Universidade de Aveiro
2022

**Marco Paulo
Caloba**

Decision support system in logistical processes management: Primus Vitória case study

Sistema de apoio à tomada de decisão na gestão dos processos logísticos: caso de estudo Primus Vitória

Universidade de Aveiro
2022

Marco Paulo Caloba

Decision support system in logistical processes management: Primus Vitória case study

Sistema de apoio à tomada de decisão na gestão dos processos logísticos: caso de estudo Primus Vitória

Relatório de Projeto apresentado à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica, realizada sob orientação científica de José Paulo Oliveira Santos, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro, de Ana Maria Pinto de Moura, Professora Auxiliar do Departamento de Economia, Gestão, Engenharia Industrial e Turismo da Universidade de Aveiro, e ainda de Paulo Almeida, resposável da Primus Vitória.

Esta dissertação teve o apoio dos projetos UIDB/00481/2020 e UIDP/00481/2020 - Fundação para a Ciência e a Tecnologia; e CENTRO-01-0145 FEDER-022083 - Programa Operacional Regional do Centro (Centro2020), através do Portugal 2020 e do Fundo Europeu de Desenvolvimento Regional.

agradecimentos / acknowledgements

Em primeiro lugar, um enorme obrigado aos meus pais, Carlos e Lúcia, por todos os esforços que fizeram para que eu tivesse a oportunidade de seguir este percurso académico até ao fim. Desde apoio psicológico e emocional até ao ubereats pessoal e à roupa lavada, tenha sido uma semana de exames, de enterro ou apenas de aulas. Por vocês, esforço-me para ser a minha melhor versão para que se possam orgulhar do homem que criaram.
À minha namorada, Elza, por há tantos anos seres o meu principal alicerce, nos bons e nos maus momentos. O tempo não parece traduzir o que já vivemos juntos e o quanto me ajudaste a crescer. Desde uma carta entregue à meia-noite até à conclusão do meu grau de mestre, vivi mais do que alguma vez pensei viver e a ti te devo isso, obrigado. Por ti, e pelo nosso futuro, continuo a dar o meu máximo todos os dias.
Ao meu irmão, Cláudio, pelo exemplo que és e que sempre mostraste ser, desempenhando na perfeição o papel de irmão mais velho. Devo-te muito pela pessoa que sou hoje e espero conseguir retribuir de alguma forma tudo o que já fizeste por mim. Obrigado por seres meu cúmplice a mostrar ao mundo que irmãos podem ser melhores amigos.
Ao André, ao Gonçalo, ao Joel e ao Miguel, porque sem vocês este curso não teria chegado ao fim. Foram centenas de horas passadas em grupo com reuniões de trabalho até o sol nascer, almoços de lancheira e convívios de caneca na mão a discutir sobre tudo o que interessa ao ser humano discutir. Devo-vos a todos um especial obrigado porque, só por vocês, este curso valeu a pena. Desejo do fundo do coração poder acompanhar-vos para o resto da vida e participar o máximo possível nelas.
Ao meu orientador, professor José Paulo Santos, um agradecimento pela oportunidade de realizar este projeto, e à minha co-orientadora, professora Ana Moura, um sentimento de gratidão por toda a dedicação e disponibilidade que me deu durante este semestre. A sua ajuda foi imprescindível e dificilmente o teria terminado sem ela, obrigado.
Gostaria também de agradecer à Primus Vitória por se terem mostrado sempre abertos a qualquer tipo de ajuda, e em particular ao Dr. Paulo Almeida, pelas horas que despendeu connosco.
Por último, deixo apenas um agradecimento a todos os familiares e amigos que de alguma maneira contribuíram para a pessoa que sou hoje.

keywords *DSS, Warehouse, Inventory management, ABC, Knapsack*

abstract

Managing a warehouse can involve a large flow of production which results in large amounts of volume to be stored. Being this the company's case, the main problem found was that there is no system for the phase of storing the products. The decision of where to store all the final products is made only by an employee which leaves no way to evaluate if the storing is done optimally and it is a very big risk for a company of it's size to have so much dependency on one person. With this, the objectives established for this project were the reorganisation of the company's external warehouse and developing a model for a decision support system, serving as a pilot project for later application to their internal warehouse. The framework designed and proposed meets the company's current needs. From this framework, are presented analyses and algorithms to classify the products according to the customer orders and to store them in the warehouse using two different methods: a direct allocation through the ABC classification method and an allocation using the Knapsack problem, an optimisation method. Some important notes on how to continue the work started here are also presented.

palavras-chave *SAD, Armazém, Gestão de inventário, ABC, Knapsack*

resumo

A gestão de um armazém pode envolver um grande fluxo de produção, o que resulta em grandes quantidades de volume para serem armazenadas. Sendo este o caso da empresa, o principal problema encontrado foi a inexistência de um sistema para a fase de armazenamento dos produtos. A decisão de onde armazenar todos os produtos finais é tomada apenas por um funcionário, não permitindo uma avaliação sobre como o armazenamento é feito e é um risco muito grande para uma empresa da sua dimensão ter tanta dependência em apenas uma pessoa. Com isto, os objetivos estabelecidos para este projeto foram a reorganização do armazém externo da empresa e o desenvolvimento de um modelo de sistema de apoio à decisão, servindo como projeto piloto para posterior aplicação ao armazém interno. A estrutura desenvolvida e proposta atende às necessidades atuais da empresa e, desta, são apresentadas análises e algoritmos para classificar os produtos de acordo com os pedidos dos clientes e para armazená-los usando dois métodos diferentes: uma alocação direta através do método de classificação ABC e uma alocação usando o problema da mochila, um método de otimização. Também são apresentados algumas notas importantes sobre como dar continuidade ao trabalho aqui iniciado.

Contents

List of Tables

Intentionally blank page.

List of Figures

Chapter 1

Introduction

Nowadays, it is possible to extract vast volumes of multidimensional data from all elements of a company, thanks to the expanding reach of contemporary business. While having so much data is excellent, organising and gleaning insights from the data extractions can take time, let alone moving forward with decision-making. A decision support system (DSS) for a data warehouse can assist in making sense of the data in time to act.

1.1 Background

Over the last few years, Primus Vitória has been working to optimise its entire production line as much as possible, mainly through automation, already involving several partnerships with projects and thesis by students at the University of Aveiro. To this end, they have been working on increasing their database with the most relevant statistics from the different stages of the production line to obtain a more qualitative view of the type of work they do and be able to intervene more quickly and assertively when necessary. However, the picking and storage of products have yet to be automated and optimised. As of today, a specific employee is in charge of establishing the section to store the products. With this new project, the company wants to face this new field, thus giving due importance to its warehouse and all its processes.

1.2 Objectives

The first meetings had with the company were to get to know better their situation, how they worked and what they were looking to improve with this project. Having established that a study of the warehouse was to be made, the company expected a layout proposal to compare to their product allocation method to determine if they were allocating the products properly and how much they would benefit from a DSS to help manage this task.

With this, the objectives established for this project were the reorganisation of the company's external warehouse and developing a model for a decision support system, serving as a pilot project for later application to their internal warehouse.

1.3 Methodology

This project's methodology consisted of five main steps. The first and longest step was the literature review, which started by searching decision support systems for warehouse management. Since there were not many useful academic findings, the search was expanded for all types of DSSs, intending to understand how these are usually structured and what kind of tools they use to try later to frame our system. While studying DSSs, papers that focused on warehouse analyses were also searched to establish the appropriate models to apply to the company's warehouse, including papers about layouts, classification methods and problem-solving models. For the second step, some product classification methods studied were applied, the PQR, FSN and ABC, and evaluated if these were right for the project's purposes. In the third step, the Knapsack problem was used to optimise the allocation of the products to the warehouse sections. The fourth step entailed the creation of algorithms that automate the second and third steps, all through python programming. The fifth and final step involved analysing the results obtained and then writing the project's report.

1.4 Structure of the document

This document is organised systematically, so the reader will learn how the final system was constructed and what ideas are required to comprehend its functioning by reading the following chapters.

The text starts with chapter 2, where the company is briefly introduced and their current situation regarding the warehouse operations, capacity and characteristics. Then, in chapter 3, the literature review, divided into two main sections, decision support systems and warehouse and storage. In the first section, the most important concepts are reviewed to understand what a DSS is, why it is important, and how to use it. In the second section, the importance of a warehouse is approached with its layout and the different types of analysis that can be made to evaluate it.

Next, in chapter 4, the case study is presented with the complete warehouse analysis through the different types of analysis and then the algorithms elaborated to automate the analyses, resulting in the final part of the chapter, which is the proposed decision support system. To finalise the document, in chapter 5, the results obtained are discussed, giving it a critical evaluation and also instructions for the continuity of this work and how it can be improved and elaborated.

Chapter 2

The company - Primus Vitória

2.1 Company introduction

Primus Vitória is a ceramics company that has been in business since 1969. The firm specialises in the production of classic tiles and offers a broad range of colours and contemporary series that are in keeping with current architectural trends. Its yearly tile output is roughly 5 million m^2, with around 75% of that going to the international market, with the French, English, and Nordic nations markets standing out in Europe, South Africa, the United States, and Canada outside of Europe. It is considered one of the most important historic Portuguese tile industrial manufacturing units.

2.2 Current situation

2.2.1 Storage procedure

Regarding storing the final products, the company has two main warehouses in the facilities we worked. The internal warehouse and older one, where they have their entire production line and the external warehouse, which they built recently to accommodate more pallets. In the internal warehouse, besides having specified zones to store complete pallets of product, they also have a designated area with shelves for the products that do not complete a pallet and are used for picking.

When the product reaches its final stage, a robot is in charge of packing and placing it in its designated pallet. Then another robot picks up the complete pallet and places it in a designated area before storing it. At this stage, an employee checks for any damages or data mistakes, and if everything is in order, the employee sets the section to store the product. This method can only be pursued with good results because the employee mentioned already has been working many years for the company and has tremendous experience when it comes to making these decisions.

Concerning the project's analyses, working with both warehouses would be a more thorough study. Nonetheless, it was decided to start only with the external warehouse and, with the external warehouse optimised and working with the DSS, it would be easier to expand it to the internal warehouse.

2.2.2 Warehouse capacity and characteristics

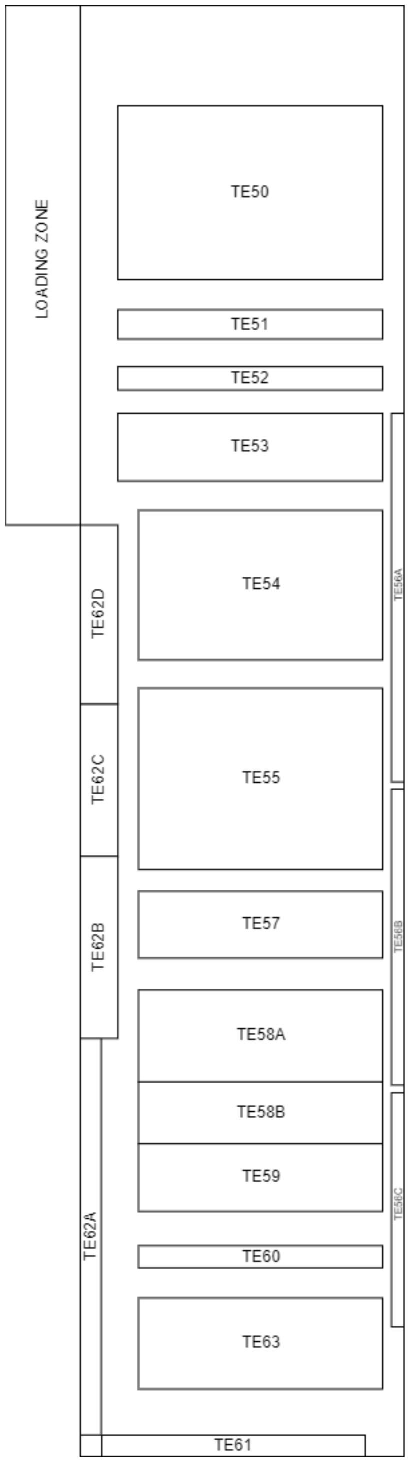


Figure 2.1: Primus Vitória's external warehouse blueprint

First, the warehouse was measured to get the areas of all sections. As shown above, in figure 2.1, the warehouse consists of a loading zone in the front left, 20 marked locations to store the product, and ample aisles for the workers to circulate with forklifts. Also, to access this warehouse, there are two openings, one at the top right, which has direct access to the end of the production line, where the product is placed before storing in either warehouse and another at the bottom right that accesses the middle of the internal warehouse storage. The measures of each section is presented in table 2.1.

Table 2.1: Primus Vitória's external warehouse measurements

Section	Length (m)	Width (m)	Floor Area (m^2)
Warehouse	173.6	38.7	6718.32
TE50	31,7	20,7	656,19
TE51	31,7	3,5	110,95
TE52	31,7	2,8	88,76
TE53	31,7	8,1	256,77
TE54	29,2	17,9	522,68
TE55	29,2	21,7	633,64
TE56A	44,1	1,5	66,15
TE56B	35,4	1,5	53,1
TE56C	28	1,5	42
TE57	29,2	8	233,6
TE58A	29,2	11	321,2
TE58B	29,2	7,4	216,08
TE59	29,2	8,1	236,52
TE60	29,2	2,5	73
TE61	31,5	2,5	78,75
TE62A	47,5	4,5	213,75
TE62B	21,8	4,5	98,1
TE62C	18,2	4,5	81,9
TE62D	21,4	4,5	96,3
TE63	29,2	10,9	318,28
		Total	4397.72

A prioritisation of the sections was also established in order to use later for the allocation of the products. This order is shown in table 2.2.

Table 2.2: Sections prioritisation

Order	Section	Order	Section
1º	TE50	11º	TE56A
2º	TE51	12º	TE56B
3º	TE52	13º	TE56C
4º	TE53	14º	TE62D
5º	TE54	15º	TE62C
6º	TE55	16º	TE62B
7º	TE57	17º	TE62A
8º	TE58A	18º	TE60
9º	TE58B	19º	TE63
10º	TE59	20º	TE61

2.2.3 Product

Ceramic powder, which makes up the tile structure, slipware, and glass, utilised in the coating, are the raw materials used in this procedure. The ceramic powder goes through the first stage of processing and pressing, whereas the other two are only added to the finished product at a later stage.

The production process starts with the raw materials mentioned above and then these go through the following operations: pressing, turning, drying, cleaning, polishing, humidification, glazing, firing, pressure test, quality control, separation, packing and storage.

All of the processes are very important for the final product but these will not be discussed because the focus of this study is the warehouse. That being said, it is important to note that the product is a very fragile, heavy product that can easily break, be damaged or hurt someone if it is not properly stored. For example, one of the safety measures the company performs in the warehouse studied, is to not store more than 5 pallets in height regarding smaller-sized products and 4 pallets regarding larger-sized products.

2.3 Problem description

After some analysis of the company, it was clear that the management of the warehouse involves a large flow of production which results in large amounts of volume to be stored. With this, the main problem found was that there is no system for the phase of storing the products. The decision of where to store all the pallets is made only by an employee who has worked at the company for many years. With her experience and knowledge, the warehouse works well because she already knows the importance and weight of each product in the company's sales. However, there is no way to check that the storing is done optimally and it is a very big risk for a company of it's size to have so much dependency on one person.

Chapter 3

Literature review

3.1 Decision support systems

The main goal of this project is to propose a framework for an inventory management decision support system to assist inventory managers in defining the parameters of inventory control rules more technically. In essence, the term "Decision Support System" (DSS) refers to any "system" that delivers meaningful information to aid in the decision-making (DM) process [1]. DSS combines human abilities with computer capabilities to enable effective data administration, reporting, analytics, modelling, and planning, as depicted in figure 3.1. This work started by reviewing the literature and analysing studies about decision support systems. We quickly became aware that there is a lack of a comprehensive approach to decision support systems for the warehouse management area. Some studies, like [2], [3] and [4], provide tailored decision support systems for inventory management, but they are the product of costly consultant work. This absence emphasises the need for a DSS to collect data from real-world scenarios and create useful heuristics to support warehouse design and management decision-making. The following subsections will present the most relevant information through the state-of-the-art phase.

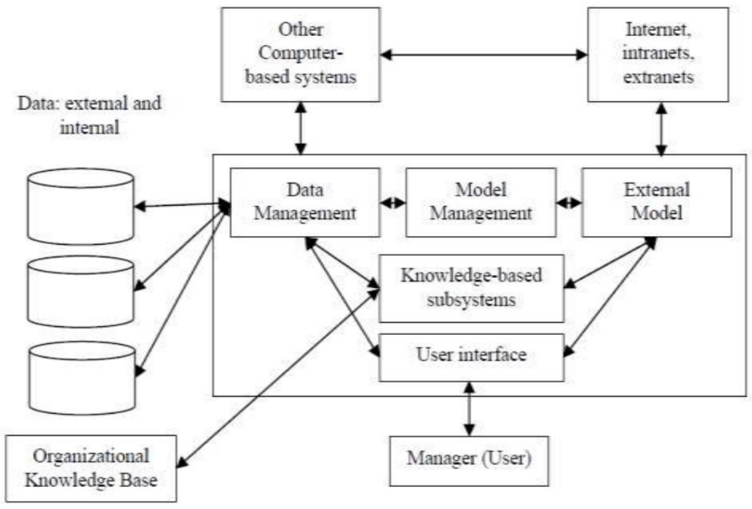

Other Computer-based systems
Internet, intranets, extranets
Data: external and internal
Data Management
Model Management
External Model
Knowledge-based subsystems
User interface
Organizational Knowledge Base
Manager (User)

Figure 3.1: Schematic view of a DSS by [5]

3.1.1 DSS characteristics

A decision support system (DSS) is an interactive, flexible, and adaptable computer-based information system that uses decision rules, models, and model bases, as well as a comprehensive database and the decision maker's insights, to produce specific, implementable decisions in solving problems that management science models would not be able to solve. As a result, a DSS aids complicated decision-making and improves its efficiency. According to [5], the following are some of the DSS's main advantages:

- Handle large volumes of data like database searches;
- Obtain and process data from several sources including internal and external data housed on mainframe systems and networks;
- Provide report and presentation flexibility to suit the decision maker's needs;
- Have both textual and graphical orientation like charts, trend lines, tables and more;
- Perform complex, sophisticated analysis and comparisons using advanced software packages;
- Support optimisation and heuristic approaches giving the decision maker a great deal of flexibility in solving simple and complex problems;
- Perform "what-if" and goal-seeking analysis.

According to [6], the three major components of a DSS are:

1. **The input databases and parameters**: It has all of the fundamental facts needed to make a decision. This might be a PC-based database extract tailored to the problem, a data warehouse storing a company's transactions, or distributed databases accessible via the internet.
2. **The analytical tools**: Operations research and artificial intelligence-based algorithms, cost calculators, simulation, flow analysis and other embedded logic procedures.
3. **The presentation mechanism**: Frequently, the output provides too much information, such as lists and tables, which the decision-maker may find challenging to comprehend. As a result, numerous data visualisation techniques help the user understand the massive amount of output data.

3.1.2 Types of DSS

As expected, there are many different DSS, but [7] established five categories that contain all of them. They are:

1. **Communication-driven DSS**: Most communication-driven DSSs are aimed at internal teams, which may include partners. Its goal is to assist in the conduct of a meeting or to allow users to collaborate. A web or client server is the most standard technology for deploying the DSS. Chat and instant messaging software, as well as online collaboration and net-meeting systems, are some examples.

2. **Data-driven DSS**: The majority of data-driven DSSs are aimed at management, employees, and product/service providers. It is used to ask particular questions of a database or data warehouse to find precise responses for specific objectives. It may be installed on a mainframe system, a client/server network, or the internet. Examples are computer-based databases with a query mechanism to check (including data incorporation to add value to existing databases).

3. **Document-driven DSS**: These are increasingly frequent and aim at a wide range of users. The goal of a DSS like this is to search web pages and locate documents based on a collection of keywords or search queries. The web and a client/server system are the most common technologies for building such DSSs.

4. **Knowledge-driven DSS**: Often known as "knowledgebases," are a catch-all term that encompasses a wide range of systems that include individuals within the company who put them up, as well as outsiders that interact with the organisation, such as customers. It is mainly used to offer managerial recommendations or select products and services. Client/server systems, the web, or applications running on stand-alone PCs are some standard deployment technologies used to set up such systems.

5. **Model-driven DSS**: These are complicated systems that aid in the analysis of choices or the selection of options. Managers and employees of a company, as well as others who engage with the company, utilise these for various reasons, depending on how the model is built up - scheduling, decision analyses, and more. These DSSs may be installed on stand-alone PCs, client/server systems, or the web using software and hardware.

Under the classification above mentioned, our case study falls in the data-driven DSS. These have progressed from simple fact-checking to data analysis to complex analysis of substantial historical data sets. Their software is also improving to better support this diverse set of tasks. Managers may double-check data with query and reporting tools that are clear and simple to use [7]. Data-driven DSSs can be presented with the following features:

Ad hoc data filtering and retrieval: The system assists users in searching for and methodically retrieving computerised data. Filtering is commonly done using drop-down menus and frequently specified queries, and users have drill-down capabilities. Users can often adjust the amount of aggregation, ranging from the most summary to the most thorough (drill-down).

Alerts and triggers: Some systems assist users in creating email notification rules and other specified actions. Create data displays & Users may generally select from various displays, such as scatter diagrams, bar and pie charts, modify the displays interactively, animate historical data on charts or other representations, and historical playback data in a time sequence.

Data management: Users have a restricted amount of "working storage" for a data subset, although they can occasionally combine data or change data formats. Users can request modifications to master data definitions and models in some systems.

Data summarisation: Pivot tables and cross-tabulations can be seen or created by users. Users can generate custom aggregations and totals, and subtotals using calculated fields. In a table structure, a pivot table highlights chosen fields and rows of data. A pivot table allows a user to see data from multiple angles and incorporate numerous fields in the table. Users can see a slice of the data or dig down for more information from a table's summary value.

Excel integration: Many data-driven DSS allow users to extract and download data for further analysis; some even allow users to submit data for analysis in their "working storage."

Metadata creation and retrieval: Users should be able to contribute metadata to their analyses and reports and update labels and descriptive material saved as metadata on the fly. Metadata in a DSS data storage is a description of the data, and it offers context for decision support and aids users in comprehending system data. To name screen displays and construct report headings, certain information is used.

Report design, generation and storage: Tables, text, pie charts, bar charts, and other diagrams are frequently used to interactively collect, create, and display information in a professional report. Once a report format has been produced, it may be stored and reused with new data. Print, web pages, and PDF documents are common ways to communicate reports.

Statistical Analysis: Users can use descriptive statistics to summarise or characterise data, draw trend lines and "mine" the data for patterns.

View predefined data displays: Displays developed by the DSS designer are standard in data-driven DSS. A dashboard display is frequently used in a system for monitoring operational performance, and a scorecard might be part of a system for measuring long-term strategic performance. A scorecard is a table that displays performance measurements and indications such as arrows or a stoplight display. Predefined data presentations can include bar and pie charts, scatter plots, and two and three-dimensional maps.

View production reports: As part of a data-driven DSS, DSS designers can build and store predetermined, periodic reports for users to view.

When creating a data-driven DSS, the valuable data get arranged and summarised in several dimensions for fast retrieval and ad hoc analysis. These systems' main purpose is to assist managers in converting data and documents into information and knowledge, so they need to be rightfully structured.

3.1.3 DSS framework

According to [8], an inventory management process must develop policies, models, and parameters for a large number of stock holding units (SKUs). Also, decision support systems are tools that can assist managers in determining how these impact inventory levels under the company's supply chain strategy. Unfortunately, because companies have many SKUs to monitor and control, it is nearly impossible to know if everything is operating at minimum inventory levels or if they will achieve the service level the company requires unless information tools facilitate and simplify these procedures available. As a result, to develop the models and parameters for the inventory management process, managers need to have information about forecasts, inventory analysis methods like the ABC method, and relevant costs and behaviours.

Regarding models overview, [8] present one that comprises the underlying inputs, a broad description of the model and the predicted outputs, allowing businesses to more precisely specify all of the data that an inventory management model requires to increase the system's efficacy. Figure 3.2 illustrates their framework.

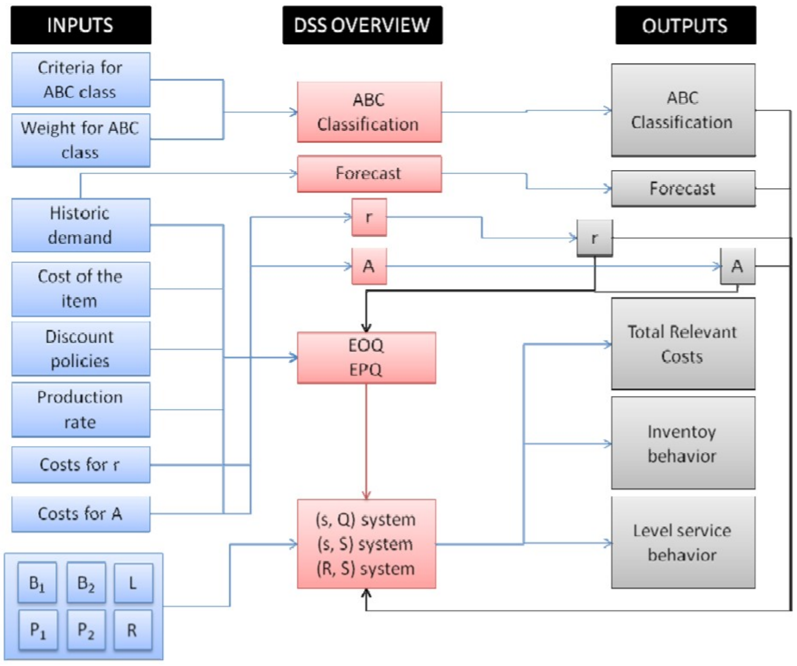


Figure 3.2: Model outline for proposed DSS, presented by [8]

As exemplified in figure 3.2, [8] state that, ideally, managers should have the following information to define the models and the parameters for the inventory management process:

- **ABC classification**: Managers can use this categorisation to prioritise their time and money resources. This classification should be used to classify inventory control models.
- **Forecast**: Inventory control models are frequently built using average demand and standard deviation rather than predicted demand. This strategy can result in excess inventory or backlogs for businesses, so different forecasting models that allow obtaining the information specified above should be included in a decision support system for inventory management.
- **Carrying costs**: This covers the money invested's opportunity cost, the costs of maintaining a warehousing and counting operation, the costs of specific storage needs, stock degradation, damage, theft, obsolescence, and taxes.

- **Ordering or setup costs**: Fixed costs that are unaffected by the quantity of replenishment and encompass all costs incurred once an order is placed.How much would the chosen inventory techniques cost, broken down into three key categories: replenishment, holding, and shortfall?

- **Total relevant cost**: How much would the chosen inventory techniques cost, broken down into three key categories: replenishment, holding, and shortfall?

- **Inventory behaviour**: The graphical behaviour of the inventory and service levels could be achieved based on the chosen models and the parameters used.

Typically, warehousing problems are non-polynomial (NP) problems and have a considerable amount of real-world data to manage. Therefore, user-friendly and timeless solutions for the warehousing issues are ambitious aims for computer-based applications, [9].

The application of this technology in a real-world scenario serves a variety of reasons. To begin with, it aids the decision-maker in making long-term strategic decisions based on the calculation of space and investment requirements for a new storage location. Second, it handles mid-term tactical decisions, such as the definition of bulk storage sections and the ability to designate multiple storage sections for distinct SKU classes. Tactical considerations also involve determining the optimal storage quantity for each SKU. Third, the DSS can help make short-term operational decisions based on SKU assignment techniques regarding creating the best-performing routing policy, [9].

3.1.4 DSS in the supply chain management

An interactive computer-based information system with an organised collection of models, people, procedures, software, databases, telecommunications and devices that assist decision-makers in solving unstructured or semi-structured business challenges is known as a decision support system. This system usually has four stages: problem input, analysis, solution, and result output.

The author of [5] states that the main benefits of a DSS are:

- Large volumes of data, such as database searches, can be handled.

- Data from various sources, including internal and external data housed on mainframe systems and networks, can be obtained and processed.

- Allows for report and presentation customisation to meet the demands of the decision-maker.

- Charts, trend lines, tables, and other graphical elements are available in textual and graphical formats.

- Using modern software applications, perform detailed, sophisticated analyses and comparisons.

- Make ”what-if” and goal-oriented analyses.

Any decision support system must perform three functions: situation evaluation, a fusion of information and generation of alternatives. With this said, according to Bozarth & Handfield (2008), the three major components of a DSS are:

1. The input databases and parameters - it has all of the essential information needed to make a decision. This could be a PC-based database extract tailored to the problem, a data warehouse storing a company's transactions or distributed databases accessible via the internet.

2. The analytical tools - operations research and artificial intelligence-based algorithms, cost calculators, simulation, flow analysis and other embedded logic procedures.

3. The presentation mechanism - frequently, the output contains too much information, such as lists and tables, which the decision-maker may find challenging to comprehend. As a result, several data visualisation techniques help the user understand the massive amount of output data. Gantt charts, for example, are used in scheduling systems to represent factory schedules.

One common thing in the approach being followed in all inventory classification techniques and their integration mechanisms is that all of them should be based on the manager's thoughts to serve best his or her goal, [10], as depicted in figure 3.3.

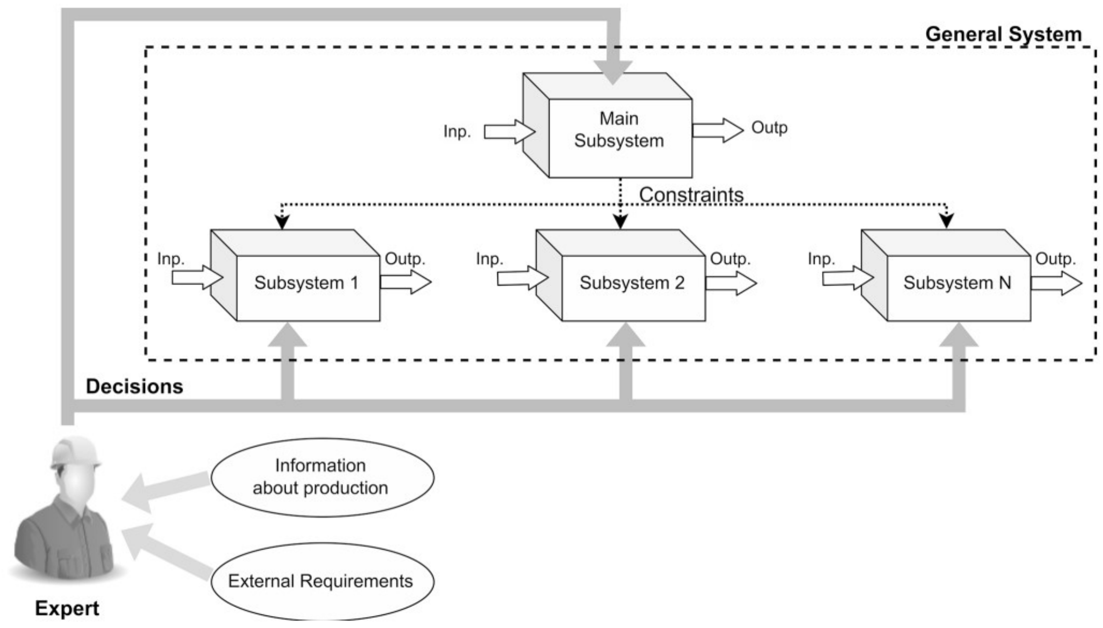


Figure 3.3: Schematic view of the decision processes in a general manufacturing system, by [11]

3.2 Warehouse and storage

In a supply chain, products must be delivered to customers in the shortest possible time, and such delivery time is only achieved with a minimum number of actions. Therefore, the warehouse is a crucial link between the customer and the company. According to [12], the main features of the warehouses are:

1. Dampening the flow of material along the supply chain in order to accommodate the variability caused by the seasonality of the product or production
2. Consolidation of products from multiple suppliers for later delivery to the customer
3. Carrying out value-added activities such as kitting, labelling and product customisation

3.2.1 Warehousing today

Rapid globalisation has ushered in many changes, boosting productivity and raising customer expectations. Customers are more demanding than ever before due to the high level of service that businesses have provided. As a result, it is critical to develop novel and practical solutions to the market's difficulties. Organisations must continue to recognise their flaws and consider turning them into assets.

A warehouse appears to be much more than a place to keep items in terms of operations: it can be a source of competitive advantage. Proper management can result in a significant reduction in the total cost of warehouse operations while also assisting in meeting the business's service level objectives, therefore optimising the warehouse and gaining a competitive edge [13]. To do this, one must make a thorough analysis of the warehouse, as depicted in figure 3.4.

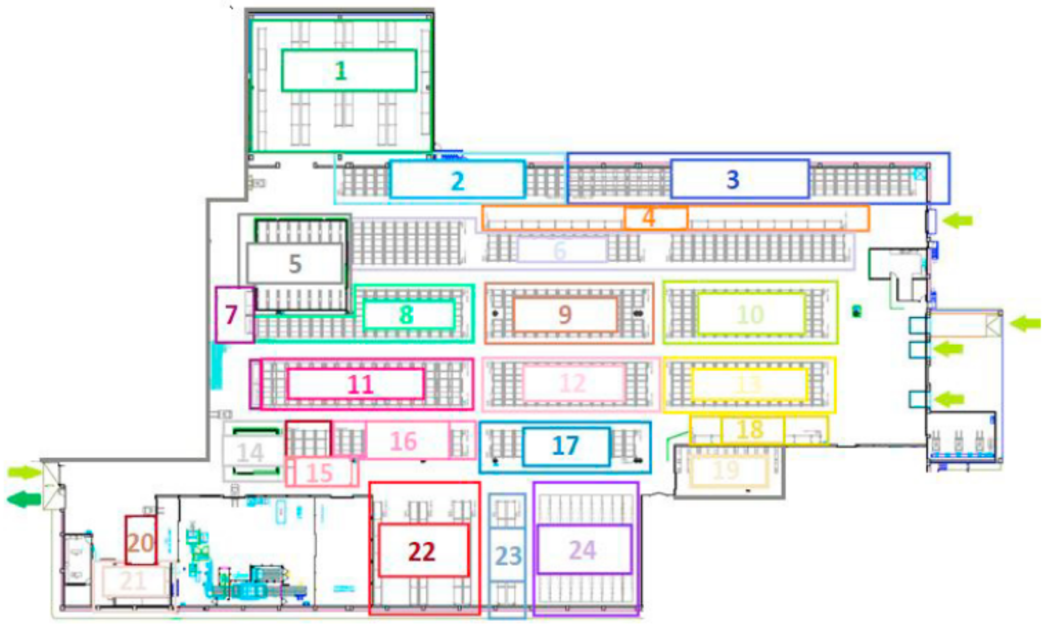


Figure 3.4: Example of a warehouse analysis, performed by [13]

According to [14], companies spend roughly 300 billion euros a year on warehousing, which can be managed in-house or through third-party logistics firms. Operating costs account for more than 85% of that total: the people, space, and equipment needed to receive, sort, store, pick, pack, and finally dispatch products. This value tends to rise as e-commerce, and worldwide supply networks develop in popularity and complexity, further complicating the operations. So, why are warehouses necessary? According to [15], they help with a variety of the company's goals, including:

- Achieving transportation economies (e.g. combine shipment, full-container load);
- Achieving production economies (e.g. make-to-stock production policy);
- Meeting changing market conditions and uncertainties (e.g. seasonality, demand fluctuations, competition);
- Overcoming the time and space differences that exist between producers and customers;
- Providing customers with a mix of products instead of a single product on each order (i.e. consolidation);
- Providing temporary storage of material to be disposed or recycled (i.e. reverse logistics).

The goal of Industry 4.0 is to employ technology to improve core logistics and warehouse activities, including loading and unloading, picking, storage management and shipping. It is not just about deploying robots to perform challenging or repetitive activities; it is also about utilising computers to collect massive quantities of data and using that data to disclose new information about the operations and make changes [16]. The use of 4.0 concepts is perfectly adapted to warehousing operations. It is a closed physical area (microenvironment) with a specified function and a wide range of automation and technology options to select when creating the company's correct model(s), as depicted in figure 3.5. With this, the company can set goals like reducing warehousing operating costs, increasing productivity, and overall improved service levels, among others, and use them to establish their work plan considering the current baseline [17].

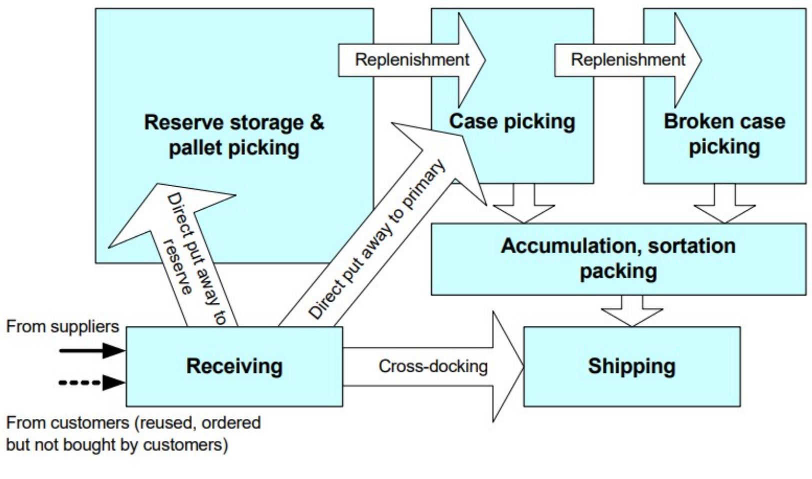

Reserve storage & pallet picking
Replenishment
Case picking
Replenishment
Broken case picking
Direct put away to reserve
Direct put away to primary
Accumulation, sortation packing
From suppliers
Receiving
Cross-docking
Shipping
From customers (reused, ordered but not bought by customers)

Figure 3.5: Typical warehouse functions and flows, by [18]

3.2.2 Layout

It is impossible to obtain an optimal generic layout that allows for an effective and efficient result in the execution of the activities of any warehouse. Each warehouse is unique, so the same layout should not be applied to two different spaces. An in-depth analysis of the characteristics and functionalities of the respective warehouse must be carried out to obtain the appropriate layout for it.

There are two types of layout most used in warehouses: the continuous flow (straight-through or straight-line) and the U-flow. Figure 3.6 shows the visual configuration of the behaviour of the two flows and the locations of the zones that constitute, in general, a warehouse.

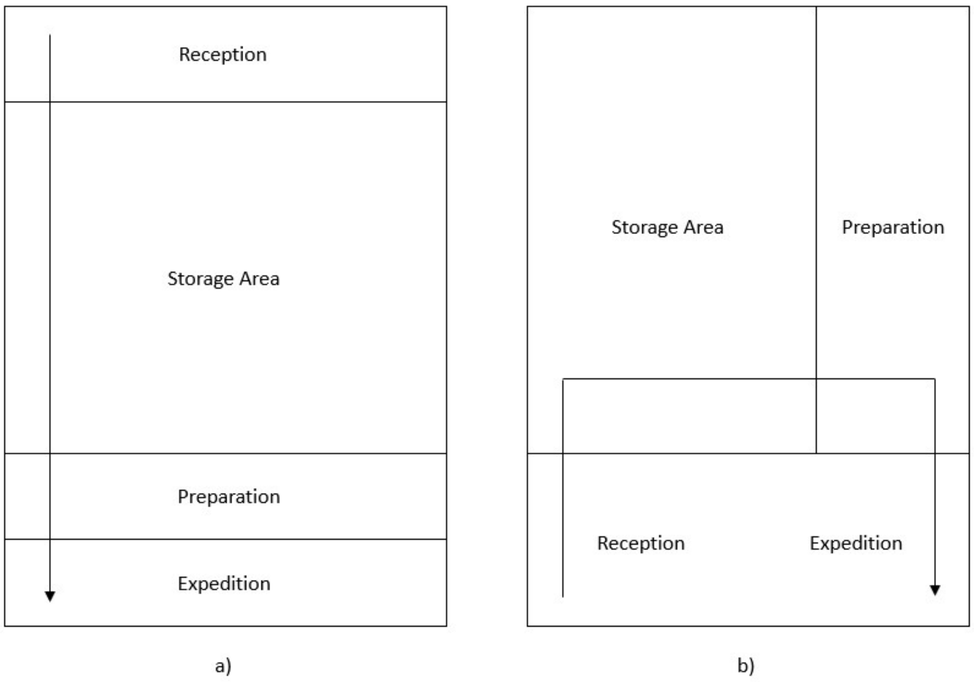


Figure 3.6: Warehouse layout examples: a) continuous flow; b) U-flow, adapted from [19]

Continuous flow layouts offer advantages in terms of travel times and congestion. They are suitable in warehouses that use cross-docking and warehouses with several manufacturing facilities (production lines).

Warehouses with a U-shaped flow have two advantages the previous flow does not have. Namely, the shorter average distance travelled within the warehouse and the proximity between the space needed for the reception and dispatch of products.

3.2.3 ABC analysis

ABC analysis is a strategy for inventory management that assesses the value of inventory items based on their importance to the company. ABC rank them based on demand, cost, and risk data, and inventory managers classify items based on those criteria. This aids business leaders in determining which items or services are most important to their company's financial performance [20]. Class A items are the most significant stock-keeping units (SKUs) in terms of sales volume or profitability. In contrast, class B items are the second most important, and class C items are the least important. The classification through this analysis is one of the most extensively used in business strategies [21].

The Pareto Principle states that 20% of causes generate 80% of the consequences, implying an uneven relationship between inputs and outputs. This principle serves as a general reminder that the input-output relationship is unbalanced [22]. ABC analysis reveals that 20% of goods deliver around 80% of the value, according to Pareto's 80/20 rule. As a result, most businesses have a small number of "A" items, a somewhat more significant group of "B" items, and a massive group of "C" things, which encompasses the vast majority of items [20], as figure 3.7 shows.

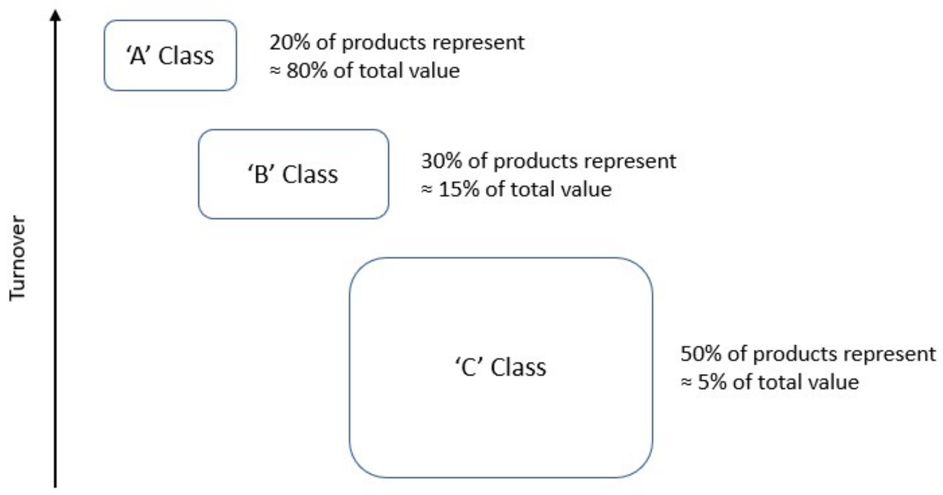


Figure 3.7: Classification of products and their turnover, adapted from [23]

The study conducted by [24] found through an ABC analysis that only 23% of total items constitute 60% of annual cost, 36% of items constitute 22% of annual costs, and 41% of items make up the rest of the 18% contribution to annual costs. Based on these findings, they recommend spending more money on supplier development and forecasting for 'A' items since these are the most important and should be given the most time and attention when placing orders. Also, cycle counting of 'A' items (physically counting a sample of total inventory regularly) is required to:

1. Eliminate the need for yearly physical inventory shutdowns and interruptions of production;
2. Eliminate annual inventory adjustments;
3. Provide qualified employees to conduct inventory audits;
4. Allow for the identification of the source of mistakes and the implementation of corrective measures;
5. Keep detailed inventory records.

Combining ABC analysis with the flow type usually gets one of the figure's 3.8 layouts for the storage area. Since, intuitively, products with a higher turnover are in the front line, next to the dispatch area, it is concluded that in this space will be class A products, followed by the placement of class B articles and, finally, class C articles.

In a warehouse with a U-shaped flow, the placement of high-turnover items is allocated near the entrance and exit of the warehouse, and the low-turnover products are placed at the back of the warehouse (across-aisle storage in figure 3.8).

Suppose the warehouse under study has a continuous flow. In that case, the products with the highest turnover are allocated in the central line of the warehouse and the products with low turnover are positioned on the sides of the warehouse (within-aisle storage in figure 3.8).

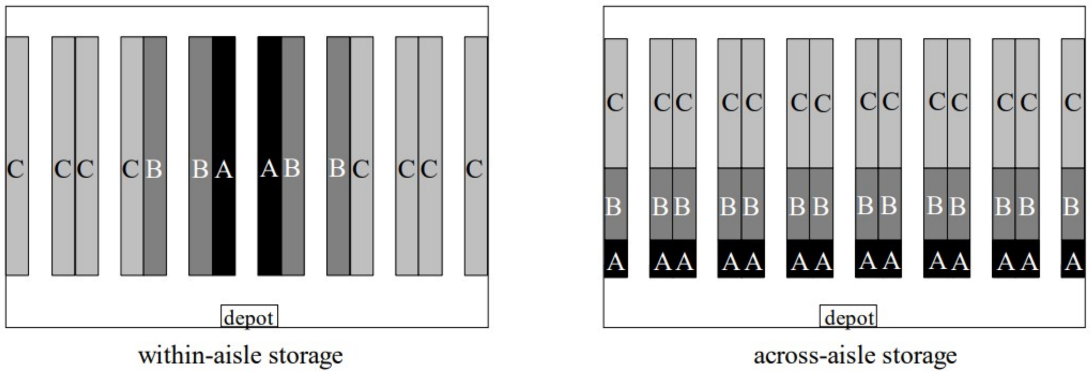


Figure 3.8: Two common ways of implementing an ABC-based storage, by [25]

Regarding inventory, ABC analysis can help better control the working capital costs. The knowledge gleaned from the research helps reduce obsolete inventory and increase inventory turnover, or how often a company must replace things once they have been sold. However, despite its many advantages for inventory management and maintenance, ABC analysis is not a one-size-fits-all solution. Customer demand patterns, classifications, systems, and other challenges vary in every business, affecting the utility of an ABC study. Knowing this, other kinds of analysis were studied.

3.2.4 PQR and FSN analysis

PQR method

The PQR classification, or popularity rating, is based on how often items are used [26]. This method considers the number of material transactions carried out in a given period [27], classifying them as one of the following three:

- **P** - items that have a high frequency of demand or consumption
- **Q** - items that have an average or intermediate frequency of demand or consumption
- **R** - items that have a low frequency of demand or consumption rate

To categorise the products as shown above, first, calculate the movement of each product, this is the percentage that product sold regarding the total amount, and then calculate the accumulation of these percentages, with the list in descending order of % movement. The 'P' products are the first 70% of accumulated movement %, the 'Q' products are the following 20%, and the 'R' products are the last 10%.

FSN method

The FSN categorisation system separates fast-moving components from slow-moving and non-moving elements. The central concept is that portions with the highest demand are classed as fast-moving, while those with the lowest demand are labelled non-moving [10]. The rest of the portions are categorised as slow-moving. The phases in the FSN technique utilised to categorise are as follows.

1. Obtain the total demand for each product;
2. Arrange the data in ascending order based on the total demand;
3. Calculate the first and third quartiles (Q1 and Q3) with equations 3.1 and 3.2, where 'n' represents the number of parts;
4. Classify the parts using the following logic:
   (a) If total demand > Q3
   Classify as Fast Moving (F)
   (b) If total demand < Q1
   Classify as Non-Moving (N)
   (c) Otherwise
   Classify as Slow Moving (S)

$$Q_1 = \frac{n+1}{4} \tag{3.1}$$

$$Q_3 = \frac{3 \cdot (n+1)}{4} \tag{3.2}$$

The analyses mentioned in this section are two different classification methods that serve the same purpose, to segregate products based on their consumption rate, quantity, and the rate at which the inventory is used. These, alongside the ABC classification method, are a well-established base to categorise and evaluate the products of any given company.

3.2.5 Knapsack problem

The knapsack problem is a combinatorial optimisation problem that determines the quantity of each item to include in a collection given a set of objects, each with a weight and a value so that the total weight is less than or equal to a given limit and the total value is as large as possible [28]. It gets its name from the problem someone with a fixed-size knapsack faces when they have to fill it with the most important items. This problem frequently arises in resource allocation, as decision-makers must choose from a collection of non-divisible projects or activities while working under a strict budget or time constraint.

There are a few variations of the Knapsack problem: 0–1 knapsack, bounded knapsack, and unbounded knapsack. The variation that suits this project is the 0-1 knapsack which restricts the number of copies of each kind of item to zero or one.

3.3 Relevant papers

All cited papers were essential to developing our work because they helped better understand concepts about DSSs and everything about warehouses. This being said, the most crucial aspect to find information on was DSS frameworks for warehouses, and the results were very scarce. The only helpful framework found was the one presented in figure 3.2. This is a good framework but a somewhat complex, so we decided to build our framework, starting also with the ABC classification as input but disregarding, for now, the historical demand and the associated costs.

Chapter 4

Analyses and proposed solution

4.1 Warehouse analysis

To start analysing how to categorise and organise the warehouse, the list of customer orders from the entire year 2021 was used to know how much the company sold through an entire year and if there were any clear distinctions between products regarding the volume of interest from the customers.

In order to do this, the company provided an excel file with data from January 2021 until March 2022, containing the relevant information of their customer orders.

Table 4.1 contains a sample of that data but all the codes were changed/omitted to keep the company's data confidential. The file contained: the type of order which, for our analysis, only mattered load orders since the other types were irrelevant; the code of the document issued for collection; the batch of the product required; the code of the product; the quantity ordered (QO) by the client; the quantity of product per pallet (Q/P) and the date of the order.

Table 4.1: Sample of the data from Primus Vitória's received orders

Type	Document	Batch	Product	QO (m^2)	Q/P (m^2)	Date
Load	LO56000984	00AA00	P*******L******VD	21.6	60.48	31/03/2022 10:48
Load	LO45000467	55CC77	P*******L******VB	1774.08	80.64	03/11/2021 15:06
...	...	...	...	...	...	...
Load	LO25000690	95CV45	P******LRU****VC	102.4	102.4	04/01/2021 12:12

Before getting to the automation of categorising the products, a direct analysis as first made to understand how to correctly approach the situation and create a baseline to compare results with the possible future algorithms.

The first step was to get the unique products that the company sells within the 43330 orders from 2021, 9846 orders from 2022 and how much volume each product sold.

A sample from the results for the entire year of 2021 is exhibited in table 4.2, and the first three months of 2022 in table 4.3. The critical numbers to retain are that 1362 products were sold in 2021 and 862 products in the first quarter of 2022, accounting for 4,972,749.97 m^2 and 1,287,867.74 m^2, respectively.

Table 4.2: Unique products in the data from 2021

	Unique Products	Quantity Ordered (m^2)
Total	1362	4972749.97
1	P*******L******VD	1448.00
2	P*******L******VD	1809.60
3	P*******L******VD	3510.70
4	P*******L******VB	2694.00
5	P*******L******VD	476606.40
6	P*******B******GB	12906.00
7	P*******L******GD	30540.80
8	PEA********	0.18
...	...	...
1355	P*******LRU****VC	7.50
1356	P******LL******GA	180.00
1357	P*******L******VC	1.50
1358	P******CL******VA	10.00
1359	P*******L*****EBB	17.00
1360	P******LL******VA	80.00
1361	P*******L*****EVB	17.00
1362	P******FLFB***EBA	132.00

Table 4.3: Unique products in the data from 2022

	Unique Products	Quantity Ordered (m^2)
Total	862	1287867.74
1	P*******H*PINK*VN	21.60
2	P*******B*****EBB	1423.00
3	P*******H*BRIG*VN	71.28
4	P*******B******G*	5867.00
5	P*******H*MARI*VN	254.16
6	P*******B******VB	117.00
7	P*******L******VC	1767.00
8	P*******L******VC	138.00
...	...	...
855	P*******L******GB	1.00
856	P*******L*SAFIEBC	15.00
857	P******FL*SAFIEBA	100.00
858	P*******L******VB	2.00
859	P*******L******X*	21120.00
860	M*******LP****EVB	6.00
861	P*******L*CT***VC	1.50
862	PEA********	0.18

Since the products with the highest volume of sales are the most relevant, the data was arranged in descending order of m^2 sold to get to the most important products, resulting in the following tables 4.4 and 4.5.

Table 4.4: Unique products in the data from 2021 by descending order of m^2 sold

	Unique Products	Quantity Ordered (m^2)
Total	1362	4972749.974
1	P*******L******VD	476606.4
2	P*******L******VC	337380.5
3	P*******L******VD	176299.2
4	P*******L******GZ	152390.52
5	P*******L******XB	128438
6	P*******L******SB	105930
7	P*******L******GZ	102438.56
8	P*******L******GD	94913.6
...	...	...
1355	PEA********	0.642
1356	PEA********	0.464
1357	PEA********	0.265
1358	PEA********	0.184
1359	PEA********	0.16
1360	PEA********	0.121
1361	PEA********	0.038
1362	PEA********	0.002

Table 4.5: Unique products in the data from 2022 by descending order of m^2 sold

	Unique Products	Quantity Ordered (m^2)
Total	862	1287867.74
1	P*******L******VD	147720.00
2	P*******L******VC	103980.00
3	P*******L******GZ	63060.48
4	P*******L******VD	55838.40
5	P*******L******GZ	54409.60
6	P*******L******VB	37227.00
7	P*******L******BB	25165.00
8	P*******LMICAN*B*	23278.80
...	...	...
855	PEA********	0.30
856	PEA********	0.30
857	PEA********	0.20
858	PEA********	0.18
859	PEA********	0.02
860	PEA********	0.02
861	PEA********	0.01
862	PEA********	0.00

Since the warehouse in analysis stores only pallets full of material and different products have different amounts of m^2 per pallet, the number of pallets each product sold was calculated. The data was then re-ordered with descending number of pallets sold, sampled in the tables 4.6 and 4.7, and, as expected, a similar order from the previous was obtained.

Table 4.6: Unique products in the data from 2021 by descending order of pallets sold

	Unique products	QO (m^2)	Q/P (m^2)	# Pallets
Total	1362	4972749.97	-	48495.09
1	P*******L******VD	476606.4	102.4	4654.359
2	P*******L******VC	337380.5	120	2811.504
3	P*******L******GZ	152390.52	80.64	1889.763
4	P*******L******VD	176299.2	102.4	1721.672
5	P*******L******XB	128438	96	1337.896
6	P*******L******GZ	102438.56	80.64	1270.319
7	P*******LMICAN*B*	51009.16	40.84	1249.000
8	P*******L******SB	105930	90	1177.000
...	...	...	...	...
1355	P************C*BA	3	5280	0.001
1356	M*******LL*****VA	12	21312	0.001
1357	P******LL******GA	10	21312	0.000
1358	M******RL**LCT*VA	4	9408	0.000
1359	P******F*****A*BA	2	5280	0.000
1360	P******F*****C*BA	2	5280	0.000
1361	M******LL******VA	8	21312	0.000
1362	P******RL******GA	1	9408	0.000

Table 4.7: Unique products in the data from 2022 by descending order of pallets sold

	Unique products	QO (m^2)	Q/P (m^2)	# Pallets
Total	862	1287867.74	-	13381.53
1	P*******L******VD	147720.00	102.40	1442.578
2	P*******L******VC	103980.00	120.00	866.500
3	P*******L******GZ	63060.48	80.64	782.000
4	P*******L******GZ	54409.60	80.64	674.722
5	P*******LMICAN*B*	23278.80	40.84	570.000
6	P*******L******VD	55838.40	102.40	545.297
7	P*******L******VB	37227.00	88.00	423.034
8	P*******L******VV	16936.56	51.84	326.708
...	...	...	...	...
855	P******FRFV****VA	4.00	3680.00	0.001
856	P******XL******GA	8.00	9408.00	0.001
857	P******FRFV****VA	2.00	3680.00	0.001
858	P******XL******VA	2.00	9408.00	0.000
859	P******VL******GA	2.00	9408.00	0.000
860	M******FLEF****VA	2.00	9500.00	0.000
861	M******FLDF****VA	2.00	9500.00	0.000
862	P******XL******GA	1.00	6720.00	0.000

With the 2021 data, from the total of 1362 unique products, 580 products sold less than one pallet and, from the 2022 data, from the total of 862 unique products, 449 also sold less than one pallet. This means that, since they have a designated area in the internal warehouse with shelves to store products that do not fulfil an entire pallet, those products never went to the external warehouse, so they are not relevant to this project. With this, the products that sold less than one pallet were removed, resulting in 782 products for 2021 and 413 for 2022.

4.1.1 PQR and FSN classification

With all the relevant products gathered, in descending order of m^2, a PQR analysis was made to decide if it is worth applying through an automated algorithm. To do this, the same historical data was used, as shown in table 4.1, and the analysis was applied for the whole year of 2021 to see if this method would suit the project's purpose. Table 4.8 contains a sample from the analysis where 0% to 50% of the aggregated % movement is classified as P, 50.01% to 80% gets classified as Q and greater than 80% classifies as R.

Table 4.8: PQR classification of the 2021 data by descending number of orders

Products	# Orders	% Mov	Agg % Mov	Quantities (m^2)	PQR	
P1	1308	3.03	3.03	337380,50	P	
P2	1169	2.71	5.74	476606.40	P	
P3	1100	2.55	8.28	54460.00	P	82
...	...	...	...	...	...	
P82	120	0.28	49.82	1701.00	P	
Q1	115	0.27	50.09	30540.80	Q	
Q2	117	0.27	50.36	1516.21	Q	
Q3	116	0.27	50.63	3248.00	Q	190
...	...	...	...	...	...	
Q190	40	0.09	79.95	74760.00	Q	
R1	40	0.09	80.04	1957.76	R	
R2	40	0.09	80.13	2955.20	R	
R3	40	0.09	80.23	220.60	R	1090
...	...	...	...	...	...	
R1090	1	0.00	100.00	132.00	R	

As discussed in section 3.2.4, the PQR classification distinguishes the fast-moving parts from the slow and non-moving parts, and the general idea is that the parts having the highest demand get classified as fast-moving and the parts having the least demand as non-moving, with the remaining parts classified as slow-moving. With this in mind, from the data presented in the table 4.8, a conclusion can be made that this kind of analysis does not add valuable distinctions to the project because all products have a relatively low percentage of the total movement, with the highest moving product representing only 3.03% of the total movement in the whole year of 2021.

Nevertheless, since the PQR analysis serves the same purpose as the FSN analysis (mentioned in section 3.2.4), in order to confirm the deduction mentioned above, the method of distinguishing the parts from the FSN analysis was adopted. This is based on calculating two quartiles, the first and third (Q1 and Q3), outlined in equations 3.1 and 3.2. Applying these to the data resulted in the following:

Table 4.9: Calculation of the Q1 and Q3 quartiles

# Products (n)	Q1	Q3
1362	340.75	1022.25

With the information from the table 4.9, the classification of the products can be made based on the following logic:

1. If total demand > 1022.25
   Classify as Fast Moving (F)
2. If total demand < 340.75
   Classify as Non-Moving (N)
3. Otherwise
   Classify as Slow Moving (S)

Table 4.10: FSN classification of the 2021 data by descending number of orders

Products	# Orders	FSN	
F1	1308	F	
F2	1169	F	3
F3	1100	F	
S1	960	S	
S2	733	S	
S3	717	S	
...	...	...	9
S7	447	S	
S8	365	S	
S9	365	S	
N1	309	N	
N2	306	N	
N3	306	N	1350
...	...	...	
N1350	1	N	

Analysing the results from the table 4.10, only three products are classified as fast-moving products, and their number of orders is barely above the Q3 quartile, meaning there are no significant outliers. Apart from those, there are nine products under the slow-moving classification and then the remaining 1350 products as non-moving. With this, the results are in accordance with the PQR analysis making it fair to say that there are no significant distinctions between the products when it comes to analysing the moving of each product concerning the total amount of products sold. Since the results showed that this type of analysis was not relevant to the project, no algorithm regarding the automation of the process was elaborated.

4.1.2 ABC classification

Moving on to the ABC analysis, this type of analysis was described in section 3.2.3. However, the most critical numbers to retain are from the Pareto principle, which suggests that 20% of the products represent 80% of the sales. In this project, since it is only a warehouse analysis regarding how and where to store the products, only the data from the volume of products sold were used, not the revenue numbers.

First, the percentage each product represented relative to the total amount sold was calculated and then those percentages were aggregated with the list in descending order of m^2 sold. The only thing left was to find the least number of products that sold the equivalent of 80% of m^2 and classify these as A, then the next 15% of m^2, classifying these as B, and finally the final 5%, getting classified as C. In the literature, different analyses were found regarding the percentage for each component of the ABC method, like 70/25/5 or 60/20/20, but the 80/15/5 was used in this project.

Table 4.11: ABC classification for the 2021 data

	Unique products	QO (m^2)	% of Total	Agg. %	ABC	
Total	782	4843781.35	-	100.000	80 / 15 / 5	
1	P2002007L000101VD	476606.40	9.840	9.84	A	
2	P1501505L000101VC	337380.50	6.965	16.80	A	
...	...	...	...	...	...	81
80	P150150FLFB2161VA	11217.00	0.232	79.60	A	
81	M2002007LP80001VB	10926.00	0.226	79.83	A	
82	P2002006L000111GE	10794.60	0.223	80.05	B	
83	P1501505L000101IC	10560.00	0.218	80.27	B	
...	...	...	...	...	...	187
267	P1500757B062001VI	1516.21	0.031	94.94	B	
268	P1500757B019601VI	1496.21	0.031	94.97	B	
269	P1003007L0QUAREBM	1469.21	0.030	95.00	C	
270	P1003007L0PALH1GM	1467.14	0.030	95.03	C	
...	...	...	...	...	...	514
781	P1001000H002001VJ	67.32	0.001	100.00	C	
782	P1001000H044101VJ	64.68	0.001	100.00	C	

Table 4.12: ABC classification for the 2022 data

	Unique products	QO (m^2)	% of Total	Agg. %	ABC	
Total	413	1236638.17	-	100.000	80 / 15 / 5	
1	P2002007L000101VD	147720.00	11.95	11.95	A	
2	P1501505L000101VC	103980.00	8.41	20.35	A	
...	...	...	...	...	...	57
80	P2002007L00010EGD	4505.60	0.36	79.52	A	
81	P2004008L00011EBZ	4497.92	0.36	79.88	A	
82	P2002007L00011EBD	4278.40	0.35	80.23	B	
83	P1501505L062511VC	4161.00	0.34	80.57	B	
...	...	...	...	...	...	124
180	P2002007L021201VD	601.60	0.05	94.94	B	
181	P1501505L019611NC	600.00	0.05	94.99	B	
182	P1001000H0MARI1VJ	599.28	0.05	95.04	C	
183	P1503007L00011EBB	595.00	0.05	95.09	C	
...	...	...	...	...	...	232
412	P1001000H019601VJ	65.34	0.01	99.99	C	
413	P1001000H0PINK1VJ	62.04	0.01	100.00	C	

Tables 4.11 and 4.12 show samples of this classification, with the relevant numbers to retain being that, for 2021, 81 products got classified as A, 187 products as B, and 514 products as C, and for 2022, 57 products got classified as A, 124 product as B and 232 products were classified as C. In other words, as shown in the following tables 4.13 and 4.14, the exact numbers of the Pareto principle were even exceeded since only 10.36% and 13.80% of the total number of products, for 2021 and 2022 respectively, represent 80% of the total m^2 sold in each time frame.

Table 4.13: Final ABC analysis for 2021

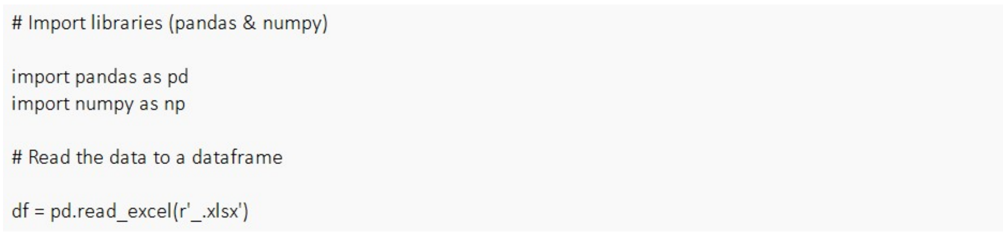
	Unique products	% of Total (#UP)	QO (m^2)	% of Total (QO)
Total	782	100.00	4843781.35	100.00
A	81	10.36	3866744.88	79.83
B	187	23.91	733222.91	15.14
C	514	65.73	243824.56	5.03

Table 4.14: Final ABC analysis for 2022

	Unique products	% of Total (#UP)	QO (m^2)	% of Total (QO)
Total	413	100.00	1236638.17	100.00
A	57	13.80	987860.56	79.88
B	124	30.02	186825.76	15.11
C	232	56.18	61951.86	5.01

Algorithm for the automation of the ABC classification

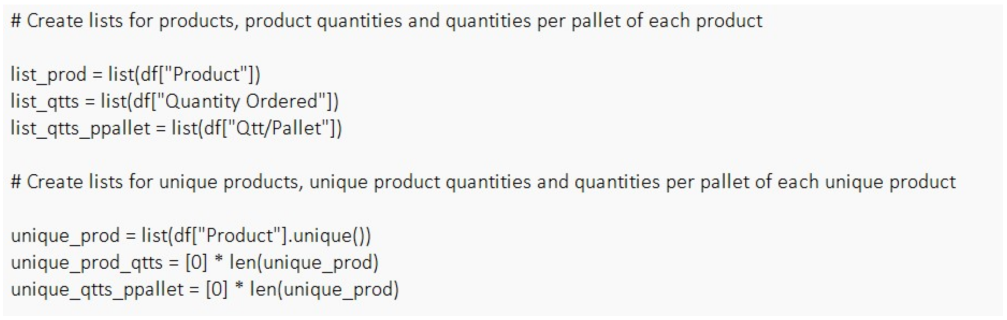
With the previous results, the ABC analysis is appropriate to this case study, so an algorithm in python to classify the products was developed to automate this process. The first step of the algorithm was to import the necessary libraries. Pandas was used for the data manipulation and analysis, and NumPy, for matrix calculus. Then, a data frame that reads the input data was created. For the input, the original Excel file (exemplified in table 4.1) that the company sent was used, using only the data from 2021. Figure 4.1 shows this part of the algorithm.



```
# Import libraries (pandas & numpy)

import pandas as pd
import numpy as np

# Read the data to a dataframe

df = pd.read_excel(r'_.xlsx')
```

Figure 4.1: ABC classification algorithm: import libraries and read the data

Next, lists were created for the products, the product quantities, and the quantities per pallet of each product from the input data. The goal of this step was to have all this information only for the unique products, so three new lists were created, the first being the list of the unique products and then two ‘empty’ lists to get their quantities and quantities per pallet. Figure 4.2 exhibits these lists.



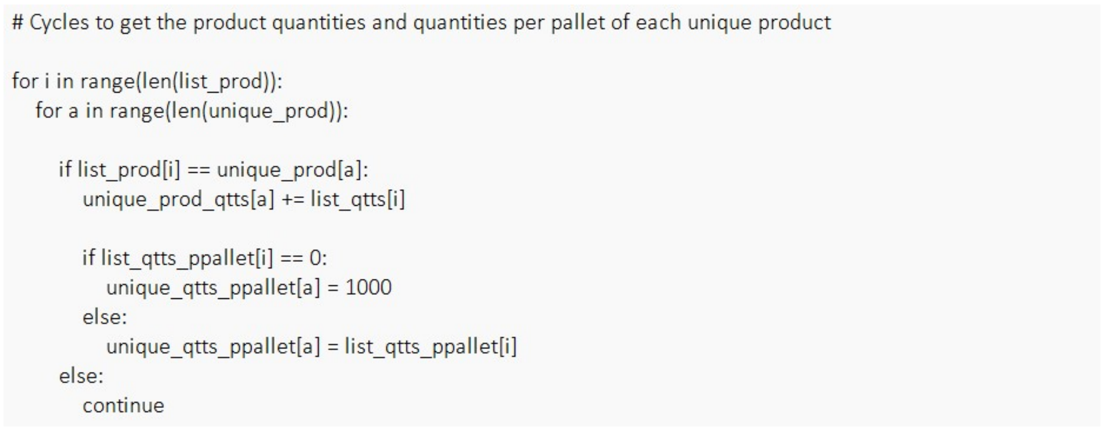
```
# Create lists for products, product quantities and quantities per pallet of each product

list_prod = list(df["Product"])
list_qtts = list(df["Quantity Ordered"])
list_qtts_ppallet = list(df["Qtt/Pallet"])

# Create lists for unique products, unique product quantities and quantities per pallet of each unique product

unique_prod = list(df["Product"].unique())
unique_prod_qtts = [0] * len(unique_prod)
unique_qtts_ppallet = [0] * len(unique_prod)
```

Figure 4.2: ABC classification algorithm: create lists for the analysis

With the lists ready, two ’for’ cycles were created, composed of two if statements. The first ’for’ cycle runs through the complete list of products, and, for each of these products, the second for cycle runs through the list of unique products.

For each iteration, if the product in the first list is the same as the one in the second list, we add the quantity of m^2 sold to the list with the unique product quantities. For the same iteration, if the value of quantity per pallet of the product in the first list equals zero, we attribute 1000 to the list of the quantities per pallet for unique products. The reason to do this is that the zero represents the lack of this information regarding the product, and keeping it as zero would result in a future error in calculating the number of pallets for the product. If the number is not zero, then we attribute the value to the list of the quantities per pallet for unique products. These cycles are laid-out in figure

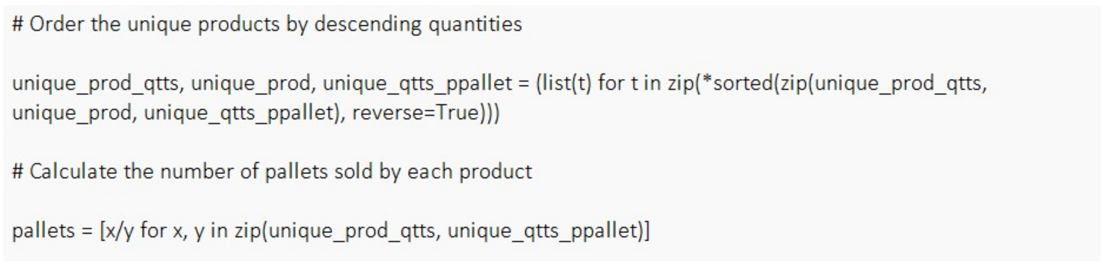
4.3.



```
# Cycles to get the product quantities and quantities per pallet of each unique product

for i in range(len(list_prod)):
   for a in range(len(unique_prod)):

      if list_prod[i] == unique_prod[a]:
         unique_prod_qtts[a] += list_qtts[i]

         if list_qtts_ppallet[i] == 0:
            unique_qtts_ppallet[a] = 1000
         else:
            unique_qtts_ppallet[a] = list_qtts_ppallet[i]
      else:
         continue
```

Figure 4.3: ABC classification algorithm: cycles to get the unique products data

With the previous steps complete, three lists were obtained composed only of the unique products and their respective total quantities sold and quantity per pallet. Afterwards, these lists were ordered to get the products in descending order of the total quantities sold and then calculated the number of pallets sold by each product. Figure 4.4 presents this part of the algorithm.



```
# Order the unique products by descending quantities

unique_prod_qtts, unique_prod, unique_qtts_ppallet = (list(t) for t in zip(*sorted(zip(unique_prod_qtts,
unique_prod, unique_qtts_ppallet), reverse=True)))

# Calculate the number of pallets sold by each product

pallets = [x/y for x, y in zip(unique_prod_qtts, unique_qtts_ppallet)]
```

Figure 4.4: ABC classification algorithm: ordering the unique products and calculating the number of pallets

Having calculated the number of pallets for each product, the products that sold less than one pallet had to be eliminated. For this, three empty lists were created with the intent of being the final ones, ready to apply the ABC classification, and then a for cycle to eliminate the products irrelevant to this project. This cycle runs through the list of unique products and is only composed of an if statement that checks if the number of pallets sold is equal to or above one and then attributes them to the final lists. Lastly, a new workbook was created, intended to make the ABC analysis for the final output of this algorithm. Figure 4.5 displays all these steps and then figure 4.6 presents a flow diagram for the 'for' cycle.

```
# Lists for the final data

keep_prod = []
keep_qtts = []
keep_pallets = []

# Cycle to eliminate products that sold 1 pallet or less

for i in range(len(unique_prod)):
    if pallets[i] > 1:
        keep_prod.append(unique_prod[i])
        keep_qtts.append(unique_prod_qtts[i])
        keep_pallets.append(pallets[i])
    else:
        continue

# Create a new workbook for the final analysis

workbook = pd.DataFrame({"Products": keep_prod, "Quantities": keep_qtts, "# Pallets": keep_pallets})
```

Figure 4.5: ABC classification algorithm: final data lists, cycle to eliminate irrelevant products and final workbook

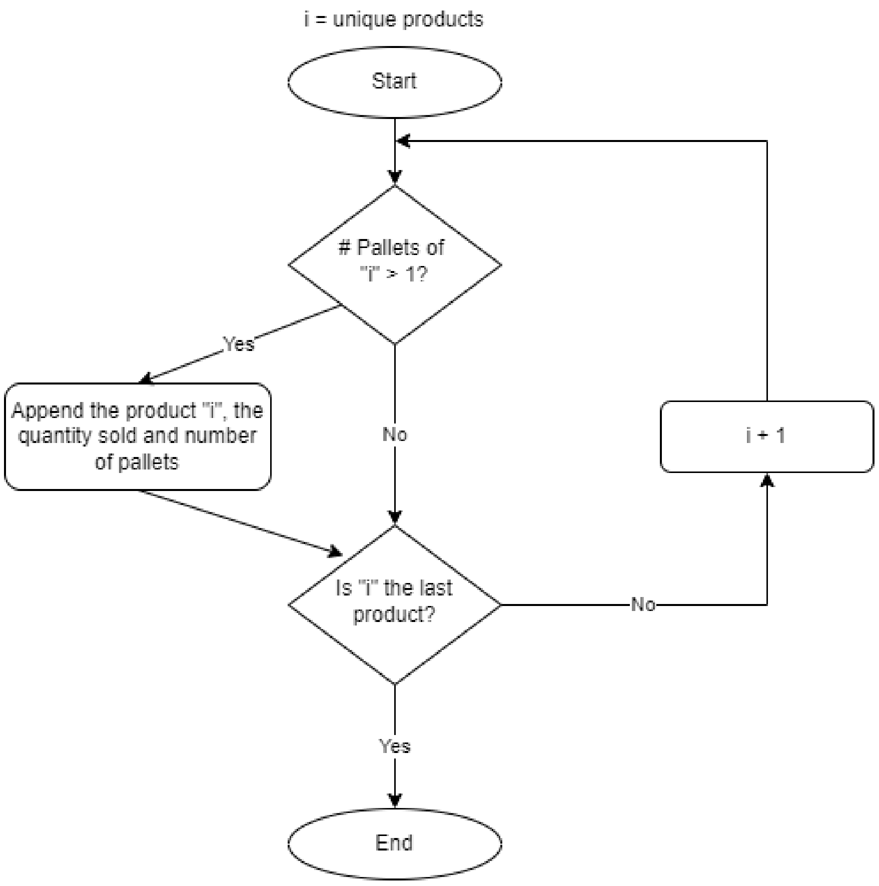
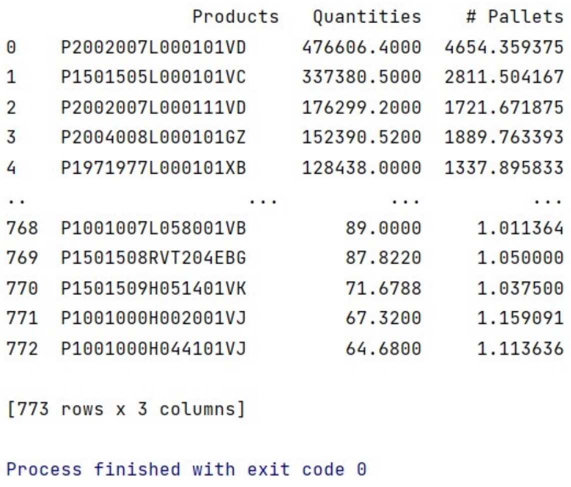


Figure 4.6: 'For' cycle flow diagram

The workbook created is the product of all the above steps, presented in figure 4.7, and matches the previous analysis made with Microsoft Excel (table 4.11).

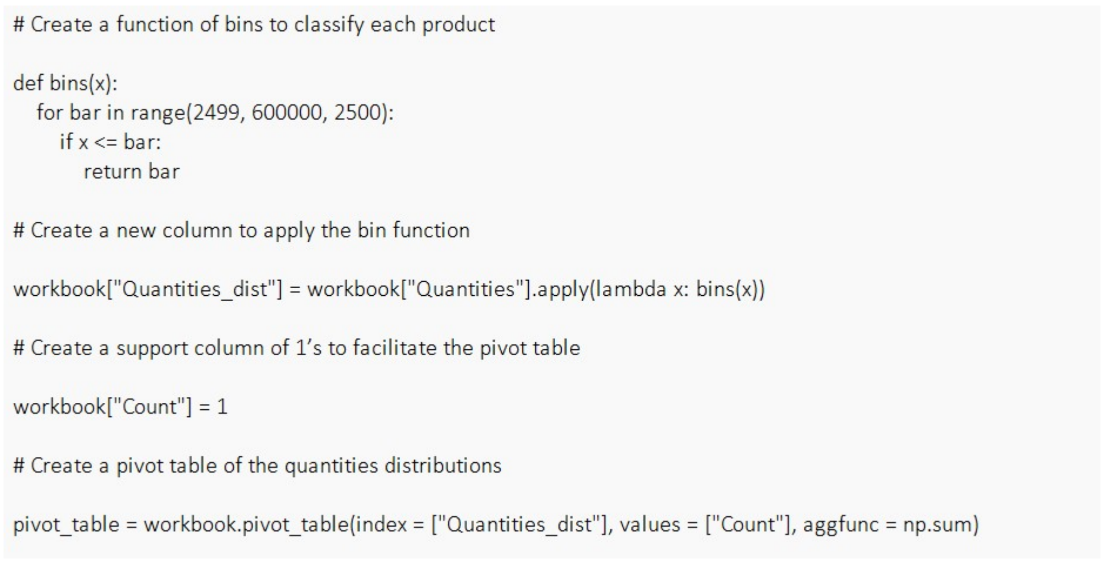


```
                  Products  Quantities    # Pallets
0    P2002007L000101VD     476606.4000  4654.359375
1    P1501505L000101VC     337380.5000  2811.504167
2    P2002007L000111VD     176299.2000  1721.671875
3    P2004008L000101GZ     152390.5200  1889.763393
4    P1971977L000101XB     128438.0000  1337.895833
..                 ...             ...          ...
768  P1001007L058001VB         89.0000     1.011364
769  P1501508RVT204EBG         87.8220     1.050000
770  P1501509H051401VK         71.6788     1.037500
771  P1001000H002001VJ         67.3200     1.159091
772  P1001000H044101VJ         64.6800     1.113636

[773 rows x 3 columns]

Process finished with exit code 0
```

Figure 4.7: Workbook for the ABC analysis

To start the ABC analysis, a function was used to classify the number of quantities each product sold into bins. If a product sold 2,499 m^2 or less, it gets classified as 2,499. If a product sold between 2,499 m^2 and 4,999 m^2, it gets classified as 4,999 m^2 and so on for each increment of 2,500 m^2. Not knowing how this type of analysis would behave in our case, the initial value for the bins and the increment started as an initial estimate and then got tweaked to get the desired effect, resulting in the values here presented.

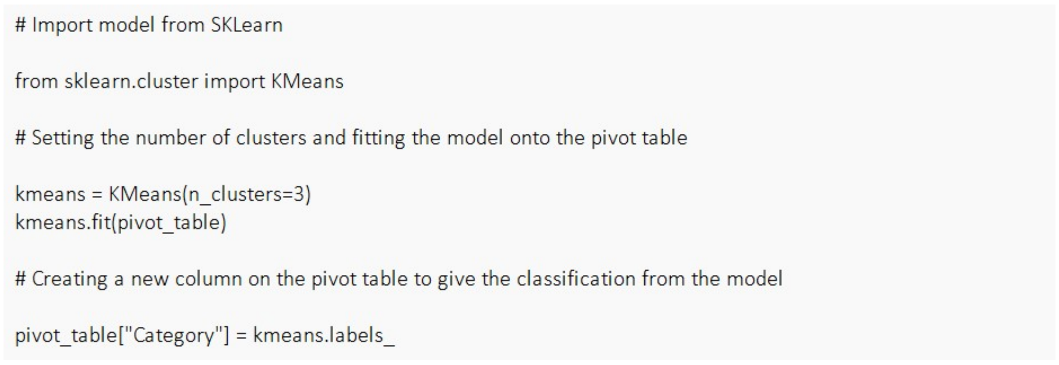
Next, a new column was created in the workbook to apply this function to the data frame and a support column composed of 1's to facilitate the pivot table. Only then creating the pivot table of the quantity's distributions, providing a very manageable data set for the model to train. These steps are demonstrated in figure 4.8.



```
# Create a function of bins to classify each product

def bins(x):
   for bar in range(2499, 600000, 2500):
      if x <= bar:
         return bar

# Create a new column to apply the bin function

workbook["Quantities_dist"] = workbook["Quantities"].apply(lambda x: bins(x))

# Create a support column of 1's to facilitate the pivot table

workbook["Count"] = 1

# Create a pivot table of the quantities distributions

pivot_table = workbook.pivot_table(index = ["Quantities_dist"], values = ["Count"], aggfunc = np.sum)
```

Figure 4.8: ABC classification algorithm: bins function, support columns and pivot table

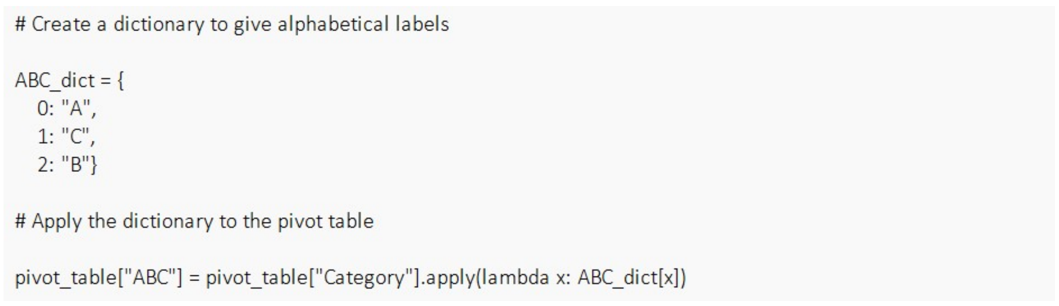
The K-means model from scikit-learn is imported and initialised. This model is one of the simplest and popular unsupervised machine learning algorithms and its training does not require labels nor split data for training or evaluation. The model's objective is simple: group similar data points together and discover underlying patterns, and to achieve this, K-means looks for a fixed number of clusters (a collection of data points aggregated together because of certain similarities) in a dataset. To process the learning data, the algorithm in data mining starts with a first group of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative calculations to optimise the positions of the centroids. The most significant drawback is that the number of clusters must be determined ahead of time. While the number of clusters is difficult to determine in many circumstances, we know we only want to divide the inventory into three clusters for this study. Thus the 'n_clusters' parameter is set to three. After that, the model was fitted onto the pivot table and created a new column to give the classification from the model. This part of the algorithm is in figure 4.9.



```
# Import model from SKLearn

from sklearn.cluster import KMeans

# Setting the number of clusters and fitting the model onto the pivot table

kmeans = KMeans(n_clusters=3)
kmeans.fit(pivot_table)

# Creating a new column on the pivot table to give the classification from the model

pivot_table["Category"] = kmeans.labels_
```

Figure 4.9: ABC classification algorithm: import the K-means model and prepare it

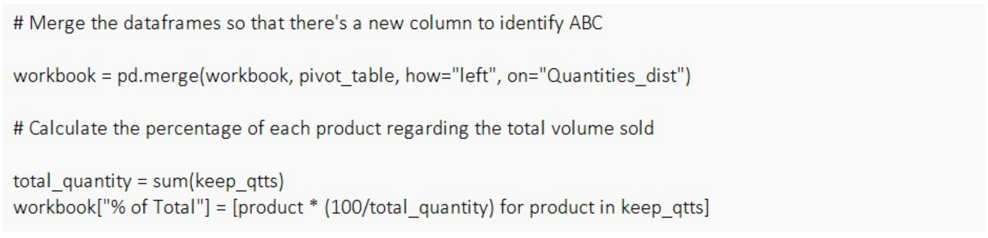
It is worth mentioning that the scikit-learn K-means model will categorise objects on a numerical scale rather than an alphabetical scale by default. As a result, the model labels each row with one of three numbers: zero, one, or two. Fortunately, while the labels may alter, the fundamental pattern stays consistent. Therefore, numerical labels will correspond to alphabetical labels, which will require further review. In this scenario, zero represents A, one represents C, and two represents B. We then wrote a dictionary and applied it to the pivot table to give each row its classification.



```
# Create a dictionary to give alphabetical labels

ABC_dict = {
   0: "A",
   1: "C",
   2: "B"}

# Apply the dictionary to the pivot table

pivot_table["ABC"] = pivot_table["Category"].apply(lambda x: ABC_dict[x])
```

Figure 4.10: ABC classification algorithm: create and apply the ABC dictionary to the model

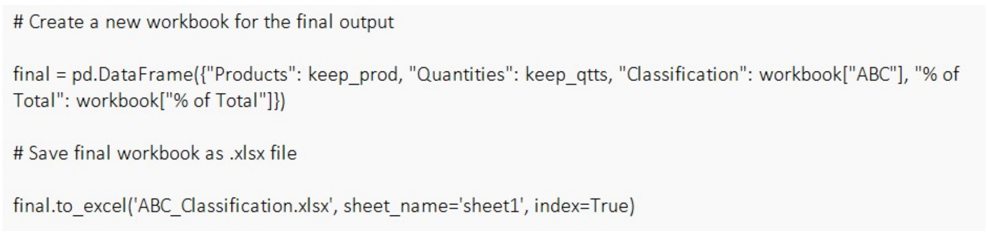
To end the analysis, since the model was trained on a pivot table, the ABC classification needs to be assigned to the actual products. For that, the main data frame and the pivot table were merged and then ended the analysis by calculating the percentage of each product sold regarding the total volume, as figure 4.11 demonstrates.



```
# Merge the dataframes so that there's a new column to identify ABC

workbook = pd.merge(workbook, pivot_table, how="left", on="Quantities_dist")

# Calculate the percentage of each product regarding the total volume sold

total_quantity = sum(keep_qtts)
workbook["% of Total"] = [product * (100/total_quantity) for product in keep_qtts]
```

Figure 4.11: ABC classification algorithm: merge the data frames and calculate the percentage regarding of each product

Finally, the last part of the algorithm is to create the final output with a workbook containing the list of products, the quantities sold, the classification and the percentage of the total. This workbook is then saved as an excel file, shown in figure 4.12.



```
# Create a new workbook for the final output

final = pd.DataFrame({"Products": keep_prod, "Quantities": keep_qtts, "Classification": workbook["ABC"], "% of
Total": workbook["% of Total"]})

# Save final workbook as .xlsx file

final.to_excel('ABC_Classification.xlsx', sheet_name='sheet1', index=True)
```

Figure 4.12: ABC classification algorithm: the final ABC workbook and extraction as an .xlsx file

With the algorithm complete, an excel file can be exported, as sampled in figure 4.15, and, comparing to the previous results (figure 4.11), the conclusion that the algorithm is correctly conducting the analysis can be done.

Table 4.15: ABC classification algorithm output

	Products	Quantities	Classification	% of Total
0	P*******L******VD	476606.40	A	9.84
1	P*******L******VC	337380.50	A	6.97
...	...	...	...	...
126	P*******L******NC	5280.00	A	0.11
127	M*******L****EVB	5183.00	A	0.11
128	P*******L******GB	4788.00	B	0.10
129	P*******B*****EBT	4762.66	B	0.10
...	...	...	...	...
191	P*******L*****EBB	2592.00	B	0.05
192	P*******L******VC	2592.00	B	0.05
193	P*******L******VC	2484.00	C	0.05
194	P*******L******VB	2474.00	C	0.05
...	...	...	...	...
779	P*******H******VJ	67.32	C	0.00
780	P*******H******VJ	64.68	C	0.00

4.1.3 Direct ABC allocation

Having the products correctly classified according to their importance, the first attempt to allocate the products was made with a direct approach. This means that the order of importance from the ABC analysis was used to directly attributed the products. Doing the first analysis empirically resulted in figure 4.13.

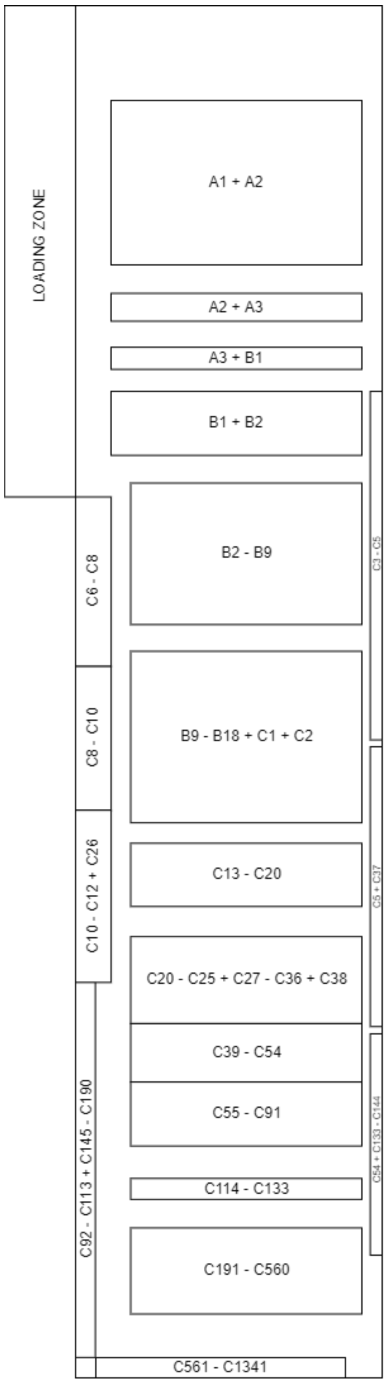
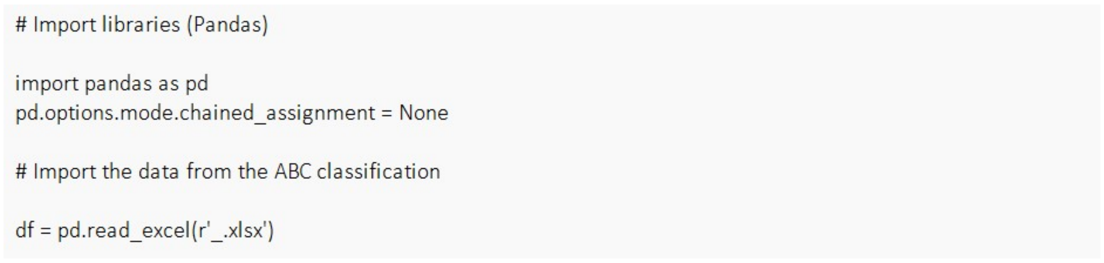


Figure 4.13: Direct ABC layout for the warehouse

Algorithm for the automation of the direct ABC allocation

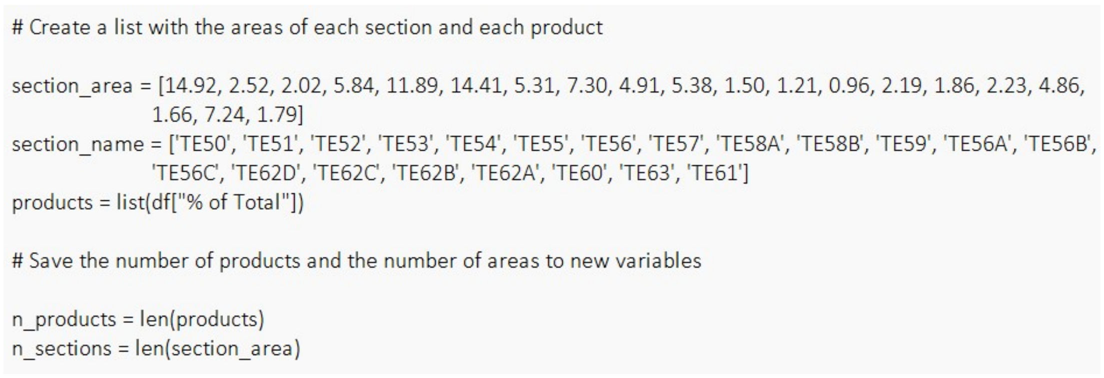
To automate the process of attributing the products directly to the warehouse areas, another algorithm was created. First, the Pandas library was imported and a line to suppress a warning was inserted. Further down the algorithm, there will be a filter to the data frame, modifying the original object. This results in the 'SettingWithCopyWarning' warning, which exists to flag potentially confusing chained assignments but is irrelevant to this case. Then, the data from the ABC classification excel file was imported. These lines are in figure 4.14.



```
# Import libraries (Pandas)

import pandas as pd
pd.options.mode.chained_assignment = None

# Import the data from the ABC classification

df = pd.read_excel(r'_.xlsx')
```

Figure 4.14: ABC algorithm: import the library, suppress a warning and read the data

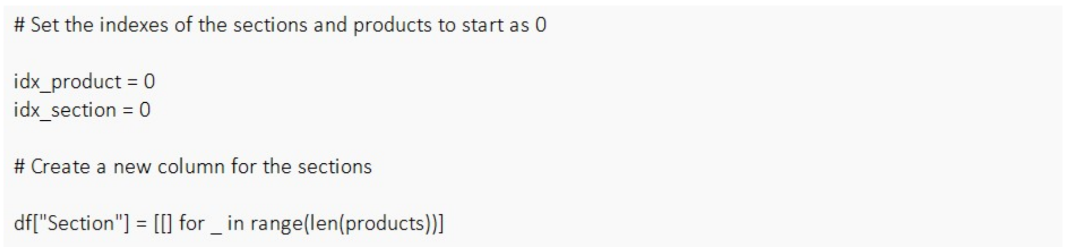
Afterwards, three lists were created: one with the percentage each section represents in the external warehouse; another with the respective names of the sections; and the last one with the product's percentage of total volume sold. With this data, two variables were created, one for the number of products and another for the number of sections. Figure 4.15 presents these steps.



```
# Create a list with the areas of each section and each product

section_area = [14.92, 2.52, 2.02, 5.84, 11.89, 14.41, 5.31, 7.30, 4.91, 5.38, 1.50, 1.21, 0.96, 2.19, 1.86, 2.23, 4.86,
                1.66, 7.24, 1.79]
section_name = ['TE50', 'TE51', 'TE52', 'TE53', 'TE54', 'TE55', 'TE56', 'TE57', 'TE58A', 'TE58B', 'TE59', 'TE56A', 'TE56B',
                'TE56C', 'TE62D', 'TE62C', 'TE62B', 'TE62A', 'TE60', 'TE63', 'TE61']
products = list(df["% of Total"])

# Save the number of products and the number of areas to new variables

n_products = len(products)
n_sections = len(section_area)
```

Figure 4.15: ABC algorithm: create lists for the products and sections, and variables for the total number of products and sections

Before placing products in the sections, indexes for both the products and the sections were created, setting them as zero. Then a new column in the data frame was created to write the name of the section where the product will be attributed. These were made as follows:



```
# Set the indexes of the sections and products to start as 0

idx_product = 0
idx_section = 0

# Create a new column for the sections

df["Section"] = [[] for _ in range(len(products))]
```

Figure 4.16: ABC algorithm: setting indexes for products and sections, and creating a column for the sections

Next, a 'while' cycle was created to place all the products in the warehouse sections. Firstly, the cycle sets the product area as the product we are analysing and subtracts its value from the evaluated section. If the result is greater than zero, then space is still available in the section, placing the product in that section and incrementing the product's index. If the result is below zero, we place the product in that section and the following and then take the quantity left out of the first section, making its value the product's new area, to be deducted from the next section. This cycle continues until the product's or section's index surpasses the total number of products or sections, respectively. Figure 4.17 shows a flow diagram for this 'while' cycle and figure 4.18 the code for this part.

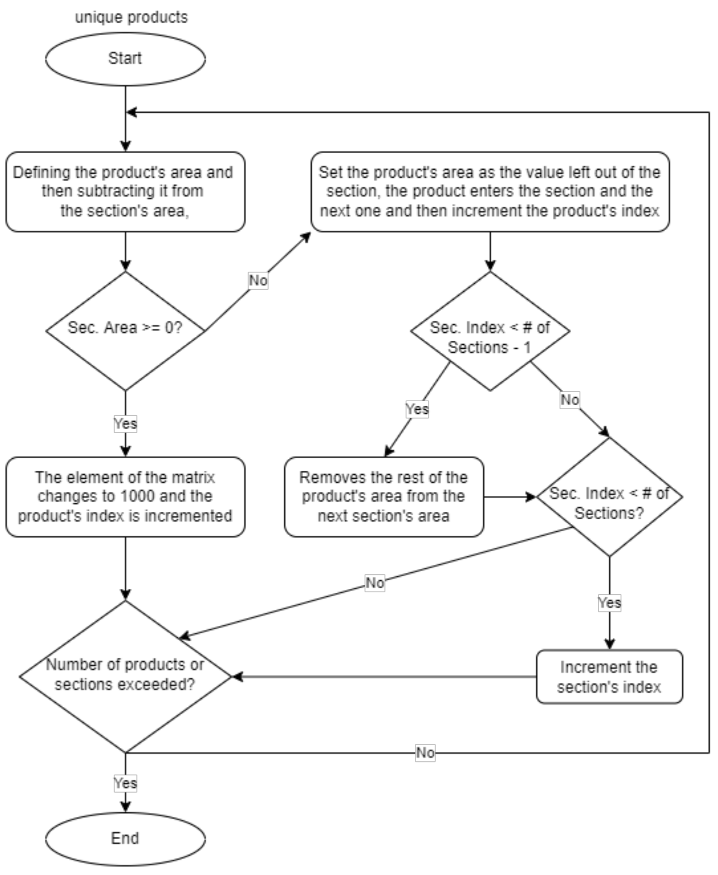
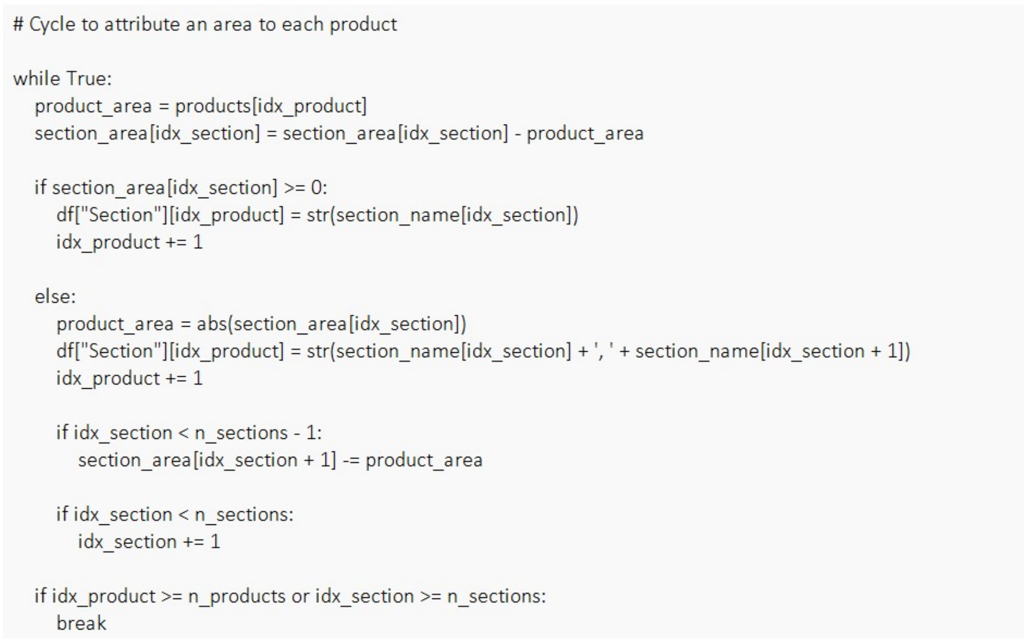

unique products
Start
Defining the product's area and then subtracting it from the section's area,
Sec. Area >= 0?
No
Set the product's area as the value left out of the section, the product enters the section and the next one and then increment the product's index
Sec. Index < # of Sections - 1
Yes
No
Yes
The element of the matrix changes to 1000 and the product's index is incremented
Removes the rest of the product's area from the next section's area
Sec. Index < # of Sections?
No
Yes
Number of products or sections exceeded?
Increment the section's index
No
Yes
End

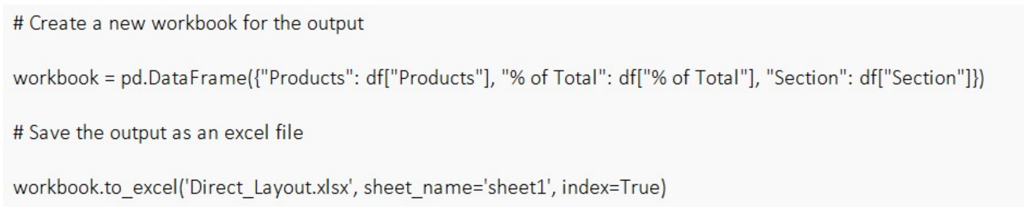
Figure 4.17: ABC allocation fluxogram



```
# Cycle to attribute an area to each product

while True:
   product_area = products[idx_product]
   section_area[idx_section] = section_area[idx_section] - product_area

   if section_area[idx_section] >= 0:
      df["Section"][idx_product] = str(section_name[idx_section])
      idx_product += 1

   else:
      product_area = abs(section_area[idx_section])
      df["Section"][idx_product] = str(section_name[idx_section] + ', ' + section_name[idx_section + 1])
      idx_product += 1

      if idx_section < n_sections - 1:
         section_area[idx_section + 1] -= product_area

      if idx_section < n_sections:
         idx_section += 1

   if idx_product >= n_products or idx_section >= n_sections:
      break
```

Figure 4.18: ABC algorithm: 'while' cycle to attribute a section to each product

To finalise this algorithm, a new workbook was created, extracting the products, the percentage of each product and the section in which the product was placed. Then, the data is placed in an excel file, resulting in figure 4.16. The algorithm for this part is in figure 4.19.



```
# Create a new workbook for the output

workbook = pd.DataFrame({"Products": df["Products"], "% of Total": df["% of Total"], "Section": df["Section"]})

# Save the output as an excel file

workbook.to_excel('Direct_Layout.xlsx', sheet_name='sheet1', index=True)
```

Figure 4.19: ABC algorithm: workbook for the output and saving as .xlsx file

Table 4.16: ABC algorithm output

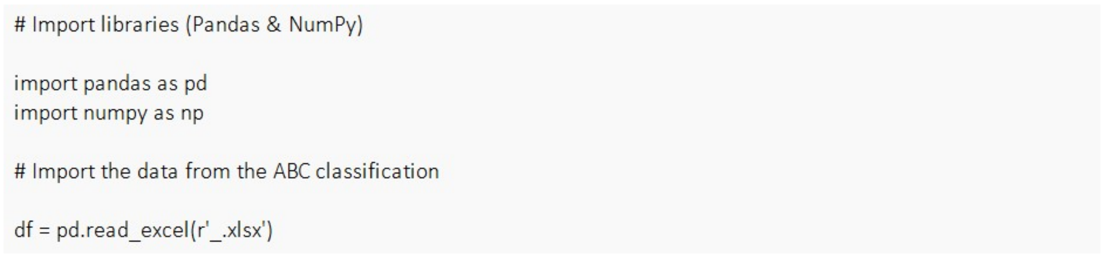
	Products	% of Total	Section
0	P*******L******VD	9.84	TE50
1	P*******L******VC	6.97	TE50, TE51
2	P*******L******VD	3.64	TE51, TE52
3	P*******L******GZ	3.15	TE52, TE53
4	P*******L******XB	2.65	TE53, TE54
5	P*******L******SB	2.19	TE54
...	...	...	...
778	P*******H******VK	0.00	TE63
779	P*******H******VJ	0.00	TE63
780	P*******H******VJ	0.00	TE63, TE61

4.1.4 Products allocation using the Knapsack problem

Now, the other approach made to the ABC classification regarding allocating the products will be described. Since it is a reality for many companies that items from a line product can not be placed in different sections of the warehouse, whether it is physical constraints or problems regarding software usage, an approach with the Knapsack problem was made. The idea is to allocate products according to their importance and consider the related capacities. This model only places the entire product's volume in each section, running many iterations to get the optimal result. The use of this method required the following five inputs:

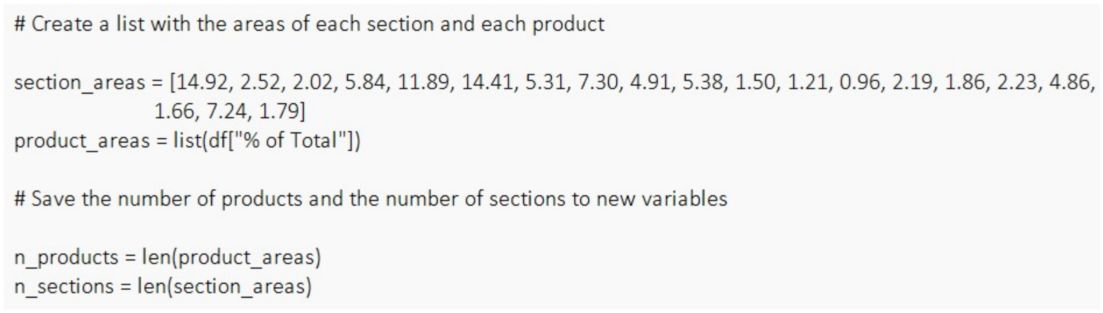
1. Total number of products;
2. Total number of sections;
3. Each product's occupation;
4. Each section's capacity;
5. Weights matrix.

To get all these, another algorithm was made. Firstly, the Pandas and NumPy libraries were imported and the excel file from the ABC classification, as shown in figure 4.20. Next, two lists were created, one with the constant section areas and another with the product areas, imported from the excel file. Then, two variables were created to store the total number of products and the total number of sections. These steps are in figure 4.21.



```
# Import libraries (Pandas & NumPy)

import pandas as pd
import numpy as np

# Import the data from the ABC classification

df = pd.read_excel(r'_.xlsx')
```

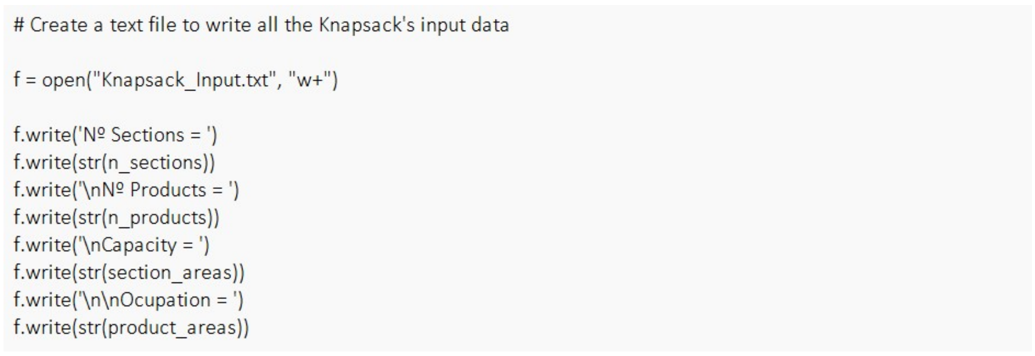
Figure 4.20: Pre-processing for the Knapsack algorithm: importing the libraries and reading the data



```
# Create a list with the areas of each section and each product

section_areas = [14.92, 2.52, 2.02, 5.84, 11.89, 14.41, 5.31, 7.30, 4.91, 5.38, 1.50, 1.21, 0.96, 2.19, 1.86, 2.23, 4.86,
                 1.66, 7.24, 1.79]
product_areas = list(df["% of Total"])

# Save the number of products and the number of sections to new variables

n_products = len(product_areas)
n_sections = len(section_areas)
```

Figure 4.21: Pre-processing for the Knapsack algorithm: lists for the section and product areas, and variables for the total number of products and sections

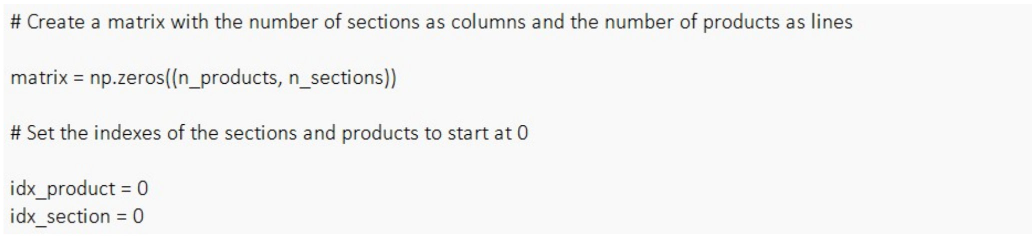
Since a text file is the desired output for this algorithm, a text file is opened in the algorithm and the following information is written: the number of sections, number of products, capacity, and occupation, and then leave the text file open to later finalise it with the weight's matrix.



```
# Create a text file to write all the Knapsack's input data

f = open("Knapsack_Input.txt", "w+")

f.write('Nº Sections = ')
f.write(str(n_sections))
f.write('\nNº Products = ')
f.write(str(n_products))
f.write('\nCapacity = ')
f.write(str(section_areas))
f.write('\n\nOcupation = ')
f.write(str(product_areas))
```

Figure 4.22: Pre-processing for the Knapsack algorithm: creating a text file to write the Knapsack's input data

Before starting the automation of the weights matrix, first a matrix with the correct size (number of sections as columns and number of products as lines) is created, and the product and section indexes are set to zero, as shown in figure 4.23.

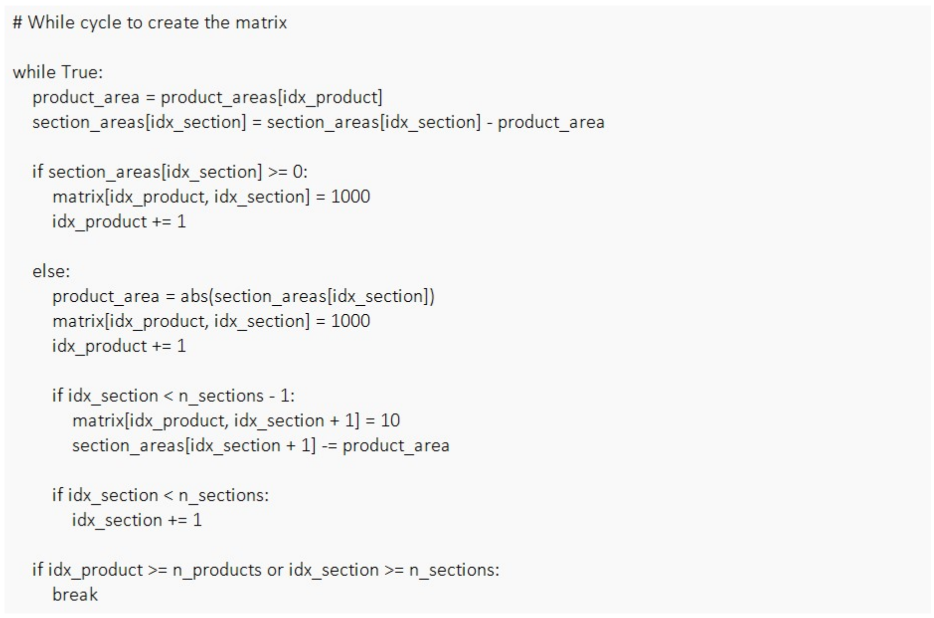


```
# Create a matrix with the number of sections as columns and the number of products as lines

matrix = np.zeros((n_products, n_sections))

# Set the indexes of the sections and products to start at 0

idx_product = 0
idx_section = 0
```

Figure 4.23: Pre-processing for the Knapsack algorithm: creating a matrix and indexes for the products and sections

Now, a 'while' cycle is created for the weights matrix. Regarding the value of each weight, only three number were used: zero, ten and a thousand. Zero is attributed if we do not want the product in the element's section, and one thousand is for the section we want the most to keep the product. Finally, ten goes to the second-best section for the product.

For the algorithm, first, the cycle sets the product area as the product we are analysing and then subtracts its value from the evaluated section. If the result is greater than zero, we use the product and section indexes to give that element a value of 1000 in the weight's matrix, meaning that it is that section the one we would prefer to place that product, and then increment the product's index. If the result is below zero, we still give that element the value of 1000, but since the product does not fit entirely in the section, we set a value of 10 for the next section regarding the same product. Since the product will not fit in the preferred section, it will be placed in the next section. This cycle runs until the product or section indexes surpass the total number of products or sections, respectively. Figure 4.24 demonstrates this part of the code and figure 4.25 shows a flow diagram for this 'while' cycle.



```
# While cycle to create the matrix

while True:
    product_area = product_areas[idx_product]
    section_areas[idx_section] = section_areas[idx_section] - product_area

    if section_areas[idx_section] >= 0:
        matrix[idx_product, idx_section] = 1000
        idx_product += 1

    else:
        product_area = abs(section_areas[idx_section])
        matrix[idx_product, idx_section] = 1000
        idx_product += 1

        if idx_section < n_sections - 1:
            matrix[idx_product, idx_section + 1] = 10
            section_areas[idx_section + 1] -= product_area

        if idx_section < n_sections:
            idx_section += 1

    if idx_product >= n_products or idx_section >= n_sections:
        break
```

Figure 4.24: Pre-processing for the Knapsack algorithm: 'while' cycle to get the weights matrix

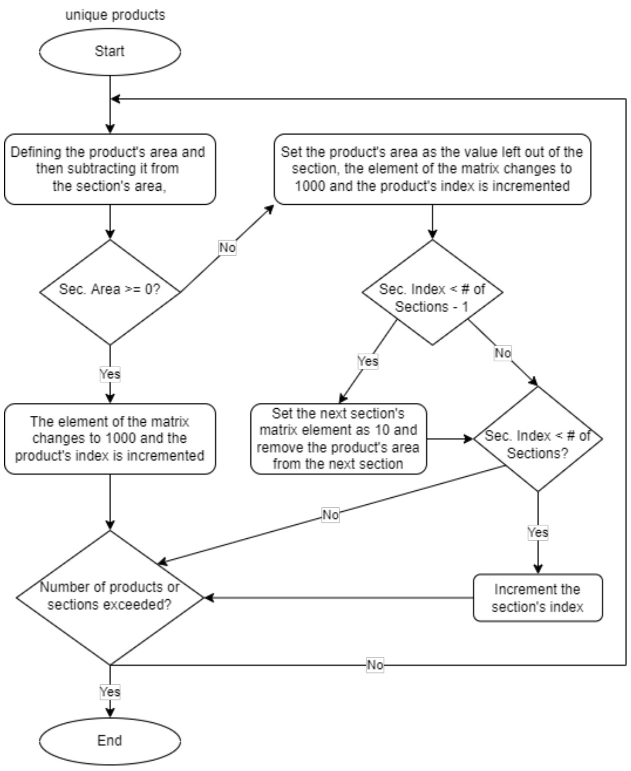
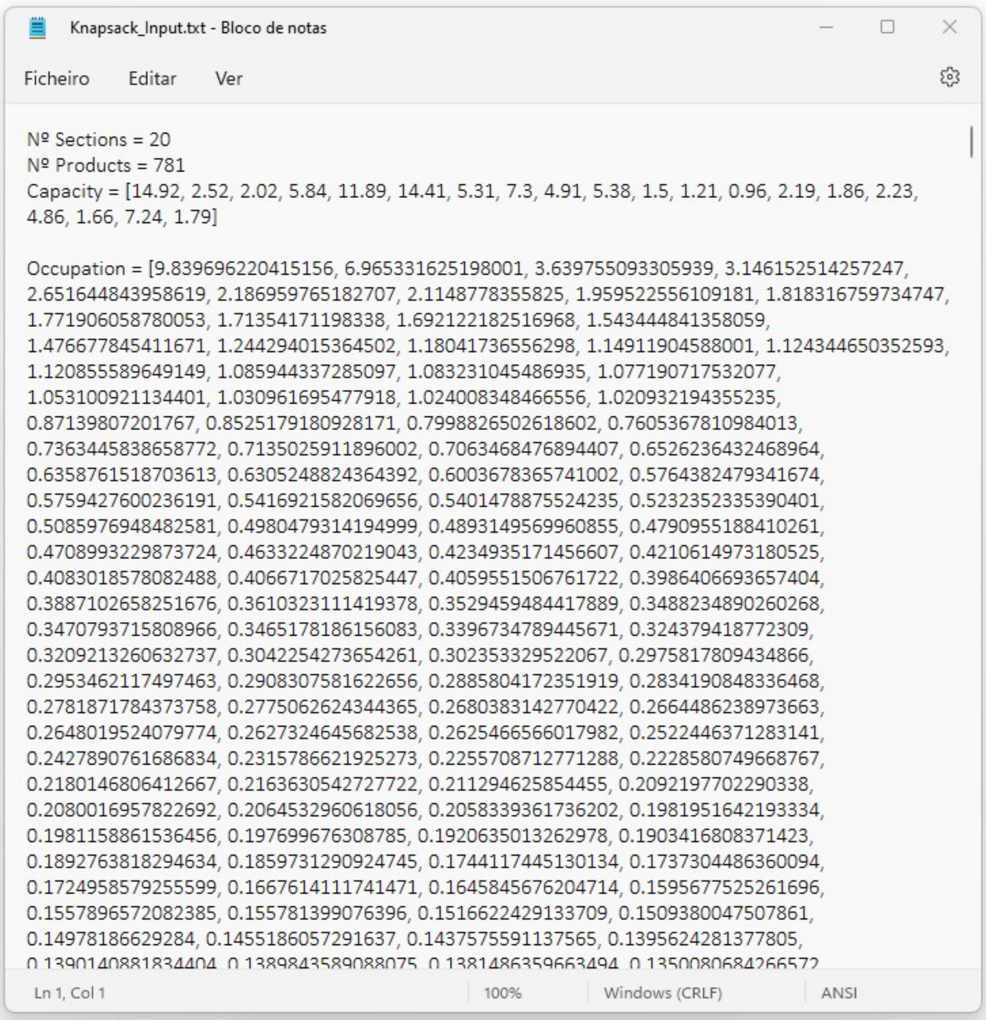

unique products
Start
Defining the product's area and then subtracting it from the section's area,
Sec. Area >= 0?
No
Yes
Set the product's area as the value left out of the section, the element of the matrix changes to 1000 and the product's index is incremented
Sec. Index < # of Sections - 1
Yes
No
The element of the matrix changes to 1000 and the product's index is incremented
Set the next section's matrix element as 10 and remove the product's area from the next section
Sec. Index < # of Sections?
No
Yes
Number of products or sections exceeded?
Increment the section's index
No
Yes
End

Figure 4.25: Knapsack allocation fluxogram

With the weights matrix automated, it is written in text file and then the file is closes. This final step of algorithm is shown in figure 4.26 and then figure 4.27 demonstrates a sample from the output text file created.

```
# Finalize the text file with the matrix

f.write('\n\nWeights = ')
np.savetxt(f, matrix, fmt='%0.0f')
f.close()
```

Figure 4.26: Pre-processing for the Knapsack algorithm: finalise the text file with the matrix



```
Nº Sections = 20
Nº Products = 781
Capacity = [14.92, 2.52, 2.02, 5.84, 11.89, 14.41, 5.31, 7.3, 4.91, 5.38, 1.5, 1.21, 0.96, 2.19, 1.86, 2.23,
4.86, 1.66, 7.24, 1.79]

Occupation = [9.839696220415156, 6.965331625198001, 3.639755093305939, 3.146152514257247,
2.651644843958619, 2.186959765182707, 2.1148778355825, 1.959522556109181, 1.818316759734747,
1.771906058780053, 1.71354171198338, 1.692122182516968, 1.543444841358059,
1.476677845411671, 1.244294015364502, 1.18041736556298, 1.14911904588001, 1.124344650352593,
1.120855589649149, 1.085944337285097, 1.083231045486935, 1.077190717532077,
1.053100921134401, 1.030961695477918, 1.024008348466556, 1.020932194355235,
0.87139807201767, 0.8525179180928171, 0.7998826502618602, 0.7605367810984013,
0.7363445838658772, 0.7135025911896002, 0.7063468476894407, 0.6526236432468964,
0.6358761518703613, 0.6305248824364392, 0.6003678365741002, 0.5764382479341674,
0.5759427600236191, 0.5416921582069656, 0.5401478875524235, 0.5232352335390401,
0.5085976948482581, 0.4980479314194999, 0.4893149569960855, 0.4790955188410261,
0.4708993229873724, 0.4633224870219043, 0.4234935171456607, 0.4210614973180525,
0.4083018578082488, 0.4066717025825447, 0.4059551506761722, 0.3986406693657404,
0.3887102658251676, 0.3610323111419378, 0.3529459484417889, 0.3488234890260268,
0.3470793715808966, 0.3465178186156083, 0.3396734789445671, 0.324379418772309,
0.3209213260632737, 0.3042254273654261, 0.302353329522067, 0.2975817809434866,
0.2953462117497463, 0.2908307581622656, 0.2885804172351919, 0.2834190848336468,
0.2781871784373758, 0.2775062624344365, 0.2680383142770422, 0.2664486238973663,
0.2648019524079774, 0.2627324645682538, 0.2625466566017982, 0.2522446371283141,
0.2427890761686834, 0.2315786621925273, 0.2255708712771288, 0.2228580749668767,
0.2180146806412667, 0.2163630542727722, 0.211294625854455, 0.2092197702290338,
0.2080016957822692, 0.2064532960618056, 0.2058339361736202, 0.1981951642193334,
0.1981158861536456, 0.197699676308785, 0.1920635013262978, 0.1903416808371423,
0.1892763818294634, 0.1859731290924745, 0.1744117445130134, 0.1737304486360094,
0.1724958579255599, 0.1667614111741471, 0.1645845676204714, 0.1595677525261696,
0.1557896572082385, 0.155781399076396, 0.1516622429133709, 0.1509380047507861,
0.14978186629284, 0.1455186057291637, 0.1437575591137565, 0.1395624281377805,
0.1390140881834404, 0.1389843589088075, 0.1381486359663494, 0.1350080684266572
```

Figure 4.27: Knapsack algorithm's text file output

Mathematical model

The Knapsack problem was briefly described in section 3.2.5. The following notations and variables were used for the model:

Notations

- P_i: Set of products, i ∈ {1,...,p};
- S_i: Set of sections, j ∈ {1,...,s};
- C_j: Section capacity area (in m^2);
- O_i: Product occupation area (in m^2);
- V_{ij}: Matrix of the values attributed to the product i for each section j, according to the ABC analysis.

Decision variable

$$x_{ij} = \begin{cases} 1, & \text{if the product } i \text{ is assigned to section } j \\ 0, & \text{otherwise} \end{cases} \quad \forall\, i \in P,\, \forall\, j \in S \tag{4.1}$$

Considering all the assumptions, variables and data, the idea is to assign all the products to the sections, ensuring all the constraints. Informally, the problem is to maximise the sum of the values of the items in the knapsack so that the sum of the weights is less than or equal to the knapsack's capacity. The mathematical model can be written as:

$$Max \sum_{i=1}^{p} \sum_{j=1}^{s} V_{ij} \times x_{ij} \tag{4.2}$$

subject to:

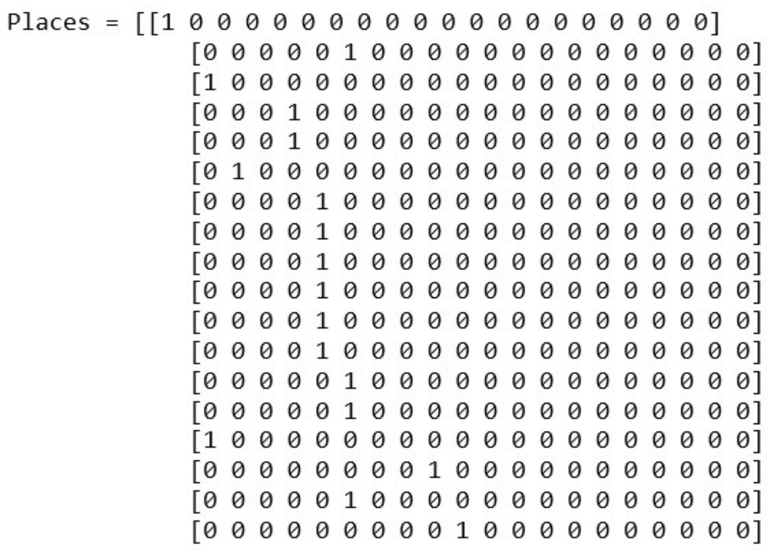
$$\sum_{j=1}^{s} x_{ij} = 1,\, \forall\, i \in P \tag{4.3}$$

$$\sum_{i=1}^{p} O_i \times x_{ij} \leq C_j,\, \forall\, j \in S \tag{4.4}$$

$$x_{ij} \in \{0,1\},\, \forall\, i \in P,\, \forall\, j \in S \tag{4.5}$$

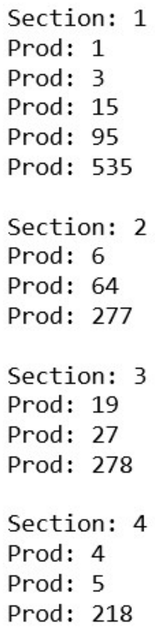
As mentioned, the objective function 4.2, according to the ABC analysis, intends to maximize the value of assigning products to sections. For this, it is necessary to ensure that each of the products is allocated to a single section (constraint 4.3) and that the capacity of the sections (in terms of area) is not exceeded (constraint 4.4). Finally, equation 4.5 defines the variables domain.

With all the data gathered for the knapsack input, the model was tested using the IBM ILOG CPLEX Optimiser. Since the analysis involves a massive amount of data and restrictions, trying to place 782 products optimally through 20 different sections, the computational processing time will be high. As expected, the first time the model was tested, it took around three hours to get the first results but only because the program was stopped when it hit a gap of 3.73%. It was taking too much time to increment from there, and a gap close to 3% gets results already very close to optimal. The program then returned a 782 by 20 matrix with ones and zeros regarding the placement of the products, and a list with the section, followed by the products that entered that section. Figure 4.28 exemplifies the model's matrix output, and figure 4.29 is the list output.



```
Places = [[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
          [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
          [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
          [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
          [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
          [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
          [0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
          [0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
          [0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
          [0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
          [0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
          [0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
          [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
          [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
          [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
          [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0]
          [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
          [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0]
```

Figure 4.28: Sample from the Knapsack's placing matrix output



```
Section: 1
Prod: 1
Prod: 3
Prod: 15
Prod: 95
Prod: 535

Section: 2
Prod: 6
Prod: 64
Prod: 277

Section: 3
Prod: 19
Prod: 27
Prod: 278

Section: 4
Prod: 4
Prod: 5
Prod: 218
```

Figure 4.29: Sample from the Knapsack's placing list output

This first attempt with the model's solution, transformed into a visual allocation of the products, got the figure's 4.30 warehouse organisation.

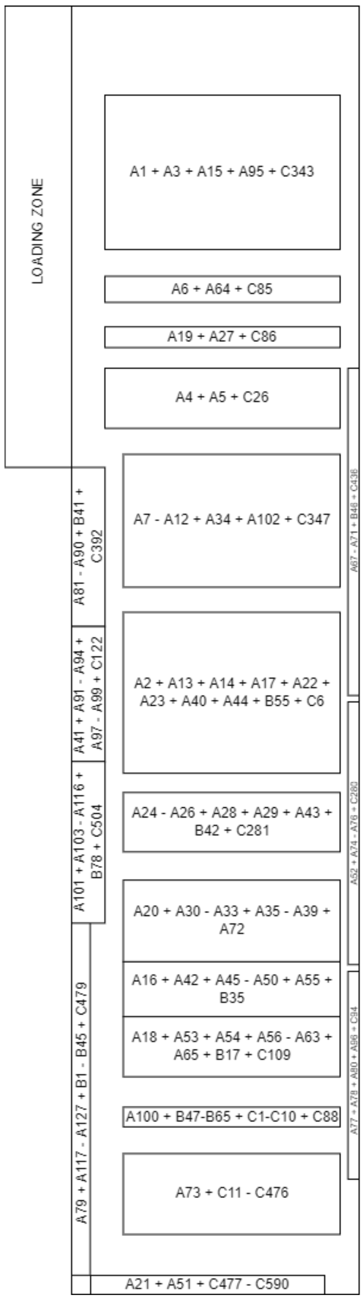

LOADING ZONE
A1 + A3 + A15 + A95 + C343
A6 + A64 + C85
A19 + A27 + C86
A4 + A5 + C26
A81 - A90 + B41 + C392
A7 - A12 + A34 + A102 + C347
A67 - A71 + B46 + C436
A41 + A91 - A94 + A97 - A99 + C122
A2 + A13 + A14 + A17 + A22 + A23 + A40 + A44 + B55 + C6
A101 + A103 - A116 + B78 + C504
A24 - A26 + A28 + A29 + A43 + B42 + C281
A52 + A74 - A76 + C280
A20 + A30 - A33 + A35 - A39 + A72
A16 + A42 + A45 - A50 + A55 + B35
A18 + A53 + A54 + A56 - A63 + A65 + B17 + C109
A79 + A117 - A127 + B1 - B45 + C479
A77 + A78 + A80 + A96 + C94
A100 + B47-B65 + C1-C10 + C88
A73 + C11 - C476
A21 + A51 + C477 - C590

Figure 4.30: Knapsack model's first layout

After the first attempt, the model was tested two more times. First, since this processing time is impracticable for an industrial application, a time limit of approximately 15 minutes was established to see how well the model could perform with a time restriction and then a test was made letting the program run without a time limit and without stopping it. With the time restriction, we hit a gap of 3.75%. Figures 4.31a and 4.31b present the results we got from these two analyses.

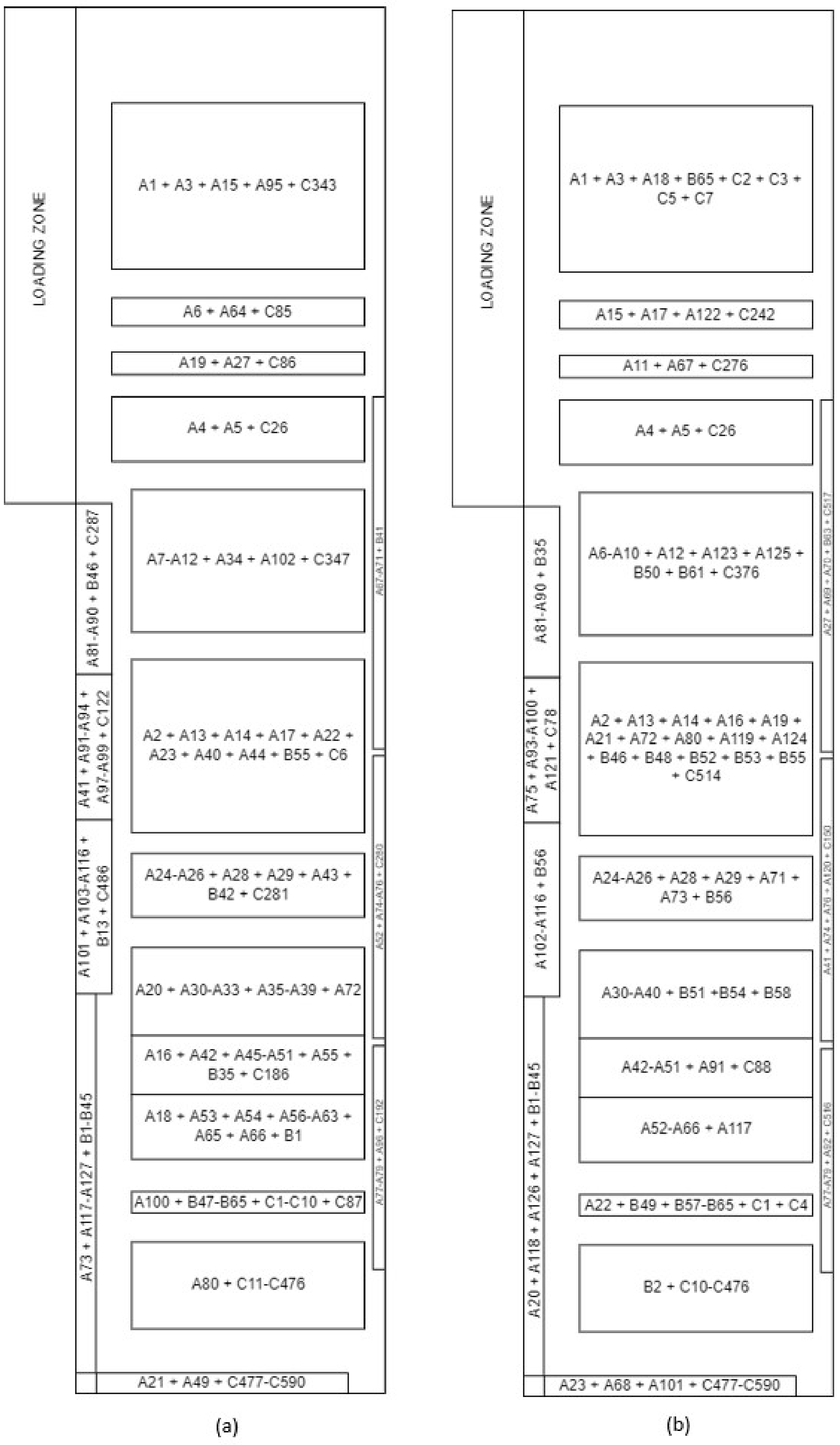

LOADING ZONE
A1 + A3 + A15 + A95 + C343
A6 + A64 + C85
A19 + A27 + C86
A4 + A5 + C26
A81-A90 + B46 + C287
A7-A12 + A34 + A102 + C347
A67-A71 + B41
A41 + A91-A94 + A97-A99 + C122
A2 + A13 + A14 + A17 + A22 + A23 + A40 + A44 + B55 + C6
A101 + A103-A116 + B13 + C486
A24-A26 + A28 + A29 + A43 + B42 + C281
A52 + A74-A76 + C280
A20 + A30-A33 + A35-A39 + A72
A16 + A42 + A45-A51 + A55 + B35 + C186
A18 + A53 + A54 + A56-A63 + A65 + A66 + B1
A77-A79 + A96 + C192
A73 + A117-A127 + B1-B45
A100 + B47-B65 + C1-C10 + C87
A80 + C11-C476
A21 + A49 + C477-C590
(a)
LOADING ZONE
A1 + A3 + A18 + B65 + C2 + C3 + C5 + C7
A15 + A17 + A122 + C242
A11 + A67 + C276
A4 + A5 + C26
A81-A90 + B35
A6-A10 + A12 + A123 + A125 + B50 + B61 + C376
A27 + A69 + A70 + B63 + C517
A75 + A93-A100 + A121 + C78
A2 + A13 + A14 + A16 + A19 + A21 + A72 + A80 + A119 + A124 + B46 + B48 + B52 + B53 + B55 + C514
A102-A116 + B56
A24-A26 + A28 + A29 + A71 + A73 + B56
A41 + A74 + A76 + A120 + C150
A30-A40 + B51 +B54 + B58
A42-A51 + A91 + C88
A52-A66 + A117
A77-A79 + A92 + C516
A20 + A118 + A126 + A127 + B1-B45
A22 + B49 + B57-B65 + C1 + C4
B2 + C10-C476
A23 + A68 + A101 + C477-C590
(b)

Figure 4.31: Knapsack model's: (a) second layout; (b) third layout

By comparing the three layouts, if time is of the essence, it is not worth to let the program run a complete analysis since it takes to much time and the first two attempts were very similar.

4.2 Results of the analyses

For many companies, storing all items from a single product in different sections of the warehouse is not possible and that is why we made the Knapsack's problem approach. If this is the case the results generated by the model were very good. Even though it is not very explicit due to the high number of products, it is clear the prioritisation of the higher class items from the front to the back of the warehouse.

If the company can set items from a product in different sections, we strongly recommend using the direct ABC allocation. If the classification of the products is done correctly, it will already give the list of products ordered from the most important to the least important. Considering that, it is straightforward that they should be placed in direct order from the section with higher priority to the section with least priority.

4.3 Proposed structure for the DSS

With all of the practical studies finalised, the DSS framework presented in figure 4.32 is proposed to meet the company's current needs. The most important input is definitely the customer orders because these dictate what the company should focus on producing and storing with greater importance, and this was the input that was able to be studied and manipulated for the DSS. Nonetheless, this input by itself leaves the analysis incomplete and unable to perform an optimal analysis.

Considering only customer orders, there is the lack of considering the actual products that are stored in the warehouse (there could be stock prior to the time frame used in the analysis). So there should be a cross-reference between the customer orders and the inventory happening inside the DSS.

Also, the production schedule is an important aspect to consider because it is what dictates the products that actually need storing and knowing in advance what the company is going to produce, combined with the orders they have, will probably result in a different classification of the importance of products.

All of our algorithms were made through python programming because the analysis presented here can all be done that way. After that, there should be a final module transforming the algorithm's output data to an actual layout, visually representing the results.

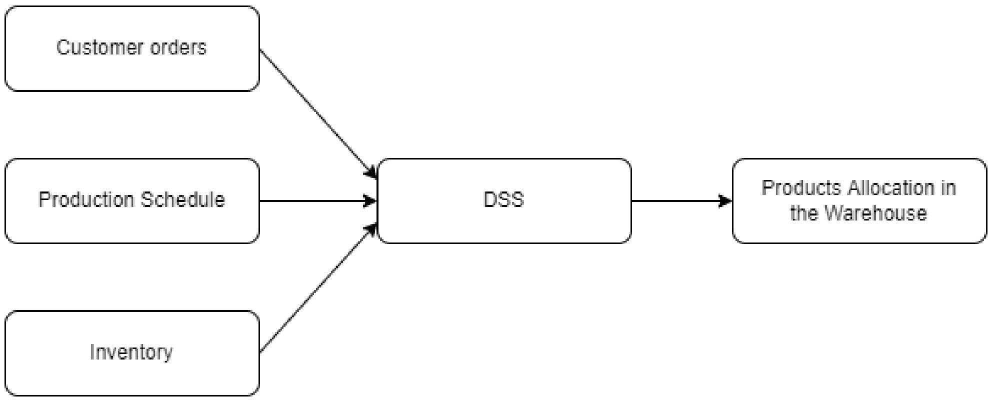


Figure 4.32: DSS framework

Intentionally blank page.

Chapter 5

Conclusions and future work

An extensive literature review as performed, narrowing it to the papers that directly served the purpose of this project. Then, several analyses were conducted to evaluate the company's products through the ABC, PQR and FSN methods, resulting in very satisfying results. This led to the creation of algorithms that automate the ABC classification and are directly compatible with the data extracted from the company's software. Finally, two more algorithms were provided that allocate the products according to the ABC classification and with two different kind of goals. With this work, the company has a clear base to continue their warehouse study and improve this system with other data they might find valuable.

The objectives established for this project were the reorganisation of the external warehouse, serving as a pilot project for later application to the internal warehouse of the company, and developing a model for a decision support system that can be applied to the external warehouse. The framework designed (presented in figure 4.32) is adequate since it that corresponds to the company's current needs. Unfortunately, there were not enough tests and experiments to get a system working with all of the proposed inputs, so in practical terms, the analysis of only one input was presented in this project. There is also an approach for categorising products and subsequent storage in the storage sections to identify the products better and improve the processes of storage and picking of the same. A proposal for integrating this approach into a decision support system and some important notes on how to continue the work started here are also presented.

It is important to mention that one aspect not achieved was the type of output of the DSS. The data can be extracted only as text/word/excel files as the final product of the system. Ideally, with this kind of work, there should be a visual representation of the output, most likely in the form of the warehouse's layout.

5.1 Limitations

As mentioned earlier in this document, a significant limitation was the lack of academic studies regarding warehouse decision support systems. This made the initial literature review work much longer than expected to gather different points of view from other fields of DSSs applications and analyse the problem with different strategies.

5.2 Future work

First, as mentioned in the previous section, the algorithms developed only consider the customer orders to make the ABC classification analysis. Future work to develop this project should lead with the existing analysis and combine it with the production schedule and the actual stock the company has to create a more robust classification.

Then, after using Knapsack's optimisation problem or other possible solution for attributing products to sections, there should be a component using the output information to generate a visual representation of the results.

Finally, further developments are expected in implementing innovative methods, models and algorithms to address the warehouse layout, storage allocation and storage assignment issues already in the presence of automated storage solutions. A helpful module integrating a cam interface for barcode reading could be implemented to support the introduction and registration of new SKUs. This feature could be used to address the issue of periodic and partial storage reorganisation rather than complete warehouse redevelopment.

Bibliography

[1] Brynne Henn. *Data Warehouses and Decision Support Systems (DSS) — Sisu Data*. URL: https://sisudata.com/blog/data-warehouses-decision-support-systems (visited on 06/12/2022).

[2] Dale D. Achabal et al. "A decision support system for vendor managed inventory". In: *Journal of Retailing* 76.4 (2000), pp. 430–454. ISSN: 00224359. DOI: 10.1016/S0022-4359(00)00037-3.

[3] Jennifer Shang et al. "A decision support system for managing inventory at GlaxoSmithKline". In: *Decision Support Systems* 46.1 (2008), pp. 1–13. ISSN: 01679236. DOI: 10.1016/j.dss.2008.04.004. URL: http://dx.doi.org/10.1016/j.dss.2008.04.004.

[4] Athanasios Spyridakos et al. "SAINC: Self-adapting inventory control decision support system for cement industries". In: *Operational Research* 9.2 (2009), pp. 183–198. ISSN: 11092858. DOI: 10.1007/s12351-008-0017-3.

[5] Tawfik M. Younis M. Tawfik El Masry. "Decision Support System in Supply Chain Management: Literature Review". In: 5.July (2016), pp. 1–23.

[6] Cecil C. Bozarth and Robert B. Handfield. *Introduction to Operations and Supply Chain Management*. Pearson Education, Inc, 2008.

[7] Daniel J. Power. *Decision Support Systems: Concepts and Resources for Managers*. 2002, p. 326. ISBN: 156720497X.

[8] Diana Cecilia Uribe Cadavid and Carlos Castro Zuluaga. "A framework for decision support system in inventory management area". In: *9th Latin American and Caribbean Conference for Engineering and Technology* (2011), WE1–7.

[9] Riccardo Accorsi, Riccardo Manzini, and Fausto Maranesi. "A decision-support system for the design and management of warehousing systems". In: *Computers in Industry* 65.1 (2014), pp. 175–186. ISSN: 01663615. DOI: 10.1016/j.compind.2013.08.007. URL: http://dx.doi.org/10.1016/j.compind.2013.08.007.

[10] Jim Lee. "A Decision Support System for Inventory". In: July (2014), pp. 513–522. URL: http://www.swdsi.org/swdsi08/paper/SWDSI%20Proceedings%20Paper%20S206.pdf.

[11] Germán González Rodríguez, Jose M. Gonzalez-Cava, and Juan Albino Méndez Pérez. "An intelligent decision support system for production planning based on machine learning". In: *Journal of Intelligent Manufacturing* 31.5 (2020), pp. 1257–1273. ISSN: 15728145. DOI: 10.1007/s10845-019-01510-y. URL: https://doi.org/10.1007/s10845-019-01510-y.

[12] Jinxiang Gu, Marc Goetschalckx, and Leon F. McGinnis. “Research on warehouse operation: A comprehensive review”. In: *European Journal of Operational Research* 177.1 (2007), pp. 1–21. ISSN: 03772217. DOI: 10.1016/j.ejor.2006.02.025.

[13] C.G.S. Rebelo et al. “The relevance of space analysis in warehouse management”. In: *Procedia Manufacturing* 55.2020 (2021), pp. 471–478. ISSN: 23519789. DOI: 10.1016/j.promfg.2021.10.064.

[14] *Warehouses: The boxes worth €300 billion — McKinsey.* URL: https://www.mckinsey.com/business-functions/operations/our-insights/warehouses-the-boxes-worth-300-billion (visited on 01/23/2022).

[15] Douglas M. Lambert, Martha C. Cooper, and Janus D. Pagh. “Supply Chain Management: Implementation Issues and Research Opportunities”. In: *The International Journal of Logistics Management* 9.2 (1998), pp. 1–20. ISSN: 17586550. DOI: 10.1108/09574099810805807.

[16] *What Industry 4.0 Means for Your Warehouse Operations – Multichannel Merchant.* URL: https://multichannelmerchant.com/blog/industry-4-0-means-warehouse-operations/ (visited on 01/30/2022).

[17] *WAREHOUSE 4.0: Accelerate Your Digital Journey — IndustryWeek.* URL: https://www.industryweek.com/sponsored/article/21146501/warehouse-40-accelerate-your-digital-journey (visited on 01/30/2022).

[18] James A. Tompkins et al. *Facilities Planning.* John Wiley Sons, 2003. ISBN: 9780470444047.

[19] José Crespo de Carvalho. *Logística e Gestão da Cadeia de Abastecimento.* Edições Sílabo, 2020.

[20] *ABC Analysis in Inventory Management: Benefits Best Practices — NetSuite.* URL: https://www.netsuite.com/portal/resource/articles/inventory-management/abc-inventory-analysis.shtml (visited on 03/16/2022).

[21] Ramakrishnan Ramanathan. “ABC inventory classification with multiple-criteria using weighted linear optimization”. In: *Computers and Operations Research* 33.3 (2006), pp. 695–700. ISSN: 03050548. DOI: 10.1016/j.cor.2004.07.014.

[22] *Pareto Principle Definition.* URL: https://www.investopedia.com/terms/p/paretoprinciple.asp (visited on 03/16/2022).

[23] Sara Raquel Miranda Martins. “Definição do layout de um armazém para a otimização de picking”. In: (2018).

[24] Numera Tahir and Muhammad Abbas Choudhary. “Development of a Decision Support System for Inventory Analysis and Control”. In: 1980 (2009).

[25] René de Koster, Tho Le-Duc, and Kees Jan Roodbergen. “Design and control of warehouse order picking: A literature review”. In: *European Journal of Operational Research* 182.2 (2007), pp. 481–501. ISSN: 03772217. DOI: 10.1016/j.ejor.2006.07.009.

[26] Ivone Silva Murakami and Dreli dos Santos Ferreira. “Estudo de caso da análise da gestão de estoques de insumos hospitalares com foco nas classificações”. In: (2013).

[27] Bruno Martins Moreira, Natan Felipe Silva, and Daniel Gonçalves Ebias. "Aplicação das Curvas PQR e ABC como base para o desenvolvimento da estratégia de gestão de estoques em uma indústria farmacêutica do centro-oeste mineiro". In: (2019).

[28] Valentina Cacchiani et al. "Knapsack problems — An overview of recent advances. Part II: Multiple, multidimensional, and quadratic knapsack problems". In: *Computers and Operations Research* 143.October 2021 (2022), p. 105693. ISSN: 03050548. DOI: `10.1016/j.cor.2021.105693`. URL: `https://doi.org/10.1016/j.cor.2021.105693`.