INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA DE INTELIGÊNCIA ARTIFICIAL




Otimização Inteligente na Gestão de Armazéns: Aplicação de Machine Learning para Previsão de Quantidades e Alocação Eficiente

ÁLVARO JOSÉ FERNANDES FRANCO
outubro de 2025

P.PORTO

Intelligent Optimization in Warehouse Management: Application of Machine Learning for Quantity Forecasting and Efficient Allocation

Álvaro José Fernandes Franco

Student no. 1141270

Dissertation for the Master's Degree in Artificial Intelligence Engineering

Advisor: Luiz Felipe Rocha de Faria, Associate Professor, Institute of Engineering, Polytechnic of Porto

Evaluation Committee:

President: Paulo Sérgio dos Santos Matos, Assistant Professor, Institute of Engineering, Polytechnic of Porto

Members:

Luiz Felipe Rocha de Faria, Associate Professor, Institute of Engineering, Polytechnic of Porto

Luís Manuel Silva Conceição, Assistant Professor, Institute of Engineering, Polytechnic of Porto

Oporto, September 2025

Dedication

To my parents, for the love, support, and values they've always instilled in me.

To my girlfriend, for her affection, patience and for always being by my side at every stage of this journey.

Resumo

Esta dissertação propõe uma framework de inteligência artificial integrada para otimizar operações de armazém através de duas tarefas complementares: a previsão diária da saída de produtos e a resolução do problema de empacotamento em 3D (3D Bin Packing Problem) para uma alocação de espaço mais eficiente. Na primeira etapa, são aplicadas técnicas de aprendizagem automática aos dados históricos de movimentações no armazém para prever a quantidade total de produtos que se espera transferir em cada dia.

Na segunda etapa, os volumes previstos são alocados em contentores tridimensionais com restrições, utilizando métodos baseados em aprendizagem por reforço. Foi desenvolvido um ambiente personalizado em OpenAI Gym para simular condições realistas de empacotamento, incluindo rotação das caixas, deteção de colisões, restrições de empilhamento e recompensas por compacidade. O agente aprende estratégias de empacotamento através da interação com o ambiente e é avaliado face a algoritmos heurísticos tradicionais.

Os principais contributos deste trabalho incluem o desenvolvimento de um ambiente baseado em aprendizagem por reforço, funções de recompensa cuidadosamente desenhadas para promover comportamentos de empacotamento eficientes e a integração da previsão de produtos com a tomada de decisão espacial. Em conjunto, estes elementos formam um pipeline completo que transforma dados históricos do armazém em decisões inteligentes e automatizadas para o planeamento diário do espaço de armazenamento.

Palavras-chave: Previsão de Séries Temporais, Empacotamento em 3D, Aprendizagem por Reforço, Optimização de Armazéns, Alocação de Inventário, Aprendizagem Automática

Abstract

This thesis proposes an integrated artificial intelligence framework to optimize warehouse operations through two complementary tasks: forecasting daily product outflows and solving the 3D Bin Packing Problem for space-efficient storage allocation. In the first stage, machine learning techniques are applied to historical warehouse movement data to predict the total quantity of products expected to be transferred each day.

In the second stage, the predicted volumes are packed into constrained three-dimensional storage bins using reinforcement learning-based methods. A custom OpenAI Gym environment is developed to simulate realistic packing conditions, including box rotation, collision detection, stacking constraints, and compactness rewards. The agent learns packing strategies through interaction with the environment and is evaluated against traditional heuristic baselines.

The main contributions of this work include the development of a reinforcement learning–based environment, carefully designed reward functions that encourage efficient packing behavior, and the integration of product forecasting with spatial decision-making. Together, these elements form a complete pipeline that turns historical warehouse data into smart, automated decisions for daily storage planning.

Keywords: Time Series Forecasting, 3D Bin Packing, Reinforcement Learning, Warehouse Optimization, Inventory Allocation, Machine Learning

Acknowledgements

I want to express my deep gratitude to my parents for their unconditional support and for the education and principles they always instilled in me, fundamental to the person and professional I am today.

To my girlfriend, for her constant emotional support and for always being by my side, even during the most challenging moments of this journey.

To my advisor, Professor Luiz Faria, for his guidance, availability, and valuable advice that greatly contributed to the development and writing of this thesis.

Finally, to the Instituto Superior de Engenharia do Porto, for the enriching academic journey and the opportunities for personal and professional growth it provided me.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos e Símbolos

Lista de Acrónimos

RFId	Radio Frequency Identification
AGVs	Automated Guided Vehicles
IoT	Internef of Things
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
ERP	Enterprise Resource Planning
PSO	Particle Swarm Optimization
RL	Reinforcement Learning
ARIMA	AutoRegressive Integrated Moving Average
XGBoost	Extreme Gradient Boosting
ML	Machine Learning
DSR	Design Science Research
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
SOM	Self-Organizing Maps
AHC	Agglomerative Hierarchical Clustering
AI	Artificial Intelligence
SPOT	Sales Production based On Time-Series
SKU	Stock Keeping Unit
GA	Genetic Algorithm
GDPR	General Data Protection Regulation

CP	Constraint Programming
MSR	Maximal Space Representation
MDP	Markov Decision Process
PPO	Proximal Policy Optimization
3D-BPP	3D Bin Packing Problem
CNN	Convolutional Neural Network
DRL	Deep Reinforcement Learning
DQN	Deep Q-Network
SAC	Soft Actor-Critic
A2C	Advantage Actor-Critic
GAE	Generalized Advantage Estimation
EWMA	Exponential weighted moving average
BLB	Bottom-Left-Back
ReLU	Rectified Linear Unit

1 Introduction

This chapter presents the context of this study, justifying the relevance of the proposed Warehouse Management System. It will present the problem definition, the objectives, the research focus and the methodology that will be applied.

1.1 Context

Nowadays, global trade and the growth of e-commerce have been placing increasing pressure on logistics efficiency. The delivery timeline, from the warehouse to the client, has become increasingly critical. According to a survey conducted by E-Commerce company Radial, over all respondents with age 18 to 24, one-third considers timeliness of delivery the dominating factor when do purchase online. Stock-outs and longer shipping time often cause an user to turn to the other competitors for better experience. (Li et al., 2019).

The rapid expansion of e-commerce has brought about an era in which the effective management of warehouses holds a central position in ensuring punctual deliveries. In this dynamic environment, optimizing warehousing operations is not just important; it's crucial to meet the growing demand for swift and precise order fulfillment. Warehousing, within the larger context of supply chain management, carries critical responsibilities that encompass the storage of diverse e-commerce products. To maintain a competitive advantage in a swiftly evolving industry and outperform competitors, warehouses must execute their operations with precision. These tasks range from receiving, put-away, cross-docking, order picking, to shipping. The execution of these operations must prioritize efficiency to guarantee the smooth flow of the supply chain while simultaneously reducing costs (Kalkha et al. 2024).

Also, as the number of distinct products grows inside, warehouse operations face challenges as they have to deal with different diversity of sizes, weights and configurations of products. This scenario requires more advanced models to allocate the forecasted quantity of the target product efficiently.

1.2 Problem Definition

Efficient goods management in warehouses is a crucial challenge for optimizing logistics processes and ensuring the adequate availability of products for shipment. In the context of picking depots, where products are prepared for dispatch, it is essential to ensure that the amount of goods placed is sufficient to meet demand, without overcrowding the available space. Accurate forecasting of the required quantity of each product for this depot, based on historical data of transfers, incoming and outgoing goods, is fundamental to improving operational efficiency and reducing costs. These operations can take up to 15% of all operational warehouse costs (Karasek, 2013).

The problem addressed in this document involves developing an algorithm to forecast the ideal quantity of each product to be placed in the picking depot over a specific period. This forecast should consider the amount already placed, the capacity of the depot, and historical records of outgoing goods from previous years.

Furthermore, once the ideal quantity of the product is determined, the algorithm should allocate that quantity to the depot positions. The allocation should take into account the dimensions of the product and the storage locations, creating optimized layouts to maximize the use of available space and facilitate the picking process.

1.3 Objectives

The proposed solution aims to optimize warehouse operations through two main objectives: forecasting product demand and efficiently allocating storage space using intelligent packing strategies. First, it seeks to predict the desired quantity of each product to be stored in the picking depot by analyzing historical data of outgoing transfers. This involves identifying temporal patterns such as sales trends, seasonality, and demand cycles using various time series forecasting techniques.

Once the future quantities are estimated, the second objective is to determine the most space-efficient way to allocate the target products within the available positions of the picking depot. To accomplish this, the solution models the allocation task as a 3D Bin Packing Problem (3D-BPP), where products and storage locations are treated as 3D entities with constraints such as size, orientation, and placement feasibility. Reinforcement learning (RL) and other machine learning techniques will be explored to learn optimal or near-optimal placement strategies over time.

Ultimately, the goal is to develop a scalable and adaptable system that reduces operational costs, improves picking speed, and increases resource utilization by combining accurate forecasting with intelligent spatial planning.

1.4 Contributions

This study aims to develop a Warehouse Management System applying and combining techniques to dettect seasonality, trending, cycles patterns on the outgoing transfers inside the warehouse.

The Warehouse Management System developed will be able to give diverse, useful, and insightful predictions, providing the user with actionable insights to enhance operational efficiency. These predictions will include demand forecasting, followed by allocation suggestions. As it is possible to leverage real-time data and using advanced algorithms, the system will improve the accuracy leading to improved accuracy regardless the warehouse dynamics or the time of the year.

This study focus mainly on the field of warehouse management. It will enhance decision-making for the user by integrating forecasting analysis on warehouse operations such as inventory management and space utilization leading to lower operational costs (Karasek, 2013). On the long run, consequently, it will improve the efficiency of processes like order picking and fastest delivery to the customer.

When incorporating ML models, the system ensures greater accuracy and reliability. With that comes its scalability and adaptability, making it suitable for businesses of all kinds and sizes. It's also very important to mention that the system promotes sustainability by reducing waste and energy usage through optimized storage and inventory practices.

Finally, the study contributes to the knowledge in the field by presenting a robust framework for integrating predictive models into Warehouse Management Systems, which can serve as a foundation for future research and practical applications.

1.5 Problem Research Methodology

This study will follow the Design Science Research (DSR) methodology. The DSR methodology is a research paradigm that focuses on the development and evaluation of artifacts to address real-world problems (Storey et al., n.d.). DSR is a highly suitable methodology for research in WMS due to its focus on creating innovative artifacts and methodologies to address practical problems (Au, 2001). In the dynamic environment of WMS, where businesses are constantly seeking new ways to enhance efficiency, effectiveness, and strategic positioning, the need for innovative solutions is prominent.

The DRS methodology defines six steps to be followed (Peffers et al., 2007):

Identification of Problem and Motivation: This involves identifying and defining a specific problem that this study seeks to address. For this study, the problem is the inefficiency and unpredictability of warehouse operations, which often result in increased operational costs, stockouts, overstocking, and suboptimal use of storage space.

Traditional Warehouse Management Systems are often limited in their capacity to provide predictive insights, relying on static rules and historical data. This lack of predictive capability hampers decision-making, particularly in areas like demand forecasting, inventory replenishment, and resource allocation.

The motivation for this study lies in addressing these challenges by leveraging advanced predictive analytics and machine learning techniques to enhance warehouse efficiency. By providing actionable insights, the proposed predictive WMS can help businesses reduce costs, improve operational performance, and achieve greater adaptability to market fluctuations. Moreover, the system's focus on sustainability contributes to reducing environmental impacts, aligning with global goals for sustainable business practices.

This problem is significant because inefficiencies in warehouse operations have a cascading effect on the entire supply chain, affecting customer satisfaction, profitability, and competitiveness. Therefore, developing an innovative solution that integrates predictive capabilities into WMS not only addresses a critical operational challenge but also contributes to advancing knowledge and practices in the field of warehouse management.

Defining Solution Objectives: Once the problem is identified, the next step is to define the objectives for a solution. For this study, the objective was to create a warehouse management system that excels in predictive capabilities, enabling businesses to address inefficiencies and improve decision-making.

Design and Development: This phase involves the conceptualization and creation of the solution. In this case, the WMS was designed and developed to address the inefficiencies and challenges identified in warehouse operations.

Demonstration: After developing the artifact, it is essential to demonstrate its functionality and effectiveness in solving the identified problem. The system was tested using a real-world scenario dataset to validate its performance.

Evaluation: This step focuses on assessing the artifact's performance against the defined objectives. For the developed system, the proposed metrics were used to prove its effectiveness and accuracy.

Communication: The final step involves disseminating the findings and contributions of the study. This research was presented as a master's thesis in Artificial Intelligence (AI), contributing to the fields of logistics, e-commerce and WMS with new insights and evidence.

Figure 1 displays a diagram of the methodology applied.

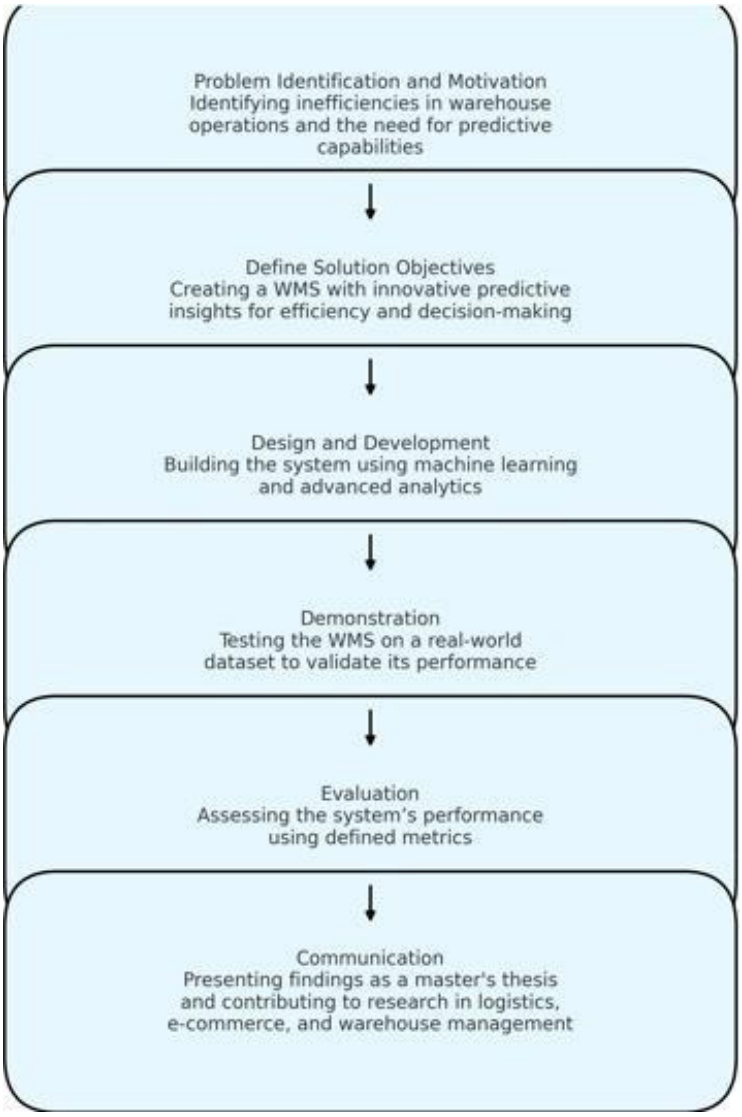

Figure 1: Methodology Diagram

1.6 Document Structure

This study is divided in 6 chapters with the following content:

The first chapter provides the context of this study and justifies the need for the proposed warehouse management system. It also establishes the contributions and problem research methodology used.
The second chapter presents the papers research methodology. It details various aspects with the purpose of answering the main research question of this study, followed by the state of art regarding warehouse management systems.

The third chapter describes the methods and tools, data extraction/experimentation used for this paper research. At the end it gives an explanation on data privacy, security and ethics concerns, how to implement a trustworthy ML system and how it would affect it, and contextualizing this to WMS environment.

The fourth chapter presents the implementation, evaluation and discussion of the the proposed models: firstly, the forecasting models to predict daily product outflows. Then, on the fifth chapter, the RL-based models to allocate these products efficiently using 3D-BPP techniques. These two chapters discuss the experimental setup, model performance, and insights gained from comparing the results with traditional baselines.

Finally, the last chapter presents the final conclusions that were drawn from the development and analysis of the results of the experiments carried out, it includes the limitations this study encountered, and future work ideas.

2 State of Art

2.1 Papers Research Methodology

This research methodology aims to provides a comprehensive analysis of existing literature relevant to this study and to answer a research question. The methodology follows established guidelines, including clearly defined search criteria, inclusion and exclusion criteria, and evaluation of the quality of selected studies, ensuring reliable basis for the concluded results.

2.1.1 Methodology

A research methodology is a mean of evaluating and interpreting all available research to a particular research question, topic area or phenomenon of interest (Kitchenham, 2004). This methodology will follow the PRISMA methodology with adaptations to gain understanding of the current state and provide insights on how to develop a novel recommendation system (Page et al., 2021). Based on its guidelines, this study will take the following steps: Formulate the research questions that best fit to answer the central question of this study. Conduct a search for relevant studies, specifying the data sources and search terms employed. Screen and select based on the defined inclusion and exclusion criteria. Evaluate the quality of studies included using the established quality assessment criteria. Perform data extraction and analysis. Compile the evidence, interpret, and present the findings, answering the proposed research questions.

2.1.2 Research Questions

The main objectives of this research is to develop a WMS for e-commerce that aims, primarly, to forecast the quantity of the product to be allocated and then to allocate that quantity efficiently. Therefore it focuses on the metrics MAE, RMSE, warehouse space, stockout rate and operational costs. Based on this goal, the following research questions were formulated:

RQ1: How can Artificial Intelligence/Machine Learning technologies enhance the efficiency and accuracy of warehouse operations, specifically quantity forecast, optimization compared to conventional methods?

RQ2: How can Warehouse Management Systems leverage technological advancements to optimize storage utilization, reduce operational errors, and enhance efficiency across warehouses of varying sizes?

RQ3: How can Artificial Intelligence and Machine Learning models overcome typical challenges in warehouse operations, such as inventory mismanagement and space optimization?

The answers for these questions must address one or more of the objectives below:

O1: Evaluate and compare the impact of Artificial Intelligence/Machine Learning technologies versus traditional methods on the efficiency of warehouse operations.

O2: Explore Artificial Intelligence/Machine Learning capabilities for enhancing decision-making accuracy in warehouse quantity forecast.

O3: Investigate the use of AI-driven/ML-driven optimization algorithms for efficient space allocation and resource management.
O4: Examine how advanced AI-driven/ML-driven WMS can reduce operational errors, reduce operational costs and improve efficiency

Table 1 shows the relationship between each research question and the objectives formulated.

Table 1 - Association between the research question and the proposed objectives

Research Questions	Objectives
RQ1	O1, O2
RQ2	O2, O3, O4
RQ3	O1, O3, O4

2.1.3 Data Sources

Three electronic databases were chosen for the research of the relevant sources: The IEEE Xplore Digital Library, ScienceDirect, and ACM Digital. IEEE Xplore, managed by the Institute of Electrical and Electronics Engineers, is renowned for its extensive resources in electrical engineering, computer science, and related fields. ScienceDirect, an Elsevier database, provides a broad spectrum of scientific and technical research in the physical sciences and life sciences among others, it has a comprehensive collection and user-friendly interface. The ACM Digital Library, from the Association for Computing Machinery, stands out for its focus on computing and information technology, hosting extensive amount of literature including pioneering work

in computer science. Collectively, these databases offer high-quality research materials, satisfying the standards to conduct this study.

2.1.4 Search Terms

The choice of the search query to be used in the database involved selecting the keywords related to the search domain. In this case, the focus was to deeper the knowledge about foreseeing demand patterns across different products and consecutively allocating them properly. To do so, the following keywords were selected:

- **Timeseries OR Time-Series OR Time Series**: these expressions are interchangeable, and both are present in the literature.

- **Sales Forecasting OR Sales Prediction OR Demand Forecasting OR Demand Prediction**: the objective of this document.
- **Warehouse Optimization OR Warehouse Enhancement OR Storage Optimization OR Storage Enhacement**: to focus the search on warehouse context.

E-commerce OR Eletronic Commerce: to filter documents more related to web businesses.

Table 2 - Number of results found in the databases for each query

Query	IEEE	ACM	ScienceDirect
(“TimeSeries” OR “Time-Series” OR “Time Series”) AND (“Sales Forecasting” OR “Sales Prediction” OR “Demand Forecasting” OR “Demand Prediction”) AND (“Warehouse Optimization” OR “Warehouse Enhancement” OR “Storage Optimization” OR “Storage Enhancement”) AND (“E-commerce” OR “Eletronic Commerce”)	48	102	61

2.1.5 Inclusion and Exclusion Criteria

The inclusion criteria to select a source to be part of the result was:

- The source objective was to predict product demand.

- The source describes the methodology used for the proposed solution.

- The source proposed solution must implement Machine Learning (ML) in some way.

The following criteria excluded sources from the selection process:

- The source does not describe the methodolgy used nor implements ML.

- The source objective does not predict product demand.

- The source was not published in the last 6 years.
- The focus of the source is not warehouse or e-commerce.

2.1.6 Quality Assessment

After the paper passed the inclusion and exclusion criteria, its quality was assessed regarding these aspects:

- The source problem context must be related to warehouse and e-commerce.

- The clarity of the results. Each paper was evaluated whether it presents clear metrics, solid discoveries, and definitive conclusions.
- The number of citations the paper has.

2.1.7 Data Extraction and Synthesis

In total, 211 sources were found with the proposed research queries in the different databases. In the abstract screening process, the sources were assessed whether it met the inclusion or exclusion criteria. 174 documents were discarded, 30 were classified as potentially relevant and 7 as relevant. The potentially relevant sources went through an additional process of results and conclusions sections reading, resulting in 13 discarded and 17 classified as relevant. The resulting 24 documents classified as relevant went through the quality assessment process, resulting in 17 documents selected for the results of this review. Figure 2 displays a flow diagram of the selection process.

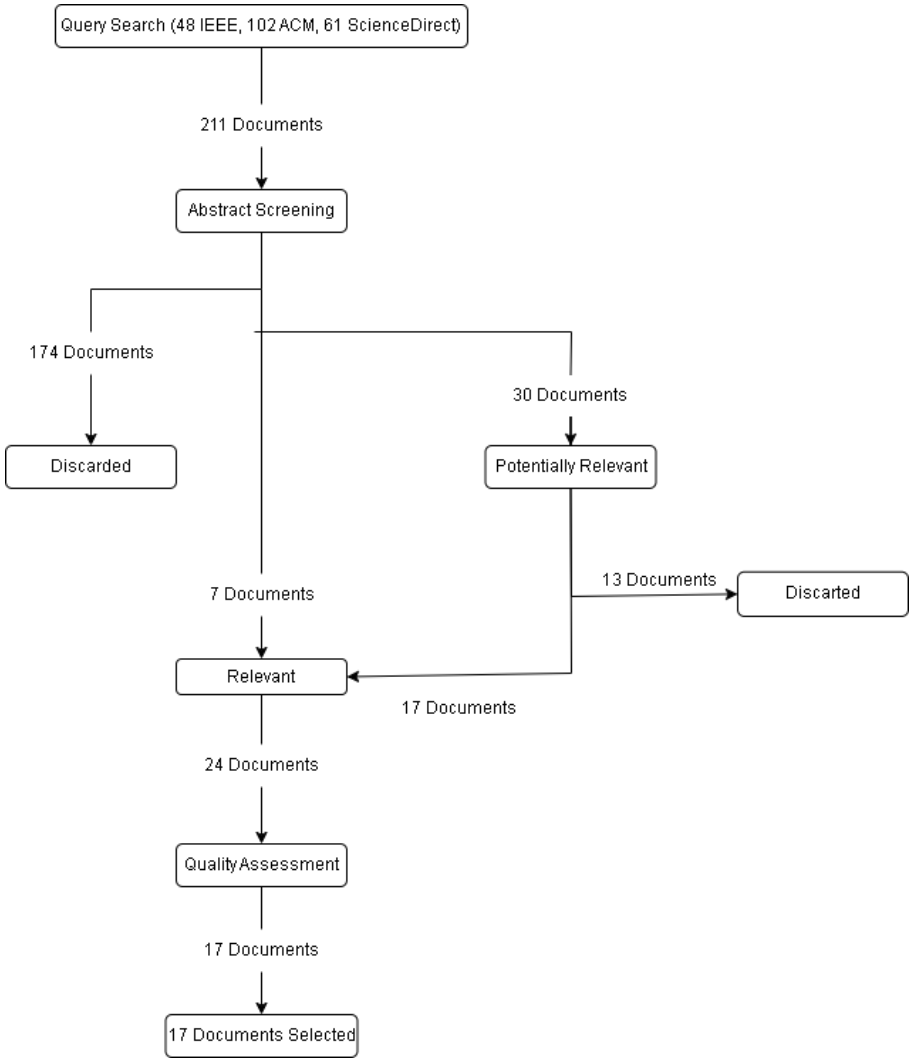

Figure 2 - Flow diagram of the paper selection process for the systematic review

Of the 17 selected documents, 8 were used to analyse the forecasting problem and 9 for the allocation problem.

2.1.8 Research Questions Answers

After completing the data extraction and synthesis, the selected papers were used to answer the three research questions. Their content was thoroughly read and analyzed, resulting in the following conclusions.

RQ1: How can AI/ML technologies enhance the efficiency and accuracy of warehouse operations optimization compared to conventional methods?

Traditional methods of optimizing logistics networks mainly rely on mathematical modeling and optimization algorithms such as linear programming, integer programming, and dynamic programming. However, traditional optimization methods often require a significant amount of

manual intervention and computation, making it difficult to cope with the challenges posed by the continuous expansion and increasing complexity of logistics networks. This calls for the exploration of new methods and technologies to enhance the efficiency and accuracy of logistics network optimization. The development of Artificial Intelligence technology has brought new opportunities for the optimization of logistics networks. AI technology can automatically analyze and process large amounts of data, uncovering hidden patterns and providing more accurate and efficient solutions for logistics network optimization. For example, machine learning technology can use historical data to establish route planning models and predict future traffic conditions and cargo demand, thereby guiding the selection of transportation routes more accurately. Deep learning technology can utilize a large amount of data in logistics networks to improve the efficiency and quality of cargo distribution. Intelligent optimization algorithms such as genetic algorithms can optimize warehouse storage and management, maximizing space utilization and reducing cargo loss and damage (Pan et al., 2024).

Based on this, countless scholars have combined big data technology to forecast the demand for e-commerce data and make a reasonable inventory strategy for enterprises, making it possible to maximize their benefits and provide consumers with good service. In this paper, we analyze the demand of an e-commerce smartphone, statistically analyze the characteristics and laws of demand data, and construct ARIMA model and XGBoost model with detailed demand time series data as examples in order to improve the accuracy of demand prediction. After the comparative analysis of model experimental results and errors, the study confirms that the XGBoost algorithm in machine learning can play its advantage of capturing nonlinear laws and improve the prediction effect (Tang, 2023).

RQ 2: How can WMS leverage technological advancements to optimize storage utilization, reduce operational errors, and enhance efficiency?

Building a successful warehouse management system encompasses solving many problems of different nature to reshape the general workflow and ensure improvements in terms of resource management. In order for such a system to be accepted and used by a logistics company, those solutions need to be presented through a simple, adaptable and most importantly, a feasible software solution. One of the aspects that needs to be covered while building a warehouse management system is the optimal product placement in the warehouse. If the products are strategically placed, all the other improvement strategies like stock to picking zone item transfer and item picking order become more efficient and easier to implement (Zunic et al., 2018).

A Warehouse management system is a software that helps to administer, handle and control the everyday operations of a warehouse. In basic words, the tenn warehousing simply means working out where and how to store the goods. Traditionally, the warehouses were labor intensive which demanded an enormous amount of workforce leading to an increased error risk. In such cases, the company many times had to face plenty of problems such as slow down or loss of orders, waste of time to find products, warehouses running out of space, boxes piling up into the aisles thereby blocking the way and wasting valuable floor space and many more.

However, the multi-fold progress in terms of technology has resulted in the warehouse operations being system-driven. This progress along with a more proactive rather than reactive approach has stressed upon better planning and operations management. It is always better to think about warehousing practices irrespective of the warehouse-size being large or small. Efficient warehouse management systems should always help us to maximize our available storage area and minimize the time and effort used to find the items. Such things reward improved efficiency and customer experience, speeding up operations, reduced cost of operations, and storage handling (Bhatia et al., 2021).

RQ3: How can AI and machine learning models overcome typical challenges in warehouse operations, such as inventory mismanagement, space optimization, and order fulfillment accuracy?

Storage is a crucial warehouse capacity. Three key decisions shape the storage capacity: how much stock should be kept in the warehouse for each Stock Keeping Unit (SKU), how frequently and when, and where the SKU should be stored in the warehouse and distributed across the various stockpile areas (Babar et al., 2022).

A general design layout must meet the requirements like maximizing available space, minimizing the time that items are handled, facilitating simple access to the stocked item, having the maximum rotation ratio feasible, allowing for the most excellent possible flexibility in product placement, and controlling the storage volumes (Babar et al., 2022).

There are numerous problems in warehouse management that need a robust solution to improve the process. For example, lack of space, time, resources, low traceability, low connectivity, excess procedures, excess stock picking, inaccurate inventory or calculation of goods kept in a warehouse, unnecessary delays in the process, damaged products, and expiry dates crossing and being unprepared for demands, etc. Nevertheless, they can be solved after adding up a few algorithms to the process. If utilized efficiently, algorithms can provide the allocations for places (Babar et al., 2022).

AI has become a powerful and influential force, bringing about significant changes in sectors through its capacity to analyze extensive data, recognize patterns, and make accurate predictions (Davenport & Kirby, 2019). AI provides a robust solution in the field of supply chain management, improving the accuracy of demand forecasts. This allows organizations to adapt more effectively to the complexities of contemporary marketplaces, with more flexibility and durability (Deloitte, 2018). AI-driven demand forecasting solutions have the capability to utilize many internal and external data sources, such as past sales data, market trends, social media sentiment, weather patterns, and even rival actions (KPMG, 2020). Artificial Intelligence systems are able to identify complex patterns and relationships through the examination of numerous data streams that conventional forecasting methods might have missed (Wamba et al., 2020). The thorough examination results in more precise and detailed predictions of demand, allowing firms to manage their inventory levels, manufacturing schedules, and marketing efforts with higher accuracy (Wamba et al., 2020).

2.2 Warehouse Management Systems

The main goal of Warehouse Management Systems is optimizing supply chain operations in different industries such as e-commerce, retail, manufacturing, and logistics. These systems help them optimize inventory management, improve order processing and ensure on-time delivery. WMS drive operational efficiencies by automating and managing warehouse operations, reducing human error, improving customer service, resulting in cost reductions and improved business performance.

WMS contains many operations being tracking inventory, managing order picking and packing processes and optimizing storage space utilization the main ones. These systems use different methods such as barcode scanning, Radio Frequency Identification (RFId), and automated guided vehicles (AGVs), to support these functions.

As we saw on the lockdown, WMS may face many challenges in adapting to dynamic market fluctuations. So, in order to address this problem, there is necessity of advanced algorithms and forecasting techniques for efficient management.

The evolution of WMS has been marked by the integration of cutting-edge technologies such as Internet of Things (IoT), AI, and ML, which have significantly improved real-time tracking, predictive analytics, and overall system performance.

2.2.1 WMS Approach

At this time, warehouse management mainly depends on the identification of consumption patterns and sales behavior. This paper conducts in-depth studies of particular product categories in terms of consumer trends, seasonality, and cycles. Such analysis is crucial for determining market trends and forecasting future demand with greater accuracy.

This allows to group products according their patterns and adjust forecasting systems to the warehouse requirements. For instance, during certain times when a product is in season, it can be stored in places that are easy to reach, but a product that is trending may have seasonal demand, and the quantity stored needs to be adjusted frequently to account for even unexpected spikes. These critical inputs are essential for feeding the forecasting and optimization algorithms, which must be capable of dealing with the breadth and complexity of product portfolios currently held within our warehouses.

2.2.2 Related Works

This section presents several related works that address forecasting and allocation problems in warehouse management. The following subsections summarize different approaches from the literature, highlighting their objectives, methods, and contributions.

2.2.2.1 Warehouse Product Forecast based on Promotions and Competitiveness

As the name says, the aim of this study is to forecast a target product during different phases over the product itself, as well as as phases of competitive products. With user features, promotional and historical sales features being fed to a Recurrent Neural Network (RNN) the study shows how the forecast value for a target product changes when there is no promotion campaing ongoing, versus when the target product is on sale, versus when one or more competitive products are on sale (Li et al., 2023).

2.2.2.2 Warehouse Product Overstocking

This study addresses the product demand in a different way. The forecast made is more specifically to prevent overstock inside the warehouse rather than the quantity estimated for outgoing movements, as in this study case.

One consequence of overstocking in some supermarkets is that the latter is obliged to make a special promotion called 'anti-waste' on products to make them more attractive. In anti-waste promotions, the company agrees to sell excess products at a loss just before their expiration date instead of throwing them away (Agbedamon et al., 2023).

Anti-waste is very different from traditional promotion which is more about attracting new customers (Agbemadon et al., 2023).
With a dataset containing over 5 million of entries of low-life expectancy products (fish and dairies) they tested many different approaches such as XGBoost, Long Short-Term Memory (LSTM) Neural Networks and different attention mechanisms (Transformer, Informer and Autoformer).

2.2.2.3 Product Allocation in Warehouses based on Demand

Differently from one of this study proposed objective, that the allocation is made considering the product and position dimensions, other studies address the allocation problem considering different features. A study introduce a novel Intelligent Storage Location Assignment (ISLA) method that utilizes advanced time series clustering algorithms specifically, Self-Organizing maps, dynamic time warping-Based k-means, and Agglomerative Hierarchical Clustering (AHC), to optimize order fulfillment and enhance warehouse efficiency. By clustering and positioning items with similar demand patterns, the approach minimizes order preparation time, reduces unnecessary warehouse movements, and improves operational flows (Kalkha et al., 2024).
Another study used a different approach to address this problem selecting the desensitized data of smartphone demand of an e-commerce company for data prediction analysis, and ARIMA and XGBoost algorithm is selected to build a demand model (Tang, 2013).

2.2.2.4 Item Arrangement in Pallets

Very similarly to the allocation objective proposed on this study, other papers address the problem of allocating several items inside a pallet considering the items dimensions and the pallet dimensions.

Storage equipment is used to support the finished product for a certain period. There are pallet storage systems that are often used in warehousing, including Block Stacking, Stacking Frames, Single-deep selective pallet rack, Doubledeep rack, Drive-in rack, Drive-thru rack, Pallet flow rack (gravity rack) and Push-back rack (Tamias et al., 2021).

Using Particle Swarm Optimization (PSO) and several parameters such as box size, pallet size, shelf capacity, type of goods, expiration date and box strength, one study performs space optimizations creating many different layouts for the target pallet and selecting the one that allocates the maximum items.

2.2.2.5 Sales Forecast based on Location of Store

Along with the historical sales data of products, some studies also focus on the location of the store where the sales meet. Sales Production based On Time-Series (SPOT) is an application to produce sales forecasts for customized input parameters – such as an individual store, state in which store is located, a particular product, category/sub-category of products, which will prove to be immensely useful for decision-making processes across a variety of concerned business units. SPOT delivers the results in the form of graphical summarization of selected data, along with its time-series forecasts (Zama et al., 2022).

2.3 Time Series Models

On the section below, the paper selects the used models retrieved from the articles found from the search query. It separates conventional approaches from ML models and refers the study from wich the model was used for development.

2.3.1 Conventional Approaches

Given the robustness and wide use over the years, these conventional approaches still serve many people to forecast sales. These approaches mainly adopt techniques of time series analysis using historical sales, however there is a downside of only considering historical sales. By considering only the historical sales records as input, these techniques onlyworkwell when the sales trend is stable. That is, these statistical models can make accurate predictions when the sales patterns exhibit strongly linear correlations (Li et al., 2019).

2.3.1.1 AutoRegressive Integrated Moving Average (ARIMA)

A statistical method that combines autoregressive components, moving averages, and differencing to handle stationary and non-stationary data. It is particularly effective for forecasting time series with linear patterns.

Autoregression means it predicts future values based on p number of past values often referred to as lags. Moving Average is very similar to Autoregression with the notable difference that instead of past values/lags, we depend on the associated error terms for these past values. We base our prediction on q number of error terms (Singh et al., 2020).

However, an autoregressive process is not sufficient to deal with non-linear sequences (Agbemadon et al., 2023).

2.3.1.2 Exponential Smoothing Models

Exponential Smoothing develops on the basis of the Moving Average Method, overcoming the shortcomings that the Moving Average can only calculate the smoothing value according to the historical values of the previous several items. Exponential Smoothing Method weights and averages all historical values of the past, puts more weights on the new prediction value, so as to eliminate the influence of random factors, identify the basic trend of economic phenomena and predict the future. The basic idea of Exponential Smoothing is to process the number of time series into a “smoothing value” and then construct a prediction model with a smoothing value, so as to calculate the predicted value (Lian et al., 2018).

2.3.1.3 Prophet

Meta also developed an open source for analysing and forecasting time series called Prophet. It mainly works with four features being trends $g(t)$, seasonality $s(t)$, holidays $h(t)$ and error $(-t)$.

$$y(t) = g(t) + s(t) + h(t) + (-t)$$

Hence, $g(t)$ reflects non-periodic variations in the time series' value. $s(t)$ reflects periodic changes (such as weekly, yearly and seasonal changes), $h(t)$ indicates the effects of holidays that occur on possibly irregular schedules that span one or more days and $(-t)$ displays information that is not described in the model (Alsaidi et al., 2023).

2.3.2 Machine Learning Models

Unlike conventional models, ML models can incorporate much more features such as economic indicators, marketing campaings and even social media trends rather than historical sales and linear trends to forecast. This leads to a better forecast accuracy as well as automation and scalability as these advanced algorithms automatically catch patterns and scales them to handle large datasets.

2.3.2.1 Linear Regression

Linear regression is a simple supervised learning algorithm used for predicting continuous outcomes. It models the relationship between one or more independent variables (features) and a dependent variable (target) by fitting a linear equation to the observed data. It's easy to implement and interpret, making it a popular choice for straightforward prediction tasks where the relationship is approximately linear.

2.3.2.2 Self-Organizing Maps

Self-Organizing Maps (SOM) preserves the temporal relationships between different orders, ensuring that orders with similar temporal patterns are grouped together on the SOM map. Simultaneously, the quantization error measures how accurately the SOM represents clusters

of orders with similar time series patterns. The goal is to uncover and visualize meaningful patterns in the order time series data, emphasizing the preservation of temporal relationships and the accurate representation of order clusters (Kalkha et al., 2024).

2.3.2.3 Agglomerative Hierarchical Clustering

Agglomerative Hierarchical Clustering (AHC) is a clustering algorithm used for unsupervised tasks on ML to cluster data points based on similarities. It begins by treating each entry as a separate cluster and slowly approximates each entry until all points belong to only one single cluster or a predefined number of clusters is reached.

For distance measurement, Euclidean metric is well known due to its efficiency, avoiding resource-intensive alternatives like Dynamic Time Warping (Kalkha et al., 2024).

2.3.2.4 LSTM Networks

LSTM networks are a type of RNN that can learn order dependencies in sequence prediction problems. In traditional Neural Networks, each input and output is independent, but in cases of predicting the next word in a sentence, the previous words are needed and therefore the previous words must be remembered. Similarly for time series, past values can influence the current value. This inspired the application of RNN for time series and the development of RNN-based architectures for time prediction. While introduced in the late 90's, LSTM models have become in last decade a viable and powerful forecasting technique for time-series (Agbedamon et al., 2023).

2.3.2.5 Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) is a powerful and efficient ensemble learning algorithm based on decision trees. It uses a boosting technique, where multiple weak learners (decision trees) are combined iteratively to improve prediction accuracy. XGBoost is known for its high performance in regression and classification tasks, speed, and ability to handle missing data. It's widely used in data science competitions and real-world applications.

The XGBoost algorithm first forms a base learner, and by judging the learner, then forms a new learner to make up for the residuals of the previous learner, and so on repeatedly forms T learners, and the residuals become smaller and smaller in the continuous iterations, and finally forms a learner with very small error and high accuracy, and then combines the T learners (Tang, 2023).

2.3.2.6 Attention-based Mechanisms

Transformer-based architectures have shown their effectiveness in natural language processing NLP and computer vision tasks. They are now also applied in time series forecasting and also show their efficiency (Agbemadon et al., 2023).

These models assign different weights to different elements in the input sequence dinamically. This allows them to focus on specific parts of the sequence while ignoring less critical information. This ability makes them particularly suited for time series tasks where relationships between two distant time points play an important role.

2.4 Packing Models

On the section below, the author selects the used models retrieved from the articles found from the search query. It separates conventional approaches from RL models and refers the study from which the model was used for development.

2.4.1 Conventional Approaches

Conventional approaches remain a common choice for solving problems like the 3D-BPP due to their simplicity, speed, and long history of use. These methods—such as rule-based heuristics, greedy strategies, and space-partitioning techniques—follow fixed logic to make decisions, without learning from past experience. While they often produce good results in predictable environments, they can struggle when conditions become more complex or dynamic. Since these models don't adapt over time, their effectiveness may drop in situations involving uncertainty or shifting constraints. Such algorithms are useful for quickly generating feasible solutions but may fall short of finding truly optimal strategies in more demanding scenarios (Martello et al., 2000).

2.4.1.1 Heuristic Algorithms

Given their simplicity and effectiveness, conventional heuristic methods have long been used to solve the 3D-BPP. These techniques often rely on predefined rules such as placing the largest item first or filling space from the bottom up. While they offer fast and reasonably good solutions, they struggle with complex or dynamic packing scenarios. Their rule-based nature limits adaptability, especially when item sizes or constraints vary significantly between instances. As a result, these approaches perform well only when problem conditions remain relatively consistent. They often fail to find optimal or near-optimal configurations in more irregular or high-variance packing tasks (Huang et al., 2022).

2.4.1.2 Constraint Programming

Constraint Programming (CP) is another technique often used to tackle the products allocation problem, especially when the specific problem at hand involves complex rules—like limiting rotations, enforcing stacking constraints, or handling item compatibility. One of the strengths of CP is that it allows these constraints to be defined clearly and directly, making it easier to model real-world warehouse scenarios. However, as the number of products increases, the computation required can grow quickly, making it harder to scale. While CP is excellent for detailed and accurate modeling, performance can become a challenge in larger, more complex packing problems (Bortfeldt et al., 2013).

2.4.1.3 Maximal Space Representation

Maximal Space Representation (MSR) is a widely used strategy for tackling the 3D-BPP, particularly valued for its efficiency in tracking available space. Instead of checking every corner of the bin for possible placements, MSR keeps a dynamic list of the largest remaining empty regions where a box could potentially fit. This makes the algorithm more efficient and strategic in how it places each item. One of the main advantages of MSR is that it reduces the number of

unnecessary placement checks, which becomes especially useful in scenarios that demand real-time decisions. That said, as more boxes are packed, the number and complexity of the tracked spaces can grow significantly, which may introduce computational overhead. MSR-based methods manage to strike a good balance between speed and solution quality, which is why they continue to be applied in many real-world industrial settings (Li et al., 2015).

2.4.2 Reinforcement Learning Models

RL models are becoming more popular to resolve complex problems like the 3D-BPP, especially in dynamic and high-dimensional environments. Unlike conventional approaches, RL agents learn from interaction with the environment, gradually improving their allocation efficiency through trial and error (Bo et al., 2022). This approach allow models to adapt to diferent product sequences, varying sizes, packing constraints and environment configurations over time. One of the main strengths of RL is its ability to come up with strategies beyond fixed rules making it most suitable for real-time situations. However, these models require significant training time, careful reward designing and their performance is very related to the quality of the simulation and states representation.

2.4.2.1 Deep Q-Network

In the Deep-Pack framework by Kundu et al., the authors tackle the 2D Online Bin Packing Problem using a vision-based Deep Reinforcement Learning (DRL) approach. They model the task as a Markov Decision Process and apply a Double Deep Q-Network (DQN) as the learning agent. This agent receives a binary image representation of the bin and incoming item as input and outputs a pixel coordinate for placement. The use of Convolutional Neural Networks (CNNs) allows the agent to process spatial layouts directly, making the model robust to sensor noise and well-suited for real-world robotic applications (Kundu et al., 2019). A carefully crafted reward function encourages the agent to form dense clusters and compact arrangements of items, maximizing usable space for future placements. The architecture is trained using experience replay and a target network to stabilize learning, achieving higher packing efficiency than several classical heuristic algorithms.

2.4.2.2 Proximal Policy Optimization

The paper “Online 3D Bin Packing with Constrained Deep Reinforcement Learning” propose a deep RL approach to address the online 3D-BPP, where items arrive sequentially and must be packed without the possibility of rearrangement (Zhao et al., 2021). To handle this dynamic and constrained environment, they frame the problem as a Constrained Markov Decision Process (MDP) and employ Proximal Policy Optimization (PPO) as the core learning algorithm. Their agent follows an actor–critic architecture, enhanced with a novel prediction-and-projection mechanism. This involves first predicting a mask of feasible actions based on physical constraints like stability and placement boundaries, and then projecting the action distribution accordingly to ensure constraint satisfaction. The observation space is constructed using a compact height map of the bin and box features, which are processed through convolutional layers to guide the policy network. PPO’s stability and sample efficiency make it particularly

well-suited to this setting, where action feasibility is highly restricted and exploration must be tightly controlled. The authors demonstrate that their architecture not only respects hard constraints during learning but also achieves competitive results against strong heuristic baselines.

2.4.2.3 Advantage Actor-Critic

In their study, Jiang et al., present a multimodal deep RL approach to solving large-scale instances of the 3D-BPP. To train their agent, the authors adopt the Advantage Actor-Critic (A2C) algorithm, enhanced with Generalized Advantage Estimation (GAE) to better balance bias and variance during learning. The packing process is broken into three sequential sub-tasks—selecting the box order, determining its orientation, and computing its final placement—each managed by a modular encoder–decoder architecture. The agent receives input from two sources: a box state encoder, implemented with a sparse Transformer to reduce computational overhead, and a view state encoder, built with convolutional layers to capture the spatial structure of the bin from a top-down perspective (Jiang et al., 2021). To address the large action space for box positioning, the authors integrate action representation learning, allowing the model to generalize from past similar placements.

2.4.2.4 Soft Actor-Critic

Zhao et al., on August 2023 approached the 3D-BPP using a deep RL setup centered around the Soft Actor-Critic (SAC) algorithm. The agent is designed to place incoming boxes into a container by learning policies that consider both spatial feasibility and packing efficiency. SAC operates with stochastic policies, allowing the agent to explore the environment more effectively by sampling actions from a learned distribution. This is particularly useful in the highly constrained 3D-BPP space, where subtle changes in placement can lead to vastly different outcomes (Zhao et al., 2023). The system represents the packing environment as a dynamic graph, capturing spatial relationships between the bin and placed boxes, and feeds this into a graph neural network to guide the SAC policy. The authors highlight that SAC's entropy-regularized learning helps maintain exploration during training, leading to more robust and generalizable strategies. The proposed method outperformed traditional heuristics and earlier RL baselines in both packing density and adaptability to unseen box sequences.

2.5 Optimization Algorithms

Optimization Algorithms are widely used to improve forecasting accuracy and packing efficiency. They are extremely helpful to automate the search for the best hyperparameter settings, ultimately enhancing forecasting accuracy and consequent allocation (Sen et al., 2023).

2.5.1.1 Particle Swarm Optimization

The particle swarm optimization algorithm (PSO) is a stochastic based optimization technique inspired by the social behavior of a flock of birds or a group of fish (Tamias et al., 2021).

Two basic PSO topologies are used in many works of literature to describe how a particle is related to other particles, namely Ring Topology and star topology. A particle is connected to two different particles in a ring topology to have a neighboring size equal to 3. While in a star topology, a particle can be connected to all other particles, called the global neighborhood (Tamias et al., 2021).

The main steps in using the PSO algorithm on optimization problems, namely the representation of the solution and its fitness function, while the parameters needed in the PSO algorithm are as follows: Pbest (Personal Best) is the best position of a particle that shows the position of the particle that is prepared to get the best solution. Gbest (Global Best) is the best position of the particle in the swarm or the best position among the existing Pbest. Velocity (V) is the speed that drives the optimization process, which determines the direction in which the particles are needed to move and fix their original position. Inertia load (0): inertia load is symbolized by w; this parameter controls the effect of velocity given by a particle. Learning Rates (c1 and c2): a constant to assess the ability of the particle (c1) and social swarm ability (c2) which indicates the load of the particle on its memory. The values of c1 and c2 are between 0-2 (Tamias et al., 2021).

2.5.1.2 Genetic Algorithms

Genetic algorithms (GA) are a type of optimization algorithm that are inspired by the principles of natural selection and genetics (Babar et al., 2022).

The GA operates by performing a series of operations such as selection, crossover, and mutation on the input solutions to produce new and potentially better solutions. The selection operation involves choosing the best solutions from the current generation based on their fitness score, which is a measure of how well they solve the problem (Babar et al., 2022).

The crossover operation involves combining two solutions to produce a new solution, while the mutation operation involves making small random changes to a solution (Babar et al., 2022). These operations are repeated multiple times, and the best solutions from each generation are selected to form the input for the next generation. This process continues until a satisfactory solution is found or a predetermined stopping criterion is met (Babar et al., 2022).

2.6 Exploration Algorithms

This section introduces common exploration strategies used in RL to balance the trade-off between exploration of new actions and exploitation of known rewarding actions. The following subsections describe two of the most widely applied methods: epsilon-greedy and softmax.

2.6.1.1 Epsilon-Greedy Exploration Algorithm

The epsilong-greedy algorithm is a RL technique that has been used in many applications. Epsilon is the probability of the agent exploring rather than exploiting (Nguyen et al., 2021).

This simple mechanism helps prevent the agent from getting stuck in suboptimal policies early on. As training progresses, epsilon is often decayed to favor exploitation.

2.6.1.2 Softmax Exploration Algorithm

The softmax exploration algorithm is another strategy commonly used in RL to balance exploration and exploitation. Instead of selecting actions randomly with a fixed probability, as in epsilon-greedy, softmax assigns a probability to each action based on its estimated value. Actions with higher Q-values have a greater chance of being selected, but suboptimal actions may still be chosen with non-zero probability. This probabilistic mechanism allows the agent to explore while still prioritising actions that appear more promising. The "temperature" parameter controls the degree of exploration: higher temperatures encourage more random behaviour, while lower temperatures make the policy greedier. This approach has been widely studied and applied in RL literature (Sutton & Barto, 2018).

2.7 Warehouse Management System Metrics

2.7.1 Forecast Accuracy

The accuracy of predictions is one of the most important metrics, as the algorithm must accurately forecast the quantity of each product required for the picking depot.

Root Mean Squared Error (RMSE) measures the average of the squared differences between the predicted values and the actual values. The lower the RMSE, the better the model's accuracy (Wu et al., 2012)(Hernández del Olmo & Gaudioso, 2008b).

$$RMSE = \frac{\sqrt{\sum_{i=1}^{N}(P(u,i) - p(u,i))^2}}{N}$$

Mean Absolute Error (MAE) evaluates the average absolute error between the predicted and actual values, helping to understand the average error in absolute terms (Wu et al., 2012)(Hernández del Olmo & Gaudioso, 2008b).

$$MAE = \frac{\sum_{i=1}^{N} |P(u,i) - p(u,i)|}{N}$$

Where $p(u,i)$ represents the real rating of user u for item i, and $P(u,i)$ represents the predicted one. N is the number of ratings available. RMSE is biased to weight large errors more than small errors. A lower value of MAE and RMSE indicates high accuracy (Anitha et al., 2013).
Mean Absolute Percentage Error (MAPE) measures the average percentage error, making it easier to compare predictions across different products with varying volumes and prices. The calculations is exactly the same as MAE except on the end the result is multiplied by 100.

2.7.2 Warehouse Space

The efficient use of space in the picking depot is crucial for maximizing storage capacity. The space utilization metric evaluates the percentage of the total available space that is being effectively used. In this context, the metric can be used to check whether the product allocations made by the algorithm are optimizing the use of the available space in the warehouse, without wasting any area.

2.7.3 Stockout Rate

The stockout rate reflects the frequency with which a product becomes unavailable in the warehouse, which can cause delays in shipments. Accurate demand forecasting and efficient product allocation should minimize this rate, ensuring that the most requested products are always available when needed.

2.7.4 Operational Costs

Operational costs are associated with the picking and inventory management processes. That includes storage costs and product transfers. Optimized allocation and accurate quantity predictions ensures a reduction of these costs, making the system more reliable and efficient.

Improvements, starting at the picking times to a lower stockout rate can all contribute to a significant reduction in operational costs.

2.8 Warehouse Management System Challenges

This section discusses the main challenges faced when implementing forecasting and allocation techniques within a WMS. The following subsections highlight issues related to demand variability, computational complexity, data integration, and the need for real-time adaptation.

2.8.1 Demand Oscilation

Product demand can vary significantly over time. It can be due to seasonal factors or due to unexpected events like the previous lockdown, marketing campaigns, changes in habits or consumer preferences. This variability makes it difficult to accurately predict the exact quantity of products needed for the picking depot. The system needs to be flexible enough to adapt to these fluctuations. This requires more advanced forecasting techniques that are capable of handling uncertainties and sudden changes.

2.8.2 Computational Complexity

Advanced forecasting models, such as neural networks or Machine Learning techniques (like XGBoost), can be computationally intensive. Processing large volumes of data and performing complex calculations in real-time requires considerable computational resources. System scalability becomes a challenge, especially when the number of products or the frequency of forecast updates increases. Ensuring that the system is resource-efficient without compromising accuracy is a significant challenge.

2.8.3 Data Integration

The effectiveness of the algorithm depends on the availability and quality of data. In many cases, data may come from various sources, such as Warehouse Management Systems, Enterprise Resource Planning (ERP), sales history, and customer information. Integrating this data into a single platform, without errors or inconsistencies, is a significant challenge. Additionally, it is necessary to ensure that the data is updated in real-time to reflect the current warehouse situation.

2.8.4 Real-Time Adaption

Warehouse conditions can change rapidly, with new product orders, changes in sales volumes, or even variations in logistical conditions. The system needs to be able to adapt in real-time to these changes to ensure that the forecasting and product allocation are always accurate. This requires quickly adjusting the algorithms to reflect the new information, without compromising the efficiency or performance of the warehouse.

3 Methods, Tools and Thecnological Challenges

This chapter begins by presenting the datasets used in this work, namely the forecasting dataset and the allocation dataset, followed by the processes applied for data extraction. Then, the methods and tools employed in the development of the proposed solution are described, together with the rationale for their selection. A data exploration phase is included to provide a better understanding of the data characteristics and support subsequent modeling decisions. Finally, the chapter discusses technological and social challenges, including ethical issues in artificial intelligence and considerations regarding data privacy and security in the context of this dissertation.

3.1 Datasets

This work relies on two different datasets, corresponding to the two main research problems addressed in the dissertation: **time series forecasting of warehouse transfers** and **allocation of boxes in the 3D-BPP**.

3.1.1 Forecasting Dataset

The forecasting dataset was built from historical records of outgoing transfers within a warehouse. Each record corresponds to a product transfer event, including product identifiers, hierarchical category information, brand, and quantitative attributes such as units transferred.

To support temporal modeling, the data was aggregated by **date**, **brand**, and **category**, resulting in a daily-level time series of transfer quantities. In total, the dataset contained:

- **Dates covered:** January 2021 to January 2025
- **Distinct products:** 55,338
- **Distinct brands:** 499
- **Distinct categories:** 681

Because many products had a very short lifecycle or low transfer activity, the dataset was filtered to retain only the most representative ones. Specifically, the **top 10 brands** with the highest average lifecycle and mean transfer quantity were selected, and within each of these brands, the **top 10 categories** were retained according to the same criteria. This filtering ensured that the training data focused on stable and relevant product flows, reducing noise from sparse or intermittent series.

From this dataset, derived variables were engineered for model training, such as time features, lag features, differences, exponential weighted moving averages (EWMA), and holiday/weekend indicators. For the LSTM model specifically, cyclical encodings (sine/cosine) were used for time-related features to better capture periodicity.

This dataset served as the basis for training, validating, and testing the **XGBoost** and **LSTM** forecasting models.

3.1.2 Allocation Dataset

For the allocation task, a synthetic dataset was generated to simulate instances of the 3D-BPP. The dataset consists of boxes characterized by their **length, width and height**, together with the bin constraints defined by maximum dimensions.

Key characteristics of the dataset include:

- **Item features:** length, width, height
- **Bin constraints:** maximum length, width and height

One bin and 50 boxes were produced through a recursive splitting process of the bin, ensuring realistic subdivisions of the bin volume and consistent variability in box dimensions. At the end of each episode, the bin is reset and a new recursive split is performed to generate a fresh set of boxes.

The dataset was created to reflect realistic warehouse constraints, enabling the training and evaluation of RL approaches for allocation. It provides sufficient variability in item dimensions and weights to test the ability of an agent to generalize across different bin-packing instances.

This dataset served as the basis for training, validating, and testing the **DQN** and **PPO** allocation agents.

3.2 Data Extraction

This section describes the procedures used to extract and prepare the datasets required for this work. Two different datasets were considered: one for the forecasting task, obtained from real warehouse movement records, and another for the allocation task, generated synthetically to simulate realistic 3D-BPP instances.

3.2.1 Forecasting

For this work, the author focused only on outgoing movements. This choice was made to simplify the initial analysis and to concentrate on the type of movement most directly related to the problem being addressed. By narrowing the scope in this way, it became possible to better understand the data structure and ensure a solid starting point before considering broader scenarios. To identify the outgoing movements, the headers of all recorded movements were extracted, and from these, the movement type identifier was retrieved. In this process, the value '601' was used as the reference ID to filter and select the records corresponding to outgoing movements.

Table 3 - Entries of warehouse movements headers

TANUM	MOVTY	MEINS
284155	601	1/7/2021
284156	601	1/7/2021
…	…	…
14172	601	1/7/2021
14173	601	1/7/2021

Regarding these column names, they are a nomenclature inside the warehouse. Below is a table that describes each of the columns name:

Table 4 - Description of column names

Column	Description
TANUM	Order Transfer Number
MOVTY	Movement Type
MATNR	Material Number
MEINS	Unit of Measure
QDATU	Date of Transfer
VLTYP	Initial Deposit
VLPLA	Initial Position
NLTYP	Final Deposit
NLPLA	Final Position
NISTM	Quantity Transfered

Because the data application framework, **SAP GUI**, struggled to handle the retrieval of all warehouse movements in a single query, the extraction process had to be adjusted. Instead of downloading the entire dataset at once—which caused long execution times and unstable results—the movements were collected month by month. This step-by-step approach made the process more reliable and ensured that the data could be gathered without errors. Below is an example of the movements retrieved for January 2021:

Table 5 - Entries of warehouse movements for January 2021

TANUM	MATNR	MEINS	QDATU	VLTYP	VLPLA	NLTYP	NLPLA	NISTM
9531	PO10148	UN	1/7/2021	A00	0010010100	916	8010000002	1
9532	PO25396	UN	1/7/2021	A00	0010010100	916	8150002311	1
…	…	…	…	…	…	…	…	…
25000	PO29190	UN	1/30/2021	902	2000000587	T01	1040010100	1
25001	PO29190	UN	1/30/2021	902	2000000587	T01	1040010100	1

As the reader can notice, all entries only contain one quantity transfered per product. So to forecast the quantity of a product we have to concatenate all the entries with the same value of **MATNR** and **QDATU**, verify if **TANUM** field is in headers data and sum the values of **NISTM.** After sorting by **QDATU** and **MATNR** resulted the following data:

Table 6 - Entries of warehouse movements for Januray 2021 concatenated

TANUM	MATNR	MEINS	QDATU	VLTYP	VLPLA	NLTYP	NLPLA	NISTM
26233	P000012	UN	1/7/2021	902	3000000966	F00	10010100	7
18993	P000033	UN	1/7/2021	A00	0010010100	916	8010001553	3
…	…	…	…	…	…	…	…	…
851916	P032349	UN	1/30/2021	902	2000001435	O00	210020100	2
85075 2	P032350	UN	1/30/2021	902	2000001435	O00	220030105	2

After concatenating all months data together, all that is left is some information regarding the product such as it's dimensions, weight and categorical information. To do that the author had access to a csv file containing all products information and had to merge the current data with the products csv file using **MATNR** field. After changing the name of columns we can look how the final data looked like on the table below:

Table 7 - Entries of warehouse movements

DATE	PRODUCT	…	WIDTH	BRAND	PRODUCTHIERARCHY	QUANTITY	…
1/7/2021	P000012	…	29	1Life	110	7	…
1/7/2021	P000033	…	38	Logitec h G	123	3	…
…	…	…	…	…	…	…	…
10/28/2022	P030354	…	70	Logitec h G	109	67	…
10/28/2022	P031164	…	0	Bosch	120	2	…
…	…	…	…	…	…	…	…

The final features of this dataset included, **DATE**, **PRODUCT**, **QUANTITY**, **INITIALDEPOSIT**, **INITIALPOSITION**, **FINALDEPOSIT**, **FINALPOSITION**, **PRODUCTHIERARCHY1**,

PRODUCTHIEARARCHY2, **PRODUCTHIERARCHY3**, **WEIGHT**, **WIDTH**, **LENGTH**, **HEIGHT**, **UNITOFMEASURE**, **WEIGHTUNIT**, **PRODUCTNAME**, **PRODUCTHIERARCHY1NAME**, **PRODUCTHIERARCHY2NAME**, **PRODUCTHIERARCHY3NAME**, **PRICE**, **CURRENCY**, **BRAND** and **BRANDNAME**.

All fields do not need description except the hierarchy ones. The hierarchy fields contain the category of the product and each hierarchy is more specific than the previous one. So in this data, hierarchy 3 is more specific than hierarchy 2 which is more specific than hierarchy 1.

3.2.2 Allocation

For the allocation task, no historical dataset was available. Instead, a synthetic dataset was created to simulate realistic instances of the 3D-BPP. The generation process was based on a recursive splitting of the bin, where the container is iteratively divided along one of its dimensions until the desired number of boxes is obtained.

This procedure guarantees that the total box volume equals the bin volume while still introducing variability in box shapes and sizes. As a result, the generated scenarios are representative of realistic bin loading conditions, with items of different proportions that need to be efficiently packed together.

For the experiments presented in this work, each episode starts with one empty bin and 50 boxes generated through recursive splits. At the end of each episode, the bin is reset and a new split is performed, ensuring variability across episodes and providing sufficient diversity for the RL agent to generalize its allocation strategy.

3.3 Method and Tools

Several Python libraries were employed to develop and analyse the data, each serving a specific purpose such as file handling, data manipulation, preprocessing, visualisation, forecasting and allocation.

Data access and preprocessing:

- **glob** was useful to read all monthly movement files following a naming convention, avoiding the need to manually specify file paths.
- **pandas**, a powerful data manipulation library, was central to merging the movement records with product information, checking movement types (e.g., outgoing movements), handling missing values and preparing the dataset for analysis.
- **numpy** supported efficient numerical operations, such as calculating correlations between product categories.

- **scikit-learn** provided preprocessing tools (scaling, encoding and imputation) and evaluation metrics (mean squared error and mean absolute error).
- **scipy** was used for statistical analysis.
- **holidays** was integrated to generate calendar-based features related to national holidays.

Data visualisation:

- **matplotlib** offered flexibility in building a wide range of plots.
- **seaborn** enhanced the clarity and aesthetics of statistical graphics.
- **plotly** enabled interactive visualisations.
- **statsmodels** supported time series diagnostics, such as partial autocorrelation plots.

Forecasting models:

- **xgboost** was used to implement gradient boosting tree model.
- Deep learning models relied on **tensorflow/keras** for LSTM network.

3D-BPP agents:

- Custom environments were developed using **gym** to define the 3D-BPP as an RL problem.
- **torch** and **lightning.pytorch** were again employed to train DQN agent.

3.4 Data Exploration

This section presents the exploratory analysis performed on the datasets used for forecasting and allocation tasks. The objective of this step was to better understand the structure, quality, and patterns of the data before applying the proposed models. The following subsections describe the main findings for each dataset.

3.4.1 Forecasting

The exploration done with the data was manipulate and visualize it. That process was done in order to visually understand the data and plan what are the next steps based on the research done as described on chapter 2.

After the data was all merged and preprocessed, the author checked the columns data type and converted it accordingly and then printed the null values. That returned a total of **63631**, **12** and **167441** null values for **FINALDEPOSIT**, **PRODUCTNAME**, **MEASUREMENTUNIT** fields accordingly. Regarding the **FINALDEPOSIT** null values, they probably can be filled with the values of entries that share the same **FINALPOSITION**. **PRODUCTNAME** is irrelevant for the objectives of this document and only is here to describe the **PRODUCT** field.
MEASUREMENTUNIT is null on the entries where fields **WIDTH**, **HEIGHT** and **LENGTH** are equal to 0.

After that the number of distinct products, categories and brands were returned. That resulted on **44213** distinct products, **17** distinct product categories of hierarchy 1, **133** distinct product categories of hierarchy 2, **656** distinct product categories of hierarchy 3 and **472** distinct brands.

Regarding outgoing movements, the first plot to be generated was to visualize the total outgoing movements over the dataset time span.

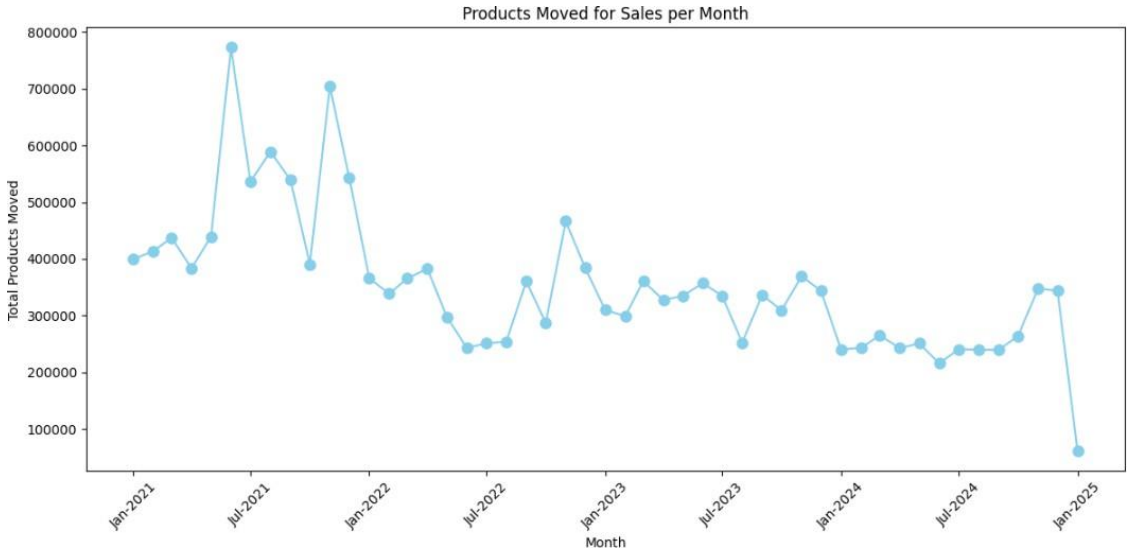

Figure 3 - Products Moved for Sales per Month over the dataset time span

Immediately after that a more close look was done in order to see the behave of outgoing movements during the months for each year, now with the mean of movements instead of total movements. In exception for year 2021 (probably due to Covid-19) and year 2025 (because the records go only to 8 January 2025) you can see kind of a pattern between April-September and September-November.

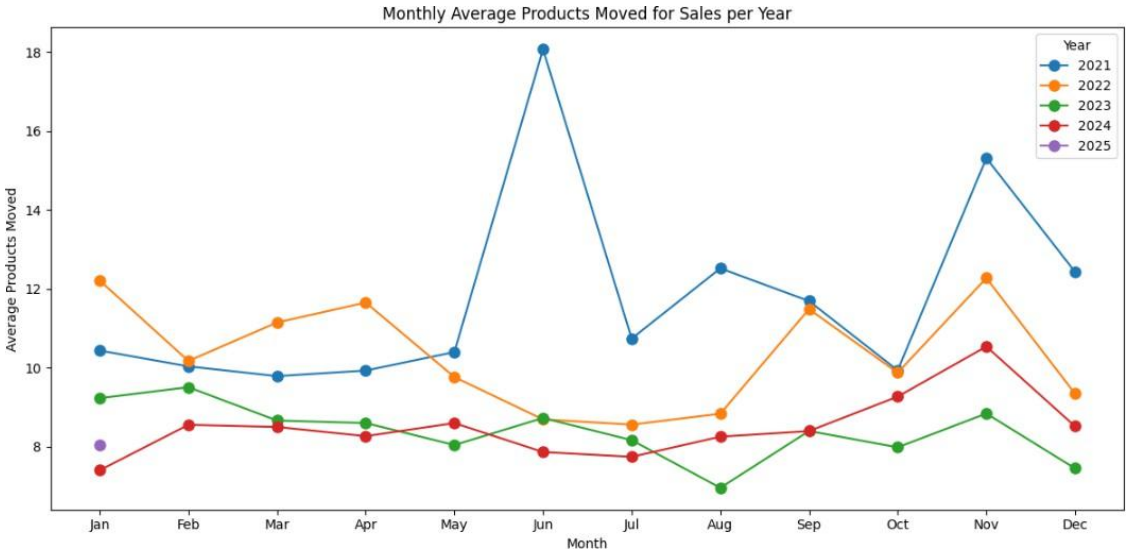

Figure 4 - Monthly Average Products Moved for Sales per Year

Another very important visualization is the movements done during the weekdays. To do so another field was added to the dataset containing the weekdays for each entry. The plot is shown below.

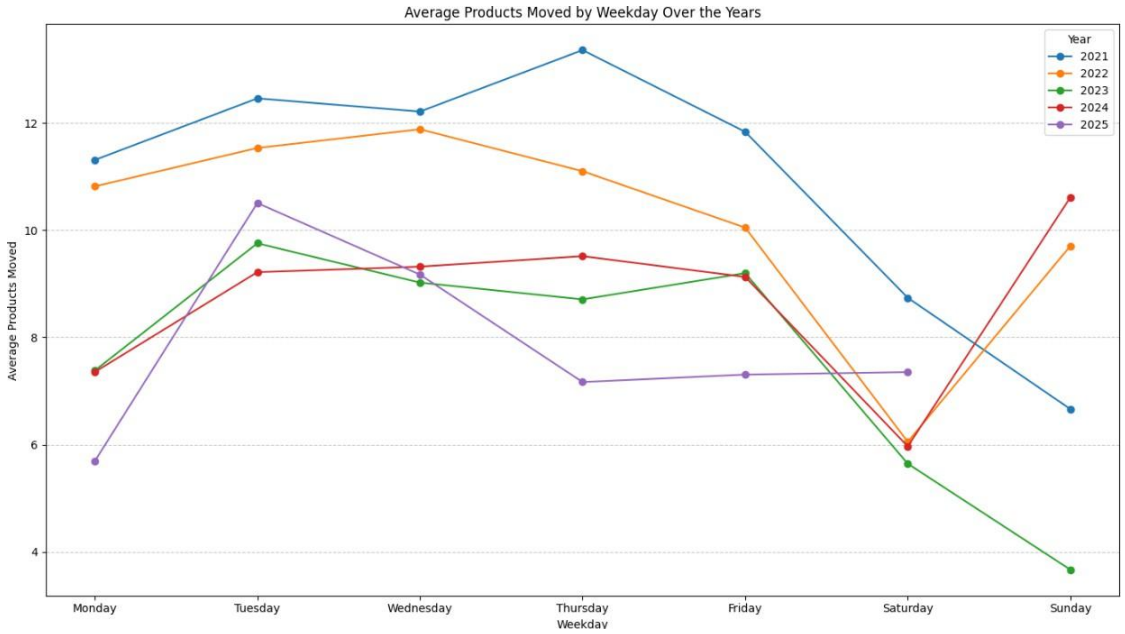

Figure 5 - Average Products Moved by Weekday per Year

Also it was generated plots to visualize the outgoing movements for each product category and brand over the years but the number of distinct values is so big that turns difficult to visually understand what stands on the plot. On the next days plots for the top categories, brands and products moved will be made in order to better visualize outgoing movements on these fields accordingly.

To conclude, it was also calculated a correlation on the different product categories over the quantity to see if different categories shared similar movements patterns. As the reader can see the only correlations between categories include Services and Services ins't a product, this category will be removed on the next days.

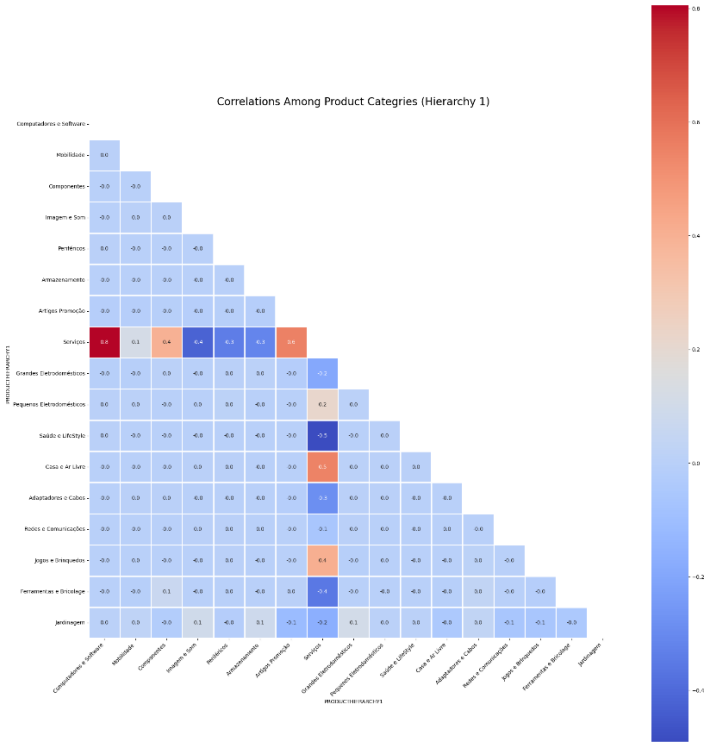

Figure 6 - Correlation between different product categories

3.4.2 Allocation

For the allocation task, the dataset does not come from historical records but was instead synthetically generated. Each episode starts with one bin and 50 boxes created through recursive splits of the container. This method ensures that the total volume of boxes is equal to the bin volume, while still producing diverse box dimensions. Before adopting recursive splitting, the environment was initially tested with fully random box generation. However, this approach caused several issues, such as boxes with extreme proportions (very thin or oversized), unbalanced scenarios that were often impossible to pack efficiently, and high variability across episodes that made it difficult to evaluate the agent's performance consistently.The exploration of this dataset was both descriptive and practical, with the following objectives:

- **Distribution of dimensions**: the width, depth and height of the boxes were examined to ensure that different size proportions were represented.

- **Volume consistency**: the sum of the volumes of all boxes was confirmed to always match the bin capacity, guaranteeing realistic load scenarios.

- **Variability across episodes**: since a new recursive split is performed at the end of each episode, different box configurations are generated, providing diversity for the training process.

A random policy agent was used to place boxes inside the bin in order to validate the environment. This made it possible to check the correct functioning of collision detection, placement rules and overall robustness of the simulation.

Overall, this exploration step confirmed that the synthetic dataset is adequate for the allocation problem, as it maintains realistic warehouse constraints while ensuring a wide range of packing scenarios for the RL environment.

3.5 Technological and Social Challenges

In this section, the ethical issues in artificial intelligence and data privacy and security are described and discussed in the context of the dissertation topics. Regarding the ethical issues the definition of ethics in AI will be given and the main requirements of European guidelines for trustworthy AI are listed. Then the ethical problems still present in the field of ML systems are also explored. In regards the data privacy and security, an overview of the concepts is given and explored in the context of machine learning and federated learning.

3.5.1 Ethical Issues in Artificial Intelligence

Ethics in AI is a topic that is very much discussed nowadays, and its importance has been rising since the creation of very powerful AI systems like robot Sophia (Hanson Robotics, 2016), ChatGPT (OpenAI, 2022), and many others. "With great power comes great responsibility" as the says goes, and that is very applicable in the AI domain. AI has been increasing its domain applicability and is surpassing the precision of human skills in many problems that a few years back were thought to be impossible to solve. But to understand what the ethics issues in AI are, there should be a definition of what ethics is and what its applicability in AI. Ethics, according to (Wang et al., 2020), is a set of moral principles that help us divide the good values from the bad values. In AI, ethics are the set of guidelines that describe ways and principles to design and develop AI systems to minimize negative impacts on the people using the AI as well as the society itself. Because of bad system design and lack of analysis in the development phase of AI systems, problems at different levels can occur in our society (Hagendorff, 2020). For instance, Amazon developed an AI-powered hiring tool that displayed bias by favoring male candidates over female ones (Lavanchy, 2018). Similarly, Microsoft introduced a chatbot on Twitter that, within 24 hours of interacting with users, began posting offensive and racist comments (Vincent, 2016).

Despite these problems not causing too much damage, as they were solved quickly, it is still a concern for AI system developers that these issues must be well analysed and solved even before the issues manifest themselves. That's why the European Commission came up with a set of guidelines on how to build Trustworthy AI (European Commission, 2019). These guidelines were established by the High-Level Expert Group on AI under the European Commission. They outline seven essential requirements that AI systems must adhere to in order to be deemed trustworthy. These principles impact all stakeholders involved with the AI system, including developers, deployers, end-users, and society as a whole. The requirements are as follows:

Human agency and oversight: AI can't harm the physical and psychological integrity of the individuals, but rather protect them. The AI system must support humans and not the other way around.

Technical robustness and safety: AI systems must be prepared against all attacks possible from malicious actors and have security mechanisms that stop the execution of AI when some action is about to cause any harm to a human. The AI systems must be reliable.

Privacy and data governance: AI systems must protect the privacy and integrity of the sensitive data that they use, from the training phases to the deployment and execution of the system.

Transparency: All the lifecycle of the system must be traceable in a way that is possible to know the reasons for any given decision made by the AI, this is possible with the help of interpretability. In addition, humans have always the right to know that they are interacting with an AI.

Accountability: The internal and external evaluations of an AI system should be made public so the trustworthiness of the system can be raised. Every negative impact that occurred in the execution of these systems should be made public as well.

These guidelines summarize all the aspects that all the persons involved must be aware of so that it is possible to create trustworthiness around AI.

Regarding this document, ethical problems can come up when using ML systems. For instance, training a model with biased data can lead to unfair forecasts. If the model, for isntance, overpredicts the quantity for certain products it can lead to massive and innefficient allocation of that product while neglecting others. Being the product transfer and allocation operations one of the initial steps of WMS this can lead to several problems and challenges on the long run.

This models handle large volumes of sensitive data. Improper use of this data could result in breaches or unauthorized access as far as privacy concerns.

When utilizing these data, it is essential to consider the implications of the General Data Protection Regulation (GDPR) in the data handling practices. GDPR, a regulation enacted by the European Union, imposes strict guidelines on the collection, processing, and storage of personal

data of individuals within the EU. To ensure compliance with GDPR, the study follows some key measures.
Also, the lack of transparency on the decision-making process can raise questions for the users as far as accuracy and efficiency performance as they may be unable to understand or challenge the rationale behind specific predictions and allocations.

3.5.2 Data Privacy and Security

Data privacy and security are two topics that are highly related to ethics regarding the AI domain. If we let data privacy and security be trespassed and do not take action to take the necessary precautions, people can act maliciously more easily. If the systems developed are prepared against that type of malicious act, the harm can be reduced or even inexistent (Bertino et al., 2021). Data privacy is everything that concerns the personal data of the users, more specifically how the data is stored, changed, removed, and shared. Data privacy protects the user from how its data is used giving him control over how its data can be managed (Dilmaghani et al., 2019). Data security, on the other hand, is everything that concerns the protection of data from theft, corruption, and unauthorised access throughout its lifecycle. This lifecycle passes through the creation, storage, usage, sharedness, archiving and destruction of data (Meurisch et al., 2022). In a summarized way security in AI is about protecting the system in a way that malicious attacks do not compromise the wellbeing of the system (Fabiano, 2019).

Data privacy and security are major concerns in the field of ML. One concern is the potential for sensitive information, such as personal data, to be accessed or misused by unauthorized parties (X. Liu et al., 2021). Another concern is the possibility of adversarial attacks, where an attacker manipulates the input data to cause an ML algorithm to make incorrect predictions. Additionally, ML algorithms can perpetuate biases present in the training data, potentially leading to discriminatory outcomes. Ensuring the security of ML systems and protecting the privacy of individuals whose data is used for training and testing is crucial to maintaining trust in the technology (Bae et al., 2018). However, measures are already being studied to mitigate these issues (Chen et al., 2022). For example, (Abadi et al. 2016) propose a technique called differential privacy that makes it difficult for an attacker to infer information about an individual data point by adding noise to the model update parameters. Another example is the work of (Steinhardt et al. 2017) where the authors test some attack and defence worst-case situations to study the impact that the attacks may have on the users. As it is possible to see, ML continues to have a lot of challenges regarding the security and privacy of data.

On the context of WMS, ensuring data privacy and security using ML systems remains a challenge. These systems rely heavily on operational data like historical transacions, sales, movements among others that if accessed or misused by unauthorized people could lead to disrupt operations as mentioned before and even business integrity. Also, this unauthorized access would affect the efficiency and accuracy leading to flawed decisions such as overpredicting/underpredicting consequently leading to overstocking/understocking and

misallocation of the products. Needless to say all this influences customer satisfaction which is probably what the business is looking for.

4 Implementation, Analysis and Results Discussion applied to Time Series Forecasting

In this chapter, the implementation of the proposed forecasting models will be presented. The forecast comparison analysis will evaluate the performance of three different approaches — a naive baseline, an XGBoost regression model, and an LSTM network — in predicting daily warehouse outflows. The proposed forecasting models will be assessed based on standard evaluation metrics. The results of each approach will be analyzed and compared to determine the most effective model for this time series forecasting task.

To ensure effective training and encoding of categorical features, the dataset was filtered to include only the most representative brand–category combinations. Specifically, the top 10 brands with the highest average lifecycle and mean quantity transferred were selected. For each of these brands, the top 10 product categories (based on the same criteria) were retained. Initially, the dataset included **55338 distinct products**, **499 brands**, and a hierarchical category structure with **18 Level 1, 136 Level 2, and 681 Level 3 categories**.

It is also very important to note that the dataset includes products with low lifecycle, product obsolescence or demand discontinuities. These occur when the launch of a new product leads to the discontinuation of older versions. These events introduce challenges to these forecasting models. To mitigate this issue, the forecasts were made at the brand and category level instead of targeting specific products.

Before selecting brand and category-level forecasting as the most reliable approach, a thorough exploratory study was conducted. This included forecasting the total number of transfers inside the warehouse, as well as transfers segmented by brand, by category, and by individual products (only for those with a stable and sufficiently long history). This step allowed for the

identification of the aggregation level that best balanced data availability, forecast accuracy, and resilience to product lifecycle disruptions.

4.1 Evaluation Metrics

The two main evaluation metrics selected were:

- **RMSE:** measures the square root of the average squared differences between predicted and actual values. RMSE penalizes large errors more strongly, making it sensitive to outliers.
- **MAE:** computes the average of the absolute differences between predictions and actual values, offering a more interpretable and robust metric against outliers.

These two metrics were selected to provide a comprehensive view of each model's forecasting performance

4.2 XGBoost

To forecast the daily quantity of items to be transferred in the warehouse, a gradient boosting model based on XGBoost was developed and optimized using PSO. XGBoost is well-suited for tabular data and offers high accuracy, speed, and scalability through its efficient implementation of gradient-boosted decision trees.

The following sections detail the proposed XGBoost architecture, training configuration, and the forecasting performance in comparison to other models.

4.2.1 Implementation Tools and Libraries

For the implementation of the XGBoost forecasting model, several Python libraries were used to support data handling, preprocessing, visualization, model training and evaluation. Table 8 below summarises the main libraries and their versions.

Table 8 - XGBoost model used libraries and versions

Library	Version	Purpose
pandas	2.2.2	Data manipulation and preprocessing
numpy	1.26.4	Numerical computations and array handling
scikit-learn	1.5.2	Preprocessing (scaling, encoding, imputation), model evaluation and splitting
scipy	1.14.1	Statistical functions and utilities
xgboost	2.1.1	Implementation of the forecasting model
matplotlib	3.9.2	Statistical data visualization
seaborn	0.13.2	Statistical data visualization
plotly	5.24.1	Interactive data visualization
statsmodels	0.14.4	Statistical analysis and time series diagnostics
holidays	0.56.0	Feature engineering for national/global holiday calendars

These libraries collectively enabled the preprocessing of historical warehouse data, the training and evaluation of the XGBoost model, as well as the generation of visualisations to interpret results and support decision making.

4.2.2 Model Architecture and Hyperparameters

This section presents the XGBoost model design and the hyperparameter optimization process. First, the tuning procedure using PSO is described, followed by the final configuration of the proposed model.

4.2.2.1 Hyperparameter Optimization with PSO

To identify optimal model hyperparameters and improve generalization performance, PSO was employed. In this approach:

- Each particle represents a potential set of XGBoost hyperparameters.
- The objective function evaluates RMSE on the validation set for each particle's configuration.
- The algorithm iteratively updates particle positions based on their own best-known performance and that of their neighbors, gradually converging to an optimal solution.

The following hyperparameters were tuned:

Table 9 - XGBoost hyperparameters to be tuned

Hyperparameter	**Description**	**Range**
n_estimators	Number of boosting rounds	[100, 10000]
learning_rate	Step size shrinkage	[0.001, 0.3]
max_depth	Maximum tree depth	[3, 12]
subsample	Row subsampling ratio	[0.5, 1.0]
colsample_bytree	Column subsampling ratio	[0.5, 1.0]
reg_alpha	L1 regularization term on weights	[0, 10]
reg_lambda	L2 regularization term on weights	[0, 20]

The PSO algorithm was configured with:

- **10 particles**, each representing a potential solution.
- **20 iterations** (to converge to optimal solution).
- Inertia weight **(w = 0.9)**, cognitive coefficient **(c1 = 0.5)** and social coefficient **(c2 = 0.3)** as Swarm behavior parameters.

The optimization returned the best performing configuration based on the lowest RMSE on the validation set.

4.2.2.2 Final Proposed Model

The best hyperparameter set found by PSO was used to instantiate the final XGBoost model. The model was trained on the full training set with early stopping enabled **(early_stopping_rounds = 50)** using the validation set as reference. The tree_method was set to hist for faster histogram-based tree construction.

The final model was persisted using joblib to ensure reproducibility, along with the fitted preprocessor and category mappings used in the one-hot encoding. This ensures that the same transformation logic is applied at inference time and avoids inconsistencies caused by categorical levels not seen during training.

At the end, the proposed XGBoost architecture looked like this:

Table 10 - XGBoost final hyperparameters

Hyperparameter	Description	Value
n_estimators	Number of boosting rounds	2277.23
learning_rate	Step size shrinkage	0.01499
max_depth	Maximum tree depth	10.46
subsample	Row subsampling ratio	0.57705
colsample_bytree	Column subsampling ratio	0.97646
reg_alpha	L1 regularization term on weights	6.11836
reg_lambda	L2 regularization term on weights	19.2545
early_stopping_rounds	Number of rounds with no improvement before stopping train	50
booster	Type of booster to use – gbtree builds decision trees	gbtree
tree_method	Algorithm used for tree construction – hist is optimized	hist
random_state	Seed for reproducibility of results	42
eval_metric	Evaluation metric used to assess model performance	rmse

4.2.3 Features and Training

To train the forecasting model, a series of preprocessing steps and feature engineering techniques were applied to the dataset. The objective was to capture both temporal patterns and product-specific behaviors that could influence future transfer volumes within the warehouse.

4.2.3.1 Target Definition

The variable to be predicted was defined as the quantity of items transferred on the next day. This target, named **QUANTITY_NEXT_DAY**, was computed by grouping the dataset by **BRAND** and **PRODUCTHIERARCHY3** (the product category) and applying a **one-day forward shift to the QUANTITY variable**. This ensured that the forecasted quantity reflected brand-specific and category-specific temporal dynamics.

After computing the target variable, rows with missing values were removed from all data splits to ensure consistency and model compatibility.

4.2.3.2 Feature Engineering

To enrich the dataset with temporal and statistical context, several sets of features were engineered:

- Lag Features: Values of QUANTITY feature from previous days, namely LAG1 (1 day before), LAG2 (2 days before), LAG7 (1 week before), LAG15 (2 weeks before) and LAG30 (1 month before).
- Differences: Difference of the QUANTITY feature between current and specific lags, namely DIFF1 (difference between today and yesterday), DIFF2 (difference between current and 2 days ago), DIFF7 (difference between current and 1 week ago), DIFF15 (difference between current and 2 weeks ago) and DIFF30 (difference between current and 1 month ago).
- EWMA: Used to capture smoothed trends
- Time encodings: Tabular representation of the time, namely DAYOFWEEK (0 to 6), DAYOFMONTH (1 to 31), DAYOFYEAR (1 to 365), QUARTER (1 to 4) and MONTH (1 to 12).
- Holiday and calendar flags: Binary indicators for Portuguese and global holidays and weekend days.

4.2.3.3 Feature Selection

After preprocessing, the final feature set included both numerical and categorical variables, listed below:

Table 11 - Numerical features for XGBoost model

Type	**Feature**
Quantity	QUANTITY
Lag	LAG1, LAG2, LAG7, LAG15, LAG30
Difference	DIFF1, DIFF2, DIFF7, DIFF15, DIFF30
EWMA	EWMA5, EWMA20, EWMA50
Time	DAYOFWEEK, DAYOFMONTH, DAYOFYEAR, QUARTER, MONTH
Holiday/Weekend	IS_PT_HOLIDAY, IS_WEEKEND

Table 12 - Categorical features for XGBoost model

Type	**Feature**
Brand	BRAND
Category	PRODUCTHIERARCHY1
Category	PRODUCTHIERARCHY2
Category	PRODUCTHIERARCHY3

4.2.3.4 Data Preprocessing and Encoding

Before model training, the data was processed using a scikit-learn ColumnTransformer to handle different data types appropriately:

- Numerical features were imputed using the mean of each column.

- Categorical features were encoded using one-hot encoding

The preprocessor was fitted on the training set only to avoid data leakage, and then applied to the validation and test sets.

4.2.3.5 Training

The dataset was split chronologically into three sets:

- Training set (70%): used to fit the models and scalers.
- Validation set (20%): used for model tuning and early stopping.
- Test set (10%): used only for final evaluation of model generalization.

The predictors (X_train, X_val, X_test) and targets (y_train, y_val, y_test) were constructed using the final transformed features and the previously computed QUANTITY_NEXT_DAY feature.

4.3 LSTM Network

To forecast the daily quantity of items to be transferred in the warehouse, a deep learning approach based on LSTM network was developed.

In order to feed the temporal data into the LSTM model efficiently, a custom WindowGenerator class was implemented. This class is responsible for transforming the raw data into sequences of fixed-length sliding windows, allowing the model to learn from both short- and long-term historical dependencies. It handles the construction of input and label windows, the generation of TensorFlow datasets for training, validation, and testing, and supports visualization of the data and predictions. The use of this class ensures consistency in window slicing across all experiments and simplifies the preparation of time series batches for model training.

The following sections detail the proposed LSTM architecture, training configuration, and the forecasting performance in comparison to other models.

4.3.1 Implementation Tools and Libraries

For the implementation of the LSTM forecasting model, a different set of Python libraries was required, focused on deep learning frameworks in addition to data handling and visualisation. Table 13 below summarises the main libraries and their versions.

Table 13 - LSTM model used libraries and versions

Library	Version	Purpose
pandas	2.2.2	Data manipulation and preprocessing
numpy	1.26.4	Numerical computations and array handling
scikit-learn	1.5.2	Preprocessing (scaling, encoding, imputation), model evaluation and splitting
tensorflow	2.17.0	Deep learning framework used to build and train the LSTM model
keras	3.5.0	High-level API for defining neural network architectures within TensorFlow
matplotlib	3.9.2	Statistical data visualization
seaborn	0.13.2	Statistical data visualization
plotly	5.24.1	Interactive data visualization
holidays	0.56.0	Feature engineering for national/global holiday calendars

These libraries allowed the preparation of input sequences, the design and training of the LSTM neural network, and the analysis of results through visualisation and statistical validation.

4.3.2 Model Architecture and Hyperparameters

This section describes the architecture of the proposed LSTM model and the hyperparameters used in its training. First, the hyperparameter search strategy based on Bayesian optimization is presented, followed by a detailed explanation of the final model configuration and training process.

4.3.2.1 Hyperparameter Optimization with Bayesian optimization

To optimize the LSTM's architecture, **Bayesian optimization** was applied using **Keras Tuner**. The search was performed over:

Table 14 - LSTM hyperparameters to be tuned

Hyperparameter	Description	Range
lstm_units	Number of boosting rounds	[32, 128]
dense_units	Number of units in dense layer	[32, 128]
learning_rate	Initial learning rate	[1e-4, 1e-1]
batch_size	Number of samples per batch	[32, 1024]

A total of 20 trials were executed, each training a different model configuration for a 10 epochs during the search phase. The best configuration, determined by the lowest validation RMSE, was then used for full training on the final model.

4.3.2.2 Final Proposed Model

The model architecture is composed of three main inputs:

- A **sequence of numerical features**, representing the time-dependent data over a fixed-length window (lookback), passed through two stacked LSTM layers (with 96 and 64 units respectively) followed by a dense layer with 64 ReLU-activated units.
- A **brand ID input**, passed through a learnable embedding layer whose dimension is dynamically determined based on the number of unique brands, followed by flattening.
- A **product hierarchy ID input**, also passed through a similar embedding layer and flattened.

The outputs of the LSTM sequence and both embeddings are concatenated and passed through two fully connected layers with 64 and 32 ReLU-activated units. The final output is a single neuron producing a continuous value representing the predicted quantity for the next day.

The model was compiled using the **Adam optimizer** with a learning rate of 0.00568, and trained with **RMSE** as the loss function. Additionally, **MAE** was used as an evaluation metric during training and validation.

To improve model generalization and convergence, the following **training callbacks** were used:

- **Early stopping**, to halt training when validation RMSE stopped improving for 10 consecutive epochs.
- **Learning rate reduction on plateau**, which reduced the learning rate by a factor of 0.5 when validation RMSE plateaued for 5 epochs.
- **Model checkpointing**, to save the best performing model based on validation RMSE.

The model was trained with a batch size of 256 for a maximum of 100 epochs.

Below is a table summarizing all proposed LSTM hyperparameters:

Table 15 - LSTM final hyperparameters

Hyperparameter	Description	Value
lookback	Number of past time steps used as input	30
num_features	Number of numeric input features	28
brand_vocab_size	Number of unique brand identifiers	10
hier_vocab_size	Number of unique category identifiers	64
lstm_layer_1_units	Number of units in the first layer	96
lstm_layer_2_units	Number of units in the second layer	64
dense_layer_1_units	Fully connected layer after LSTM output	64
dense_layer_2_units	Second dense layer after embeddings + LSTM concat	32
embedding_dim (brand)	Dimension of brand embedding	min(32, max(4, brand_vocab_size//2 + 1))
embedding_dim (category)	Dimension of category embedding	min(32, max(4, hier_vocab_size//2 + 1))
optimizer	Optimization Algorithm	Adam
learning_rate	Initial learning rate	0.00568
loss_function	Objective function	RMSE
metrics	Additional evaluation metrics	MAE
epochs	Maximum number of training epochs	100
batch_size	Number of samples per batch	256
early_stopping	Patience for early stopping	10
reduce_lr_on_plateau	Factor to reduce learning rate if no improvement	0.5
checkpoint	Best model selection based on	Validation RMSE

This combination of LSTM-based temporal modeling, embedding layers for categorical context, and Bayesian hyperparameter optimization allowed the model to effectively learn from structured time series data and produce competitive forecasts.

Below there is a representation of the proposed LSTM final architecture:

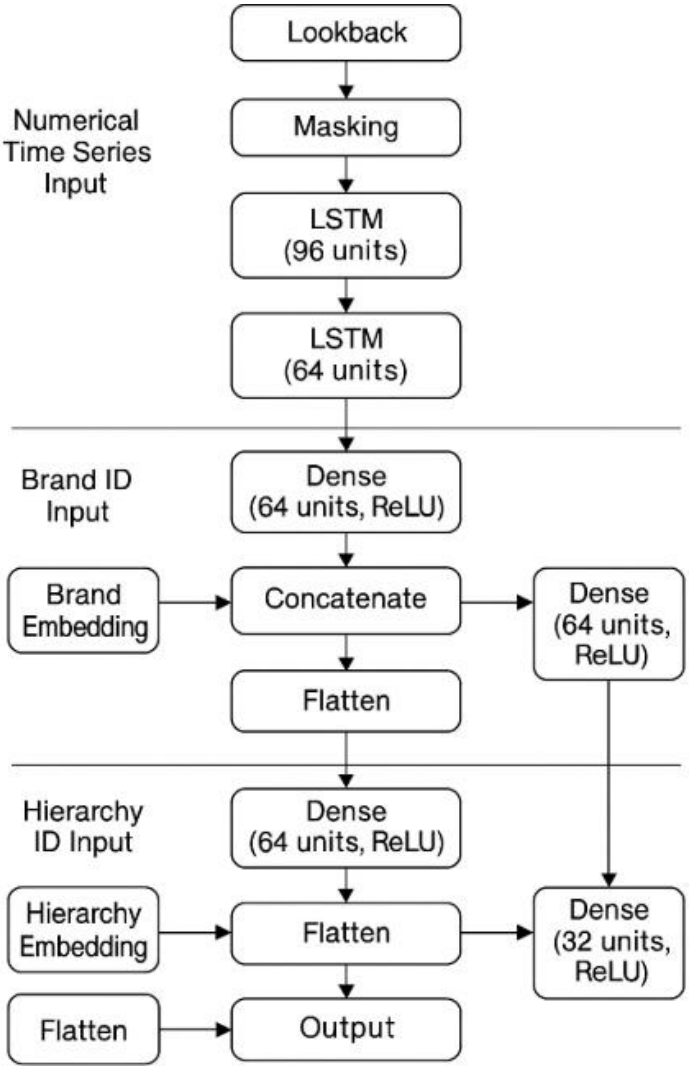

Figure 7 - Proposed LSTM final architecture

4.3.3 Features and Training

To train the LSTM-based forecasting model, the same series of preprocessing steps and feature engineering techniques applied to the XGBoost model were reused. The goal remained to represent both temporal patterns and item-specific behavior that could influence the volume of transfers within the warehouse.

4.3.3.1 Target Definition

The target variable for the LSTM model was also defined as the quantity of items transferred on the following day, referred to as **QUANTITY_NEXT_DAY**. This was computed by grouping the dataset by **BRAND** and **PRODUCTHIERARCHY3** and **applying a one-day forward shift** to the

QUANTITY variable. This ensured the LSTM model would learn to forecast future transfers using both **brand-specific and category-specific historical patterns**.

As with the XGBoost model, rows with missing values in the input or target columns were removed after shifting, and this cleaning was applied separately for the training, validation, and test splits.

4.3.3.2 Feature Engineering

The LSTM model reused the same engineered features as the XGBoost model to maintain consistency and allow for fair performance comparison. These included lag variables, difference features, exponential weighted moving averages (EWMAs), and binary flags for holidays and weekends.

The only exception was the representation of temporal features. Instead of using tabular encodings such as day of the week or month index, the LSTM model employed cyclical time encodings using sine and cosine transformations. This approach is more suitable for neural networks, as it allows the model to recognize the periodic nature of time-based variables (e.g., the continuity from Sunday to Monday or December to January).

4.3.3.3 Feature Selection

The final set of features for the proposed LSTM model included the following:

Table 16 - Numerical features for LSTM model

Type	**Feature**
Quantity	QUANTITY
Lag	LAG1, LAG2, LAG7, LAG15, LAG30
Difference	DIFF1, DIFF2, DIFF7, DIFF15, DIFF30
EWMA	EWMA5, EWMA20, EWMA50
Time	WEEK_SIN, WEEK_COS, MONTH_SIN, MONTH_COS, YEAR_SIN, YEAR_COS
Holiday/Weekend	IS_PT_HOLIDAY, IS_WEEKEND

Table 17 - Categorical features for LSTM model

Type	**Feature**
Brand	BRAND
Category	PRODUCTHIERARCHY3

4.3.3.4 Data Preprocessing and Encoding

For the LSTM model, the dataset was normalized using a Min-Max scaler applied to the numerical features. Categorical features (brand and category) were transformed into embeddings through dedicated embedding layers within the model architecture, allowing it to learn dense representations of brand and category identifiers.

A custom WindowGenerator class was used to convert the raw time series into overlapping input sequences (windows) and corresponding future labels. This class ensured consistent slicing of windows for training, validation, and testing, and was also used for plotting model predictions.

4.3.3.5 Training

The dataset was chronologically split into:

- Training set (70%): used to fit the LSTM weights and embedding layers.
- Validation set (20%): used to monitor generalization and control early stopping.
- Test set (10%): held out completely until final evaluation.

Each input to the LSTM consisted of a fixed-length sequence of time steps (lookback), including numeric features and the corresponding brand and category identifiers. The target value was the next-day quantity (QUANTITY_NEXT_DAY).

The model was trained using the Adam optimizer with RMSE as the loss function. Evaluation metrics also included MAE, a combination of early stopping, learning rate scheduling and model checkpointing was used to improve generalization and prevent overfitting.

4.4 Models Evaluation and Comparison

On this section the models are evaluated in terms of predictive performance. The objective is not only to measure accuracy but also to compare how effectively each model captures the temporal dynamics of warehouse movements.The XGBoost regression, the LSTM network and Prophet heuristic are assessed using the evaluation metrics defined earlier, providing both error magnitude and interpretability. To ensure the generalizability of the results, **the evaluation was carried out across different brand–category combinations, specifically two distinct brands, each tested on two categories with consistent transfer activity**, namely:

- Brand “1487” (Xiaomi) and category “1060000100001” (Smartphones Android).
- Brand “1487” (Xiaomi) and category “1060000800001” (Earbuds and Earphones).
- Brand “1791” (Logitech) and category “1090000600001” (Mouses).
- Brand “1791” (Logitech) and category “1090000600002” (Keyboards).

For each brand–category pair, all models were trained using historical data and tested using a test set. The forecasts were compared against actual values using the RMSE and MAE metrics.

The tables below summarize the evaluation results:

Table 18 - Evaluation results for brand Xiaomi

Brand	Category	Model	RMSE	MAE
Xiaomi (1487)	Smartphones Android (1060000100001)	XGBoost	45.65	31.07
Xiaomi (1487)	Earbuds and Earphones (1060000800001)	XGBoost	40.56	29.01
Xiaomi (1487)	Smartphones Android (1060000100001)	LSTM	52.64	41.86
Xiaomi (1487)	Earbuds and Earphones (1060000800001)	LSTM	44.99	34.68
Xiaomi (1487)	Smartphones Android (1060000100001)	Prophet	59.66	42.96
Xiaomi (1487)	Earbuds and Earphones (1060000800001)	Prophet	78.02	60.29

Table 19 - Evaluation results for brand Logitech

Brand	Category	Model	RMSE	MAE
Logitech (1791)	Mouses (1090000600001)	XGBoost	42.28	30.28
Logitech (1791)	Keyboards (1090000600002)	XGBoost	12.46	9.89
Logitech (1791)	Mouses (1090000600001)	LSTM	49.21	39.25
Logitech (1791)	Keyboards (1090000600002)	LSTM	17.47	12.11
Logitech (1791)	Mouses (1090000600001)	Prophet	51.50	41.15
Logitech (1791)	Keyboards (1090000600002)	Prophet	16.08	13.92

When visualizing the plots we can conclude:

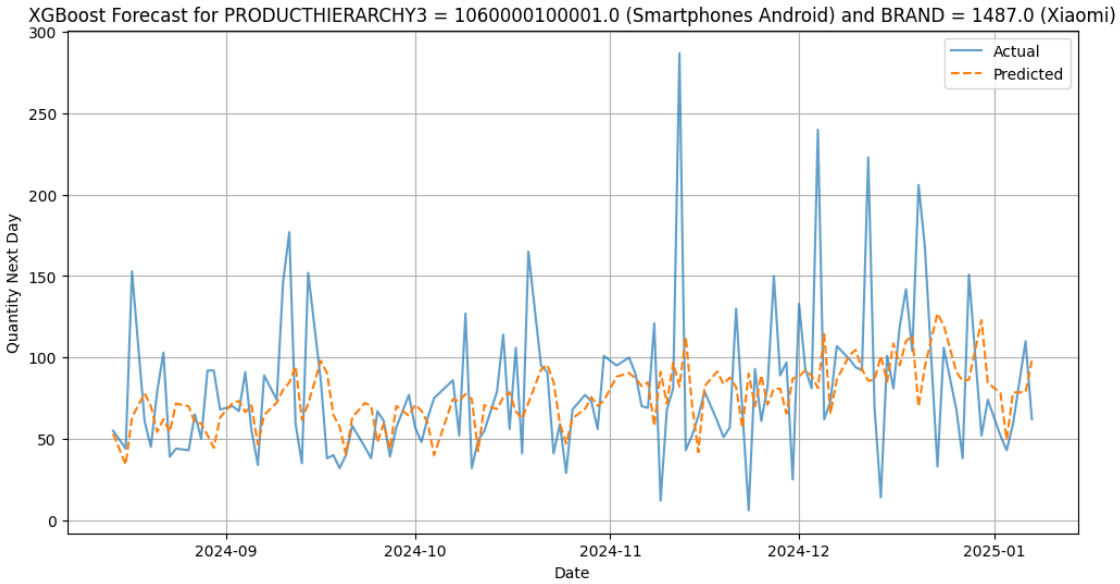

Figure 8 - XGBoost predictions vs actual values for Smartphone Android

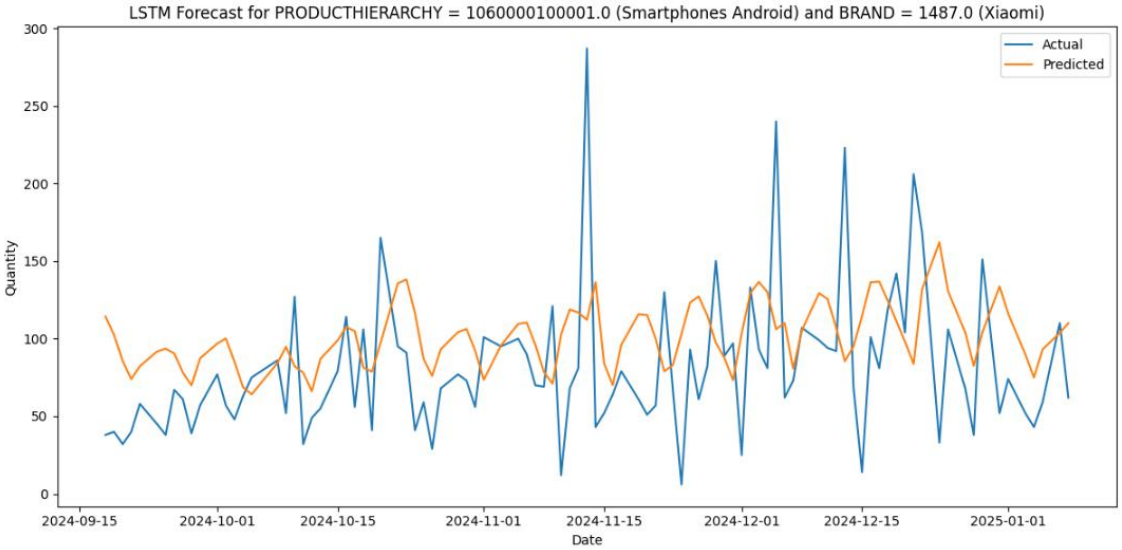

Figure 9 - LSTM predictions vs actual values for Smartphones Android

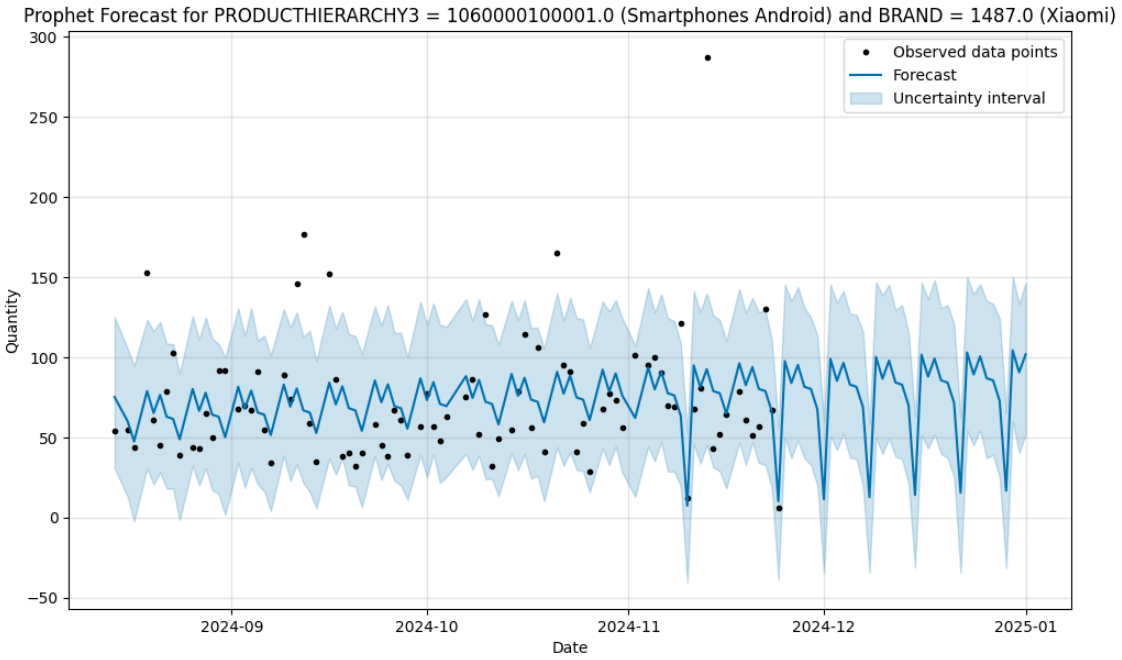

Figure 10 - Prophet predictions vs actual values for Smartphones Android

XGBoost achieved the lowest errors (both RMSE and MAE), making it the most effective model for this category. LSTM and Prophet underperformed, particularly Prophet which had the highest error values.

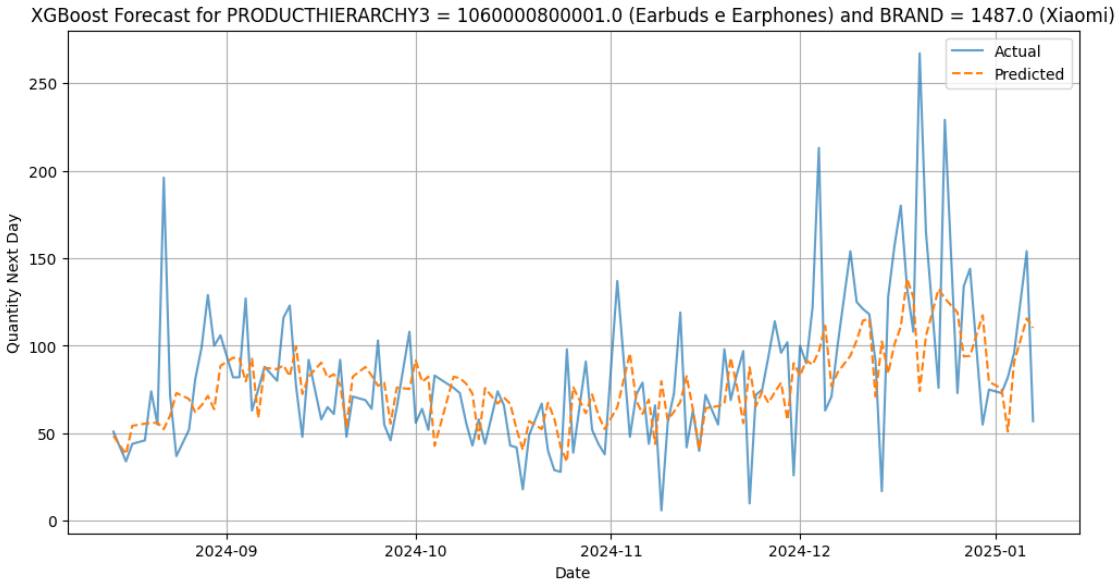

Figure 11 - XGBoost predictions vs actual values for “Earbuds e Earphones”

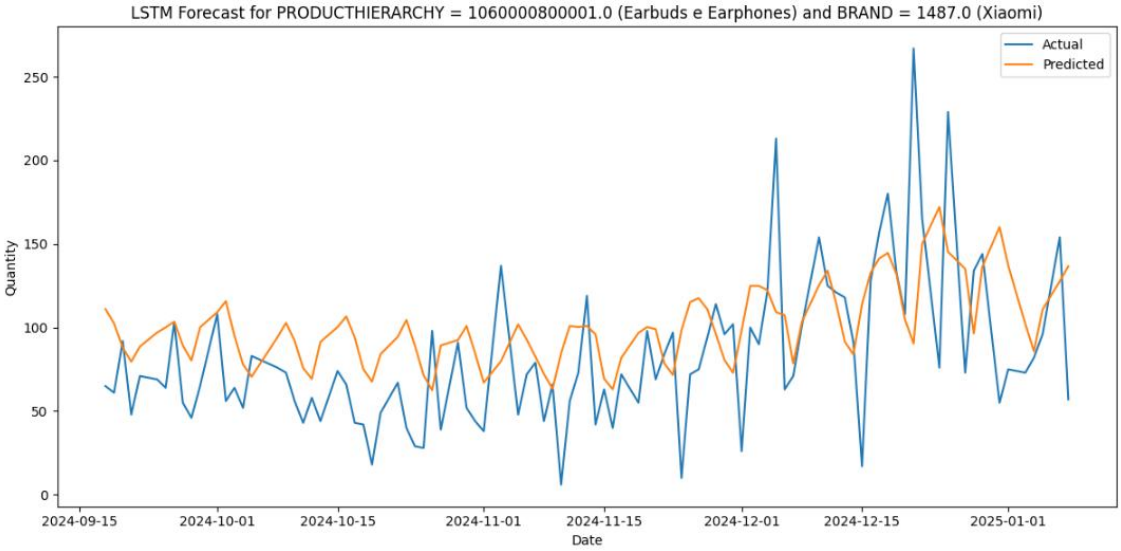

Figure 12 - LSTM predictions vs actual values for Earbuds and Earphones

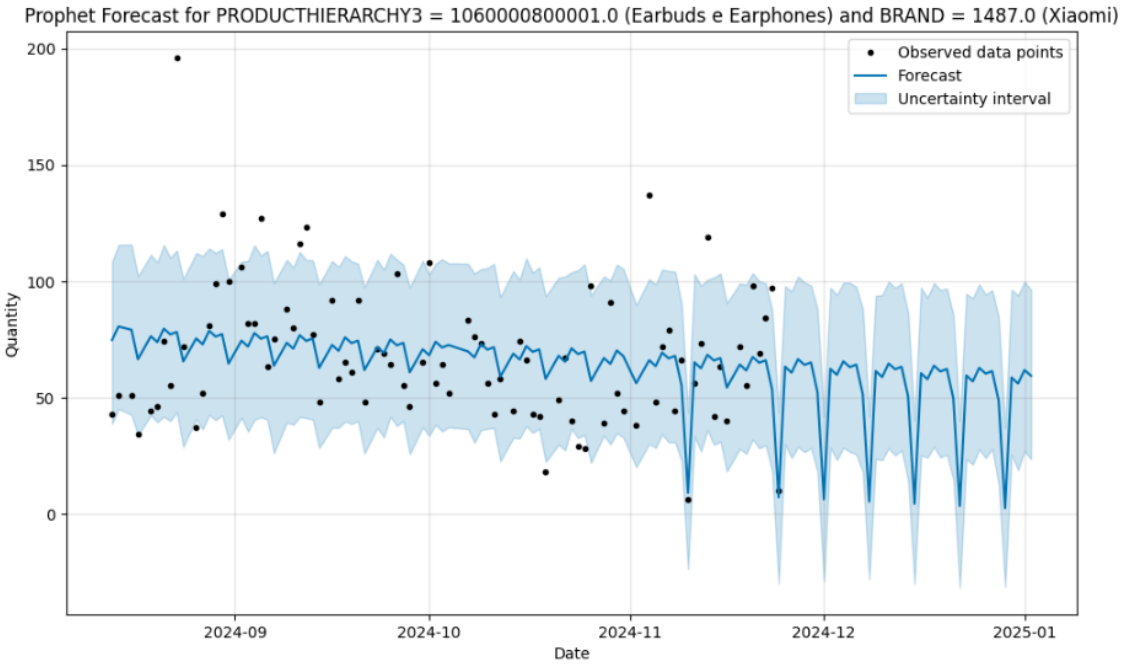

Figure 13 - Prophet predictions vs actual values for Earbuds and Earphones

Once again, XGBoost clearly outperformed the other models, with the lowest error scores. LSTM performed moderately but still worse than XGBoost, while Prophet exhibited the weakest performance by a large margin.

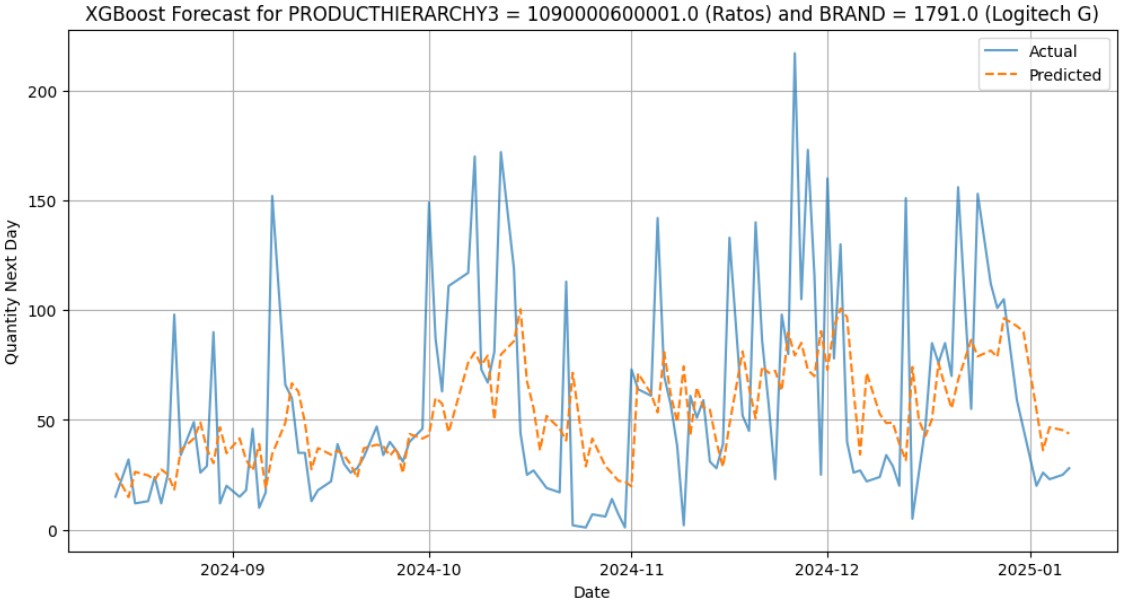

Figure 14 - XGBoost predictions vs actual values for Mouses

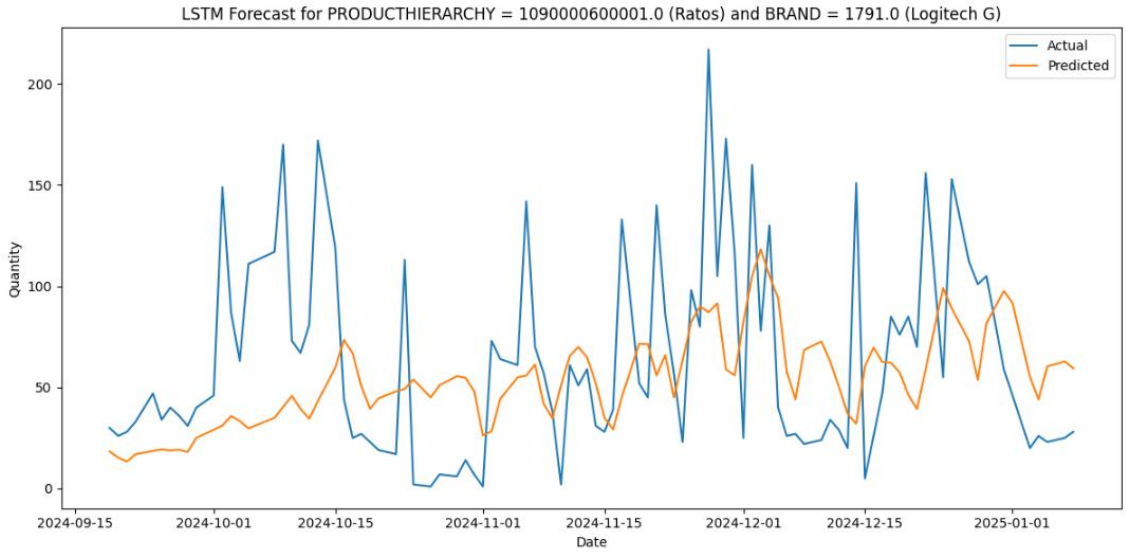

Figure 15 - LSTM predictions vs actual values for Mouses

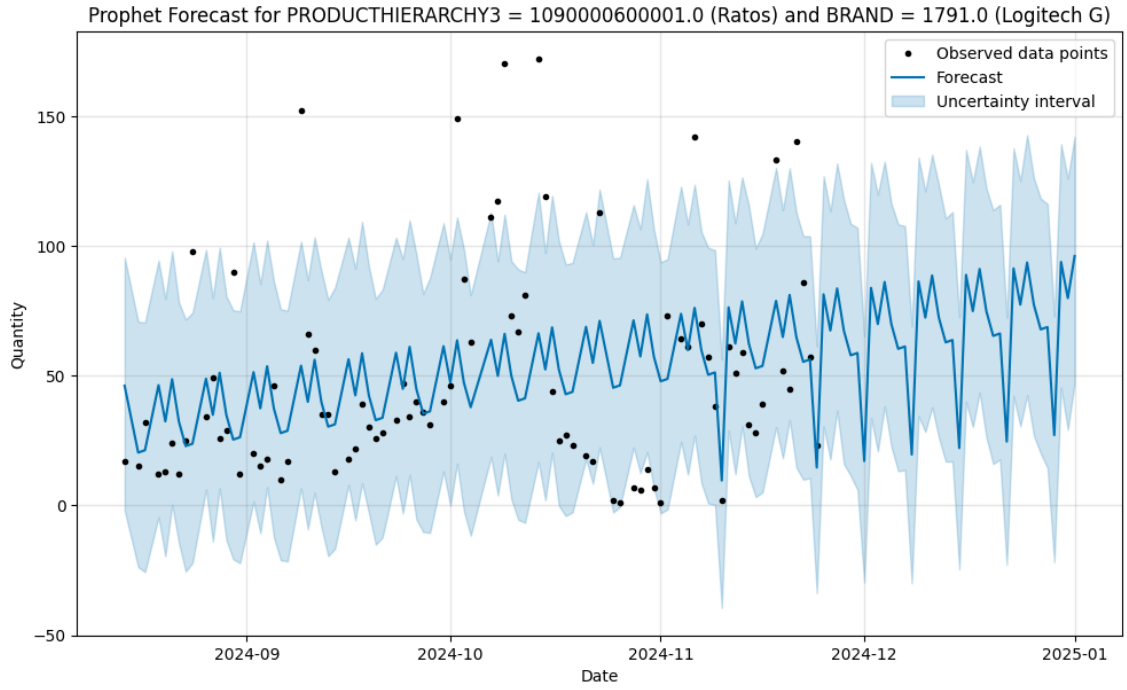

Figure 16 - Prophet predictions vs actual values for Mouses

XGBoost provided the most accurate forecasts, outperforming LSTM and Prophet. Both LSTM and Prophet had relatively close results, but still lagged behind XGBoost in both RMSE and MAE.

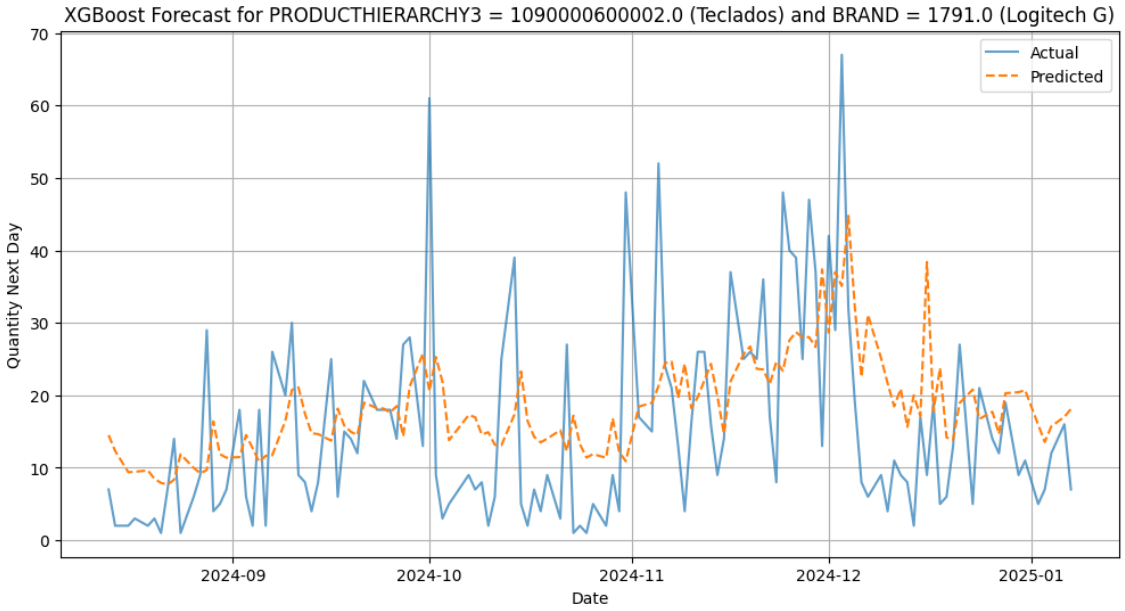

Figure 17 - XGBoost predictions vs actual values for Keyboards

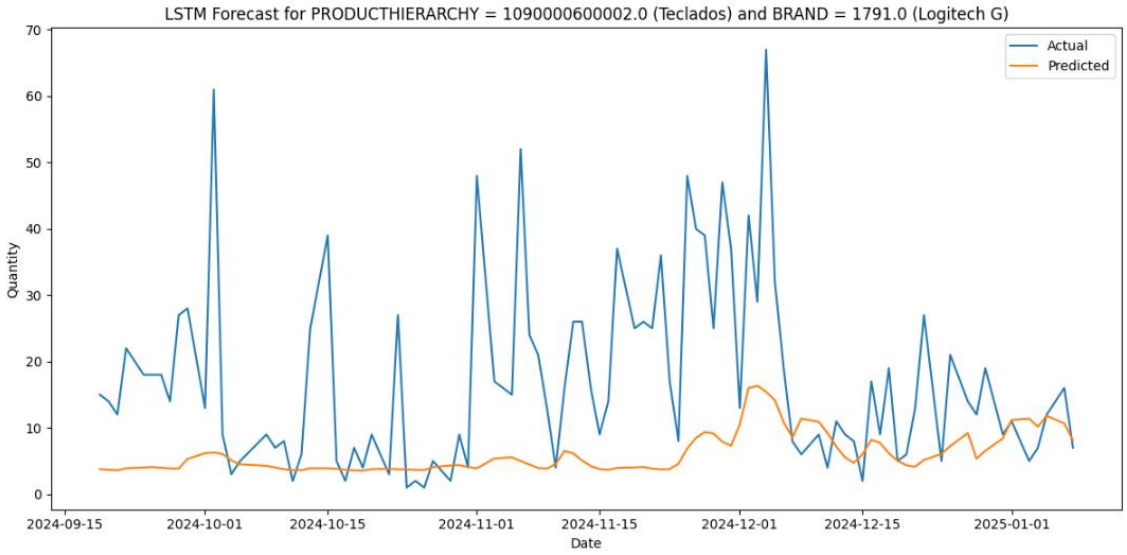

Figure 18 - LSTM predictions vs actual values for Keyboards

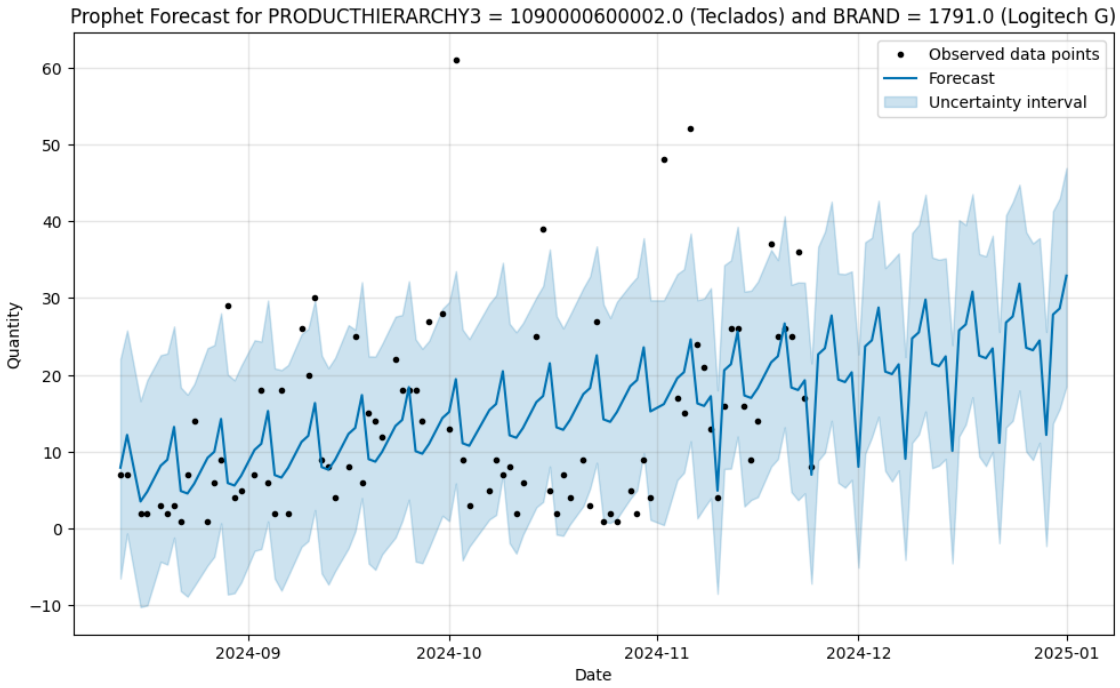

Figure 19 - Prohet predictions vs actual values for Keyboards

XGBoost once again achieved the best performance, with notably lower errors. LSTM came in second, slightly ahead of Prophet in RMSE but behind in MAE.

Across all brand–category combinations, **XGBoost consistently provided the most accurate forecasts with lower RMSE and MAE**. LSTM offered competitive results in some cases, while Prophet generally performed the worst.

5 Implementation, Analysis and Results Discussion applied to 3D Bin Packing Problem

In this chapter, the implementation of the proposed RL solution for the 3D-BPP will be presented. The analysis will focus on the development of a custom simulation environment and the training of agents to learn efficient allocation strategies under realistic warehouse constraints. The main goal of this section is to evaluate whether RL can improve space utilization and box placement compactness compared to a heuristic baseline.

A synthetic dataset was created through recursive splitting of the bin, ensuring realistic box dimensions and consistent variability across episodes. Each episode begins with one empty bin and 50 boxes to be allocated, with new configurations generated at every reset. This setup ensures that the agents experience a wide range of allocation scenarios, allowing the evaluation of their generalization capabilities.

As a baseline for comparison, the Bottom-Left-Back (BLB) heuristic with sorted boxes (largest-first ordering) was implemented. This classical approach ensures that the available space is filled systematically and serves as a strong non-learning benchmark. Two RL agents were then developed: a **DQN** and a **PPO** agent. Both were trained and evaluated within the same environment to allow a fair comparison with each other and with the heuristic baseline.

The results of the allocation task will be assessed using evaluation metrics specifically designed for packing problems, including bin utilization and compactness. The analysis will compare the performance of the BLB heuristic baseline, the DQN agent and the PPO agent, highlighting the improvements achieved by RL in the allocation task.

5.1 Implementation Tools and Libraries

The implementation of the RL agents for the 3D-BPP allocation task required several Python libraries to support environment design, model training, and evaluation. Table 20 summarises the main libraries and their versions.

Table 20 - Libraries and versions used for the allocation task

Library	Version	Purpose
pandas	2.2.2	Dataset preparation and statistical analysis
numpy	1.26.4	Numerical computations and matrix operations
gym	0.26.2	Interface for the custom 3D-BPP environment and action/state handling
torch	2.4.1	Deep learning framework used to implement neural networks for DQN and PPO
stable-baselines3	2.3.2	Reinforcement learning library providing PPO implementation and utilities
matplotlib	3.9.2	Visualisation of training curves and allocation results
plotly	5.24.1	Interactive 3D visualization of box placements inside the bin

Together, these tools enabled the design of the RL environment, the training of DQN and PPO agents, and the comparison against the BLB heuristic baseline.

5.2 Problem Formulation and Environment Design

The allocation task addressed in this work is formulated as a variation of the 3D-BPP. In this problem, a set of boxes must be placed inside a finite container while respecting its spatial constraints. The objective is to maximize the utilization of the available space, while promoting compact and stable packing arrangements.

5.2.1 Problem Formulation

Formally, the problem can be defined as follows:

- **Input**: one bin with fixed dimensions *(W,D,H)* and a set of n boxes, each with dimensions *(w,d,h)*.
- **Constraints**:
    1. Boxes must be placed within the bin boundaries.
    2. Boxes cannot overlap.

3. Boxes must be placed either on the bin floor or on top of previously placed boxes.

- **Objective**: maximize a weighted combination of bin utilization, compactness, and placement stability.

This formulation captures the main requirements of a realistic warehouse allocation scenario, where efficient use of pallet space and the stability of the stacked products are critical.

5.2.2 Environment Design

To simulate this problem, a custom environment was developed based on the OpenAI Gym interface. Each episode begins with one empty bin and 50 boxes generated synthetically through recursive splits of the container. This recursive split strategy for box generation was **adapted from the 3D-BPP repository by Luis Garcia** (Garcia, 2022), which is cited in the References section. This generation method guarantees that the total volume of the boxes matches the bin capacity, while still producing a variety of shapes and sizes across episodes.

The environment manages the bin state, updates the placement of boxes, and ensures that invalid actions are rejected. It was also used to implement the baseline BLB heuristic with sorted boxes, which serves as a benchmark for comparing RL agents. The environment is fully compatible with different agents, supporting the training and evaluation of both the DQN and PPO approaches.

The state space is represented by the current status of the bin and the next boxes to be placed. This includes:

- The dimensions, position, and occupancy status of placed boxes.
- The remaining free space regions inside the bin.
- The dimensions of the next boxes waiting to be placed.
- Additional indicators such as current volume utilization and the number of boxes successfully packed.

This formulation provides the agent with a complete view of the packing progress, enabling it to make informed placement decisions at each step.

The overall interaction between the agent and the environment is illustrated in Figure 20.

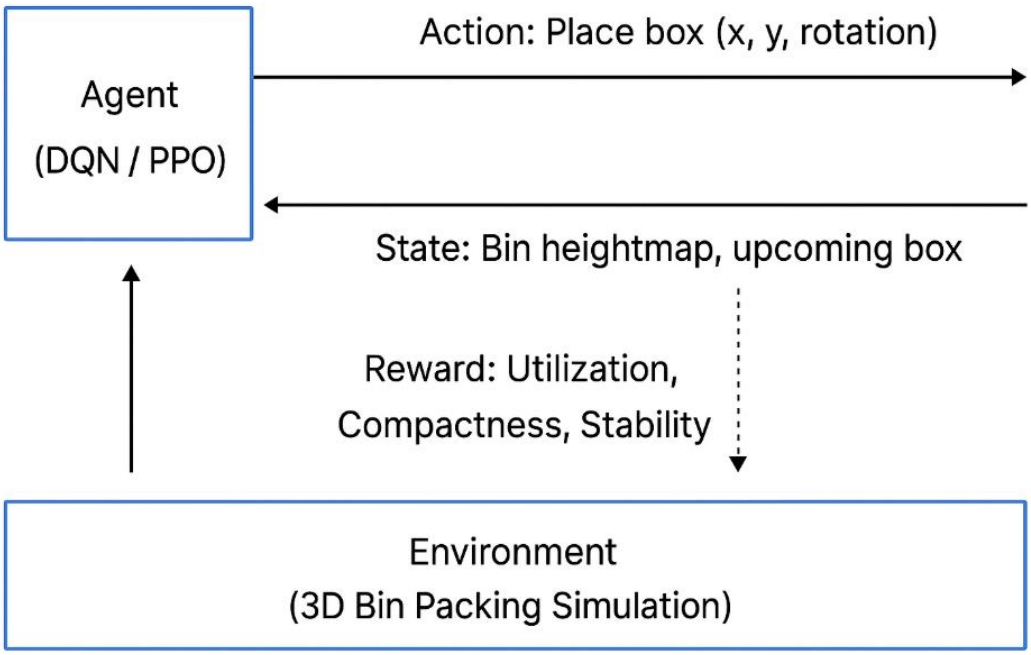

Figure 20 - Interaction between environment and RL agents

5.3 Reward Design and Action Space

This section presents the design choices regarding the reward function and the action space for the RL environment. These elements are crucial to define how the agent perceives progress towards efficient allocations and how it can interact with the environment to achieve this goal.

5.3.1 Reward Design

The reward function plays a central role in guiding the agent towards efficient allocation strategies. In this work, a hybrid reward was designed combining both incremental and terminal components:

- **Incremental reward**: at each step, the agent receives a signal based on the change in compactness after placing a box. Positive changes indicate a denser arrangement, while negative values penalise placements that create additional voids. This provides local feedback during the episode.

- **Terminal reward**: once all boxes are placed or no further actions are possible, a final reward is added proportional to the bin utilization (ratio of packed volume to bin volume). This encourages the agent to maximise space efficiency over the entire episode.

This reward design balances short-term guidance (compactness) with long-term objectives (utilization).

5.3.2 Action Space

The action space was discretised to make the problem tractable for RL. Each action corresponds to a candidate placement of the current box, defined by:

- **Position**: a pair of coordinates *(x,y)* representing the bottom-left location of the box on the bin's base grid.
- **Rotation**: a discrete choice among the valid orientations of the box dimensions.

Thus, each action can be represented as *(x,y,r)*, where r denotes the selected rotation type. To avoid exploring infeasible actions, a **valid action mask** is used: only positions and orientations that respect bin boundaries and avoid collisions are made available to the agent.

This formulation reduces the complexity of the action space while ensuring that the agent learns to make spatial decisions that are both valid and effective.

5.4 Exploration Algorithms

Exploration is a critical aspect of RL, as it allows agents to discover effective strategies instead of prematurely converging to suboptimal behaviours. DQN and PPO, each rely on distinct exploration mechanisms.

5.4.1 DQN Exploration

For the DQN agent, exploration was handled through two alternative strategies:

- **Epsilon-greedy**: with probability epsilon, the agent selects a random valid action instead of the action with the highest predicted Q-value. This mechanism ensures continuous exploration of the action space. During training, epsilon was decayed from a high initial value to a small final threshold, gradually shifting the balance from exploration to exploitation. This simple yet effective approach prevents the agent from becoming stuck in suboptimal allocation patterns at early stages of learning.
- **Softmax exploration**: instead of making random choices uniformly, actions were selected probabilistically according to a softmax distribution over Q-values. Actions with higher predicted values had greater probability of being chosen, while still allowing exploration of less promising actions. A temperature parameter controlled the exploration–exploitation balance: higher temperatures encouraged more uniform

exploration, while lower temperatures made the policy greedier. This method provides a smoother and value-aware exploration process compared to epsilon-greedy.

5.4.2 PPO Exploration

For the PPO agent, exploration is inherent to its stochastic policy. PPO models the probability distribution over actions directly, sampling actions according to their likelihood during training. This ensures that the agent naturally explores the action space while simultaneously adjusting the policy through gradient updates. Entropy regularization was used to maintain sufficient exploration by preventing the policy from collapsing prematurely into deterministic behaviours.

In summary, while DQN relies on explicit exploration strategies (epsilon-greedy or softmax) applied to its Q-value estimates, PPO integrates exploration as part of its stochastic policy formulation. Both approaches were tested in the 3D-BPP environment, enabling a comparison between off-policy value-based learning and on-policy policy-gradient exploration mechanisms.

5.5 Evaluation Metrics

The two main evaluation metrics selected for the allocation task were:

- **Bin Utilization**: measures the ratio between the total volume of packed boxes and the total volume of the bin. This metric captures how efficiently the available space was used.

- **Compactness**: evaluates how tightly the boxes are packed inside the bin. It is defined as the ratio between the total volume of the placed boxes and the volume of the minimal bounding cuboid that contains them all. Higher compactness indicates fewer voids and more stable arrangements.

These two metrics were selected to provide a comprehensive view of the performance of each allocation strategy, balancing overall space usage with the quality and stability of the packing.

5.6 DQN Agent

This section details the implementation of the DQN agent used to solve the 3D-BPP environment. The subsections present the neural network architecture designed to approximate the action–value function and the training strategy with its associated hyperparameters.

5.6.1 Model Architecture

The DQN agent was implemented with a fully connected neural network that approximates the action–value function *Q(s,a)*. The input to the network is the state vector described in Section 5.2, which combines the bin heightmap, the lookahead features of upcoming boxes, and global statistics. This representation encodes both the spatial configuration of the bin and contextual information required for decision making.

The architecture consists of:

- **Input layer**: receives the flattened state vector of dimension *W×D+3k+3*, where *W* and *D* are the bin width and depth, *k* is the lookahead horizon, and the additional three features correspond to global statistics.

- **Hidden layers**: two fully connected layers with 128 neurons each, activated by the Rectified Linear Unit (ReLU) function. These layers provide the capacity to learn non-linear interactions between box dimensions, bin occupancy, and global state indicators.

- **Output layer**: produces one Q-value for each discrete action *(x,y,r)*, where *x,y* represent placement coordinates and *r* the rotation type. Invalid actions are masked during training to prevent infeasible placements.

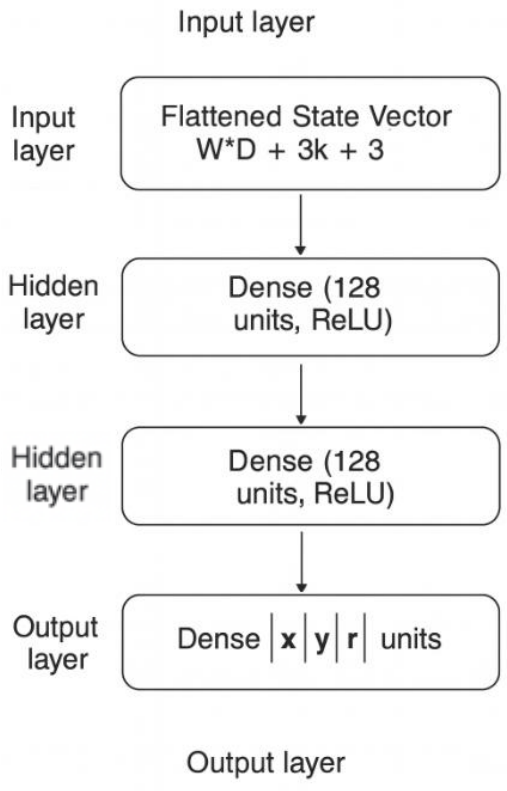

Figure 21 - DQN agent architecture

This relatively compact architecture was chosen to balance expressiveness and training stability. Larger networks were avoided to reduce the risk of overfitting and excessive computational cost, while still allowing the agent to capture meaningful patterns in packing strategies.

5.6.2 Training Strategy and Hyperparameters

The DQN agent was trained in the custom 3D-BPP environment using an **off-policy learning approach**. Transitions *(s,a,r,s')* were stored in a replay buffer and sampled uniformly to update the Q-network, while a separate target network was used to stabilize learning by providing fixed Q-value estimates during each update period.

The main hyperparameters and training settings are summarized in Table 21.

Table 21 - DQN agent hyperparameters

Hyperparameter	**Description**	**Value**
Replay buffer size	Maximum number of transitions stored for experience replay	50000
Batch size	Number of transitions sampled per training update	64
Discount factor γ	Weight assigned to future rewards	0.99
Optimizer	Algorithm used for gradient descent	Adam
Learning Rate	Step size for weight updates during training	0.001
Target update frequency	Steps between synchronizing online and target networks	100
Exploration strategy	Mechanism for balancing exploration vs exploitation	Epsilon-greedy / Softmax
Epsilon schedule	Probability of random action in epsilon-greedy from 1.0 to 0.1 over 50k and 200k steps	Linear Decay
Softmax temperature	Controls action probability distribution in softmax, annealed 1.0 to 0.1	Annealed
Training episodes	Total number of episodes used for training	1000 / 5000

This configuration was selected to balance exploration and exploitation, provide stable learning updates, and ensure the agent experienced sufficient variability across training episodes.

5.7 PPO Agent

This section describes the implementation of the PPO agent for the 3D-BPP environment. The subsections present the actor–critic network architecture adopted and the training strategy with its corresponding hyperparameters.

5.7.1 Model Architecture

The PPO agent uses a fully connected actor–critic network that jointly models the policy $\pi(a|s)$ and the value function $V(s)$. The input is the flattened state vector described in Section 5.2 (bin heightmap, lookahead of upcoming boxes, and global statistics), i.e., a vector of size $W \times D + 3k + 3$.

- **Shared trunk:** two dense layers with 128 units each (ReLU), shared by actor and critic to learn compact, non-linear features from both spatial and contextual signals.
- **Policy head (actor):** a dense layer that outputs logits over all discrete actions (x, y, r). A mask zeroes out infeasible actions at every step so the policy only assigns probability to valid placements.
- **Value head (critic):** a separate dense layer that outputs a single scalar $V(s)$ used as a baseline for variance reduction and stable advantage estimation.

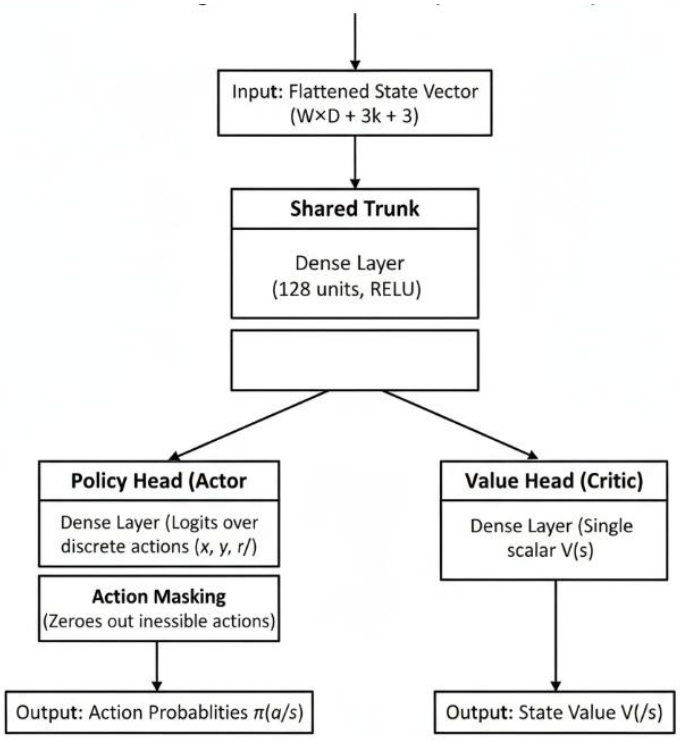

Figure 22 - PPO agent architecture

This compact architecture balances expressiveness with training stability and runtime efficiency, while the action mask enforces feasibility directly at the policy level.

5.7.2 Training Strategy and Hyperparameters

The PPO agent was trained in the same custom 3D-BPP environment, following an **on-policy learning approach**. Instead of replay buffers, PPO collects rollouts of transitions directly from the current policy and updates the network using multiple epochs of minibatch stochastic gradient descent. The clipped surrogate objective stabilizes training by preventing excessively large policy updates, while GAE provides low-variance and less biased estimates of expected returns.

The main hyperparameters and training settings are summarized in Table 22.

Table 22 - PPO agent hyperparameters

Hyperparameter	**Description**	**Value**
Discount factor γ	Weight assigned to future rewards	0.99
GAE λ	Parameter for bias–variance tradeoff in advantage estimation	0.95
Clipping ε	Trust region for policy updates	0.2
Optimizer	Algorithm used for gradient descent	Adam
Learning Rate	Step size for weight updates during training	3*e*-4
Value loss coefficient *(vf_coef)*	Weight of value function loss in total loss	0.5
Entropy coefficient *(ent_coef)*	Encourages exploration by penalizing certainty	0.01
Epochs per update	Number of epochs per policy update	4
Minibatch size	Number of transitions per minibatch	1024
Training episodes	Total number of episodes used for training	1000 / 5000

This configuration was chosen to ensure stable policy optimization, maintain sufficient exploration through entropy regularization, and achieve a balance between sample efficiency and robustness across different training horizons.

The PPO implementation used in this work was **inspired by the work of Hang Zhao** (Zhao, 2021), whose repository link is included in the References section.

5.8 Agents Evaluation and Comparison

In this section, the RL agents are evaluated in terms of their ability to allocate boxes efficiently in the 3D-BPP. The objective is not only to measure performance but also to compare how effectively each agent learns strategies for maximizing bin utilization and ensuring compact and stable placements.

The agents under evaluation are:

- **DQN Agent** – a value-based approach that estimates Q-values for each discrete action.
- **PPO Agent** – a policy-gradient method that directly optimizes the policy using clipped objectives.
- **BLB Heuristic Baseline** – a simple deterministic strategy that sorts boxes by volume before placement and starts placing from the bottom left back positions.

To ensure comparability, all agents were trained and tested under the same environment configuration, consisting of one bin and 50 boxes generated synthetically through recursive splits of the container.

The experiments were run for a fixed number of episodes. The baseline heuristic serves as a reference point, allowing us to assess the gains achieved by reinforcement learning over a rule-based approach.

5.8.1 Learning Curves

Figure 23 shows the learning curve of the **DQN agent trained for 1000 episodes**.

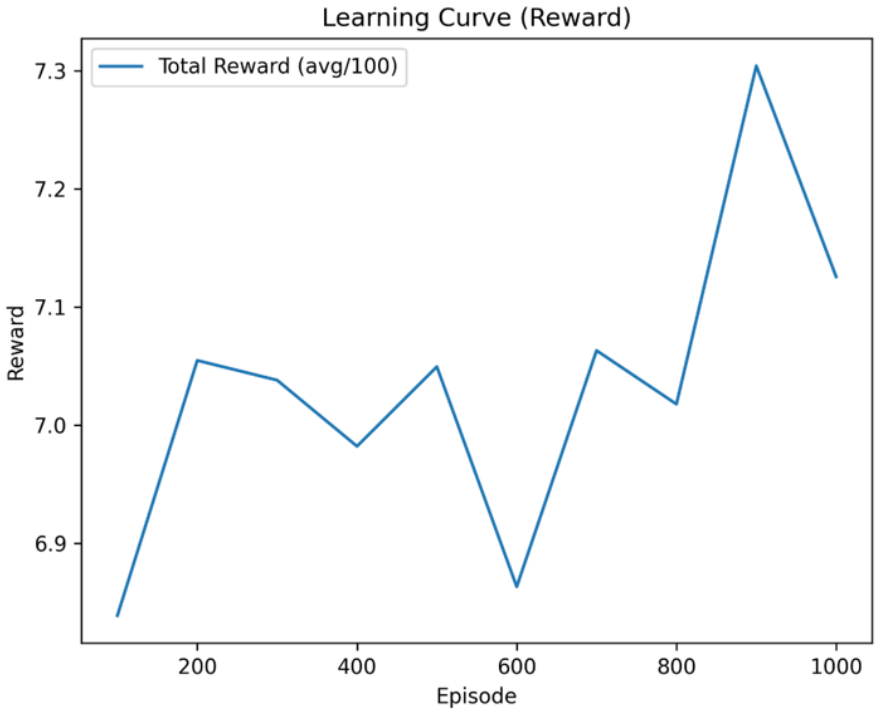

Figure 23 - DQN 1000 episodes learning curve

The DQN 1000 episodes training curve seems irregular and exhibits strong fluctuations. While rewards oscillate around the 7.0 mark, there is no clear upward trend that would indicate

consistent policy improvement. Occasional peaks, such as near episode 900, suggest that the agent sometimes discovers better strategies, but these gains are not stably maintained.

This unstable behavior is characteristic of value-based methods like DQN when applied to complex environments such as the 3D-BPP. The reliance on Q-value estimation makes the agent sensitive to exploration choices and reward shaping. Despite this, the agent manages to maintain performance slightly above the baseline reward, showing that it is learning some valid placement strategies.

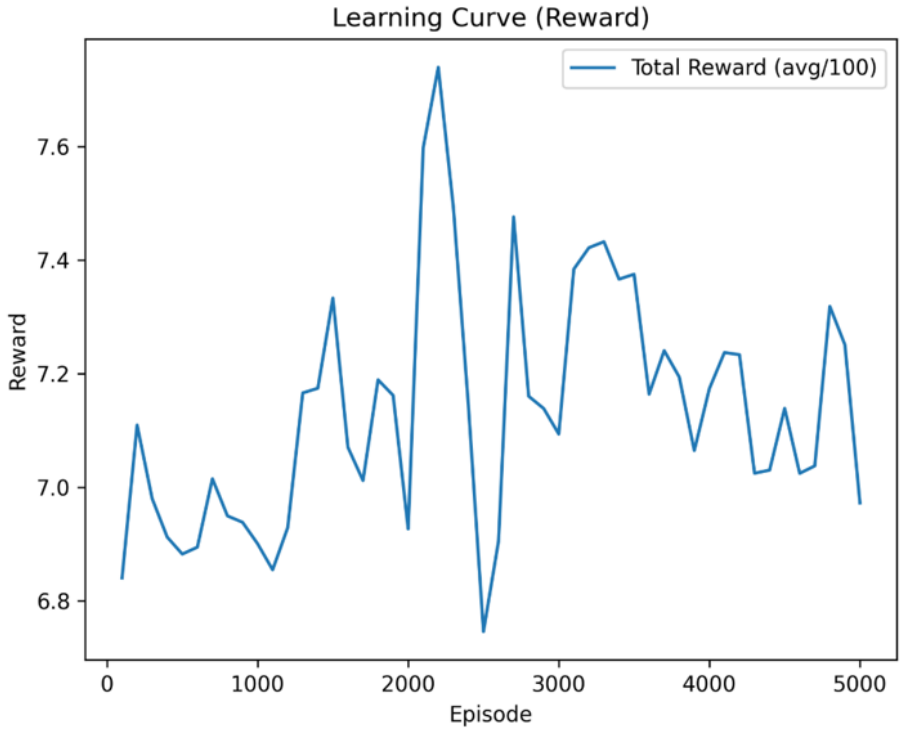

Figure 24 - DQN 5000 episodes learning curve

When extending the training of the DQN agent to 5000 episodes, the learning curve shows higher variability compared to the 1000-episode run. While there are peaks of improved performance around episodes 2000–2500, with rewards reaching above 7.6, these gains are not sustained, and the curve fluctuates considerably afterwards. The average reward stabilizes at a level only slightly higher than the 1000-episode case, suggesting that prolonged training did not translate into consistent improvements. This instability is a typical limitation of value-based methods such as DQN when applied to complex environments like 3D-BPP, where exploration and credit assignment are challenging.

Figure 25 presents the learning curve of the **PPO agent trained for 1000 episodes**.

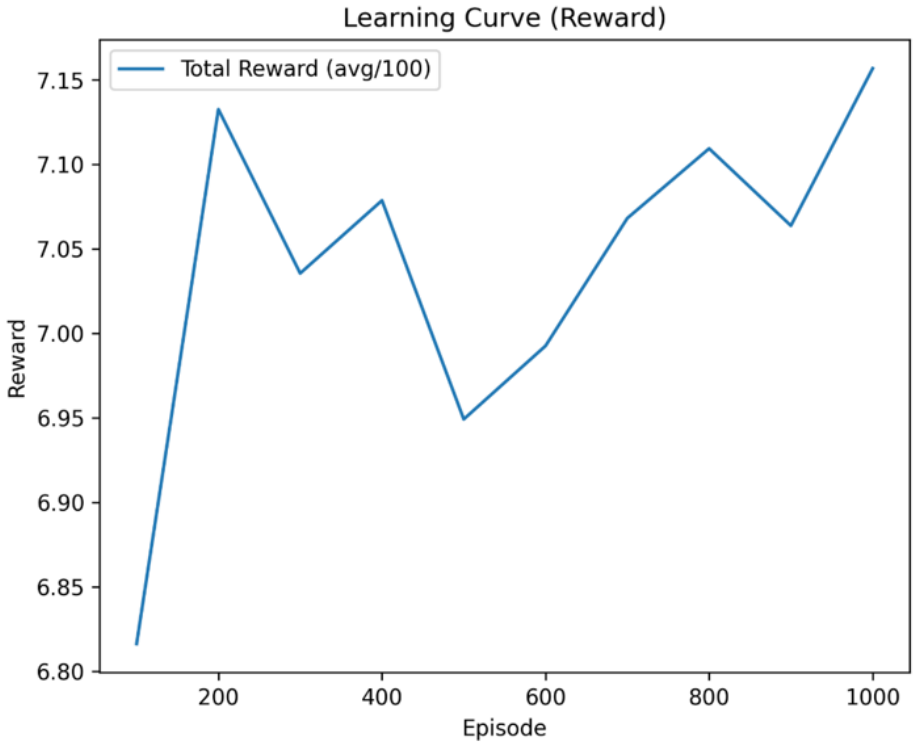

Figure 25 - PPO 1000 episodes learning curve

The PPO agent shows a steady and stable training process. After an initial rise in reward within the first 200 episodes, the curve maintains a consistent level slightly above 7.0. This indicates that the agent was able to learn meaningful allocation strategies relatively quickly, without major oscillations or instability.

Unlike the DQN curves, which exhibit higher variability and slower improvement, PPO shows more robustness in converging towards a stable policy. The stability of the curve suggests that the clipped objective of PPO helped prevent large updates and contributed to a smoother training process.

Still, the curve shows limited upward growth after convergence, suggesting that while PPO outperforms DQN in stability and consistency, additional training steps or tuned hyperparameters might be required to push performance beyond this plateau.

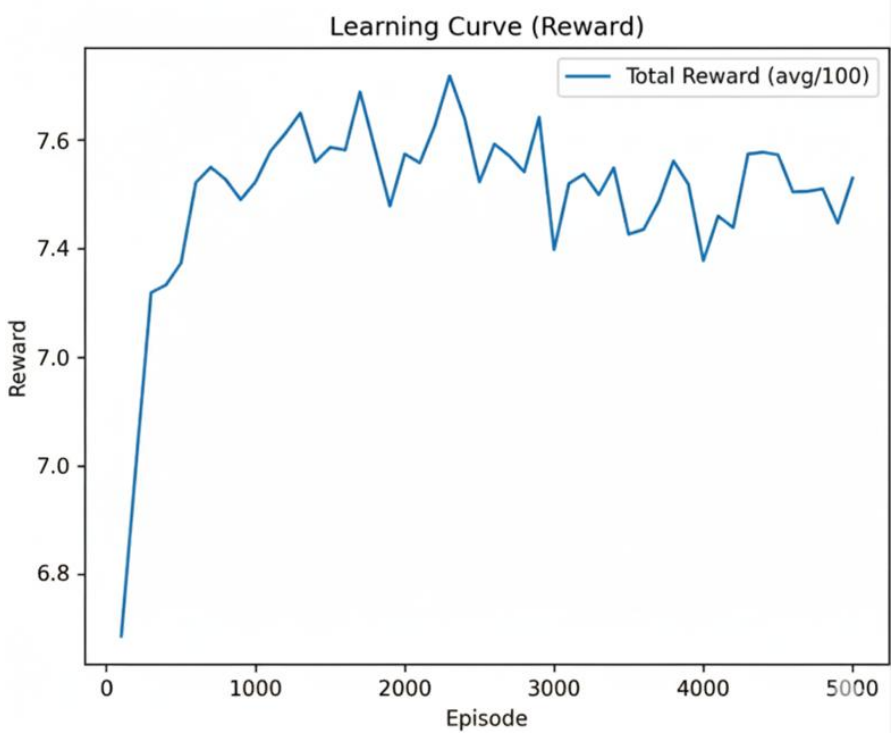

Figure 26 - PPO 5000 episodes learning curve

The learning curve for the **PPO agent trained over 5000 episodes shows a clear improvement in stability and overall reward** compared to the shorter run. After an initial steep increase in the first few hundred episodes, the curve stabilizes around an average reward between 7.4 and 7.6. While some oscillations remain throughout training, the variance is relatively contained, indicating that the agent has learned consistent placement strategies. Importantly, the PPO agent maintains a reward level above that of the DQN agents, reflecting its stronger ability to generalize across different packing scenarios. The plateau in later episodes suggests that the policy converged, with only minor fluctuations due to the stochastic nature of the environment and exploration.

5.8.2 Evaluation

After training, all agents and the baseline heuristic were tested using the same test set of boxes (each test generated a different set of boxes), resulting in a total of **20 evaluation runs per method to evaluate the bin volume's percentage utilization**. This setup ensures fairness in the comparison by exposing each approach to identical packing scenarios.

The results of the 20 test runs will be summarized in a comparative table, allowing a clear visualization of performance differences across the heuristic and RL approaches.

The DQN agent shown below was trained with softmax exploration since the train with epsilon-greedy exploration did not achieve satisfactory results, with bin utilization consistently below 70%.

Table 23 - Bin volume percentage evaluation results

Test	DQN 1000 ep	DQN 5000 ep	PPO 1000 ep	PPO 5000 ep	BLB Heuristic
1	73.20%	68.00%	65.00%	81.00%	76.30%
2	81.50%	63.60%	86.70%	82.30%	86.90%
3	78.20%	81.50%	85.30%	82.70%	78.50%
4	76.50%	82.70%	92.10%	94.00%	91.90%
5	89.90%	89.00%	83.70%	90.40%	90.50%
6	80.70%	88.00%	84.00%	91.20%	80.50%
7	77.10%	73.00%	77.90%	85.80%	81.60%
8	82.30%	83.30%	90.60%	89.10%	93.00%
9	81.10%	74.20%	93.80%	95.30%	90.90%
10	70.80%	69.80%	83.00%	88.50%	87.80%
11	75.70%	79.00%	83.10%	88.80%	82.60%
12	87.80%	74.50%	90.90%	90.10%	89.00%
13	70.80%	70.00%	82.00%	73.80%	84.60%
14	77.60%	76.90%	73.10%	70.40%	75.60%
15	78.50%	74.50%	97.90%	83.10%	82.80%
16	80.30%	76.00%	87.70%	84.40%	83.20%
17	75.40%	74.30%	83.60%	83.40%	74.50%
18	87.80%	63.10%	87.20%	85.40%	76.50%
19	70.70%	72.10%	92.70%	83.40%	81.60%
20	79.00%	75.90%	92.20%	87.80%	88.20%

On average across the 20 test runs, the agents and the heuristic achieved the following bin volume utilization percentages:

- **DQN (1000 steps)**: 78.75%
- **DQN (5000 steps)**: 75.47%
- **PPO (1000 steps)**: 85.63%
- **PPO (5000 steps)**: 85.55%
- **BLB Heuristic**: 83.82%

These results highlight that the PPO agent clearly outperformed both versions of the DQN agent and even surpassed the BLB heuristic, achieving the best average bin utilization. The DQN agent, while able to learn viable strategies, remained below the heuristic baseline in both configurations.

6 Conclusions

In this chapter, an overview of the study developed, and the objectives achieved are presented. Following the study results, the limitations encountered, and future work suggestions are presented.

6.1 Summary and Objectives Achieved

The main objective of this work was to design and implement a complete pipeline capable of addressing two interconnected warehouse management challenges: **time series forecasting of product outflows** and **optimal allocation of boxes in the 3D-BPP**. By integrating predictive modeling with RL, the proposed approach provides both a forward-looking view of warehouse demand and an adaptive method for space optimization.

For the forecasting task, two machine learning models were implemented — **XGBoost regression** and **LSTM networks**. In addition, the Prophet model was included as an external benchmark to contrast its results with the implemented methods. The evaluation across different brand–category combinations demonstrated the ability of XGBoost and LSTM to capture temporal dynamics while highlighting the trade-offs between accuracy, interpretability, and computational cost.

For the allocation task, a custom RL environment was implemented to simulate realistic palletization scenarios. The environment incorporated carefully designed reward functions, discrete action spaces, and synthetic datasets generated through recursive splitting. Two RL agents — **DQN** and **PPO** — were trained and evaluated against a heuristic baseline. The results highlighted the advantages of PPO in learning stable packing strategies, achieving performance levels comparable to or surpassing the heuristic approach.

Overall, the research objectives were successfully achieved:

- A reproducible forecasting pipeline was developed and tested with real warehouse data.
- An RL-based allocation environment was designed and implemented, with support for multiple agents and evaluation protocols.
- Extensive experiments validated the effectiveness of the proposed methods, highlighting both the potential and the limitations of combining forecasting with allocation for warehouse planning.

6.2 Limitations and Future Work

Although this work achieved its objectives, several limitations were identified throughout the development and experimentation process. These limitations also open opportunities for future improvements and extensions.

In the forecasting task, one of the main challenges lies in predicting the demand for new products with little or no historical data. Since new items are constantly introduced into the warehouse, traditional time series models struggle to provide accurate forecasts in these scenarios. Future research could address this by exploring transfer learning, similarity-based forecasting methods, or hybrid approaches that combine statistical and ML techniques. In addition, the LSTM network implementation can be further improved through the exploration of alternative architectures such as GRUs or Transformers, as well as more advanced training strategies, including attention mechanisms and regularization, to better capture long-term dependencies. Another limitation concerns the way categorical features were encoded. While this work relied mainly on detailed product-level categories, future studies could experiment with more general encodings to reduce sparsity and improve generalization. A promising direction is the cross-encoding of similar categories and brands, allowing the model to leverage relationships across products and mitigate issues caused by short product life cycles. Finally, while RMSE and MAE were used to evaluate performance, future research could integrate probabilistic forecasting metrics to better capture the uncertainty of demand predictions.

For the 3D-BPP, a key limitation is that the DQN and PPO agents did not converge to globally optimal solutions. Improving convergence stability remains a challenge, and may benefit from the exploration of alternative RL algorithms such as A3C, SAC, or hybrid value-policy approaches. Another limitation is the reliance on manually defined hyperparameters. Automated hyperparameter optimization techniques, such as Bayesian optimization or population-based training, could help improve training stability and final performance. Reward design also requires refinement. While this work tested compactness and utilization rewards, future work could incorporate penalties for misplaced boxes and stronger incentives for stability, along with adaptive weighting schemes that evolve throughout training. The state representation is another aspect that could be improved: replacing handcrafted heightmaps with 3D voxel grids or convolutional encoders may provide agents with a more expressive understanding of spatial relationships. Furthermore, the experiments were limited to synthetic datasets generated

through recursive splits. Although these datasets ensured variability and realism, future work should integrate real warehouse data to validate the models under operational conditions. Another open challenge is computational efficiency, as training RL agents for the 3D-BPP is resource-intensive. Parallelized training or the use of simplified surrogate environments could significantly reduce experimentation time and make the approach more practical.

Looking ahead, one of the most promising directions is the integration of both tasks into a single system. A full pipeline where the time series model forecasts daily product outflows and the RL agent allocates the predicted quantities into warehouse space would close the loop between forecasting and allocation. This integration would not only strengthen the realism of the approach but also provide a practical decision-support tool for daily warehouse operations.

In summary, while this work provides a solid foundation for combining forecasting and allocation in warehouse planning, addressing these limitations will further enhance accuracy, efficiency, and applicability in real-world scenarios.

6.3 Final Remarks

This thesis explored two complementary challenges in warehouse management: forecasting daily product outflows and optimizing the allocation of goods through the 3D-BPP. By approaching forecasting with ML and deep learning models, and allocation with RL agents, the work demonstrated how artificial intelligence can support both strategic planning and operational decision-making in logistics.

The results confirmed that traditional ML models, such as XGBoost, are highly effective for forecasting in data-rich settings, while neural network approaches like LSTM offer potential for capturing more complex temporal dependencies. In parallel, the development of a custom RL environment enabled the training and evaluation of DQN and PPO agents, showing that policy-gradient methods in particular are capable of learning stable allocation strategies that compete with or surpass heuristic baselines.

Most importantly, the research laid the groundwork for an integrated system in which forecasting models inform RL agents, enabling warehouse managers to not only anticipate demand but also allocate space optimally based on these forecasts. Such a system would represent a meaningful step toward intelligent, data-driven warehouse operations, where decisions are continuously adapted to both expected and unexpected dynamics of supply and demand.

In conclusion, while challenges remain in terms of model generalization, training stability, and computational efficiency, this work highlights the significant potential of combining time series forecasting with RL to create robust solutions for warehouse planning. It provides a foundation upon which future research can build, moving closer to practical, AI-driven decision-support systems for real-world logistics.

References

Wu, W., He, L., & Yang, J. (2012). Evaluating recommender systems. 7th International Conference on Digital Information Management, ICDIM 2012, 56–61. https://doi.org/10.1109/ICDIM.2012.6360092 Hernández del Olmo, F., & Gaudioso, E. (2008b). Evaluation of recommender systems: A new approach. Expert Systems with Applications, 35(3), 790–804.

Anitha, L., Devi, M. K. K., & Devi, P. A. (2013). A Review on Recommender System. In International Journal of Computer Applications (Vol. 82).

Han, J., Liang, R., Yao, H., & Yao, H. (2023). Intelligent Warehousing based on numerical heuristic planning. In *2023 4th International Conference on Electronic Communication and Artificial Intelligence (ICECAI)* (pp. 353-357). IEEE.

Priyanga, P., Sridevi, S., Ashwini, K., & Deepa, S. R. (2023). The Smart Factory of Tomorrow: Artificial Intelligence and Machine Learning Reshaping Manufacturing Processes. In *2023 Second International Conference On Smart Technologies For Smart Nation (SmartTechCon)* (pp. 1477-1481). IEEE. Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele University*, *33*(2004), 1-26.

Zama, M., Jotawar, A., Du, J., Anik, F. I., Anjum, N., Saha, B., ... & Shahriar, H. (2022). SPOT (Sales Production based On Time-Series): A Comprehensive Approach to Sales Forecasting using Contextually- tailored Time Series Analysis. In *2022 4th International Conference on Sustainable Technologies for Industry 4.0 (STI)* (pp. 1-6). IEEE.

Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., ... & Moher, D. (2021). The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. bmj, 372. Storey, V. C., Robinson, J. M., & Baskerville, R. L. (n.d.). Computational Science: A Field of Inquiry for Design Science Research. https://hdl.handle.net/10125/80043

Au, Y. A. (2001). Design Science I: The Role of Design Science in Electronic Commerce Research. Communications of the Association for Information Systems, 7. https://doi.org/10.17705/1cais.00701 Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. Journal of Management Information Systems, 24(3), 45–77. https://doi.org/10.2753/MIS0742-1222240302

Babar, M. M., Syed, A. J., Asim, S., Khan, A. R., & Hai, M. A. (2022). Development of an Algorithm based on the Optimization of space, time and resources of the warehouse using Genetic Algorithms. In *2022 3rd International Conference on Innovations in Computer Science & Software Engineering (ICONICS)* (pp. 1-8). IEEE.

Qi, Y., Li, C., Deng, H., Cai, M., Qi, Y., & Deng, Y. (2019). A deep neural framework for sales forecasting in e-commerce. In *Proceedings of the 28th ACM international conference on information and knowledge management* (pp. 299-308).

Singh, B., Kumar, P., Sharma, N., & Sharma, K. P. (2020). Sales forecast for amazon sales with time series modeling. In *2020 first international conference on power, control and computing technologies (ICPC2T)* (pp. 38-43). IEEE.

Lian, J., & Li, L. (2018). Predictive analysis of e-commerce enterprises soft operating costs based on exponential smoothing technique. In *2018 9th International Conference on Information Technology in Medicine and Education (ITME)* (pp. 911-915). IEEE.

Davenport, T. H., & Kirby, J. (2019). Artificial intelligence and the future of work. Technology Review, 122(5), 26-31.

Alsaidi, M., & Alhindi, A. (2023). Time Series Forecasting Model for E-commerce Store Sales Using FB- Prophet. In *2023 14th International Conference on Information and Communication Systems (ICICS)* (pp. 1-6). IEEE.

KPMG (2020). Artificial intelligence in the supply chain: A new frontier. Retrieved from: https://kpmg.com/us/en/articles/2023/supply-chaingenerative-ai-potential.html.

Kalkha, H., Khiat, A., Bahnasse, A., & Ouajji, H. (2024). Enhancing Warehouse Efficiency with Time Series Clustering: A Hybrid Storage Location Assignment Strategy. *IEEE Access*.

Agbemadon, K. B., Couturier, R., & Laiymani, D. (2023). Overstock Prediction Using Machine Learning in Retail Industry. In *2023 3rd International Conference on Computer, Control and Robotics (ICCCR)* (pp. 439-444). IEEE.

Tamias, R., Setianingsih, C., Irawan, B., Dityantomo, S. R., Putra, R. A., Wulandari, R. S., ... & Ruriawan,M. F. (2021). Particle Swarm Optimization Algorithm for Optimizing Item Arrangements in Storage Warehouse. In *2021 3rd International Conference on Electronics Representation and Algorithm (ICERA)* (pp. 167-172). IEEE.
Bhatia, G., Khole, R., Sawant, S., & Vazirani, S. (2021). Entrepôt Optimisé-A Storage Optimizer. In *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)* (pp. 01-05). IEEE.
Deloitte (2018). The future of retail: How artificial intelligence is transforming the industry. Retrieved from: https://www2.deloitte.com/us/en/pages/consulting/articles/future-ofretail-stores.html
Pan, H., Li, M., & Yang, C. (2024). Research on Logistics Network Optimization Based on Artificial Intelligence. In *2024 Third International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)* (pp. 1-6). IEEE.
Zunic, E., Hasic, H., Hodzic, K., Delalic, S., & Besirevic, A. (2018). Predictive analysis based approach for optimal warehouse product positioning, in 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2018-Proceedings.
Tang, T. (2023, May). Analysis and Demand Forecasting Based On e-Commerce Data. In *2023 6th International Conference on Artificial Intelligence and Big Data (ICAIBD)* (pp. 64-68). IEEE. Wamba, S. F., Akter, I. H., Uddin, M. A., & Gunasekaran, M. (2020). How artificial intelligence can improve supply chain management? Journal of Business Research, 123, 239-254.
Hanson Robotics. (2016). Sophia. https://www.hansonrobotics.com/sophia/
OpenAI. (2022). ChatGPT: Optimizing Language Models for Dialogue. https://openai.com/blog/chatgpt/ Siau, K., & Wang, W. (2020). Artificial Intelligence (AI) Ethics. Journal of Database Management, 31(2), 74–87. https://doi.org/10.4018/JDM.2020040105
Hagendorff, T. (2020). The Ethics of AI Ethics: An Evaluation of Guidelines. Minds and Machines, 30(1), 99–120. https://doi.org/10.1007/s11023-020-09517-8
Lavanchy, M. (2018). Amazon's sexist hiring algorithm could still be better than a human. https://www.imd.org/research-knowledge/articles/amazons-sexist-hiring-algorithmcould-still-be- better-than-a-human/
Vincent, J. (2016). Twitter taught Microsoft's AI chatbot to be a racist asshole in less than a day. https://www.theverge.com/2016/3/24/11297050/tay-microsoft-chatbot-racist European Commission (2019). Ethics guidelines for trustworthy AI. https://digital- strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai
Bertino, E., Kantarcioglu, M., Akcora, C. G., Samtani, S., Mittal, S., & Gupta, M. (2021). AI for Security and Security for AI. Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy, 333–334. https://doi.org/10.1145/3422337.3450357
Dilmaghani, S., Brust, M. R., Danoy, G., Cassagnes, N., Pecero, J., & Bouvry, P. (2019). Privacy and Security of Big Data in AI Systems: A Research and Standards Perspective. 2019 IEEE International Conference on Big Data (Big Data), 5737–5743. https://doi.org/10.1109/BigData47090.2019.9006283 Meurisch, C., & Mühlhäuser, M. (2022). Data Protection in AI Services. ACM Computing Surveys, 54(2), 1–38. https://doi.org/10.1145/3440754
Fabiano, N. (2019). Robotics, intelligent systems, ethics and data protection. Proceedings of the 2nd International Conference on Applications of Intelligent Systems, 1–5. https://doi.org/10.1145/3309772.3309787
Liu, X., Xie, L., Wang, Y., Zou, J., Xiong, J., Ying, Z., & Vasilakos, A. V. (2021). Privacy and Security Issues in Deep Learning: A Survey. IEEE Access, 9, 4566–4593. https://doi.org/10.1109/ACCESS.2020.3045078 Bae, H., Jang, J., Jung, D., Jang, H., Ha, H., Lee, H., & Yoon, S. (2018). Security and Privacy Issues in Deep Learning.
Chen, H., & Babar, M. A. (2022). Security for Machine Learning-based Software Systems: a survey of threats, practices and challenges.
Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Zheng, X. (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. ArXiv.
Steinhardt, J., Koh, P. W., & Liang, P. (2017). Certified Defenses for Data Poisoning Attacks.
Sen, A., Mazumder, A. R., Dutta, D., Sen, U., Syam, P., & Dhar, S. (2023). Comparative evaluation of metaheuristic algorithms for hyperparameter selection in short-term weather forecasting.
Nguyen, Q., Teku, N., & Bose, T. (2021, September). Epsilon greedy strategy for hyper parameters tuning of a neural network equalizer. In *2021 12th International Symposium on Image and Signal Processing and Analysis (ISPA)* (pp. 209-212). IEEE

Huang, Y., Lai, L., Li, W., & Wang, H. (2022). A differential evolution algorithm with ternary search tree for solving the three-dimensional packing problem. *Information Sciences*, *606*, 440-452.
Martello, S., Pisinger, D., & Vigo, D. (2000). The three-dimensional bin packing problem. *Operations research*, *48*(2), 256-267.
Bortfeldt, A., & Wäscher, G. (2013). Constraints in container loading–a state-of-the-art review. *European Journal of Operational Research*, *229*(1), 1-20.
Li, X., & Zhang, K. (2015). A hybrid differential evolution algorithm for multiple container loading problem with heterogeneous containers. *Computers & Industrial Engineering*, *90*, 305-313.
Zhao, H., She, Q., Zhu, C., Yang, Y., & Xu, K. (2021, May). Online 3D bin packing with constrained deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 35, No. 1, pp. 741-749).
Kundu, O., Dutta, S., & Kumar, S. (2019, October). Deep-pack: A vision-based 2d online bin packing algorithm with deep reinforcement learning. In *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)* (pp. 1-7). IEEE.
Bo, A., Lu, J., & Zhao, C. (2022, November). Deep Reinforcement Learning in POMDPs for 3-D palletization problem. In *2022 China Automation Congress (CAC)* (pp. 577-582). IEEE.
Chien, S. Y., & Wong, C. C. (2023, August). Online 3D Bin Packing for Novel Objects Based on Deep Reinforcement Learning. In *2023 International Conference on Advanced Robotics and Intelligent Systems (ARIS)* (pp. 1-4). IEEE.
Jiang, Y., Cao, Z., & Zhang, J. (2021). Solving 3D bin packing problem via multimodal deep reinforcement learning.
Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (Vol. 1, No. 1, pp. 9-11). Cambridge: MIT press.
Garciar, E. (2022). https://github.com/luisgarciar/3D-bin-packing
Zhao, H. (2021). https://github.com/alexfrom0815/Online-3D-BPP-DRL