decision-making during the route construction.

Strategies to directly solve the routing problem were also explored, leveraging diverse methodologies. For example, Zhao et al. [55] employ SL to train decision-making models using labeled traffic data, relying on features such as vehicle density, road density, and speed limits to dynamically optimize routing strategies. The SL model maps real-time traffic conditions to optimal routing strategies, facilitating adaptive multi-hop routing in vehicular networks. The emphasis lies on adaptive decision-making within wireless vehicular networks, aiming to enhance performance using dynamic, global traffic data. Another approach, at Ladha and Chaubey [31], integrates ANNs with optimization algorithms, specifically the Cuckoo Search algorithm, to predict the best route in public transport systems. That algorithm identifies multiple optimal routes, which are then used to train the ANN, considering traffic density, routing time, and vehicle load. The model predicts the most efficient route for passenger flow and dynamically updates routes in real-time to address demand fluctuations and reduce inefficiencies.

Specific problem dimensions in routing and inventory, traditionally modeled as static parameters, can be learned through advanced SL techniques instead. For instance, Choi, Yu, and Choi [12] use a Deep Neural Network (DNN) in the Global Path Approximation Model to dynamically estimate travel times. This model considers edge congestion and load information to generate congestion-aware predictions, enabling more efficient and balanced vehicle routing decisions in highly dynamic environments. Similarly, Ahn et al. [2] leverage a sequence-to-sequence graph convolutional gated recurrent unit model to predict future edge-traveling times, using attributes such as the number of vehicles, their velocity, and idle states. Other types of problem parameters attributes can also be learned. To solve capacity challenges, Küp et al. [30] apply regression-based SL to predict delivery capacities for courier fleets, using features such as delivery counts, historical performance metrics, and time slot data. The approach provides input for optimizing fleet allocation and ensures that delivery schedules are balanced between standard and instant orders. To solve sequence-related challenges, Zhou et al. [56] utilize SL to derive service sequences for couriers. By analyzing spatial and temporal patterns in historical data, their model predicts the most efficient next step in the delivery process, later integrating RL to refine routes.

Other approaches have focused on the request side, aiming to develop learning-based models that support routing and decision-making in dynamic settings. For instance, Baty et al. [7] propose a supervised learning framework tailored to a DVRP with time windows. The model predicts a request prize vector by leveraging spatial coordinates, demand, service times, and travel durations. The prize vector directly influences CO layers, enabling dynamic routing and dispatching decisions. In a complementary approach, Han et al. [25] explore LSTM models to forecast energy consumption, accounting for variables such as driving profiles, traffic conditions, and state of charge. The predicted energy consumption guides routing and charging decisions, enhancing the

efficiency of delivery operations. Finally, Wei, Xu, and Yang [52] forecast demand from historical customer data to inform vehicle routing and rental choices. Using models like LGBM, the framework predicts customer demand for specific time windows and forward warehouses, which are used to optimize routing and reduce delivery costs. Distinct from prior approaches, this work ties demand forecasting with inventory planning, as predictions guide delivery quantities and resource allocation decisions.

Table 2.2: Routing applications of Supervised Learning

Author	**Year**	**Model**	**Input**	**Output**
[27] Jemai and Mellouli	2008	FFBPNN	Request in Binary	Edge Selection for Route
[5] Arnau, Juan, and Serra	2018	MLR	Vehicle and Route Properties	Cost of Edge
[55] Zhao et al.	2021	OS-ELM	Traffic Features	Routing Strategy
[2] Ahn et al.	2022	GCNN, GRU	Past Traffic Flow Data	Edge Travelling Times
[53] Xiang et al.	2022	RBF	Node Pairwise Proximity	Binary Visiting Order
[31] Ladha and Chaubey	2023	ANN	Optimal Routes	Best Passenger Flow Route
[56] Zhou et al.	2023	FourNN, Ptr-Net	Current Location, Request Type	Service Sequence
[30] Küp et al.	2024	GBDT	Historical Delivery Metrics	Delivery Capacity
[7] Baty et al.	2024	FFNN	Request Features State	Request Prize Vector
[52] Wei, Xu, and Yang	2024	DT, RF, GBDT, XGBoost, AdaBoost, LGBM	Historical Customer Data	Predicted Demand
[12] Choi, Yu, and Choi	2024	DNN	Edge Congestion, Load Information	Travel Times
[25] Han et al.	2024	LSTM	Driving Profiles, Traffic Information, Energy Consumption Data	Energy Consumption

R - Routing Decisions, P - Problem Parameters.
FFBPNN - Feedforward Backpropagation Neural Network, MLR - Multiple Linear Regression, OS-ELM - Online Sequential Extreme Learning Machine, GCNN - Graph Convolutional Neural Network, GRU - Gated Recurrent Unit, RBF - Radial Basis Function Network, ANN - Artificial Neural Network, FourNN - Four-layer Neural Network, Ptr-Net - Pointer Network, GBDT - Gradient Boosted Trees, FFNN - Feedforward Neural Network, DT - Decision Tree, RF - Random Forest, XGBoost - Extreme Gradient Boosting, AdaBoost - Adaptive Boosting, LGBM - Light Gradient Boosting Machine, DNN - Deep Neural Network, LSTM - Long Short-Term Memory

2.6 State-of-the-Art Analysis

Most DSIRP formulations share similar characteristics, such as one-to-many network structures, single and non-perishable items, and a focus on exclusive deliveries instead of pickup and delivery or emergency transshipments, contrasting with real-world scenarios. However, different stock-out management strategies and inventory replenishment policies have been consistently explored across the literature.

Research on inventory management showcases a wide use of SL models to tackle forecasting, optimization, and policy-driven challenges. NNs, such as feed-forward and LSTM models, dominate due to their complex, multimodal data capacity. Demand forecasting is a primary application,

with models like LGBM [19] and GBDT [54] predicting future demand based on sales trends, promotions, and transaction features. SL is also applied to policy decisions, such as reorder probabilities and value function estimation, supporting dynamic inventory adjustments. A key focus has been balancing understock and overstock risks, with methods predicting optimal inventory thresholds or levels for efficient stock management [43].

The literature on vehicle routing demonstrates a broad range of methodologies and models tailored to diverse challenges. NNs, especially feed-forward architectures, dominate due to their capacity to handle complex patterns, while simpler models like linear regression or decision trees are occasionally employed for specific tasks. Advanced techniques, such as GBDT [30] and LSTM [25] networks, stand out for tasks involving temporal or non-linear dependencies, such as forecasting energy consumption and demand. The models address various objectives: edge selection, cost estimation, route sequencing, and delivery capacity planning. A significant trend is the learning of problem parameters previously treated as fixed, such as congestion levels, energy requirements, and delivery capacities [12].

Despite promising advances in using SL across inventory and routing problems, its integration into the DSIRP remains limited. Most notably, existing DSIRP applications of SL focus almost exclusively on customer prioritization [22], with little to no research addressing delivery quantity optimization or route planning using learning-based methods. However, complementary inventory and routing research findings reveal promising opportunities to address this gap. Advanced models explored in that research, along with various strategies for dynamic routing, inventory threshold prediction, and reordering decisions, were developed using different input features (e.g., traffic data for routing and market data for inventory). These efforts lay the foundation for this work to connect with the DSIRP and address existing gaps in the literature. This thesis builds on those complementary findings by introducing an SL framework for customer selection in a DSIRP context, while proposing different policies for quantity and routing to support integration.

Chapter 3

Problem Definition

The Dynamic and Stochastic Inventory Routing Problem (DSIRP) is a complex challenge in operations research that integrates inventory control, delivery scheduling, and vehicle routing in uncertain environments. This problem commonly arises in supply chain systems, such as those found in grocery retail, pharmaceutical distribution, or multi-store replenishment networks, where a central supplier must distribute goods to multiple geographically dispersed customer locations. In such cases, the supplier must jointly decide how much and when to deliver to each customer, and how to combine them into routes, without knowing future customer demand in advance. This chapter lays the groundwork for addressing the DSIRP by formalizing its essential components and decision-making structure. It begins with a mathematical description of the delivery network, cost components, and constraints. Then, it frames the problem as a sequential and stochastic process in which each decision affects immediate costs and future feasibility. By framing the problem as a sequential decision-making process with state transitions that depend only on current information, a Markov Decision Process (MDP), defined by states, decisions, and transition dynamics, the chapter establishes a foundation for learning-based approaches, such as reinforcement learning or policy learning methods. Finally, a visual example illustrates the flow of decisions and the role of uncertainty over time.

3.1 Problem Description and Notation

We consider a general DSIRP over a finite planning horizon of $\mathscr{T}$ discrete periods, indexed by t, $\mathscr{T} = \{0, 1, \ldots, T-1\}$. The delivery network consists of a single supplier and a set of N customer entities. It is represented as a complete directed graph $G = (\mathscr{N}, \mathscr{A})$, where the node set is $\mathscr{N} = \{0, 1, 2, \ldots, N\}$, with node 0 denoting the supplier and nodes $i \in \mathscr{N}' = \mathscr{N} \setminus \{0\}$ representing the customers. The set of arcs is $\mathscr{A} = \{(i, j) : i, j \in \mathscr{N}, i \neq j\}$, and each arc $(i, j) \in \mathscr{A}$ is associated with a non-negative routing cost c_{ij}.

A single vehicle with capacity V is available to perform deliveries. The vehicle starts and ends its route at the supplier in each period and can visit a subset of the customers in each period to replenish their stock. The binary variable x_{ijt} equals one if the vehicle travels directly from

location i to j during period t, and zero otherwise. Likewise, the binary variable y_{it} equals one if customer $i \in \mathcal{N}'$ is visited in period t, and zero otherwise.

At the beginning of each period t, the planner must determine the delivery quantities $\{q_{it}\}_{i \in \mathcal{N}'}$ without knowing the real demand for that period. Each customer i faces a stochastic demand D_{it}, modeled as a random variable with a known probability distribution, which is only revealed at the end of that period. Each customer starts the planning horizon with an initial inventory level I_{i0}. In case of stockout, the shortfall is recorded as lost sales l_{it}, which are not backlogged but rather lost permanently. Each unit of stockout incurs a penalty cost z_i.

The product is assumed to be non-perishable. The supplier has capacity enough that can satisfy any delivery quantities up to each customers capacities. However, the total amount delivered in each period must not exceed the vehicle capacity V. Furthermore, inventory held at a given customer incurs a unit holding cost h_i per period, while the supplier does not incur holding costs. A summary of all sets, parameters, and decision variables used in this model is provided in Table 3.1.

The planner aims to minimize the total expected cost over the entire horizon $\mathcal{T}$. This total cost includes three components:

$$\sum_{t=0}^{T-1} \sum_{i \in \mathcal{N}'} h_i \cdot I_{it} \tag{3.1}$$

$$\sum_{t=0}^{T-1} \sum_{i \in \mathcal{N}'} z_i \cdot l_{it} \tag{3.2}$$

$$\sum_{t=0}^{T-1} \sum_{(i,j) \in \mathcal{A}} c_{ij} \cdot x_{ijt} \tag{3.3}$$

Equation (3.1) represents the total inventory holding costs at customer locations, Equation (3.2) the penalties for lost sales due to unmet demand, and Equation (3.3) the total transportation cost incurred by the single vehicle. In each period, the planner must make joint decisions on whether to visit a customer (y_{it}), how much to deliver (q_{it}), and which route to take (x_{ijt}). These decisions must anticipate uncertain future demand while satisfying inventory and routing constraints, thereby carefully balancing the trade-offs between overstocking, understocking, and delivery performance.

Table 3.1: Notation summary

Sets	
$\mathscr{N}$	Set of customers and supplier, with node 0 as the supplier
$\mathscr{N}'$	Set of customers, $\mathscr{N}' = \mathscr{N} \setminus \{0\}$
$\mathscr{A}$	Set of arcs (i, j) between nodes
$\mathscr{T}$	Planning horizon indexed by t
Parameters	
V	Capacity of the vehicle
c_{ij}	Cost of traversing arc (i, j)
D_{it}	Demand of customer i in period t
$\bar{U}_i$	Maximum inventory capacity at customer i
h_i	Unitary inventory holding cost of product at customer i
z_i	Unitary shortage cost of product at customer i
Decision variables	
I_{it}	Inventory level at customer i at the beginning of period t
q_{it}	Quantity delivered to customer i in period t
l_{it}	Lost sales at customer i in period t
y_{it}	1 if customer i is visited in period t, 0 otherwise
x_{ijt}	1 if arc (i, j) is traversed by the vehicle in period t, 0 otherwise

3.2 Sequential decision process

As a stochastic and dynamic problem, the DSIRP can naturally be formulated as a MDP, which provides a structured framework for modeling sequential decision-making under uncertainty. An MDP is defined by a set of states S_t, a decision a_t, a stochastic process characterized by the realization of random variables, such as demand, represented by the information W_{t+1}, and a cost function that evaluates each decision following a policy π. In this context, the planner observes a state at the beginning of each period, selects a decision, and then observes new information in the form of realized customer demands. This realization triggers a transition to the next state S_{t+1} while also generating costs that accumulate over the planning horizon.

3.2.1 State (S_t)

At the beginning of each period t, the system is characterized by a state S_t, representing all information observable to the planner prior to making delivery and routing decisions. The primary state variable is the vector of inventory levels $I_t = (I_{0t}, I_{1t}, \ldots, I_{Nt})$, where I_{it} denotes the stock available at customer $i \in \mathscr{N}'$ at the start of period t. These inventory levels reflect the cumulative effect of past decisions and realized demands and directly influence the feasibility and outcome of future delivery decisions. In addition to the inventory, the state might incorporate auxiliary information that characterizes each customer and the network. These components fit into four

categories across two dimensions. The first dimension is the component nature, with components being either *Static*, where they do not change their value across the periods, such as customer capacity, or *Dynamic*, where they hold a different value in different periods, such as the util rate, which is the ratio between customer inventory and customer capacity. The second dimension is the domain, where the components can fall either into the *Inventory* category, such as the mean demand, or the *Routing* category, such as the distances between customers. Derived from core components, the state includes metrics like the estimated number of periods to stockout given the current inventory and expected demand. The initial state S_0 is defined by the known inventory levels at each customer at the beginning of the planning horizon.

$$I_{0t+1} = I_{0t} - \sum_{i \in \mathscr{N}'} q_{it}, \quad \forall t \in \mathscr{T} \tag{3.4}$$

$$I_{it+1} = I_{it} + q_{it} - D_{it} + l_{it}, \quad \forall i \in \mathscr{N}',\, t \in \mathscr{T} \tag{3.5}$$

$$I_{it} \leq \bar{U}_i, \quad \forall i \in \mathscr{N}',\, t \in \mathscr{T} \tag{3.6}$$

$$q_{it} \leq V y_{it}, \quad \forall i \in \mathscr{N}',\, t \in \mathscr{T} \tag{3.7}$$

$$\sum_{i \in \mathscr{N}'} q_{it} \leq V y_{0t}, \quad \forall t \in \mathscr{T} \tag{3.8}$$

$$\sum_{j \in \mathscr{N}: j \neq i} x_{jit} = y_{it}, \quad \forall i \in \mathscr{N},\, t \in \mathscr{T} \tag{3.9}$$

$$\sum_{j \in \mathscr{N}: j \neq i} x_{ijt} = y_{it}, \quad \forall i \in \mathscr{N},\, t \in \mathscr{T} \tag{3.10}$$

$$\sum_{i \in B} \sum_{\substack{j \in B \\ j \neq i}} x_{ijt} \leq \sum_{i \in B} y_{it} - y_{\ell t}, \quad \forall B \subseteq \mathscr{N}',\, |B| \geq 2,\, \ell \in B,\, t \in \mathscr{T} \tag{3.11}$$

$$l_{it} \geq 0, \quad \forall i \in \mathscr{N}',\, t \in \mathscr{T} \tag{3.12}$$

$$I_{it} \geq 0, \quad \forall i \in \mathscr{N},\, t \in \mathscr{T} \tag{3.13}$$

$$q_{it} \geq 0, \quad \forall i \in \mathscr{N}',\, t \in \mathscr{T} \tag{3.14}$$

$$y_{it} \in \{0,1\}, \quad \forall i \in \mathscr{N},\, t \in T \tag{3.15}$$

$$x_{ijt} \in \{0,1\}, \quad \forall (i,j) \in A,\, t \in \mathscr{T} \tag{3.16}$$

3.2.2 Decision (a_t)

A decision is made at each time period t, before demand is revealed. This decision is represented by a_t. This consists of three types of variables that jointly decide the delivery and routing strategy. A continuous variable q_{ti} denotes the quantity of product delivered from the supplier to customer $i \in \mathscr{N}'$. The updated inventory is obtained by adding the inventory available at the beginning of the period to the delivered quantity. Two sets of binary variables represent the routing component. First, the variable y_{it} takes the value of one if customer i is visited during period t, and zero otherwise. This allows us to identify which customers are part of the route and require delivery. Secondly, the variable x_{ijt} indicates whether the arc (i, j) is traversed by the vehicle during period

t. A decision a_t is feasible if it satisfies the constraints given in Equations (3.4) to (3.16). In order to identify the progress of the system after a decision is made, we introduce the concept of a *post-decision state*, denoted by $S_t^a = (S_t, a_t)$. This is the intermediate state resulting from the planner's decision a_t, but before the stochastic information W_{t+1} is revealed. For simplification purposes, we represent the decisions y_{it}, x_{ijt}, and q_{it} as $y_t = (y_{it})_{i \in \mathcal{N}'}$, $x_t = (x_{ijt})_{(i,j) \in \mathcal{A}}$, and $q_t = (q_{it})_{i \in \mathcal{N}'}$ in the action $a_t = (y_t, x_t, q_t, I_t^a)$. The inventory for the post-decision state, I_t^a, is obtained according to Equations 3.17 and 3.18.

$$I_{0t}^a = I_{0t} - \sum_{i \in \mathcal{N}'} q_{it}, \quad \forall t \in \mathcal{T} \tag{3.17}$$

$$I_{it}^a = I_{it} + q_{it}, \quad \forall i \in \mathcal{N}', t \in \mathcal{T} \tag{3.18}$$

3.2.3 Stochastic Information (W_{t+1})

Stochastic information W_{t+1} represents the customer demands revealed at the end of each period t, after the planner has executed the delivery and routing decisions, denoted by the decision a_t. This demand is modeled as a random variable with a known probability distribution, characterized by parameters such as mean and variance.

We denote the demand realization as $D_t = \{D_{it}\}_{i \in \mathcal{N}'}$, with $W_{t+1} = D_t^\omega$ capturing the stochasticity. Here, $\omega \in \Omega$ represents a scenario or outcome in the underlying probability space, and D_t^ω denotes a specific realization of the random demand at time t under scenario ω. These realizations are assumed to be exogenous, independent of both the current state S_t and decision a_t. The system transitions from the post-decision state S_t^a to the next state S_{t+1} according to a deterministic transition function Γ, defined as $S_{t+1} = \Gamma(S_t^a, W_{t+1})$, where inventory levels are updated based on realized demands. This transition is represented by:

$$I_{it+1} = I_{it}^a - D_{it}^\omega + l_{it}, \quad \forall i \in \mathcal{N}', t \in \mathcal{T} \tag{3.19}$$

The resulting cost function for each period includes inventory holding, stockout, and routing costs:

$$C(S_t^a, W_{t+1}) = \sum_{i \in \mathcal{N}'} h_i I_{it+1} + \sum_{i \in \mathcal{N}'} z_i l_{it} + \sum_{(i,j) \in \mathcal{E}} c_{ij} x_{ijt} \tag{3.20}$$

3.2.4 Policy

A solution to the DSIRP is defined as a policy $\pi \in \Pi$, which maps each state S_t to a decision $a_t = A^\pi(S_t)$ at each period $t \in T$. The decision a_t is selected from the action space A, which defines the set of all feasible actions. Upon observing the current state S_t, the planner selects a decision a_t, after which demand is realized and the system transitions to a new state.

The objective is to find an optimal policy $\pi^* \in \Pi$ that minimizes the expected cumulative cost over the planning horizon:

$$\pi^* = \arg\min_{\pi \in \Pi} E\left[\sum_{t \in \mathcal{T}} C(S_t, A^\pi(S_t)) \middle| S_0\right] \tag{3.21}$$

This formulation captures the sequential nature of the problem under uncertainty, where decisions influence not only immediate costs but also future inventory levels and delivery feasibility, and represent the core trade-offs of the DSIRP, namely, inventory holding, lost sales, and routing efficiency.

3.3 Example

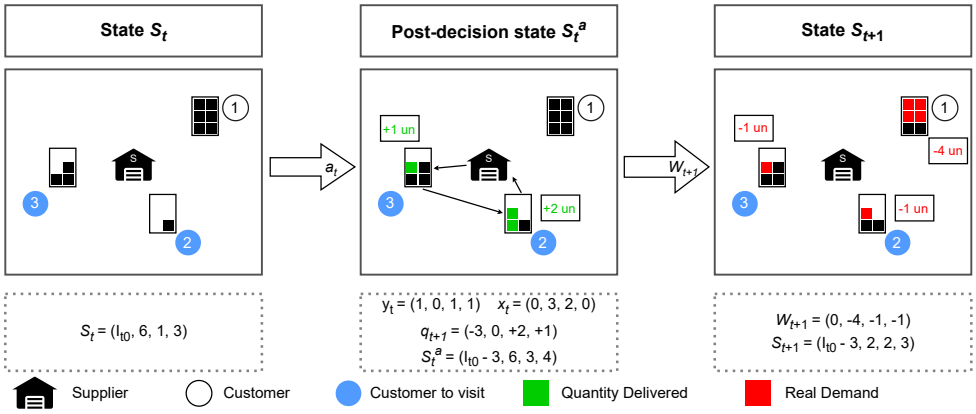


Figure 3.1: Illustration of the MDP structure for the DSIRP

To illustrate the sequential structure of the DSIRP, we provide a simplified example in Figure 3.1, modeled using the MDP framework. It is visually structured into three steps: the current state S_t, which evolves through a decision a_t to a post-decision state S_t^a, and finally transitions to the next state S_{t+1} following the realization of stochastic demand $\{D_{it}\}_{i \in \mathcal{N}'}$.

In this example, and for illustrative purposes, we assume that the state S_t includes only the inventory levels of the supplier and customers. Specifically, the initial state $S_t = (I_{0t}, I_{1t}, I_{2t}, I_{3t}) = (I_{0t}, 6, 1, 3)$ comprises the inventory at the supplier and three customers. Customers highlighted in blue (2 and 3) are selected for delivery based on the current state.

The decision a_t is divided in representation into the partial decisions y_t, q_{t+1} and x_t. The decision $y_t = (1, 0, 1, 1)$, indicates that customers 2 and 3 are visited. In green, we observe that customer 2 receives two units and customer 3 receives one, while the supplier decreases three units, corresponding to the partial decision vector $q_{t+1} = (-3, 0, 2, 1)$. The routing path selected is $0 \to 3 \to 2 \to 0$, as shown in the delivery route, at $x_t = (0, 3, 2, 0)$. The post-decision state reflects the inventory after the decision of delivery quantities and route. The updated inventory becomes $S_t^a = (I_{0t} - 3, 6, 3, 4)$.

The transition to state S_{t+1} incorporates the realization of stochastic demand, given by $W_{t+1} = (0, -4, -1, -1)$. As highlighted in red, customer 1 consumes four units, customer 2 consumes one, and customer 3 consumes one. The resulting inventory becomes $S_{t+1} = (I_{0t} - 3, 2, 2, 3)$, showing updated values after demand is subtracted from S_t^a.

If the realized demand at any customer exceeds the available inventory in S_t^a, the difference would be recorded as lost sales, incurring additional penalty costs as per the model.

Chapter 4

Methodology

This chapter presents the methodology developed to train, validate, and assess supervised learning-based policies for the DSIRP addressed in this work. The structure adopted is inspired by the Cross Industry Standard Process for Data Mining (CRISP-DM) [48] framework, following a systematic process from problem understanding to model deployment and iterative refinement. A dual-environment setup separates model training and validation from deployment and evaluation in a simulated operational context. This design allows for isolated tuning of learning parameters and controlled testing under dynamic and stochastic conditions.

The methodology is organized around a development pipeline illustrated in Figure 4.1, which encapsulates the core steps and feedback loops that drive the improvement cycle. The process iteratively refines model performance through targeted feature engineering and simulation-informed adjustments, such as adding features that capture patterns observed in simulation runs or represent edge-case scenarios. This chapter also outlines how the inventory-routing problem is approached within the simulator using a structured three-step framework. Each step is treated as a distinct decision point, allowing the application of either heuristics or learning-based methods, which simplifies the resolution of the problem.

4.1 Approach Overview

The methodology begins with a series of initial steps that encompass the identification of the problem context, the delineation of the work structure, and the organization of the data to be used throughout the process. Subsequent to this, the approach transitions into an iterative cycle in which the experiment evolves through model refinement and evaluation.

This iterative loop is inspired by the CRISP-DM methodology but extends it by incorporating a simulation component for policy testing in dynamic conditions. The loop captures the progression from feature engineering, through model training, to simulation-based evaluation, and back to analysis-driven improvements.

The pipeline is comprised of two primary environments. The first, designated as the Policy Generator stage, encompasses all activities related to model development, including feature

engineering, hyperparameter tuning, and validation. The second is the Simulation stage, which evaluates the trained model in a controlled environment and collects performance metrics, i.e., total costs. These two environments, while distinct, are interdependent: insights drawn from performance analysis inform the next cycle of feature engineering, thereby effectively closing the loop.

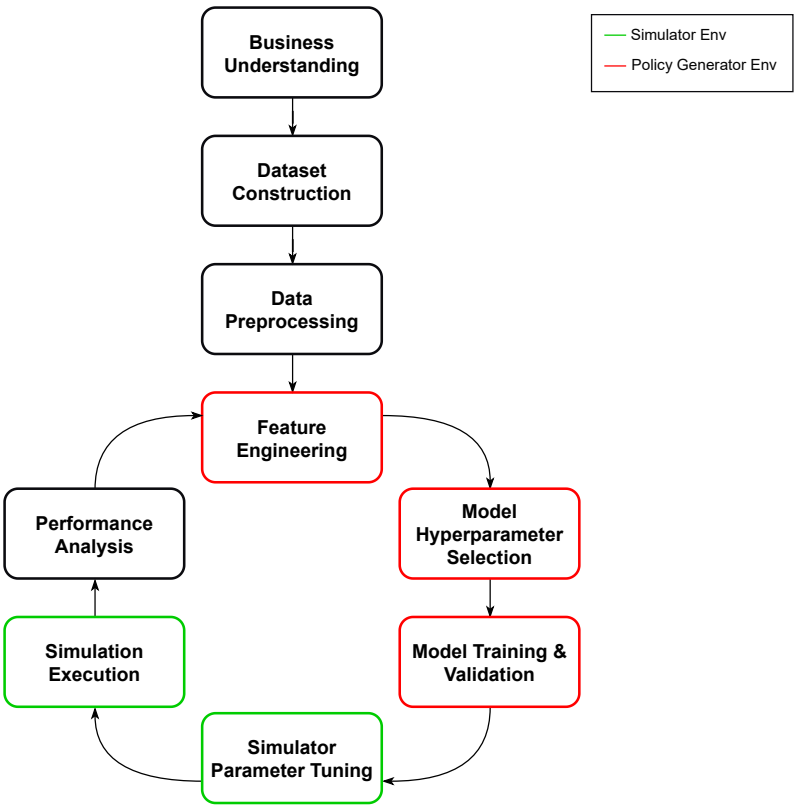


Figure 4.1: Development Pipeline

4.1.1 Initial Steps

The identification of the needs for the project was achieved through a comprehensive literature review and an in-depth analysis of the DSIRP problem characteristics. Three core decisions (customers to visit, quantities to deliver, vehicle route), inherent to the structure of the DSIRP, were identified and assigned distinct methodological approaches. The strategies adopted for each decision are further detailed in Section 4.2. The literature review also guided the selection of commonly used models and approaches in similar problems, with a view to adapting them to our specific problem. Following the completion of the business understanding stage, the subsequent

step was to create the datasets. First, a set of instances was generated, each one containing a group of customers and their characteristics, as well as periods and actual demands which are revealed periodically. Then, a construction was made with a set of features related to each customer and period, using the instances generated, which are detailed in Section 5.1. In total, three datasets were created from different instance sets: one for training the models, one for validating them, and a third for testing and comparing the proposed approach against other benchmark policies. These datasets were specifically designed to support the learning of a priority rule, aimed at deciding which customers to visit in a given period. For each dataset, a data preprocessing method was applied, including feature normalization and the selection of the most relevant features for use in the learning models.

4.1.2 Feature Engineering and Policy Generator Environment

The Policy Generator environment encompasses the whole process including exploring features selection, feature engineering, model development and tuning, and model validation. Feature engineering supported the development of the model across iterations. Starting with an initial set of features, which included fundamental operational features such as mean demand and customer distance, the process evolved through insights obtained from the Performance Analysis step. This feedback guided the creation of new features by combining existing variables (resulting in relational features) and the removal of features flagged as redundant by correlation analysis.

The initial step in model development is the selection of the hyperparameters of the Multi-Layer Perceptron employed for Supervised Learning. In general, default network parameters were used, with different sizes of hidden layers used for different models. For the Learning rate init and Momentum, in order to select better values than default, instead of using a traditional grid search, a more adaptive search strategy was also explored to efficiently search near-optimal parameter values, tailored to the specific feature set. A complete list of the hyperparameters explored and their respective tested intervals is presented in Table 4.1. Once the optimal parameters were selected, the model was trained using the training and then validated on the validation to ensure robustness and generalization. Performance was evaluated using the Root Mean Squared Error (RMSE) for regression models, given its tendency to penalize higher discrepancies between prediction and ground truth, while the Area Under the Curve (AUC) served as the primary measure for classification models.

Table 4.1: Hyperparameters Values

Parameter	**Value / Interval**
Hidden layer size	(16), (32), (64,64)
Momentum	0.1 – 0.9
Maximum iterations	200 – 700
Learning rate init	0.00001 – 0.01

4.1.3 Simulation Environment

The simulation environment is designed to mirror real-world operational constraints and evaluate the performance of decision policies under dynamic demand scenarios. The SL model is deployed to make the partial decision on which customers to visit, integrated alongside complementary policies, within a controlled and repeatable setting. The simulation operates with some fixed parameters across experiments, while others, such as decision thresholds and sensitivity settings within specific policies, require tuning. Parameter tuning is guided by performance on the validation to evaluate the capacity of generalization to the test set. Subsequent to the calibration of the model's parameters, its evaluation is conducted on the test dataset. The primary performance indicators collected include routing cost, inventory holding and stockout costs. These metrics are computed for each instance in the test set and stored individually, enabling comparison against the PH benchmark and alternative policy baselines.

4.1.4 Perfect Hindsight Analysis

The PH solution is obtained by solving a deterministic Mixed-Integer Program (MIP) in which all sources of uncertainty, such as customer demands, are assumed to be fully known. It serves as an ideal benchmark for policy evaluation. Using the results obtained from the testing dataset, we performed comparisons with a set of baseline decision policies. However, since computing the PH solution is intractable for large instances, direct comparisons with PH were restricted to a subset of smaller test instances, i.e., 135 out of the 1080 on the test set, which are the ones with only five customers and ten periods. In order to assess the improvement of the model across development cycles, we compared the validation dataset results from successive iterations, using total cost as the primary evaluation metric. While these aggregate comparisons provide a high-level view of model progress, they do not fully explain where or why the model diverges from optimal behavior. To better understand which aspects of the PH policy the model fails to capture, a more detailed analysis is made, by identifying the X instances with the highest gaps in the total costs between PH and the learning policy. Then, for each of these cases, an instance-level analysis of customer visit and delivery quantity decisions is performed to derive insights for the next iteration, guiding the refinement of feature representations or model configurations in subsequent development cycles, which may help the model better approximate the PH policy's behavior, a process detailed in Section 5.3.

4.2 Decision Policy Decomposition

While it is theoretically possible to solve the DSIRP holistically using an integrated optimization solver in rolling horizon, the computational complexity of such approach renders it impractical for large-scale instances, as some can take days to solve to optimality, making it unsuitable when decisions are required on a daily basis. As a result, the decision-making process within the simulation environment follows a structured three-step decomposition, illustrated in Figure 4.2, aligned

with the classical formulation of IRPs. Each step in this process corresponds to a distinct decision, including the selection of customers to be visited, the determination of the quantity of items to be delivered to each, and the calculation of a delivery route. This configuration facilitates the integration of diverse policies or models at each stage, allowing for flexible experimentation.

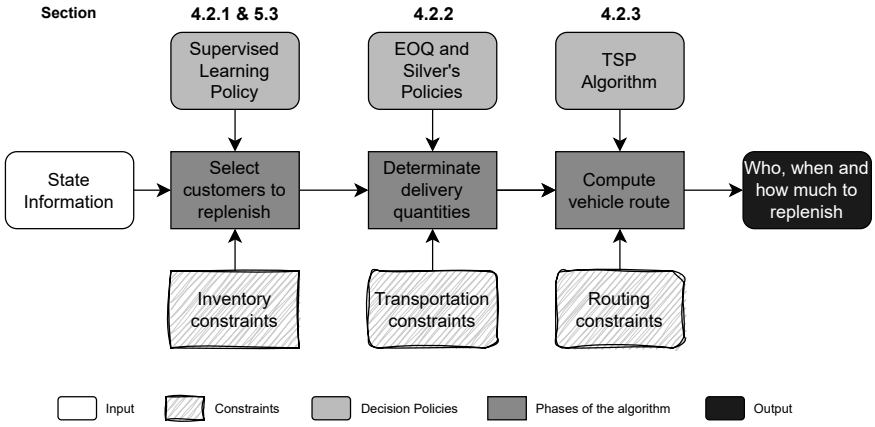


Figure 4.2: DSIRP Three Step Policy (adapted from Roldán, Basagoiti, and Coelho [45])

4.2.1 Customer Visit Decision

The decision of which customers to visit in each period plays a critical role in reducing the decision space of the problem. However, it has to be defined carefully, as it involves the possibility of disregarding customers who may experience stockouts. While a preliminary selection of customers can be made, it is equally important to establish a prioritization among the selected customers. This is because, in practice, limited vehicle capacity may prevent full adherence to the planned inventory policies, which means we might not be able to visit some of the selected customers. As noted by Roldán, Basagoiti, and Coelho [45], "It is possible that after having selected the customers and an inventory policy, the capacity of the vehicle is not sufficient to guarantee that the policy is fully respected," which reinforces the need for prioritization under vehicle capacity constraints.

Traditional heuristics, such as Lowest Inventory First (LIF) [11] or the ratio of current inventory to mean demand, are commonly used in inventory problems. However, these approaches primarily focus on inventory-related objectives and do not account for other relevant dimensions such as routing and cost trade-offs. Given that our problem integrates inventory, routing, and lost sales costs, these traditional heuristics could be insufficient.

To address this, we propose the use of a SL model to guide the customer visit decision. By incorporating features that reflect both inventory and routing characteristics, the model enhances decision flexibility, potentially improving overall system performance compared to static rule-based policies. A detailed analysis of the model's variations and results is presented in Chapter 5.

4.2.2 Delivery Quantity Decision

In accordance with the previously calculated priority order, we determine, for each selected customer, the amount of inventory to be delivered. In the domain of Inventory Management literature, there exists a wide range of policies for quantity selection. Among these, some are relatively simple and static, such as the fixed quantity policy, which is characterised by the delivery of a constant percentage of the customer's capacity. For example, in Roldán, Basagoiti, and Coelho [45], it is applied a fixed replenishment rule to deliver a predetermined fraction of each customer's inventory capacity. In this work, we consider two alternatives: Economic Order Quantity (EOQ) and a set of decision rules for (R, s, S) systems Silver, Pyke, and Peterson [49]. Each of these policies required adaptations to accommodate the nature of our problem, which included the routing component. Furthermore, the delivery decision is bounded by the vehicle's capacity and the customer's available inventory space.

4.2.2.1 EOQ Policy

EOQ is a classical approach in inventory theory, first introduced by Harris [26] over a century ago. It aims to find a trade-off between holding and ordering costs, minimizing total costs. The original formula is presented in Equation 4.1, where μ represents the expected demand, S is the fixed setup cost per order, and H is the per-unit holding cost.

$$Q^* = \sqrt{\frac{2\mu S}{H}} \tag{4.1}$$

The EOQ policy requires specifying a fixed setup cost per order, traditionally representing the administrative or logistical expense incurred each time an order is placed. In our context, we reinterpret this concept to reflect the cost of serving an individual customer. Specifically, we model the setup cost using a measure of customer isolation: the average distance from a given customer to its τ nearest neighbors, expressed in percentage terms. This captures the intuition that customers located further from others are more costly to reach, and thus resemble a higher setup cost when considered as individual ordering points. We test multiple values of τ, treating it as a tunable parameter, as different neighborhood sizes can yield complementary insights into spatial customer distribution and its effect on cost estimation. A scaling parameter α is introduced as a factor to maintain comparable magnitude with other terms in the formula. The adapted EOQ formula is shown in Equation 4.2, with all other components aligned to our problem's notation.

$$Q = \sqrt{\frac{2 \cdot \alpha \cdot \mu \cdot \text{Isolation}_{\tau}}{H}} \tag{4.2}$$

Here, μ is the customer's mean demand, H is the unit holding cost, and $\text{Isolation}_{\tau\%}$ is the average distance to % nearest customers. α and τ are both examples of tunable parameters within the simulator in the process described in Section 4.1.3.

4.2.2.2 (R, s, S) policies

The (R, s, S) policies are a set of decision rules proposed in Silver, Pyke, and Peterson [49], which provides a three sequential steps approach for determining inventory replenishment decisions under uncertainty. For simplicity purposes, we will address across the rest of the dissertation this policies as Silver's Policy. The first step, shown in Equation 4.3, computes the base order quantity Q_p, based on the customer's mean demand μ, demand standard deviation σ, holding cost h, shortage cost z, and a setup cost component. This setup cost is traditionally treated as a fixed value.

$$Q_p = a \cdot \mu^{1-b} \cdot \left(\frac{\text{SetupCost}}{h}\right)^b \cdot \left(1 + \frac{\sigma^2}{\mu^2}\right)^c \tag{4.3}$$

From this quantity, and using additional components such as customer's demand standard deviation σ, and the cost trade-off between holding and shortage penalties, the reorder point s_p is computed as seen in Equation 4.4.

$$s_p = c_1 \cdot \mu + \sigma \cdot \left(\frac{c_2}{z} + c_3 - c_4 \cdot z\right) \tag{4.4}$$

A third conditional step is introduced. If $Q_p/\mu \leq c_5$, a capped reorder-up-to level S_0 is calculated, as shown in Equation 4.5, and used to adjust the final s and S values. Otherwise, a simpler base policy applies using s_p and Q_p directly, as shown in Equation 4.6.

$$S_0 = \mu + k \cdot \sigma, \quad s = \min(s_p, S_0), \quad S = \min(s_p + Q_p, S_0) \tag{4.5}$$

$$s = s_p, \quad Q = Q_p \tag{4.6}$$

To adapt this policy to our problem setting, we replaced the fixed setup cost with the same isolation measure used in the EOQ adaptation, computed as the average distance to the nearest $\tau\%$ of customers. This leads to the updated base quantity formula shown in Equation 4.7.

$$Q_p = a \cdot \mu^{1-b} \cdot \left(\frac{\text{Isolation}_\tau}{h}\right)^b \cdot \left(1 + \frac{\sigma^2}{\mu^2}\right)^c \tag{4.7}$$

For the second step, the review and lead time horizon $R+L$ was adapted to the decision period of the simulation. The shortage cost term B_3, which originally represents penalty cost per unit short at the end of a review interval, was adapted to our defined shortage cost per period. The resulting adjusted reorder point formula is presented in Equation 4.8.

$$s_p = c_1 \cdot \mu + \sigma \cdot \left(\frac{c_2}{z} + c_3 - c_4 \cdot z\right) \tag{4.8}$$

The third step remains unchanged, following the conditional logic from the original formulation. The original parameters $a, b, c, c_1, c_2, c_3, c_4, c_5$ proposed by Silver [49] are presented in Table 4.2.

Table 4.2: Silver's Policy Parameters

Parameter	a	b	c	c_1	c_2	c_3	c_4	c_5
Original Value	1.300	0.506	0.116	0.973	0.183	1.063	2.192	1.500

4.2.3 Routing Decision

After determining which customers will be visited and the respective quantities to deliver, the final step involves optimizing the delivery route. Since this is a single-vehicle, single-product distribution scenario, the routing problem can be modeled as a TSP, where the goal is to determine the shortest route that begins and ends at the supplier (depot), visiting each selected customer exactly once.

While other methods, such as heuristics or ML-based approaches, could be employed to approximate a solution, the problem's limited size at each decision step allows us to rely on an exact solver. This ensures the computation of an optimal route with low computational cost, and avoiding additional approximation errors introduced by more complex techniques.

Chapter 5

Results

This chapter presents the experimental setup where the iterative development of the supervised learning model was made, and the simulator where it was tested. For that experiment, three sets of instances and respective datasets were created, which are detailed in the first section (5.1). For each iteration, the comparison with the PH is made, and metrics are analyzed. At the end, a general comparison between the models' results is made, as well as a comparison with the considered literature benchmarks, and the time performance during training and test is evaluated.

5.1 Dataset Overview

The creation of the dataset was made across two major steps. At first, instances are generated with different dimensions according to training, validation, and testing purposes. Then, from those instances, problem features were selected to reflect inventory and routing dimensions for the SL model to understand and predict targets like which customer to visit.

5.1.1 Problem Instances

All problem instances used in this work were synthetically generated, following the structure and characteristics of the benchmark instances proposed by Coelho, Cordeau, and Laporte [15] available at the author's website [13]. Three distinct sets of instances were created to support the SL pipeline. The training set is the smallest, designed to provide sufficient diversity for learning while keeping computational effort low. The validation set is slightly larger and used to tune parameters and prevent overfitting. Finally, a larger testing set was generated to evaluate the model's generalization performance.

To ensure model robustness, the three sets vary in key problem characteristics, including the number of customers, the planning horizon, and the cost parameters, such as inventory holding and position. While the training set covers a narrower range to facilitate learning consistency, the testing set introduces greater variability in both scale and parameter ranges to better evaluate model adaptability. Table 5.1 summarizes the distribution of key parameters across each subset.

Table 5.1: Parameter Ranges by Instance Sets

Characteristic	**Training**	**Validation**	**Testing**
Number of customers (N)	6, 8, 10	5, 7, 9	5, 10, 25, 50
Planning periods (T)	10, 8, 6	10, 8, 6	10, 20
Holding cost factor (hcf)	3, 6, 8	4, 7	1, 5, 10
Stockout cost factor (scf)	25, 70, 150	40, 100	20, 50, 200
Vehicle capacity factor (vcf)	1.3, 1.5, 1.7	1.4, 1.6	1.25, 1.5, 1.75
Number of Instances	$2 \cdot 81 = 162$	$10 \cdot 24 = 240$	$5 \cdot 216 = 1080$

The number of customers defines the size of the distribution network, and the number of periods sets the planning horizon over which delivery and inventory decisions are made. These two values jointly determine the dimensionality of each instance. Due to computational complexity, particularly in the resolution of the PH benchmark, the number of customers and periods is chosen to remain relatively small in training and validation instances, while the testing set includes more diverse and larger-scale combinations. Instance counts are determined as follows. For training and validation sets, N and T values are used in predefined pairs, resulting in 3 distinct (N,T) combinations in each set. For the testing set, all possible pairings of the selected N and T values are considered, resulting in $4 \times 2 = 8$ (N,T) combinations. Each pair is then combined with all combinations of the hcf, scf, and vcf, which results in:

- Training: $3 \times 3 \times 3 \times 3 = 81$ base configurations, each evaluated under 2 random seeds, in total 162 instances.
- Validation: $3 \times 2 \times 2 \times 2 = 24$ base configurations, each evaluated under 10 random seeds, in total 240 instances.
- Testing: $8 \times 3 \times 3 \times 3 = 216$ base configurations, each evaluated under 5 random seeds, in total 1080 instances.

A subset of the testing with the lowest numbers of customers and periods, i.e., n=5 and T=10, is used for some control comparison with the PH values. The holding cost factor influences the holding cost at customer i, which is computed as:

$$h_i = hcf \times U_i, \quad \text{where } U_i \sim \mathscr{U}(0.02, 0.1) \tag{5.1}$$

which allows for heterogeneous yet controlled holding costs between customers of the same instance. The stockout cost factor influences the penalty associated with lost sales by scaling the customer's holding cost:

$$z_i = scf \times h_i \tag{5.2}$$

Shortage and holding penalties remain proportionally connected, which might influence the inventory policy learning. The vehicle capacity factor is applied on top of the total expected demand across all customers:

$$V = vcf \times \sum_{i \in \mathcal{N}'} \mu_i \tag{5.3}$$

Lower vehicle capacity factors result in tighter routing constraints, increasing the challenge of meeting demand efficiently.

5.1.2 Supervised Learning Datasets

Since the SL model learns from the training data and is evaluated on the validation set, it is important that both datasets are of the highest possible quality. To ensure this, all instances in training and validation were solved to optimality using the PH method, providing the best possible reference values for learning. While solving the optimal version of each instance, we highlighted a set of features that can be categorized across two dimensions. The first dimension refers to the dynamism of the features. The feature can either be *static*, if its value is the same across all periods, or *dynamic*, if it varies from period to period. The second dimension indicates whether a feature is more relevant to routing or inventory management within the DSIRP. Either they fall into the group that includes features related to positional characteristics, which helps give insights into possible routing costs, or into the set that provides demand-specific features, allowing for the verification of the potential risk of stockout if no delivery occurs. We can also define a set of calculated features from the first two sets, which capture derived relations between demand, inventory, and routing characteristics, such as nPeriodsSinceLastDelivery, which relates holding costs with the isolation from one customer to all others. All those initial features are listed in Table 5.2, divided by their category.

Table 5.2: Feature categorization by domain and variable nature

	Inventory	**Routing**
Static	customerCapacity	distanceToDepot
	customerHoldingCost	isolation
	meanDemand	maxDistance
	shortageCost	minDistance
	stdDemand	nearCustomer
	vehicleCapacity	
Dynamic	customerInventory	isolationToPriorityCustomers
	nPeriodsToStockout	maxDistanceToPriority
	predictedLostSales	minDistanceToPriority
	utilRate	
		nPeriodsSinceLastDelivery

Most of these features have their normalized version, which tends to be used for most models to scale the features. The features that were not normalized are those computed from existing features. Since they often represent ratios or percentages, their values inherently lie between zero and one. The features represent what their names display. When they mention priority customers, it is related to customers whose current inventory is lower than or equal to the mean demand. We also store both the optimizer's decision regarding whether to visit or not and the quantity delivered in the case of a visit. In this work, we focus the classification task on the visit decision variable, while the delivery quantity is not explored further. For the regression task, we introduce a new target variable called nPeriodsToDeliver, which represents the number of periods until a customer is next visited. This variable is computed for each customer and based on the PH solution, by identifying the next period in which a visit occurs and subtracting the current period from it. A lower value of nPeriodsToDeliver indicates higher priority, with zero meaning the customer should be visited in the current period. Table 5.3 presents a sample of the dataset, where the target variable corresponds to either a classification or regression task. In the classification setting, the target indicates whether a visit occurs in a given period (1 for a visit, 0 otherwise). Accordingly, in the regression setting, the target reflects the number of periods remaining until the next delivery. For instance, since a visit occurs in period 2, the classification labels are 0 and 1 for periods 1 and 2, respectively, while the regression labels are 1 and 0, indicating one period to deliver in period 1 and none in period 2.

Table 5.3: Example dataset illustrating input features and output labels for classification (Visit) and regression (nPeriodsToDeliver) tasks

Type	**Customer**	**Period**	**Customer Capacity**	**Distance To Depot**	⋯	**Customer Inventory**	**Output**
Classification	1	1	372	356.7071533	⋯	87	0
	1	2	372	356.7071533	⋯	24	1
Regression	1	1	372	356.7071533	⋯	87	1
	1	2	372	356.7071533	⋯	24	0

5.2 Experimental Setup

The experimental setup consists of two main environments: the policy generator environment and the simulator environment. Each plays a distinct role in developing, optimizing, and evaluating the policies proposed in this work.

In the policy generator environment, all predictive models were implemented using the *scikit-learn* library in Python. After preliminary experimentation with several algorithms, including Support Vector Machines (SVM), Random Forests, and XGBoost, we selected the *MLPClassifier* and *MLPRegressor* classes to train multi-layer perceptrons for classification and regression tasks,

respectively, as they have consistently given superior or comparable performance across evaluation metrics. To identify suitable hyperparameter configurations, we combined manual tuning with the use of Optuna [3], a Bayesian optimization framework. Unlike exhaustive grid search, Optuna applies efficient sampling strategies to explore the search space and identify promising configurations based on past evaluations.

The simulator environment is developed in Java and is designed to emulate the DSIRP under a configurable three-step decision-making policy. At each decision point, it sequentially determines which customers to visit, the quantity to deliver to each, and the optimal delivery route, allowing the testing of different policy designs under realistic operational constraints. Although the simulator primarily operates through heuristic and learning-based policies, it occasionally incorporates exact optimization to evaluate or support specific decisions. For this purpose, the Gurobi Optimizer [23] (version 12.0) is used to compute optimal solutions when needed. For the training and validation datasets, Gurobi was used to solve full PH instances. In the testing phase, due to scalability constraints, only a subset of instances with five customers and ten periods were fully solved to obtain PH benchmarks. For the remaining testing instances, Gurobi was applied to solve the routing component within the three-stage decision-making procedure. Another tool used in this environment is Optuna, which is also employed to tune the simulator parameters associated with each trained model. One of those parameters is the threshold (e.g., with a threshold of 0.3, we decide to visit all customers which probability of visiting is over 30%), a decision boundary applied to the value predicted by the model for each customer, determining whether that customer should be visited in a given period. Additional parameters include the scaling parameters (α and τ) used in the policies to determine the quantity to deliver.

Regarding evaluation metrics, each environment adopts metrics aligned with its purpose. For classification models, we report the AUC, a robust measure that captures the model's ability to distinguish between classes across varying thresholds. For regression models, the RMSE is used, due to it placing greater emphasis on larger deviations and thus penalizing high-magnitude errors more heavily. The simulator-based evaluations rely on cost metrics computed for each instance. These include Inventory Costs, which account for the expense of holding inventory at customer locations, Lost Sales, representing the cost associated with unmet demand, and Routing Costs, defined by the vehicle distance traveled across the planning horizon. The sum of these components returns the Total Cost, which serves as the primary measure of policy effectiveness during simulation-based evaluation.

5.3 Iterative Model Development

During the development stage, the model suffered small improvements guided by conclusions made from previous iterations. For each of the four iterations made, we trained a model, compared its performance with PH, and noted the results and the gaps to be explored.

5.3.1 Initial Feature Set

The first iteration explores the use of SL to predict customer delivery decisions based on the initial feature set. We considered two types of models: a classification model that predicts whether a customer should be visited in a given period, and a regression model that estimates the number of periods until the customer should be visited. For both tasks, several candidate models were evaluated, with neural networks ultimately selected. The classification model uses a single hidden layer with 32 nodes, while the regression model uses a smaller architecture with one hidden layer of 16 nodes. Both models were trained using the training dataset and tuned on the validation dataset using Optuna, which also refines values for the simulator parameters: threshold, isolation τ, and alpha α. These values are summarized in Table 5.4.

Table 5.4: Model and simulator parameters for the first iteration

Type	**Parameter**	**Classification Model**	**Regression Model**
Model	Hidden layer size	32	16
	Learning rate	0.001	0.000117
	Max iterations	700	700
Simulator	Threshold	0.23	0.85
	Isolation	0.64	0.89
	Alpha	0.885	0.445

The comparison between model outputs and PH benchmarks is presented in Tables 5.5 and 5.6. Each table shows the top ten instances with the highest gaps in total cost between the SL model and PH solution, broken down into inventory holding, stockout, and routing costs. The instances vary across different configurations of holding and stockout cost factors, and vehicle capacity, which are shown as separate columns to support interpretation.

Table 5.5: Top 10 PH comparison gaps – Classification Initial model

		Total Costs		**Inventory Costs**		**Lost Sales**		**Routing Costs**	
hcf-scf-vcf	**Gap%**	**PH**	**SL**	**PH**	**SL**	**PH**	**SL**	**PH**	**SL**
10–200–1.25	172	7028	19101	1437	1724	0	11880	5592	5497
10–200–1.5	171	6984	18958	2028	1972	0	10880	4955	6105
10–200–1.5	171	6758	18321	1412	1895	0	10600	5346	5826
1–20–1.25	161	2779	7241	46	127	1308	152	1424	6962
10–200–1.5	141	7729	18662	1426	1322	0	10480	6302	6860
1–20–1.75	123	3383	7540	41	166	1710	38	1632	7336
10–200–1.25	123	7397	16478	2122	1836	140	9240	5136	5402
10–200–1.75	104	6618	13504	2875	2660	140	6520	3604	4324
1–20–1.5	102	3030	6113	42	276	2021	78	967	5760
10–200–1.5	99	6042	12021	1884	1748	120	5400	4038	4873

Table 5.6: Top 10 PH comparison gaps – Regression Initial model

		Total Costs		**Inventory Costs**		**Lost Sales**		**Routing Costs**	
hcf-scf-vcf	**Gap%**	**PH**	**SL**	**PH**	**SL**	**PH**	**SL**	**PH**	**SL**
1–20–1.25	171	2779	7536	46	161	1308	49	1424	7325
1–20–1.5	121	3030	6697	42	244	2021	90	967	6362
10–200–1.5	117	6042	13100	1884	1520	120	6800	4038	4781
10–200–1.25	111	7028	14843	1437	1287	0	7320	5592	6236
10–200–1.75	109	6618	13853	2875	2031	140	6860	3604	4963
1–20–1.75	109	3383	7077	41	166	1710	50	1632	6861
1–20–1.25	102	3104	6277	158	338	1048	41	1898	5898
1–20–1.5	90	4013	7615	114	243	1756	36	2143	7336
1–20–1.25	88	4548	8541	87	231	1560	15	2902	8294
10–200–1.5	84	9608	17707	2386	2291	0	6120	7222	9296

Two contrasting patterns emerge based on the magnitude of the shortage cost. For low values (e.g., scf = 20), the models tend to make unnecessary deliveries, incurring routing costs that the PH solution avoids by choosing not to deliver at all, since the cost of lost sales is lower in those scenarios. For high shortage costs (e.g., scf = 200), the models occasionally fail to identify critical delivery needs, leading to lost sales that the PH solution would have prevented. These insights

motivate the second iteration, which introduces features that relate shortage and holding costs with isolation, to better capture cost sensitivity across diverse scenarios and instances.

5.3.2 Relative Feature Set

The second iteration aims to improve the model's ability to adapt its delivery decisions according to specific characteristics of each instance. This is attempted by introducing two new features, *holdingToStockout* and *holdingToIsolation*, which incorporate information about the relationship between holding costs, shortage costs, and isolation factors. The first fits into inventory features and the second into both inventory and routing features. Both of the features are static. The goal is to better guide the model in distinguishing contexts where deliveries are necessary versus those where it may be optimal to accept lost sales, particularly in low-shortage-cost instances (e.g., scf = 20), where avoiding deliveries is often more cost-effective. Once again, several models were evaluated for both classification and regression tasks. The best-performing configurations are shown in Table 5.7, which includes both model hyperparameters, e.g., hidden layer size, among others and simulator control parameters, e.g, threshold, isolation and alpha.

Table 5.7: Model and simulator parameters for Relative features

Type	**Parameter**	**Classification**	**Regression**
Model	Hidden layer size	16	32
	Learning rate	0.000117	0.000117
	Max iterations	*default*	700
	Momentum	0.222	*default*
Simulator	Threshold	0.27	0.85
	Isolation	0.72	0.61
	Alpha	0.52	0.855

Tables 5.8 and 5.9 present the updated results compared to PH for the classification and regression models, respectively. In general, the gaps were reduced in instances with a shortage cost equal to 20. This improvement is especially visible in the regression model, as summarized in Table 5.10, which compares total costs across stockout cost factor levels for both iterations.

Table 5.8: Top 10 PH comparison gaps – Classification Relative model

		Total Costs		**Inventory Costs**		**Lost Sales**		**Routing Costs**	
hcf-scf-vcf	**Gap%**	**PH**	**SL**	**PH**	**SL**	**PH**	**SL**	**PH**	**SL**
10–200–1.25	192	7028	20521	1437	1176	0	13320	5592	6026
1–20–1.25	160	2779	7212	46	141	1308	63	1424	7008
5–200–1.25	132	8384	19419	1761	1087	0	10740	6624	7592
1–20–1.75	119	3383	7392	41	175	1710	36	1632	7181
1–20–1.25	102	3104	6277	158	338	1048	41	1898	5898
1–20–1.5	92	3030	5825	42	307	2021	2	967	5516
1–20–1.25	91	4548	8665	87	229	1560	13	2902	8422
1–20–1.75	90	4009	7632	121	282	2193	42	1694	7307
1–20–1.5	90	4013	7615	114	243	1756	36	2143	7336
1–20–1.25	86	3338	6208	136	199	465	1	2736	6008

Table 5.9: Top 10 PH comparison gaps – Regression Relative model

		Total Costs		**Inventory Costs**		**Lost Sales**		**Routing Costs**	
hcf-scf-vcf	**Gap%**	**PH**	**SL**	**PH**	**SL**	**PH**	**SL**	**PH**	**SL**
10–200–1.25	185	7028	20028	1437	1704	0	12720	5592	5604
1–20–1.25	181	2779	7813	46	151	1308	49	1424	7613
1–20–1.75	139	3383	8072	41	151	1710	102	1632	7819
10–200–1.5	125	6758	15178	1412	2282	0	6820	5346	6076
1–20–1.5	103	3030	6141	42	237	2021	90	967	5815
1–20–1.25	98	4548	9007	87	237	1560	14	2902	8757
10–200–1.5	79	9608	17227	2386	2616	0	6580	7222	8031
10–200–1.75	78	8284	14756	2405	3215	200	3160	5679	8381
1–20–1.25	77	4718	8351	198	461	2115	102	2405	7788
1–20–1.5	77	4013	7089	114	223	1756	53	2143	6813

Table 5.10: Total cost per shortage cost level and model type - Initial vs Relative

Model	**Shortage Cost**	**Initial**	**Relative**
Classification	SC 20	333477	347408
	SC 50	352938	347807
	SC 200	437515	404928
Regression	SC 20	348286	335773
	SC 50	345997	346585
	SC 200	396444	400813

While the regression model shows consistent improvement for scf = 20 instances, its predictions remain volatile under high-penalty (scf = 200) scenarios. To further investigate this behavior, we analyze in detail the instance with the largest gap to the PH in regression, focusing on both quantity and visit decisions, in Table 5.11. It includes all customer-period pairs where either the demand exceeded the mean demand, such as customer two in the first period, or the vehicle reached full capacity, such as customers three and four in the second period. These cases were specifically selected because they represent conditions that challenge the decision not to visit a customer, either due to unexpectedly high demand or because delivery was constrained by limited vehicle capacity.

Table 5.11: Individual Instance Analysis - SL Regression Relative model

T	N	Demand	Mean Demand	Demand > Mean	Shortage	Vehicle Capacity Full	Expected Delivery
1	1	78.0	75.0	Yes	0.0	No	No
1	2	98.0	63.0	Yes	-14.0	No	No
1	3	45.0	35.0	Yes	0.0	No	No
1	4	62.0	58.0	Yes	0.0	No	No
1	5	84.0	78.0	Yes	0.0	No	No
2	1	92.0	75.0	Yes	-43.0	No	No
2	2	82.0	63.0	Yes	0.0	No	No
2	3	26.0	35.0	No	0.0	Yes	Yes
2	4	67.0	58.0	Yes	-14.0	Yes	Yes
2	5	79.0	78.0	Yes	0.0	No	No
3	2	82.0	63.0	Yes	-15.0	No	No
3	3	60.0	35.0	Yes	0.0	No	No
3	4	69.0	58.0	Yes	0.0	No	No
3	5	102.0	78.0	Yes	0.0	No	No
4	1	93.0	75.0	Yes	0.0	No	No
4	3	36.0	35.0	Yes	0.0	No	No
4	4	60.0	58.0	Yes	0.0	No	No
4	5	92.0	78.0	Yes	0.0	No	No
5	1	81.0	75.0	Yes	0.0	No	No
5	5	121.0	78.0	Yes	0.0	No	No
6	3	60.0	35.0	Yes	-14.0	No	No
6	5	106.0	78.0	Yes	0.0	No	No
7	2	83.0	63.0	Yes	0.0	No	No
7	5	83.0	78.0	Yes	0.0	No	No
8	4	66.0	58.0	Yes	0.0	No	No
8	5	83.0	78.0	Yes	0.0	No	No
9	1	93.0	75.0	Yes	0.0	No	No
9	4	60.0	58.0	Yes	0.0	No	No
10	2	66.0	63.0	Yes	0.0	No	No
10	5	109.0	78.0	Yes	-5.0	No	No

While most cases with above-average demand do not result in shortages, there are outliers that lead to significant costs. Additionally, some stockouts occur when customers are not visited due to the vehicle reaching full capacity. While the first case is difficult to address, since the model already receives the mean and standard deviation of demand, the second could potentially be mitigated by adjusting delivery quantities for other customers or by delivering a period earlier for some of them.

This motivates the next two iterations, which introduce a Quantity Rationing heuristic to better balance delivery priorities when faced with capacity limitations and a few more features to try to capture the need to deliver earlier to some customers to avoid overloading the vehicle.

5.3.3 Quantity Rationing Heuristic

Both models so far operate under the assumption that quantities calculated by EOQ are delivered up to the point the vehicle becomes full, even if that leads to skipping some customers. This

can result in suboptimal delivery allocations, especially when sometimes it would be preferred to deliver less quantities to more customers. To mitigate this, we introduce the *Quantity Rationing Heuristic*, similar to Equal Quantity Discount (EQD) by Roldán, Basagoiti, and Coelho [45], a post-processing step that modifies the delivery quantities after customers have been selected, ensuring all selected customers are visited. Instead of dropping customers when total demand exceeds vehicle capacity, this heuristic proportionally scales down all delivery quantities. Let q_i denote the recalculated quantity delivered to customer i and q'_i the original estimated quantity. We define a scaling factor ε as:

$$\varepsilon = \min\left(1, \frac{V}{\sum_{i \in \mathcal{N}'} q'_i}\right) \tag{5.4}$$

$$q_i = \varepsilon \cdot q'_i, \quad \forall i \in \mathcal{N}' \tag{5.5}$$

This ensures total deliveries respect the vehicle capacity V, while allowing all selected customers to be served. The heuristic is integrated into the simulation without altering the trained models and simulator parameters. Tables 5.12 and 5.13 demonstrate improvements in most instances, indicating a generally consistent positive trend. The impact is particularly noticeable in high-shortage-cost scenarios. For example, in the classification model, the instance `hc5-sc200-vc1.75-1` experiences a reduction in its gap from 132%, in the absence of rationing, to 103%. In the context of regression, instance `hc10-sc200-vc1.25-4` improves from a gap of 185% to 107%.

Table 5.12: Top 10 PH comparison gaps – Classification with Quantity Rationing

		Total Costs		**Inventory Costs**		**Lost Sales**		**Routing Costs**	
hcf-scf-vcf	**Gap%**	**PH**	**SL**	**PH**	**SL**	**PH**	**SL**	**PH**	**SL**
1–20–1.25	159	2779	7200	46	130	1308	6	1424	7065
1–20–1.75	124	3383	7567	41	144	1710	2	1632	7422
1–20–1.25	117	3104	6733	158	251	1048	38	1898	6443
5–200–1.75	103	7173	14557	896	977	0	6150	6278	7430
1–20–1.5	96	3030	5934	42	281	2021	0	967	5653
1–20–1.25	89	3338	6313	136	160	465	3	2736	6150
1–20–1.25	84	4718	8704	198	423	2115	23	2405	8258
1–20–1.75	84	3519	6480	185	315	1294	0	2041	6165
1–20–1.25	82	4548	8288	87	256	1560	8	2902	8024
10–50–1.75	82	8296	15082	2183	2438	0	3885	6113	8759

Table 5.13: Top 10 PH comparison gaps – Regression with Quantity Rationing

		Total Costs		Inventory Costs		Lost Sales		Routing Costs	
hcf-scf-vcf	**Gap%**	**PH**	**SL**	**PH**	**SL**	**PH**	**SL**	**PH**	**SL**
1–20–1.25	184	2779	7883	46	95	1308	63	1424	7724
1–20–1.75	116	3383	7296	41	143	1710	78	1632	7075
10–200–1.25	107	7028	14564	1437	1428	0	6960	5592	6176
1–20–1.25	90	3104	5908	158	314	1048	25	1898	5569
1–20–1.25	90	4548	8627	87	207	1560	41	2902	8378
5–200–1.75	88	7173	13459	896	934	0	4850	6278	7675
1–20–1.5	82	3030	5525	42	264	2021	67	967	5195
10–200–1.5	79	9608	17227	2386	2616	0	6580	7222	8031
1–20–1.25	75	4718	8280	198	381	2115	22	2405	7877
1–200–1.5	75	4114	7195	112	107	144	548	3858	6540

To quantify the impact at scale, we summarize the total cost results by shortage cost group in Table 5.14, showing clear improvement in both classification and regression models compared to the previous iteration.

Table 5.14: Total cost per shortage cost level and model type - Vehicle Capacity vs Quantity Rationing

Model	**SC**	**Vehicle Capacity**	**Quantity Rationing**	**Improvement**
Classification	20	347408	333311	14097
	50	347807	343441	4366
	200	404928	382248	22680
Regression	20	335773	333311	2462
	50	346585	343441	3144
	200	400813	382248	18565

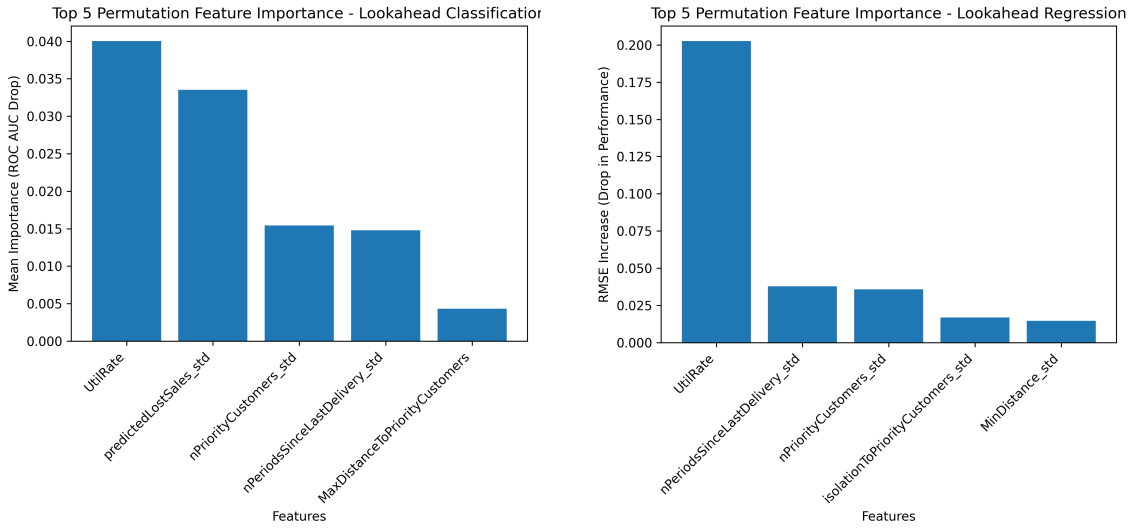
5.3.4 Lookahead Feature Set

The lookahead iteration seeks to enhance model performance by attempting to anticipate future delivery bottlenecks caused by vehicle capacity constraints. To achieve this, three additional features were introduced, *demandToCapacity*, *nPriorityCustomers*, and *nPriorityCustomersSecondPeriod*. The first feature fits in the static and inventory categories, while the other two fit in the dynamic and inventory and routing categories. The last feature counts how many customers are expected to reach a critically low inventory within the next two periods, i.e., below twice their mean demand. These features aim to allow the model to plan more proactively and prevent accumulation

of critical deliveries in a single period. Table 5.15 presents the selected hyperparameters for both classification and regression models. While all three features were included in the training set, only *nPriorityCustomers* proved to be impactful, as reflected in the permutation importance analysis displayed in Figures 5.1a and 5.1b.

Table 5.15: Model and simulator parameters for lookahead iteration

Category	**Parameter**	**Classification**	**Regression**
Model	Hidden layer size	16	16
	Learning rate	0.000117	0.000117
	Max iterations	*default*	700
	Momentum	0.222	*default*
Simulator	Threshold	0.35	0.74
	Isolation	0.70	0.65
	Alpha	0.97	0.81



(a) Classification Features

(b) Regression Features

Figure 5.1: Permutation Importance Top Five - Lookahead Features

Full performance comparisons against the PH benchmark are detailed in Table 5.16 and Table 5.17. Overall, the results show marginal variation from the previous iteration with quantity rationing, suggesting that the added features had limited additional impact. The results remain consistent with earlier trends, but improvements were instance-specific rather than systematic. For example, `hc10-sc200-vc1.25-4` and `hc10-sc200-vc1.5-1` reduce the gap for regression while `hc1-sc20-vc1.75-3` increases.

Table 5.16: Top 10 PH comparison gaps – Classification Lookahead model

		Total Costs		Inventory Costs		Lost Sales		Routing Costs	
hcf-scf-vcf	**Gap%**	**PH**	**SL**	**PH**	**SL**	**PH**	**SL**	**PH**	**SL**
1–20–1.25	211	8633	118	14	8501	2779	46	1308	1424
1–20–1.25	121	6845	286	7	6551	3104	158	1048	1898
1–20–1.75	119	7410	148	54	7208	3383	41	1710	1632
1–20–1.25	94	8835	232	26	8577	4548	87	1560	2902
1–20–1.25	93	6425	153	0	6272	3338	136	465	2736
1–20–1.5	88	5701	285	23	5393	3030	42	2021	967
1–20–1.25	86	8756	417	22	8317	4718	198	2115	2405
10–200–1.25	84	12900	1923	5120	5857	7028	1437	0	5592
5–200–1.75	79	12844	1239	4500	7105	7173	896	0	6278
5–200–1.5	74	10192	1456	3600	5135	5851	1170	120	4561

Table 5.17: Top 10 PH comparison gaps – Regression Lookahead model

		Total Costs		Inventory Costs		Lost Sales		Routing Costs	
hcf-scf-vcf	**Gap%**	**PH**	**SL**	**PH**	**SL**	**PH**	**SL**	**PH**	**SL**
1–20–1.25	185	7926	172	14	7740	2779	46	1308	1424
1–20–1.75	130	7782	184	59	7539	3383	41	1710	1632
1–20–1.25	99	6647	154	6	6488	3338	136	465	2736
1–20–1.25	98	6154	352	2	5800	3104	158	1048	1898
10–200–1.25	95	13720	2090	5280	6350	7028	1437	0	5592
1–20–1.25	91	9028	390	43	8596	4718	198	2115	2405
1–20–1.75	85	6519	294	14	6211	3519	185	1294	2041
10–200–1.5	83	12782	3365	3640	5777	6984	2028	0	4955
1–20–1.5	81	5480	297	34	5149	3030	42	2021	967
1–20–1.25	75	7960	206	28	7726	4548	87	1560	2902

Table 5.18 summarizes the total costs grouped by shortage cost category, allowing comparison between the lookahead model and the previous iteration.

Table 5.18: Shortage Cost Rationing Comparison: Relative vs Lookahead Model

Model Type	**SC**	**Relative**	**Lookahead**
Classification	20	348072	347857
	50	350288	347405
	200	382948	383467
Regression	20	333311	350312
	50	343441	354523
	200	382248	378257

In summary, this lookahead iteration shows that while feature augmentation can refine predictions, its benefits are conditional and context-dependent. Most gains came from the earlier structural improvements with the relative adaptation and delivery quantity heuristics.

5.4 Literature Benchmarks Comparison

In general, from iteration to iteration, there were slight improvements on the model's attempt to replicate the PH behaviour. However, it would be interesting to have a model capable of generalizing its behaviour to larger instances. For that, we compared the whole testing set for each of the obtained models and two literature benchmarks. The first follows one of Roldán, Basagoiti, and Coelho [45] approaches, which allows more freedom in customer visit decisions. The second explores Silver's Policy [49] with two different possibilities, either using its reorder point and reducing the SL to a simple customer priorisation decision, or not using the reorder point and compare with the EOQ results.

5.4.1 Roldán's Benchmark

In this benchmark, customers are ordered and visited by priority on Lower Inventory First (LIF), where the reorder point is 25%, which means customers are visited if they fall under 25% of their capacity. The quantity selected to deliver for each customer is calculated by the EOQ, the same way as our models, to allow better comparison of the SL performance.

Table 5.19 reflects the gaps from each model to our benchmark. For the validation set of instances, which was our decision set, all models beat Roldán [45]. If we select the best-performing model according to the validation, we also outperform the benchmark model on the testing set, even thought this model is not the top performer on that set. However, there are some models that, despite performing well on validation, lack performance on the testing set, which displays some overfitting to validation. We also noticed that the introduction of the rationing method leads to a decrease in performance on the training set. This can be attributed to the fact that the model is optimized based on the decision-making process used during training. Therefore, any alteration to this process, such as replacing the original delivery selection mechanism with the rationing method, can negatively impact performance, as the model is no longer aligned with the structure it was

trained to predict. Since the models were only affected by validation when tuning the parameters for the simulator, we explored partial parts of the testing set. Testing $T = 10$ includes all testing instances with ten periods only, Testing $\mathscr{N}\backslash\{50\}$ removes from the testing set the instances with fifty customers, and Testing $n = 5, T = 10$ just includes the five customers and ten-period instances. The largest improvements are noticed for the Testing $T = 10$, which means the main bottleneck identified is the instances with twenty periods. There are slight improvements on the $\mathscr{N}\backslash\{50\}$ instances, but lower.

Table 5.19: Percentual Gap Comparison across Models to Roldán

Set	**Metric**	**Classification**				**Regression**				**Roldán**
		Initial	Relative	Relative Rationing	Lookahead Rationing	Initial	Relative	Relative Rationing	Lookahead Rationing	Value
Train	Total	-0.60	-0.55	6.54	9.13	-1.19	-0.62	5.98	8.15	1180123
Valid	Total	-0.40	-0.49	-1.66	-0.88	-1.21	-0.68	-1.94	-1.13	1756050
	$T = 10$	2.70	-4.50	-5.16	3.34	1.43	-1.83	-3.95	2.57	10646155
	$\mathscr{N}\backslash\{50\}$	7.38	-1.52	-2.52	5.74	4.50	-0.11	-2.01	3.20	17666100
	$n = 5, T = 10$	4.34	2.14	0.39	0.15	1.26	0.56	-1.68	0.55	1077135
Test	Total	8.73	-2.57	-3.54	8.78	6.68	0.53	-0.88	7.08	32088825
	Inventory Costs	-8.79	-12.93	-14.17	-3.46	-22.17	-7.36	-8.56	-10.12	9803587
	Routing Costs	-6.38	-1.13	0.74	-4.78	-1.71	-4.55	-3.25	-3.95	18062600
	Lost Sales	114.06	15.30	2.87	95.17	109.58	40.55	27.08	94.17	4222638

When analyzing differences between models, there is a general improvement from Initial to Relative, then a slight improvement to Relative with Rationing. The Lookahead model doesn't generalize well, despite good performance on the 135 small instances, i.e, n=5, T=10, compared with other models. All models tend to incur larger lost sales values compared with Roldán [45], gaining on the Routing Costs in most cases and all cases on Inventory Costs. The best tradeoff identified is the changes between Relative with the normal approach and Rationing, where in both Classification and Regression, there is improvement in both Inventory Costs and Lost Sales at a smaller loss on Routing Costs, which displays better management of quantities distributed.

Table 5.20 displays how each model trained behaves across the training and validation sets. The increase of RMSE from model to model displays an improvement with the addition of new features. However, using the knowledge from the percentual gap comparison, it can be concluded that there was an overfit to smaller instances on the lookahead model. A similar situation happened with the classification models, where all models have strong AUC performance, but the model with the lowest AUC managed to generalize the best.

Table 5.20: Performance metrics across feature types

Task	Metric	Initial	Relative	Lookahead
Regression	RMSE Training	0.7667	0.7571	0.7472
	RMSE Validation	0.7311	0.7229	0.7197
Classification	AUC Training	0.9540	0.9183	0.9272
	AUC Validation	0.9345	0.9092	0.9172

5.4.2 Silver's Benchmark

We attempted to apply Silver's quantity policy as an alternative to the EOQ. For that, we considered two options: using the reorder point defined in Silver's policy, or ignoring it and applying only the quantities calculated by the policy. The first option, which included the reorder point, produced the best results, as shown in Table 5.21.

Table 5.21: Percentual Gap Comparison across Models to Silver with Reorder Point

Metric	**Classification**			**Regression**			**Silver**
	Initial	Relative	Lookahead	Initial	Relative	Lookahead	Value
Testing Total	1.99	2.00	1.78	0.90	2.01	2.17	29233151
Inventory Costs	-0.34	-0.88	-0.57	1.53	-0.68	-1.22	9741211
Routing Costs	0.42	0.43	0.38	-0.56	0.36	0.58	17240290
Lost Sales	24.07	26.50	22.72	9.42	26.31	29.01	2251650

The SL policy serves only as a customer prioritization mechanism, as the decision of which customers to visit is dictated by the fixed reorder point policy. As a result, the SL model cannot influence visit selection directly and underperforms compared to the LIF approach used in Silver's benchmark policy. Additionally, the differences between results are constrained by the strong reorder point policy, which overrides many of the customer priorities designed by the SL logic.

The second option, where the reorder point is omitted, is presented in Table 5.22, comparing EOQ and Silver's policies strictly in terms of quantity calculation. Only one model performed better with Silver's quantities, suggesting that those quantities are most effective when paired with the reorder point. In any case, our best model using the EOQ policy remains 5.87% behind the Silver policy with a reorder point.

Table 5.22: EOQ vs Silver: Percentual Gap Comparison across Models

Metric	Classification				Regression			
	Initial	Relative	Relative Rationing	Lookahead Rationing	Initial	Relative	Relative Rationing	Lookahead Rationing
Testing Total	34890315	31262693	30953983	34905012	34233458	32258146	31807453	34360372
	36043164	32916060	32189715	34832281	35041134	33078015	32634240	34494825
	3.30	5.29	3.99	-0.21	2.36	2.54	2.60	0.39
Inventory Costs	8941764	8536037	8414213	9464102	7630057	9081613	8964827	8811562
	8875931	9482848	9272732	9409634	9016631	9273999	9145920	9155977
	-0.74	11.09	10.20	-0.58	18.17	2.12	2.02	3.91
Routing Costs	16909763	17858064	18195848	17199679	17753576	17241490	17476417	17349666
	16833088	17238288	17585031	17228219	17041280	17175702	17427272	17334463
	-0.45	-3.47	-3.36	0.17	-4.01	-0.38	-0.28	-0.09
Lost Sales	9038788	4868592	4343922	8241230	8849825	5935044	5366209	8199145
	10334144	6194922	5331952	8194427	8983222	6628314	6061047	8004385
	14.33	27.24	22.75	-0.57	1.51	11.68	12.95	-2.38

5.5 Time Performance

The time performance of the SL pipeline is closely tied to the scale of the dataset used for model training and evaluation. In contrast, the benchmark methods used throughout this work rely on lightweight heuristic policies, which naturally result in faster execution times. The values to compare were obtained from Roldán's Benchmark. Table 5.23 provides a comparative summary of execution durations across both simulation and model tasks.

Table 5.23: Time performance comparison between Supervised Learning and Benchmark approaches

Type	Task	Supervised Learning	Benchmark
Simulator	Single Instance Run	75 milliseconds	61 milliseconds
	Validation Tuning	26.7 minutes (8s $\times$ 200)	11.7 minutes (7s $\times$ 100)
Model	Hyperparameter Tuning	30–180 minutes	—
	Model Training	6–60 seconds	—

The increase in total simulation time from 61 milliseconds to 75 milliseconds represents an approximate 23% overhead. Despite existing a relatively modest increase, the majority of the time for each case is mainly attributable to the use of Gurobi to solve the TSP for the routing decisions in both. For the validation phase, the runtime nearly doubles, rising from 700 to 1600 seconds, primarily due to the additional need to optimize the decision threshold parameter, which is not present in the heuristic benchmarks. Both include multiple runs to represent the tuning process of the parameters present.

In the model development process, hyperparameter tuning is the most time-consuming step, with durations ranging from 30 minutes to 3 hours depending on the architecture complexity (e.g., number and size of hidden layers). This is because tuning involves training and evaluating the model multiple times across different combinations of parameters to find the optimal configuration. The more complex the model, the longer each training run takes, and the more combinations need to be tested. In contrast, training a model with the selected parameters is highly efficient, often completed in under one minute. As such, the majority of the computational time in the SL approach is concentrated in the tuning stage. Once completed, subsequent training and simulation runs proceed rapidly, supporting scalable and repeatable experimentation.

Chapter 6

Conclusions and Future Work

This chapter presents the conclusions drawn from the development of the dissertation project. It reflects on whether the proposed objectives were met and outlines further developments that could be done to the project and for the broader research area.

6.1 Conclusions

Solving the DSIRP requires balancing multiple interconnected decisions under uncertainty, making it a particularly challenging problem for traditional optimization methods. While RL has shown promise for sequential decision-making tasks, its high computational requirements and training complexity pose practical limitations. In this context, SL emerges as a possible alternative, offering scalability by learning directly from expert or optimal decisions.

The main objective of the project was to explore the application of SL to the DSIRP, with the aim of minimizing total costs. This objective is particularly challenging due to the inherent trade-off between routing costs, associated with visiting a customer, and the potential lost sales when customers are not visited. The literature review highlighted a clear gap in the application of SL to customer visit decisions. This gap made the project focus on both customer prioritization and visit selection, effectively addressing RQ 1.

The development of the project relied on an iterative process across two main environments. The Policy Generator environment focused on model training and evaluation using decision-specific metrics. The Simulation environment combined the SL model for deciding to visit or not to visit specific customer, the Economic Order Quantity heuristic and Silver's Policy to determine delivery quantities and a MIP solver to solve a TSP to optimize routing decisions. This setup allowed us to replicate multi-period decisions.

In each iteration, distinct classification and regression models were generated through the refinement of features or customer selection strategies. These models were evaluated using both simulator metrics, such as total cost, inventory cost, lost sales, and routing cost, and ML metrics, including AUC for classification and RMSE for regression. Each iteration was guided by analysis of Perfect Hindsight benchmarks for a small, representative set of instances.

Results show improvements over a benchmark policy from the literature, particularly for smaller instances with fewer periods. However, due to limited vehicle capacity, it was not always possible to serve all customers. To address this, a Rationing Heuristic was introduced to enable visiting all selected customers by proportionally reducing delivered quantities. This approach led to a notable reduction in lost sales and associated costs. Despite these gains, the lookahead model performed worse than previous iterations, likely due to the introduction of additional features that did not generalize well.

These findings, combined with the analysis of AUC and RMSE across iterations, suggest some degree of overfitting to the set of training and validation instances. The models showed improved decision-making during training with the best values of AUC and RMSE at the end, but failed to generalize to the broader testing set. This gives mixed answers to RQ 2. While improvements over the benchmark policy confirm the potential of SL when applied with flexibility, a significant performance gap remains relative to Silver's policy. This indicates that, although promising, SL-based approaches still require refinement before they can be considered reliable alternatives for the DSIRP.

6.2 Future Work

To further advance this line of research, it is essential to understand the behavior of the whole system. Several directions for future work are proposed below, focusing on dataset design, feature engineering, model optimization, and simulation strategy. First, a more diverse and representative dataset could substantially improve model accuracy and robustness. The fact that the main performance drop occurs in instances with longer planning horizons indicates that the models lack sufficient exposure to such cases during training. Although computationally intensive, expanding PH computations to cover longer horizons, even with fewer customers, could help bridge this gap. In terms of feature engineering, further improvements could arise from identifying features that better capture the cost trade-offs inherent to the DSIRP. For example, refining or replacing the *holdingToRouting* feature with a more expressive indicator could help models learn when deliveries are strategically avoidable.

Regarding model training, although hyperparameters for the Multi Layer Perceptron models were selected via both grid search tuning and Optuna, the latter may have contributed to overfitting on the training set. Future work could explore more robust feature selection and regularization techniques to mitigate this risk. Finally, alternative simulation policies could be explored. These might involve enhancing the three-step policy within individual decision layers, establishing a linkage between SL predictions across multiple layers, or formulating an integrated approach in which SL is used to learn parameters that guide or accelerate mathematical optimization routines.

References

[1] Fatima Ezzahra Achamrah, Fouad Riane, and Sabine Limbourg. "Solving inventory routing with transshipment and substitution under dynamic and stochastic demands using genetic algorithm and deep reinforcement learning". In: *International Journal of Production Research* 60.20 (2022), pp. 6187–6204. DOI: 10.1080/00207543.2021.1987549. eprint: https://doi.org/10.1080/00207543.2021.1987549. URL: https://doi.org/10.1080/00207543.2021.1987549.

[2] K. Ahn et al. "Congestion-aware dynamic routing for an overhead hoist transporter system using a graph convolutional gated recurrent unit". In: *IISE Transactions* 54.8 (2022), pp. 803–816. DOI: 10.1080/24725854.2021.2000680.

[3] Takuya Akiba et al. *Optuna: A Next-generation Hyperparameter Optimization Framework*. 2019. arXiv: 1907.10902 [cs.LG]. URL: https://arxiv.org/abs/1907.10902.

[4] F. Akkerman, D. Prak, and M. Mes. "Dynamic reordering and inspection for the multi-item Inventory Record Inaccuracy problem". In: *European Journal of Operational Research* (2024), pp. 1–17. DOI: 10.1016/j.ejor.2024.09.033.

[5] Q. Arnau, A. A. Juan, and I. Serra. "On the use of learnheuristics in vehicle routing optimization problems with dynamic inputs". In: *Algorithms* 11.12 (2018). DOI: 10.3390/a11120208.

[6] A. Asadi and S. N. Pinkley. "A Monotone Approximate Dynamic Programming Approach for the Stochastic Scheduling, Allocation, and Inventory Replenishment Problem: Applications to Drone and Electric Vehicle Battery Swap Stations". In: *Transportation Science* 56.4 (2022), pp. 1085–1110. DOI: 10.1287/trsc.2021.1108.

[7] L. Baty et al. "Combinatorial Optimization-Enriched Machine Learning to Solve the Dynamic Vehicle Routing Problem with Time Windows". In: *Transportation Science* 58.4 (2024), pp. 708–725. DOI: 10.1287/trsc.2023.0107.

[8] Walter J. Bell et al. "Improving the Distribution of Industrial Gases with an On-Line Computerized Routing and Scheduling Optimizer". In: *Interfaces* 13.6 (Dec. 1983), pp. 4–23. DOI: 10.1287/inte.13.6.4. URL: https://ideas.repec.org/a/inm/orinte/v13y1983i6p4-23.html.

[9] Richard Bellman. *DYNAMIC PROGRAMMING*. Cited by: 9943. 2021, pp. 1–346. DOI: 10.2307/j.ctv1nxcw0f. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85012688561&doi=10.2307%2fj.ctv1nxcw0f&partnerID=40&md5=6f5422f08230215c13b8e8ba6c572900.

[10] Jan Brinkmann, Marlin W. Ulmer, and Dirk C. Mattfeld. "Dynamic Lookahead Policies for Stochastic-Dynamic Inventory Routing in Bike Sharing Systems". In: *Computers Operations Research* 106 (2019), pp. 260–279. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2018.06.004. URL: https://www.sciencedirect.com/science/article/pii/S0305054818301588.

[11] Ann Melissa Campbell and Martin W.P. Savelsbergh. "A decomposition approach for the inventory-routing problem". In: *Transportation Science* 38.4 (2004), pp. 488–502. DOI: 10.1287/trsc.1030.0063.

[12] J. Choi, T. Yu, and D. G. Choi. "Dynamic OHT Routing Using Travel Time Approximation Based on Deep Neural Network". In: *IEEE Access* 12 (2024), pp. 6900–6911. DOI: 10.1109/ACCESS.2024.3351225.

[13] Leandro Coelho. *Stochastic and Dynamic Inventory Routing Instances*. https://www.leandro-coelho.com/instances/stochastic-and-dynamic-inventory-routing/. 2024.

[14] Leandro C. Coelho, Jean François Cordeau, and Gilbert Laporte. "Heuristics for dynamic and stochastic inventory-routing". English. In: *Computers and Operations Research* 52.PART A (Dec. 2014), pp. 55–67. ISSN: 0305-0548. DOI: 10.1016/j.cor.2014.07.001.

[15] Leandro C. Coelho, Jean-François Cordeau, and Gilbert Laporte. *Dynamic and Stochastic Inventory-Routing*. Tech. rep. CIRRELT-2012-66. CIRRELT, 2012. URL: https://www.cirrelt.ca/documentstravail/cirrelt-2012-37.pdf.

[16] Leandro C. Coelho, Gilbert Laporte, and Jean-François Cordeau. *Dynamic and Stochastic Inventory-Routing*. Tech. rep. CIRRELT-2012-37. Montreal, Canada: CIRRELT, 2012. URL: https://www.cirrelt.ca/DocumentsTravail/CIRRELT-2012-37.pdf.

[17] Maximiliano Cubillos, Remy Spliet, and Sanne Wøhlk. "On the effect of using sensors and dynamic forecasts in inventory-routing problems". In: *INFOR: Information Systems and Operational Research* 60.4 (2022), pp. 473–490. DOI: 10.1080/03155986.2022.2073110. eprint: https://doi.org/10.1080/03155986.2022.2073110. URL: https://doi.org/10.1080/03155986.2022.2073110.

[18] George B. Dantzig and John H. Ramser. "The Truck Dispatching Problem". In: *Management Science* 6.1 (1959), pp. 80–91. DOI: 10.1287/mnsc.6.1.80.

[19] L. Fan et al. "Change is safer: a dynamic safety stock model for inventory management of large manufacturing enterprise based on intermittent time series forecasting". In: *Journal of Intelligent Manufacturing* (2024). DOI: 10.1007/s10845-024-02442-y.

[20] Awi Federgruen and Paul Zipkin. "A Combined Vehicle Routing and Inventory Allocation Problem". In: *Operations Research* 32.5 (Oct. 1984), pp. 1019–1037. DOI: 10.1287/opre.32.5.1019. URL: https://ideas.repec.org/a/inm/oropre/v32y1984i5p1019-1037.html.

[21] Michel Gendreau, Alain Hertz, and Gilbert Laporte. "A tabu search heuristic for the vehicle routing problem". In: *Management Science* 40.10 (1994), pp. 1276–1290.

[22] Toni Greif et al. "Combinatorial Optimization and Machine Learning for Dynamic Inventory Routing". In: *arXiv preprint arXiv:2402.04463* (2024).

[23] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2024. URL: https://www.gurobi.com.

[24] G. Haixiang et al. “Period sewage recycling vehicle routing problem based on real-time data”. In: *Journal of Cleaner Production* 288 (2021), p. 125628. ISSN: 0959-6526. DOI: 10.1016/j.jclepro.2020.125628. URL: https://www.sciencedirect.com/science/article/pii/S0959652620356742.

[25] D. Han et al. “Optimal Electrical Vehicle Charging Planning and Routing Using Real-Time Trained Energy Prediction With Physics-Based Powertrain Model”. In: *IEEE Access* 12 (2024), pp. 123250–123266. DOI: 10.1109/ACCESS.2024.3438993.

[26] F. W. Harris. “How many parts to make at once”. In: *The Magazine of Management* 10.2 (1913). Reprinted in Operations Research, 38(6), 1990, pp. 947–950, pp. 135–136, 152.

[27] J. Jemai and K. Mellouli. “A neural-tabu search heuristic for the real time vehicle routing problem”. In: *Journal of Mathematical Modelling and Algorithms* 7.2 (2008), pp. 161–176. DOI: 10.1007/s10852-008-9082-0.

[28] Menglei Jia, Albert H Schrotenboer, and Feng Chen. “Scenario Predict-then-Optimize for Data-Driven Online Inventory Routing”. In: *arXiv preprint arXiv:2401.17787* (2024).

[29] Astrid S Kenyon and David P Morton. “Stochastic vehicle routing with random travel times”. In: *Transportation Science* 37.1 (2003), pp. 69–82.

[30] E. T. Küp et al. “An Integrated Framework for Dynamic Vehicle Routing Problems with Pick-up and Delivery Time Windows and Shared Fleet Capacity Planning”. In: *Symmetry* 16.4 (2024). DOI: 10.3390/sym16040505.

[31] A. Y. Ladha and N. K. Chaubey. “PAARGAMAN: Passenger Demand Provoked (On-The-Fly) Routing Of Intelligent Public Transport Vehicle with Dynamic Route Updation, Generation, and Suggestion”. In: *International Journal on Recent and Innovation Trends in Computing and Communication* 11 (2023), pp. 391–405. DOI: 10.17762/ijritcc.v11i8s.7219.

[32] D. Li et al. “Dynamic sales prediction with auto-learning and elastic-adjustment mechanism for inventory optimization”. In: *Information Systems* 119 (2023). DOI: 10.1016/j.is.2023.102259.

[33] J. Li et al. “Dynamic inventory allocation for seasonal merchandise at Dillard’s”. In: *INFORMS Journal on Applied Analytics* 51.4 (2021), pp. 297–311. DOI: 10.1287/INTE.2020.1068.

[34] H. Liang. “A self-adaptive prediction model for dynamic pricing and inventory control”. In: *Metallurgical and Mining Industry* 7.12 (2015), pp. 80–94.

[35] Francisco Maia, Gonçalo Figueira, and Fábio Neves-Moreira. “Stochastic Dynamic Inventory-Routing: A Comprehensive Review”. Manuscript under review at Computers & Operations Research. 2025.

[36] Iliya Markov et al. “Waste collection inventory routing with non-stationary stochastic demands”. In: *Computers Operations Research* 113 (2020), p. 104798. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2019.104798. URL: https://www.sciencedirect.com/science/article/pii/S0305054819302400.

[37] R.S. McKenna, M.J. Robbins, B.J. Lunday, et al. “Approximate dynamic programming for the military inventory routing problem”. In: *Annals of Operations Research* 288 (2020), pp. 391–416. DOI: 10.1007/s10479-019-03469-8.

[38] Seyed M.J. Mirzapour Al-e-hashem, Yacine Rekik, and Ebrahim Mohammadi Hoseinhajlou. "A hybrid L-shaped method to solve a bi-objective stochastic transshipment-enabled inventory routing problem". In: *International Journal of Production Economics* 209 (2019). The Proceedings of the 19th International Symposium on Inventories, pp. 381–398. ISSN: 0925-5273. DOI: https://doi.org/10.1016/j.ijpe.2017.06.020. URL: https://www.sciencedirect.com/science/article/pii/S0925527317301895.

[39] Fábio Neves-Moreira et al. "The multi-product inventory-routing problem with pickups and deliveries: Mitigating fluctuating demand via rolling horizon heuristics". In: *Transportation Research Part E: Logistics and Transportation Review* 164 (2022), p. 102791. ISSN: 1366-5545. DOI: https://doi.org/10.1016/j.tre.2022.102791. URL: https://www.sciencedirect.com/science/article/pii/S1366554522001806.

[40] Pamela C. Nolz, Nabil Absi, and Dominique Feillet. "A stochastic inventory routing problem for infectious medical waste collection". In: *Networks* 63.1 (2014), pp. 82–95. DOI: https://doi.org/10.1002/net.21523. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.21523. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.21523.

[41] Warren B Powell. *Reinforcement learning and stochastic optimization*. 2021.

[42] Christian Prins. "A simple and effective evolutionary algorithm for the vehicle routing problem". In: *Computers & Operations Research* 31.12 (2004), pp. 1985–2002.

[43] X. Ren et al. "Anticipatory shipping versus emergency shipment: data-driven optimal inventory models for online retailers". In: *International Journal of Production Research* 62.7 (2024), pp. 2548–2565. DOI: 10.1080/00207543.2023.2219343.

[44] C. C. Reyes-Aldasoro et al. "A hybrid model based on dynamic programming, neural networks, and surrogate value for inventory optimisation applications". In: *Journal of the Operational Research Society* 50.1 (1999), pp. 85–94. DOI: 10.1057/palgrave.jors.2600658.

[45] Raúl F. Roldán, Rosa Basagoiti, and Leandro C. Coelho. "Robustness of inventory replenishment and customer selection policies for the dynamic and stochastic inventory-routing problem". In: *Computers Operations Research* 74 (2016), pp. 14–20. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2016.04.004. URL: https://www.sciencedirect.com/science/article/pii/S0305054816300740.

[46] Stefan Ropke and David Pisinger. "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows". In: *Transportation Science* 40.4 (2006), pp. 455–472.

[47] Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. *A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning*. 2011. arXiv: 1011.0686 [cs.LG]. URL: https://arxiv.org/abs/1011.0686.

[48] Colin Shearer. "The CRISP-DM model: The new blueprint for data mining". In: *Journal of Data Warehousing* 5.4 (2000), pp. 13–22.

[49] Edward A. Silver, David F. Pyke, and Rein Peterson. *Inventory and Production Management in Supply Chains*. 4th. CRC Press, 2016. ISBN: 9781466558618.

[50] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. 2nd. MIT Press, 2018.

[51] A. Violi et al. “An age-based dynamic approach for distribution of perishable commodities with stochastic demands”. In: *Soft Computing* 27.11 (June 2023), pp. 7039–7050. DOI: 10.1007/s00500-023-07917-3.

[52] J. Wei, X. Xu, and B. Yang. “Data-driven vehicle rental and routing optimization: An application in online retailing”. In: *Computers and Industrial Engineering* 197 (2024). DOI: 10.1016/j.cie.2024.110588.

[53] X. Xiang et al. “A Pairwise Proximity Learning-Based Ant Colony Algorithm for Dynamic Vehicle Routing Problems”. In: *IEEE Transactions on Intelligent Transportation Systems* 23.6 (2022), pp. 5275–5286. DOI: 10.1109/TITS.2021.3052834.

[54] B. Yang et al. “Data-driven optimization models for inventory and financing decisions in online retailing platforms”. In: *Annals of Operations Research* 339.1–2 (2024), pp. 741–764. DOI: 10.1007/s10479-023-05234-4.

[55] L. Zhao et al. “Novel Online Sequential Learning-Based Adaptive Routing for Edge Software-Defined Vehicular Networks”. In: *IEEE Transactions on Wireless Communications* 20.5 (2021), pp. 2991–3004. DOI: 10.1109/TWC.2020.3046275.

[56] C. Zhou et al. “Reinforcement Learning-based approach for dynamic vehicle routing problem with stochastic demand”. In: *Computers and Industrial Engineering* 182 (2023). DOI: 10.1016/j.cie.2023.109443.