DEFINING A MACHINE LEARNING IMPLEMENTATION FOR DEMAND FORECASTING IN DENTAL TRACKING SYSTEM®

Camilo Andres Macias Vargas
Student number: 41170

Master of Science Thesis
Supervisor: Dr. Sepinoud Azimi
Åbo Akademi University
Faculty of Science and Engineering
Department of Information Technology
December 2018

Camilo Andrés Macias Vargas

ABSTRACT

In this thesis we conduct a comparison between the non-linear SVR (Support Vector Regressor) and the LSTM (Long-Short-Term Memory) Recurrent Neural models, for the prediction of dental assets demand on institutions using the DTS (Dental Tracking System) software developed by the company LM-Instruments. DTS is a Java programming language application that tracks usage of dental assets based on RFID (Radio Frequency Identification) technology. This work describes the generation of the data in DTS, defines what data is considered relevant to be used in the demand forecast; and presents the process of data preparation, as well as the development of the models and the validation process. Finally, the best results are presented, analysed and compared. This comparison reviews the performance, implementation simplicity and computational power required. After the work performed, the suggestion for LM-Instruments is to use SVR with a polynomial kernel of degree one, using low penalty, epsilon and coefficient values. This implementation proved to have the best performance when using weekly-based data sets over daily data sets.

Keywords: LM-Instruments, dental, tracking, SVR, LSTM, forecast, demand, DTS

CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

ANN
Artificial Neural Network. Technique of ML.

API
Application Programming Interface.

BPTT
Back Propagation Through Time. Technique used by NN.

DL
Deep Learning. Type of learning performed by DNN.

DNN
Deep Neural Network. Type of NN.

DTS
Dental Tracking System. Software that will make use of the implementation of this thesis.

FFNN
Feed Forward Neural Network. Special type of NN.

GUI
Graphic User Interface.

JIC
Just-In-Case. Strategy of inventory management.

JIT
Just-In-Time. Strategy of inventory management.

JSON
JavaScript Object Notation. Protocol used for data transferring between computing systems.

LMI
Short for LM-Instruments; Company developing DTS.

LSTM
Long-Short Term Memory. A special type of RNN.

MAE
Mean Absolute Error. Measure of difference between two continuous values.

ML
Machine Learning.

MLP
Multi-Layer Perceptron. A class of FFNN.

MLR
Multiple Linear Regression. Technique of ML with linear representation that uses more than one predictor variable.

NN
Neural Network. Technique of ML.

OECD
Organization for Economic Co-operation and Development.

RBF
Radial Basis Function. Kernel available for use in SVR.

RFID
Radio Frequency Identification. Technology used by DTS for tracking dental assets.

RNN
Recurrent Neural Network. Special type of NN.

SC
Supply Chain.

SCM
Supply Chain Management. Set of practices that add business value to customers.

SVC
Support Vector Classification. Technique of ML derived from SVM.

SVM
Support Vector Machines. Well known technique of ML.

SVR
Support Vector Regression. Technique of ML derived from SVM, used for development during the thesis.

UI
User Interface.

VMI
Vendor Managed Inventory. Strategy of inventory management.

1 INTRODUCTION

This chapter provides a review of the background, concepts and methodologies relevant to this study. In a similar way, it presents the problem we intend to solve, as well as what has been done, where we are and how we can use that previous work to solve our problem.

1.1 Background

It is universally acknowledged that the ultimate goal of every business is to receive revenue for a given service or product. As a part of the normal evolution, every company in every single industry intends to optimize its business process to offer the best quality product or service with the least effort and cost.

There are many fronts at which a company fights different battles to achieve this type of improvement or evolution. Some of these are external to the company itself, but some others are internal and depend on the way the company and its internal processes work and interact with each other. Furthermore, the fronts, battles, external and internal entities, and therefore processes, all of them vary depending on the nature of the company and the industry in which it is involved.

This work will be focused on the Healthcare industry as the proposed case study demands.

1.1.1 Health Care Industry

Health is defined as the state of well-being in three different aspects: physical, mental and social [5]. It is not difficult to understand how important the role of the health care industry is in our society and our lives. Due to its importance, the performance and development of all the companies and entities involved in this industry are critical for the whole humankind as end customers.

Besides the well-known criticality of the industry, another major characteristic to highlight is its size. A sub-categorization of industries can be created based on each company's business role inside the health care industry. The possible categories are "pharmaceuticals, biotechnology, equipment, distribution, facilities, and managed health care" [6]. Among these industries, the present work will be focusing mainly on the distribution industry, which is represented by "distributors and wholesalers of health care products" [6], as well as the facilities industry, which is represented by the health care provider's locations where the patients receive direct treatments, such as hospitals, nursing homes, among others [6].

In general, all sub-industries have a significant and important role to play for the correct functioning of the industry. However, the importance of these two sub-industries is based on the role they play in the chain of the health service. Facilities, as explained previously, are the physical locations in which patients receive treatments and, therefore, they could be defined as the heart of the healthcare industry. In a similar way, as facilities are dependent on the distribution to obtain the required resources to serve the patients, i.e. medicines, equipment and instrumentation, it is possible to state that based on the purpose and the criticality of its role, the distribution sub-industry resembles the veins of the human body for the healthcare sector. According to OECD's (Organisation for Economic Co-operation and Development) report of the healthcare sector performance for 2016 [7], one of the actions to be taken in order to achieve a healthcare service more accessible especially for ageing populations and improve the quality of service, is to have a more resilient healthcare system that makes use of new technologies, for which changes need to be done [7]. Furthermore, they state that due to previous economic crisis, "health spending growth has slowed significantly across Europe" and because of that, many companies intend to diminish the money expenses "by reducing the lengths of stays in hospital and pharmaceutical costs, and also by lowering administrative costs" [7]. It is worth highlighting the importance that the adoption of technology in health services has for the OECD when stating that "Health care is an information-intensive endeavour, and adoption of digital technology and eHealth can enable such improvement" [7], when talking about challenges such as "demographic change, rising chronic disease and multi-morbidity, along with fiscal pressures" [7] and the sustainability of the health system all across Europe.

Furthermore, the OECD signals "improved quality of care, better planning and resource allocation, and enhancing the evidence base for health service delivery and

policy making" as some of the benefits that the adoption of technology in the health services and, what they define as eHealth, brings to the industry [7]. The final purpose of this thesis work is to provide better planning and resource allocation by means that will be explained later in this paper.

1.1.2 Supply Chain Management (SCM)

Supply Chain Management is defined as "the integration of key business processes from end user through original suppliers" that produce and provide products, services and information which represent added value for customers and stakeholders [8].

Based on this definition, SCM can be perceived as another step in the evolution of the business world, in which companies have started to see the benefits of cooperating and interacting in the market as partners, leaving aside the individualism and trying to achieve better customer experiences through collaboration.

For a better understanding of SCM and its practices, the eight supply chain processes identified by the global supply chain forum are presented below:

1. Customer relationship management
2. Customer service management
3. Demand management
4. Order fulfilment
5. Manufacturing flow management
6. Supplier relationship management
7. Product development and commercialization
8. Returns management

Customer Relationship Management

This process defines how the relation and interaction with the customer are maintained; besides identifying "customers and customer groups to be targeted as part of the firm's business mission." [9].

For every customer, the company performs an analysis and defines a product and service agreement trying to mold it as much as possible to the characteristics of each customer, without completely losing the standardization of the offered product or service. The final purpose is to achieve profitability as high as possible, for both the company and the customer.

Customer Service Management

It is presented as the provider of "the single source of customer information, such as product availability, shipping dates and order status" [9]. The firm intends to respond in the best way possible to any event or action, triggered by the customer. This response should be as effective as possible, as no disruption should take place in the customer's business process or in the firm's own process.

Demand Management

This process refers to the manner in which the demand should be handled, thus finding a balance between the requirements of the customers and the supply capabilities of the company. Some useful tasks include "forecasting demand and synchronizing it with production, procurement and distribution" [9].

This specific process in the SCM is of high relevance, as the current thesis work intends to implement a demand forecasting solution based on machine learning (ML) which will be embedded in a software application to provide customers with tools that improve the demand process as part of their internal SCM practices.

An additional key matter in the demand management process is the need for a contingency plan, which defines how to react to unexpected events that break the balance between customer requests and supply capabilities.

Order Fulfillment

According to [9], an integration between the manufacturing process and logistics, together with marketing, is key for achieving an optimal order fulfillment by stating that the "firm should develop partnerships with key members of the supply chain to meet customer requirements" besides reducing the total cost for customers [9].

Similarly, a less formal definition of fulfillment logistics describes it as "what happens after the order is placed" [10]. It also proceeds to highlight the importance of

order fulfillment in e-commerce today. The authors do so, through the presentation of the Amazon success story, and explain why its success is an excellent proof of how important the order fulfillment process is as part of the SCM. "Amazon.com envisions a 'killer' supply chain that can deliver virtually any product —not just books— directly to customers better than its competitors" [10].

In the order fulfillment process, the firm must define how an order is handled, from its generation from the customer side, to the delivery at the customer's facilities.

Manufacturing Flow Management

As stated by [9], this process establishes the management of the product's flow through the manufacturing stages including transportation. The final objective of this process is to establish the manufacturing flexibility for the firm to be able to serve different market targets, considering the requirements of each of them.

Supplier Relationship Management

A very informal but precise way to describe this process, is as a mirror image of customer relationship management [9]. This is because, whereas customer relationship management defines how the interactions with the customers are maintained, the supplier relationship management defines the interaction of a firm with its supplier. Likewise, as it is done in the customer relationship management, a product and service agreement should be defined for the supplier relationship management.

The prior is important due to the fact that relationships with clients and suppliers define how fluent and dynamic the supply chain is.

Product Development and Commercialization

As part of its strategic process, [9] suggest that the definition of product development and commercialization should be aligned with the sourcing, manufacturing and marketing strategies, as they might have an impact on the product development process and its commercialization.

A company can benefit both internally and externally from using teams in its product development process. On one hand, "successful teams have accelerated the product development cycle, reduced development costs, and increased the quality of

new products" inside a company [11]; on the other hand "NPD (New Product Development) teams are associated with increased product success in the market, more frequent introduction of new products, and higher customer satisfaction" [11].

Returns Management

This process defines the way a firm should handle returned products. As some authors suggest, the first step to ensure a working process for handling returned products is the return avoidance, which can be defined as "manufacturing and selling the product in a manner such that returns are minimized" [9], in other words, ensuring product quality and good supply process for customers requests.

Furthermore, [9] mention that the returns management process should consider laws for treating a product for disposal in a proper way, as well as decide whether a returned product can be recycled, repaired, or even directed to a market of second-hand or used "goods".

Based on the review of the definition of SCM and the processes that it comprehends, it becomes clear what its purpose is and why it is crucial for companies to implement SCM processes and practices to achieve optimal performance in this respect.

One of the reasons for this work and its case study gaining value is the implementation of a solution for the improvement of SCM in the healthcare sector; given that according to [12], the healthcare sector, in general, has not made as much profit or improved as much as other sectors have with the adoption of SCM practices.

In the same work, the authors conduct a research showing that one of the barriers for healthcare provider institutions to adopt SCM practices is the existence of conflicting goals between the stakeholders of the supply chain, as well as the existence of intra-institutional conflicts, by stating that "participants commented that conflicts usually arise due to requests for reference items and unrealistically high inventories" [12]. Based on that, it stands to reason that some of these conflicts could be avoided by having a system which presents the correct numbers to the material manager and prevents making decisions based on misinformation and incorrect data about the items in stock and their specific location. Moreover, the authors also suggest that "computer software applications should be used to calculate order quantities based on demand forecasting and safety stock levels" [12], which would allow institutions to use personnel to more important tasks, such as patient procedures, instead of wasting human resources

on tasks that do not relate to the main service purpose and that could additionally be automatized.

1.1.3 Inventory Management

Inventory Management is defined as the mix of "theories, functions and management skills involved in controlling an inventory" [13]. However, its main purpose is not only to control an inventory but to do it in an efficient way, minimizing the cost and maximizing the profit out of the inventory items involved in a given business.

By virtue of the prior definition, inventory management can solve the necessity of the European health sector according to the OECD; which is better planning and resource allocation, since better management of the items in an inventory would not only help to continue attending the demand for the items but also dispose of inventory items only when and where it is required, which allows the healthcare service providers to plan better and allocate resources in a more efficient way. For this reason, inventory management is an essential part of the supply chain, as it plays an important role in the process flow of assets, equipment and medicines into the facilities.

After reviewing the role of inventory management in the supply chain, a simple yet very clear and natural definition presents inventory management as "supervision of supply, storage and accessibility of items in order to ensure an adequate supply without excessive oversupply" [14].

From a more generalized point of view, [15] reviews the inventory management technique and analyzes what the objectives and constraints in it are. Below, the objectives of inventory management, according to [15], are presented.

1. Cost minimization
2. Profit maximization
3. Maximization of rate of return on stock investment
4. Determination of a feasible solution
5. Keeping at an acceptable level the amount of human effort expended in the management and control of inventories
6. Ensuring flexibility to cope with an uncertain future

7. Minimizing political conflicts (in terms of the competing interests) within the organization

8. Maximizing the chance of survival of the individual manager's position or of the firm itself

Even though all the objectives explain clearly what benefits the inventory management techniques can bring to the organizations and why are they so important for different type of businesses nowadays, there is one objective that might not be addressed by most of the enterprises when handling inventories and stock item levels, as it is the amount of human effort. It is universally acknowledged that employees are if not the most, one of the most important resources that any company could have for many different reasons. However, it might not have enough relevance in some cases the fact that human effort as a resource, could be more profitable when it is used to perform core-related tasks instead of controlling and managing an inventory. On the one hand, in the ideal scenario, it is possible to talk about a self-managing inventory that requires no human effort at all, which takes care of all the steps in the replenishment cycle; however, it is not a secret that such a solution is not easy to achieve, if achievable at all, with the current technological advances. However, if such a solution exists, it will be hardly affordable by the vast majority of enterprises worldwide, due to its high cost. On the other hand, in a more realistic scenario, it is key to implement practices that allow the employees to focus in more business-core related tasks than the managing and controlling of the inventory, as the latter type of tasks can be automated; resulting in better resource utilization.

In a similar way, the author in [15] defines the constraints of inventory management as follows:

1. Supplier constraints – minimum order sizes, restrictions to certain pack sizes, maximum order quantities, restrictions on replenishment times.

2. Marketing constraints – minimum tolerable customer service levels, where service can be measured in a number of ways.

3. Internal constraints – storage space limitations, maximum budget to be used for purchases during a particular period, maximum workload (number of replenishments per period), personnel involved (capabilities and attitudes).

The previous are limitations that might arise and have to be considered when implementing inventory management practices.

One of these practices is explained in [16], where the author refers to Just-In-Time (JIT), by stating that it "is an idealized concept of inventory management wherein we are able to supply" any material at any time "just in time with 100% supply assurances without keeping any inventory on hand". The author goes further and explains this system with a perfect analogy when comparing JIT to "the supply of oxygen to the human body" [16].

As opposed to JIT, another strategy has also been widespread around companies of many industries. This strategy is Just-In-Case (JIC) which consists of keeping large inventory stocks to minimize the probability of not being able to serve a customer request due to an item being out of stock.

These two strategies reflect the challenge that every company faces when deciding on the right size of the stock: keeping too much or too low.

The authors in [17], not only explain these two problems but also explain what consequences both could bring to the enterprise. First, according to [17], a too low stock can lead not only to a customer request unfulfilled, but also to a big impact on the business, "since lost orders may lose a customer altogether, and dissatisfied customers often complain about the lack of availability. If this dissatisfaction spreads, the long-term impact causes a reduction in profitability". Moreover, the human effort might increase, considering that "firefighting, although often accompanied by a feeling of success, uses a great deal of time and energy" [17]. Second, a too high stock level can lead to an unnecessarily big amount of money spent on holding and managing the items. Furthermore, overstock "impacts on the space where inventory is held, whether on the shop floor or in a warehouse" [17]. Moreover, overstock could bring difficulties when reacting to changes, due to "the number of parts that need to have changes made to them is correspondingly large" [17]. This is true, especially when a change in the market demand or as a security policy needs to be done.

1.1.4 Machine Learning

Machine learning (ML), is defined as one of the branches of computational algorithms, which is designed to emulate human intelligence by the process of "learning" from previous experience and the surrounding environment [18]. The term "learning" can be explained as "the process of converting experience into expertise or knowledge" [19].

Furthermore, a mapping between the general process of a ML algorithm and the process of learning can be constructed by stating that "the input to a learning algorithm is training data, representing experience, and the output is some expertise, which usually takes the form of another computer program that can perform some task" [19]. A more technical definition of ML states that "a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" [20].

However, ML comprehends different types of learning. The division can vary according to the authors [19–24]. Nonetheless, there are two types of learning that are the most popular ones and are recognized by every author, which are supervised and unsupervised learning.

Types of Learning

A definition of supervised and unsupervised learning will be presented, in order to have a better understanding of where the focus of this work relies on, and the characteristics of the problem it is intended to solve.

According to [19], the difference between these two types of learning can be described as a characteristic of the nature of the interaction between the learner, i.e. ML model, and the environment. In this specific case, the interaction related to the "experience" provided to the learner with the training data, which for **supervised learning** contains the labels or the output data that the model will try to predict for the samples in the test set. On the other hand, for **unsupervised learning** the "right answer" is not provided in the training data, and the objective is for the learner to interpret the data in its own way to detect patterns or any other characteristic that can be valuable to understand the data.

More official definitions of supervised and unsupervised learning will be reviewed below.

- Supervised Learning: algorithms of this type are composed of a dependent/target variable, and a set of independent variables/predictors. The aim of this type of learning is to "learn a mapping from the input to an output whose correct values are provided by a supervisor" [21].

- Unsupervised Learning: algorithms of this type do not require a dependent/target variable. The objective is to "find the regularities in the input" [21], which the

author defines as patterns that occur more often in the input data.

Other types of learning exist, which stand in the middle of supervised and unsupervised learning, but those will not be reviewed in depth as they are not considered relevant for the current work.

Types of Problems

Similarly, the problems that can be solved with ML exhibit a considerable variation, based on the existent techniques and application fields.

However, in order to understand in a more clear way the problem that will be addressed and concepts that will be used later in this work, only the two most well known and most general types of problems will be presented.

- Regression Problem: can be defined as the task of predicting a numeric value [22]. The most important characteristic of this type of problem, as mentioned before, is the type of data of the output y. Such type of data is a continuous value, generally quantities; such as amounts, prices or sizes.
- Classification Problem: is defined as the task of predicting classes [22] or labels. The type of output value in this type of problems is defined as discrete.

It is important to notice that these two types of problems can belong to supervised or unsupervised learning, though it is more common to address them as supervised learning.

Support Vector Regressor

SVR (Support Vector Regressor) is a technique developed and presented by Vladimir Vapnik together with other researchers, based on the concept of support vectors, developed by Vapnik in previous years [25].

A more in-depth review of SVR can be found in [26], which details the development of the technique and its role as the basis for the creation of new models.

SVM (Support Vector Machines), is widely known for outperforming other techniques as it has been demonstrated in [27–30]. This is due to the benefits provided by the way SVR achieves a better generalization when performing the learning process [26].

The good generalization of this technique together with its proven good performance over time series data predictions, support the choice of SVR as one of the techniques to use in this work's case study. Furthermore, one of the first challenges faced during this work was the transformation of the data to make it valuable for the ML model to interpret and be able to predict accurately. This problem can be further addressed by SVR, as "SVMs are state-of-the-art tools for nonlinear input-output knowledge discovery" [26], which means SVR has the ability to easily discover the relationship between the features and the target output.

Similarly, [30] perform some tests on SVR with a chaotic time series dataset used also in [31], where "different approximation techniques (polynomial, rational, local polynomial, Radial Basis Functions with multiquadrics as basis function, and Neural Networks)" [30] were put to test in order to find which model performed better for chaotic time series data prediction. As a result, [30] found that SVR outperformed most of the techniques previously tested. Likewise, tests were also conducted to identify how sensitive SVM is regarding the hyper parameters ϵ and the penalty constant C, as well as the dimensions of the predictor. According to [30], the parameter C has little effect on the generalization error as well as the number of support vectors, whereas the value of ϵ has a big impact and can cause overfitting when the dimensionality of the feature space is too big. This risk will be reviewed later in this paper.

Recurrent Neural Networks

A NN (Neural Network) can be defined as "a wide class of flexible nonlinear regression and discriminant models, data reduction models, and nonlinear dynamical systems" [32].

However, a non-technical explanation, such as the one presented by [33], describes the NN as devices used for solving different problems that neurobiological systems can solve effortlessly.

More precisely, a Recurrent Neural Network (RNN) is a type of neural network used in ML for complex problems solving, mainly related with sequential data, i.e. data related with other data in space or time. RNNs are defined as "connectionist models with the ability to selectively pass information across sequence steps, while processing sequential data one element at a time" [34].

Another definition, slightly more complex yet more complete, states that an RNN is similar to a standard multilayer perceptron, "with the distinction that we allow con-

nections among hidden units associated with a time delay" [3]. These connections are used by the model to retain past information, making the network capable of discovering "temporal correlations between events that are far away from each other in the data" [3].

The term LSTM stands for Long-Short Term Memory, and could be considered as an "extension" of RNN, which allows it to remember information even for a longer period of time, i.e. more time steps than it would otherwise do.

The prior means that, by having additional recurrent connections among the hidden neurons, RNNs are able to store information from the past in the state of the model, which is passed from neuron to neuron. In this state, information from different points in time can be stored together and therefore the model is able to define relationships between them. In case the model is an LSTM RNN, the time steps between the pieces of data stored could differ to a larger extent.

Another benefit of LSTM over regular RNN is its ability to avoid a well-recognized and common issue during the training process, which is the fact that it is considered "hard to train properly" [3], even though it is also recognized as a simple and powerful model. The problems addressed in [3], are the "vanishing gradient" which is solved by LSTM, and the "exploding gradient" which might, however, be present in an LSTM network as well. These issues will not be reviewed as they are not in the scope of this work; however, the implemented solution will be presented later in this thesis.

1.1.5 Machine Learning in SCM

Throughout the years, many different techniques have come to the scene in order to help the companies increase the profit of each resource while lowering the management costs. One of those techniques is machine learning, which can be used in many different ways to solve a wide variety of problems. Demand forecasting is a traversal issue that affects many different industries and companies around the world, and machine learning has emerged as one more option for the companies to solve this problem. If a company has the possibility to know in advance a given product or service demand, it is possible to align the business process to serve this demand, which could result in lowering costs, thus increasing the revenue.

A good example of how ML can solve this, is presented in [35] in which different machine learning techniques such as multiple linear regression, neural networks, recurrent neural networks and support vector machines are used "to forecast the distorted

demand at the end of a supply chain (bullwhip effect)" [35]. The results of the already mentioned ML techniques are then compared with the result of more traditional processes such as "naive forecasting, trend and moving average" [35]. For the previously mentioned work, the authors used two different data sets, the first one provided by a simulated supply chain, and the second one taken from Canadian Foundries orders. Both data sets were divided into a set of training data and another set for testing which would enable performance measurement and comparison through the MAE (Mean Absolute Error) for the predictions of each forecasting model. The result of this work showed, as described by the authors, that "there was no statistically significant differences in terms of the accuracy of forecasts among RNN, SVM, NN, and MLR" [35]. The prior means that even though RNN and SVM provided the most accurate predictions, the difference was not significant and, therefore, based on the predictions' accuracy and the implementation complexity, MLR could be more worth to use in this case.

Although similar, the problem intended to be solved in this thesis differs in many points from the one tackled by [35], due to the nature of the supply chain, as no external entities are involved in our case study. As will be reviewed later, the target forecasted demand does not correspond to that of a supply chain, but to a cyclic process flow instead. Similarly, the nature of the items or assets differs as well, taking into account that the demand will not be forecasted on individuals but on sets or groups. Additionally, some of these groups of items can be defined as "single-use", i.e. every time the previously mentioned process flow is completed, the item will not be part of future predictions any more; whereas some others can have multiple uses, which means once they complete a cycle of the process flow, they will start a new cycle.

Furthermore, during the process of reviewing previous studies to use as basis for the present work, it was found that even though there are different publications related to inventory and supply chain management in healthcare provider institutions [12], very few implement practical solutions for real case studies [36] [37] and even fewer or none involve machine learning among the solutions used, which comes to be of primary importance as ML has proved to be one of the more widely spread solutions for Supply Chain Management (SCM) implementations in other industries.

However, even though [36] do not involve any ML solution, it results in great use, due to the fact that it provides a good insight into what problems healthcare institutions face if no SCM policies are used, and how to start implementing them in order to de-

crease the operational cost of managing assets such as pharmaceutical products. The research is conducted in a healthcare institution in Malaysia, which consists of a central distribution center or wholesaler and a chain of clinics scattered throughout the country. As mentioned, one of the main issues in the distribution process of assets from the wholesaler to the chain of clinics is the implementation of demand forecasting, as "one of the issues is the availability of accurate data on consumption. However, the lack of standard nomenclature for healthcare products, plus the preferences of clinicians creates further uncertainties" [36]. After the analysis was performed, [36] suggests the implementation of a VMI (Vendor Managed Inventory) strategy in which "the supplier assumes responsibility for the management of inventory at the customer, and takes decisions regarding replenishment" [36].

In addition to that, [37] give additional guidance on how to improve logistics (bed logistics, hospital cleaning, and pharmaceutical distribution) in a healthcare institution in Denmark and the United States while making a comparison between them. After reviewing the current status of the logistics management in healthcare industry, the author signals RFID (Radio Frequency Identification) and barcodes as technologies that "can enable the flow of information and track items in the SC" [37]; which highlights the importance of the role that DTS plays in the dental industry. Furthermore, [37] states that "hospitals have adopted stockless and JIT strategies to reduce inventory levels and inventory costs", which were also mentioned in [36] as the inventory management approaches that are more widely spread along the healthcare industry. However, [36] states that these "should only be used for high volume products, with a more traditional approach for low volume medical supplies".

Nevertheless, one of the few works that use machine learning approaches for inventory management of healthcare assets is [38] in which, an artificial neural network is used to classify assets in three different categories: very important, moderately important and least important; more commonly known as the ABC classification. This classification allows the company to treat assets in different ways, according to their importance, and therefore have a better management of resources, according to the importance of the item in stock. The ANN (Artificial Neural Network) is trained by using two different approaches and compared afterwards between them, as well as with the performance of Multiple Discriminate Analysis (MDA). The result showed that ANN performed significantly better than MDA, whereas there was no significant difference between the two different models used to train the ANN.

Even though the previously referred work addresses inventory management of assets in a pharmaceutical company, the nature of a classification problem differs from ours which is a regression problem.

Even so, [38] has been reviewed by other researchers to develop their work. One of these is [39] which investigates the causes of the gap between research and real-life application of classification and demand forecasting of spare parts in different industries. In the findings related to the demand forecasting presented in [39], out of ten companies analyzed, only two have absent or almost absent implementation and eight implement traditional demand forecasting techniques such as simple average, moving average or exponential smoothing, out of which only four have a formalized forecasting process.

Similarly, and as a complement to [39], some studies and work have been done on demand forecasting, as an SCM implementation approach in industries other than the healthcare. One of these is [40] in which the authors review the benefits of neural networks (NN) and implement one in order to forecast the demand of products considered to have a "lumpy" demand. The term lumpy is used by [40] to reference intermittent or irregular demand, which is defined by [41] as "random demand with a large proportion of zero values". In [40], the performance of the NN model and that of more traditional models such as single exponential smoothing, the Croston's method and its improved version of the Syntetos–Boylan approximation are compared. As a result, [40] found that NN outperformed all the other methods of prediction in all but some special cases in which the training data contained longer zero-demand periods than the ones present in the test data. A more in-depth review of [40] will be done later in this paper, as it has some similarities with the characteristics of the problem the present work tries to solve and, therefore, it will be used as a guide on the NN model building as well as into the data transformation process. In the same way, [31] and [30] will be considered in the following chapters as they also implement ML solutions for "chaotic" demand forecasting which, as will be detailed further, is also a characteristic of the data that is being used in our case study.

1.2 Our Case Study

In this work, the demand forecasting of a variety of dental assets will be addressed with the help of machine learning techniques. The data will be provided by DTS

(Dental Tracking System), a system capable of tracking a wide range of dental assets by the use of RFID technology, thus being able to depict the flow of the assets and their current state, among other information. This system can be perceived as a big data type of system, as it generates information out of the behavior and actual flow processes of institutions and makes it available for the users to be able to make organizational decisions based on it as well as improve the assets management itself. As specified by the company, one of the drivers of this software's development, among many others, is to give the customers an aggregate value of decreasing the operational costs with the use of DTS.

DTS is developed by LM-Instruments Oy, a company founded in 1973, based in Parainen (Sw.: Pargas) in southwest Finland, part of the Planmeca Group that produces high technology dental instruments and other supplementary products as well. Their goal is to provide users of DTS with the possibility of setting alerts for the existing tracking locations in the institution, such that the clinic or university can be aware of the stock level of every given asset for each of the aforementioned locations, as well as be automatically informed when the stock level is less than the expected demand for a given period of time. It is this expected demand for every type of asset that we intend to predict with the ML implementation proposed by the end of this thesis work. However, in the scope of this thesis work will be included the analysis of the data and the implementation of the models in Python version 3.5.2, but not the actual implementation inside DTS, any notification system, any UI implementation related nor tests on DTS real-life environments.

LM-Instruments wants to improve DTS in such a way that the institution making use of the system can rely on the application to improve its inventory management process, therefore having a positive impact on the SCM policies by minimizing the cost of managing instruments and materials. Because of all that has been exposed previously, machine learning has been considered as the way-to-go for implementing demand forecasting in DTS.

1.3 Research Problem Statement

The purpose of this thesis work is to find the best machine learning model based on SVM (Support Vector Machines) and LSTM-RNN (Long-Short Term Memory Recurrent Neural Networks) to forecast the demand of dental assets (materials and in-

struments) tracked by DTS, for every customer procedure location configured in the system.

The company seeks to provide the users of DTS with the ability to improve material and instruments management, by reducing costs and optimizing resources.

Due to the aforementioned lack of articles published involving inventory management of health/dental care assets and the use of machine learning models for addressing demand forecasting, this thesis work will take the first step towards it, based on the results obtained by [35], explained previously, as well as [31], [30] and [40].

1.4 Thesis structure

The thesis is divided into four parts. The first part defines the data used for the implementation and its characteristics, in chapter 2. Chapter 3 presents the models used and the practical implementation for the case study. The third part is composed of chapter 4, in which the best results acquired are presented, and chapters 6 and 5 in which the development process of the thesis is reviewed and learnings are presented, as well as the challenging issues and their solution.

2 DATA MANAGEMENT

The process of creation of the business data will be presented at first, in order to achieve a better understanding of the data that will be handled in this work.

2.1 Data Generation

DTS is a tracking system based on RFID technology, which uses tags to store a unique identification string. This string can be read with the help of RFID antennas through radio frequency waves. The instruments manufactured by LM-Instruments contain RFID tags, which are registered into the system by using a "reader" device which has an RFID embedded antenna. These reader devices are part of the system and play a key role in the process of data generation when tracking the instruments. Likewise, LM provides RFID tags that can be used to track different types of materials, which due to their nature, do not contain embedded RFID tags.

The use of the reader device can be divided into four phases: authentication, activity selection, assets reading and information transmission.

- Authentication Phase: it allows the user to create a session in the device by using his/her credentials, allowing the device to identify the user that is performing any action on it.

- Activity Selection Phase: each reader device has a given set of activities configured through the server application. These activities are displayed immediately after the authentication phase is finished. The activities presented in the reader are logical entities, which match the stages of the process that dental assets go through in a specific institution. This process and its activities can be configured in the application according to the needs of the institution itself, thus, the DTS application fits the way every institution works and not the other way around. When the configured activities are presented to the user, he/she selects

the activity that wants to perform, which will be transferred to the server application in the very last phase. Together with the activity, additional required business information will be transferred.

- Assets Reading Phase: in this phase the user must place the RFID-tagged assets involved in the activity, over the reader. While this happens, the reader collects in memory all the read tags. Once the user presses the "send" button on the interface of the reader, the information transmission phase is started.

- Information Transmission Phase: in this phase all the information gathered in the previous phases is assembled in a JSON (JavaScript Object Notation) object and sent to the server via Web Service. The required parameters for ensuring the connection and correct communication with the server, are previously stored in a configuration file on the reader.

Once the information is received by the application server, it is processed to verify the validity in terms of data type and process flow, including permissions among others. After the data is processed, it is stored in the database.

2.2 Relevance of Data

The relevance of the information in the database will be dictated by the problem intended to be solved. Considering that the purpose of this work is to predict the frequency of use of assets registered in the system, any data provided by external systems will be discarded. Similarly, any data stored in the database, that is not related to the data generation process can also be discarded. An example of this is the system configuration data.

We can easily identify all the transaction and asset's data stored as highly relevant for our solution. However, the data defining the process flow in the institutions will not be considered as relevant information, due to the fact that the purpose of this work is not predicting the usage of the assets at all stages of the process flow, but only at the end where the assets will be used in a dental procedure, which is the moment in the process flow in which the institution requires to have full availability of the assets in advance.

Similarly, the activities existing in the system as an entity will not be considered in its entirety, but as a filter instead, considering that the activity for registering dental

procedures in the system is identified by the name of "Clinical Use". Furthermore, the unexpected readings during the maintenance or execution of activities other than "Clinical Use", which might not follow the defined process flow, could bring noise data and affect the performance of the models. Therefore, only the transactions related to the "Clinical Use" activity, will be considered as relevant for the implementation.

Likewise, DTS stores entities that map real-life locations into the system. Inside the system the locations are bound to reader devices, which at the same time are bound to the activities. Due to the inclusion of the filter for the "Clinical Use" activity, only locations that can perform that activity will be considered. Moreover, the intended predictions will be performed in such a way, that the user can identify how many assets would be required per location, thus, the predictions should include the location in the dataset.

Based on this, altogether, the data considered relevant for this work will be composed by the transactions related data, i.e. the time stamp, the activity, the location, and the assets read, for every transaction registered in the system. This data will be extracted accordingly from the respective entities existent in the database engine. More detailed information is skipped due to being considered confidential information for LM-Instruments and the DTS project.

2.3 Data Set Creation

Once the data relevance for the machine learning models implementation has been identified, the next step is to build the dataset. The process of preparing the data and creating a meaningful dataset is more important than it seems, as "data cleaning and preparation takes approximately 80% of the total data engineering effort" [42]. The data preparation process was performed many times as it suffered several improvements and modifications throughout the development of the current work. The prior in accordance to what [42] describes as KDD (Knowledge Discovery in Databases), an iterative sequence composed by four steps; which are **defining the problem, data pre-processing (data preparation), data mining, and post data mining**.

In order to discover what are the most used articles in the process of every institution and begin the process of dataset preparation, the first step to follow is to extract the relevant data from the database. Let us remember the transaction data we labeled as relevant:

- Timestamps
- Articles
- Locations
- Activities

In addition to the mentioned items, the tables holding relations among the entities referenced will also be considered. However, the nature of the relation will not be described in order not to reveal the structure of the database.

The data mentioned is used to store the details of the transactions performed in the system; defining a transaction as every time a user has gone through all of the four phases of the use of a reader; previously detailed in the data generation section.

The extraction was performed by dumping the data into csv files with help of the GUI of the tool Heidi SQL, which is currently being used by the DTS development team. The first need when preparing the data and generating the dataset that will be used to train the models is to define the target value, i.e. the value to be predicted, in this case, the demand of every article based on the existent data. With this step, the data transformation sub-process starts. As explained by [42], "data collecting, data integration, data transformation, data cleaning, data reduction, and data discretization", all make part of the data preparation process. Therefore, once the csv dump files are generated, the historical record of the transactions performed in a given location during a specific lapse is reviewed to count how many times a single asset was involved in "Clinical use" transaction activity. That number will be used as the demand. As part of the data transformation, it is also necessary to define what is the time unit, or the length of the period of time for which the demand would be defined. The database of the environment provided by LM-Instruments contained transactions since 2015, which causes that working with yearly-based data produces no more than two or three data points, which is not enough to train a machine learning model. It was decided then, to work with months, but a new limitation found was that the use of locations entities in the selected environment, started only after September-October 2017, which represented a limitation for working with months; as at the early stages of the current work, there were only six months of transactions involving locations; and due to the fact that the predictions would be performed per location, it is vital to be able to assign a demand to a location entity. This finding hampered the work on monthly datasets

and made it impossible to do it on an annual basis. For this reason, and to be able to work with datasets of considerable sizes, days was defined as the time unit to use, thus having approximately 180 data samples per type of asset and location.

After discussing with the software team in charge of developing DTS at LM, an improvement for the small number of data samples per type in a location; older transactions which did not have a location associated, were modified to reference directly the location belonging to the reader device that was associated with that transaction. This means, for transactions prior to September 2017, the location to be used would be that of the reader device involved in the transaction. With this adjustment, it was possible to use transactions as old as of January 2015, therefore increasing the number of data samples available for use from 180 days to over 800 days.

In the same way, after increasing the number of daily data samples per type in a location to 800, it became possible to use a different period's size other than one day. The first alternative was seven-days periods.

The use of a longer period of time was considered based on the short period of reaction that an institution would have if a predicted demand would require the institution to not only perform maintenance to some instruments or take them through the cleaning and sterilization process but also if due to the end of life of older instruments or articles such as materials, it is required to perform a purchase. In that case, it is more probable that the delivery will not happen until a few days later and that would probably cause a lack of assets for the institution, which would represent an issue for its business process.

It is possible that the selection of one-day and seven-day period predictions will not be suitable for all of the institutions and will not represent the final solution. Nonetheless, these periods are used as a first approach towards providing the customers with a single general suitable solution.

Based on this, it was included into the generation of the data set the possibility to have data samples per seven-day periods, which allowed to have data sets with $m = 150$, with m being the number of samples in a dataset, per type in a location.

As it is commonly done when running machine learning models, these data samples will be later divided into training and testing subsets. However, depending on the approach we take on the data with which the models are run, the number of data samples would vary. If one-day long data samples are used; running the models on grouped data per type of asset as mentioned before, would give us 800 samples for every single

type of asset in the system per single location; whereas considering every asset as an individual data sample would give us 800 samples per individual asset per location; which means if there are around 30 thousand assets in the system, a total of approximately 24 millions of data samples model per location would be used; whereas by grouping the data we would only have 800 data samples, but the models should be run once per type, which in this case is around 400 times.

In summary, on the one hand, if we consider having one data sample per day for every type of article in a location, it would be necessary to perform around 400 predictions (combination of types of articles and locations) with 800 samples in every data set. On the other hand, a single prediction should be performed if no separation is performed over the main data set, but the number of data samples in the data set would be of around 24 million.

The prior reflects how much impact the shape of the data set can have on the performance of any ML model, not only in terms of accuracy but also regarding the time and the amount of data that is required to process.

Based on the fact that our predictions are intended to be performed per location and type, the seven-days long periods approach is preferred, besides of having a big boost performance-wise due to the division of a big training-predicting task into many small ones.

Another important variation was the removal of those data samples equal to zero in the target output as part of the data cleaning step in the preparation process. There were mainly two different reasons to include this optional alteration to the datasets. On the one hand, during the first tests a considerable number of datasets (after being separated as explained previously), were found to have the target value with only zero values. This, representing that the specified type of article was never used in the specified location, thus it has a demand equal to zero. These types of datasets were preventing the predictions to be performed in a faster and more practical way. On the other hand, a brief analysis over the per-type-and-location-separated datasets, it was clear that due to the lumpiness of the data, no linear models could describe in a more accurate way the pattern in the data. Furthermore, the intermittent data makes it more difficult, even for non-linear models, to produce accurate predictions; which is the reason why it was decided to introduce the possibility of removing the zero-demand samples in order to predict more accurately.

When these samples were removed, not only an improvement on the performance

of the models was perceived, as the prediction accuracy increased, but also that the prediction given to the institutions would come more in handy even if the article type is not used during the predicted period of time in a specific location, as the institution will be ready with the necessary number of articles for the upcoming period of time. Thus, even if the predicted value ${y_t}^* \neq 0$, where ${y_t}^*$ is the predicted demand of an article type for a period t; whereas the real value $y_t = 0$, where y_t is the real demand of an article type for the same period t; for the next prediction y_t would be ignored for being zero, yielding ${y_{t+1}}^* \approx {y_t}^*$, where ${y_{t+1}}^*$ is the predicted demand of an article type for the next period $t + 1$; in which case the institution would have the availability for y_{t+1} since the end of y_{t-1}.

2.3.1 Improvements to the Data Set

One of the biggest improvements included in the data set generation process was the possibility to separate the data samples in smaller datasets. This separation is performed according to the combination of different characteristics such as the type of an article, the location, and the code. This allowed the predictions to be performed in smaller tasks which would improve the performance not only regarding the execution time but also regarding the accuracy of the prediction. The latter, due to the data points used by the model having a closer relationship between each other, than the one they have with other article type's data. This would improve the model interpretation of the data when performing separate prediction tasks for every article type, instead of one prediction for all the articles at once.

However, one big issue arose in the early stages, which hindered the execution of predictions in separate tasks. The issue consisted of the data containing values that differ from one article to the other, even if these belong to the same type of article. It is important to consider that the demand predictions are to be performed at the type of article level and not at an individual level. Based on the prior, these differing data values should be combined in one, which created a challenge on how to merge different values that articles of the same type had for the same attribute. On the other hand, the possibility of using the data at an individual article level; that means with no grouping of data, but using the type of article as one more feature or predictor, seemed to be easier to implement. Nonetheless, the number of data points when running a prediction and the wide range of values in attributes such as expiration dates and count of cycles completed, among others; did not produce the desired performance. This issue

was solved with the generation of time series data sets, which allowed the data to be grouped without using the attributes that could not be merged.

Besides of the prior variations and improvements, during the generation of the dataset; new predictors were created as part of the feature transformation step, defined by [43] as "a process through which a new set of features is created". This, in an attempt of providing the models with the most meaningful data as possible.

2.3.2 Generation of New Features

The created predictors, as well as target columns, are detailed below:

Point In Time

This feature is a sequential number used to keep the data samples in time order. This provided a first approximation on the use of time series, which can be defined as a "set of observations X_t , each one being recorded at a specific time t" [44].

The rationale behind the assignation of a number to a data point is related to its age, thus having a value of one means that the data point was the first period of time of collected data, i.e. the oldest. The bigger the number, the closer to the current point in time.

Time To Expiration

This feature was created to allow the possibility of merging all the different expiration dates that every single article belonging to the same article type has, into a single value, in the same way the demand was grouped.

The reasoning behind the computation of the values was the following:

$$t = (d - e)/s$$

for every data sample, where d is the start date of a period (might be a one day period or a seven days period), e is the expiration date and s is the size of the period in days.

Frequency Variance

This feature is used as an alternative for the output. This target column represents the difference between the $t-1$ frequency and t.

$$v_t = f_t - f_{t-1}$$

Where v_t is the frequency variance and f is the frequency, while t represents a given time and $t-1$ the previous period.

This way instead of predicting the actual demand, the behavior of the demand for $t+1$ would be predicted. However, after analyzing the data, it was found that the values of the frequency did not have that type of distribution, whereas the frequency variance should have the desired distribution with values close to zero.

Previous Cyclecount

It represents the value of the cycle count feature for the previous time step.

$$pcc_t = cc_{t-1}$$

This purpose of this feature was to be used as a predictor in a time series shaped dataset.

Previous Time to Expiration

Similarly, this predictor represents the value of the feature "Time to Expiration" previously explained, with the purpose of being used in time series dataset.

$$pte_t = te_{t-1}$$

Periods To Last Demand

Because of the lumpiness of the registered demand for the articles, in the separated dataset it was common to find many zero and non-zero demands.

This feature intends to reflect the length in periods of time since the last non-zero demand was registered for an article type in a specific location.

If the data sample for time t has a value of one in this feature, it means that the data sample $t-1$ has a non-zero demand; whereas if the data sample t has periods to last

demand equal to 3, it means that the data sample $t-3$ has a non-zero demand, but data samples $t-1$ and $t-2$ have a demand of zero. A similar implementation was used by [40] when dealing with an intermittent demand dataset.

Frequency Time Series

Frequency Time series is not a single feature but a set instead. During the creation of the dataset, it was introduced the option of creating a customizable-size chain of features representing the frequency of previous time steps.

As already mentioned, the size of this chain of features is selected when launching the process of dataset generation. Therefore if six is entered as the size, the same number of features (six) will be created, storing the frequency for the time step α, as follows:

$$t - \alpha_{freq} = freq_{t-\alpha}$$

where $t - \alpha$ represents the number of previous time steps, considering t as the reference to a current time step; and $freq$ being the frequency of a given time step.

This technique is used for time series prediction, considering that SVR and LSTM have been widely recognized as being accurate with this type of dataset.

The purpose of this feature creation is being able to use univariate time series, taken as an example from [45] and other tutorials online.

Periods To Last Demand Time Series

Similar to the previous set of predictors, the process is repeated for generating another chain of features with the same size as the "Frequency Time Series", but in this occasion, the feature "Periods To Last Demand" is used instead.

$$t - \alpha_{periods} = periods_{t-\alpha}$$

where $t - \alpha$ represents the number of previous time steps, considering t as the reference to a current time step; and $periods$ being the periods to last demand for a given time step.

The difference between the terms in the previous equality is that $t - \alpha_{periods}$ is the name of the new feature created for the data sample t and $periods_{t-\alpha}$ is the already existent feature of the data sample $t - \alpha$ from which the value is taken.

2.3.3 Dataset Shape

In ML, n is widely used to refer to the number of features or predictors used as inputs for a dataset, whereas m is used to define the number of data samples in a dataset.

In this section, the optional variations included when choosing the shape of the dataset will be related to the value of these two variables n and m.

Number of Data Samples

On the first place, altering the number of data samples was highly limited by the amount of data available in the system. As it was already mentioned in a previous chapter, at first it was possible to use only 120 days of data per article type in a location.

However, the database (a copy of the original environment) was altered so that the location tied to the reader devices was used as the location referred by those transactions performed in DTS before the use of location entities was introduced in the software. Once this modification was performed, more data became available, to the point of being able to use as much as 800 days or 150 weeks. These numbers (800 and 150) became then the number of data samples m for every separated data set, depending on how big was the count of days that the demand should represent.

Therefore, if it was decided that the demand should be presented on a per-day basis, 800 would be the sum of samples. Whereas if it was decided that the demand should be presented on a per-seven-days basis i.e. a week, then only 150 samples would be available to be used in every article-per-location data set.

Regarding the size of the periods, i.e. how many days should compose every period; it was decided that only one and seven would be used, due to the necessity of limiting the scope of the tests performed to make it finite, and because these were the two periods proven to fit better the requirements of the application and the institution whose data was used for the development of this work.

However, even though the goal is to find one single solution for either of these periods length, the period of seven days long was preferred as predicted in advance; considering that it would allow more response time for the institution to be ready with the required assets to supply the predicted demand.

Predictor Features

On the second place, altering the number of features or predictors used is a basic yet key step in the process of building a dataset to use with an ML model.

As a first approach, the basic information of every article was used; basic information being the "type" or "code" of the articles, which work as an identifier of specific types of asset; the cycle count which is a counter of how many cycles has the asset completed in the process flow; the expiration time, among others. This, because the characteristics of every asset might dictate the importance of every asset in a location and therefore, how high or low is its demand.

However, on the very last version of the dataset, twenty-nine predictors and two target outputs were available to be selected according to the technique chosen to be applied and used to fit and predict a given model.

Because of that, the number of possible combinations to be tested is already considerably big; without taking into account the possible combinations of the hyperparameters for SVR and LSTM.

The final objective of being able to select the predictors in a dynamic way, by using parameters in the source code, was to be able to apply data reduction by removing predictors that did not perform well and did not have a strong relation with the target output; as explained by [46] when stating that feature subset selection, as part of the data reduction phase of the data preparation process, consists on identifying "informative features from the original set and delete the others".

2.3.4 Target Output

As mentioned before, in the final data set, two different target outputs were available to be used in the predictions.

On the one hand, the frequency which represents the demand, interpreted as the number of transactions an article was read on or used in DTS; detailing a procedure with a customer at a specific location, in a given period of time.

The drawback of using the frequency as target output was that it does not have a Gaussian distribution, which was closer to be achieved by the frequency variance. One characteristic that affects the distribution of this target output, is that, for obvious reasons the demand can be only represented either by positive numbers or zero, but never by a negative number, so there exists a limit on $y = 0$. In the same way, the

distribution was the same disregarding the length of the periods selected (one or seven days), as values closer to zero are more common than big numbers.

On the other hand, frequency variance, which was explained already before, made it possible to have a distribution closer to Gaussian and was expected to boost the performance of the models, besides of providing the opportunity of predicting the trends of the data instead of fixed values.

3 BUILDING OF MODELS

As the first step for building the models, the analysis of the data provides hints on what the models should be and the features they require for performing as optimal as possible.

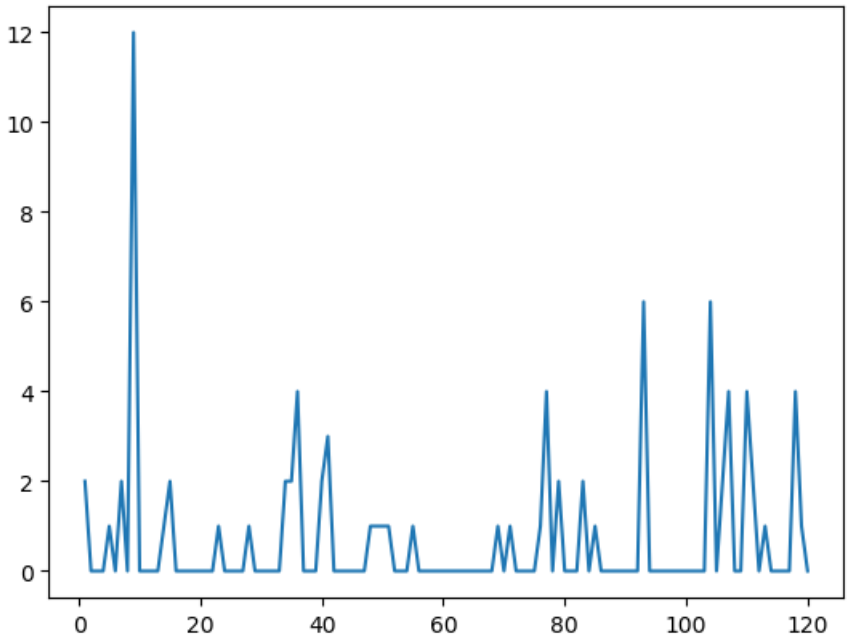

Figure 3.1: Frequency of use of an article in a location for 120 days.

Looking at figure 3.1, the graphical representation of the data suggests that no linear model would perform well enough. This figure presents the frequency of use of an article (which is not specified in order not to break the confidential information policy agreed with LM-Instruments) in a given location. This frequency presents a dynamic variation of values such that, fitting a linear model could bring close guesses about its future demand, but it would not be as accurate as required. Thus, it was

decided to use a non-linear model as the base model, after discovering that a single straight line would not be able to describe the behavior of the frequency of use for the articles, throughout the time.

Another challenge that was faced in this work was creating a model that generalize well enough, in order to be able to predict accurately with different types of articles and different locations, each combination providing different behavior.

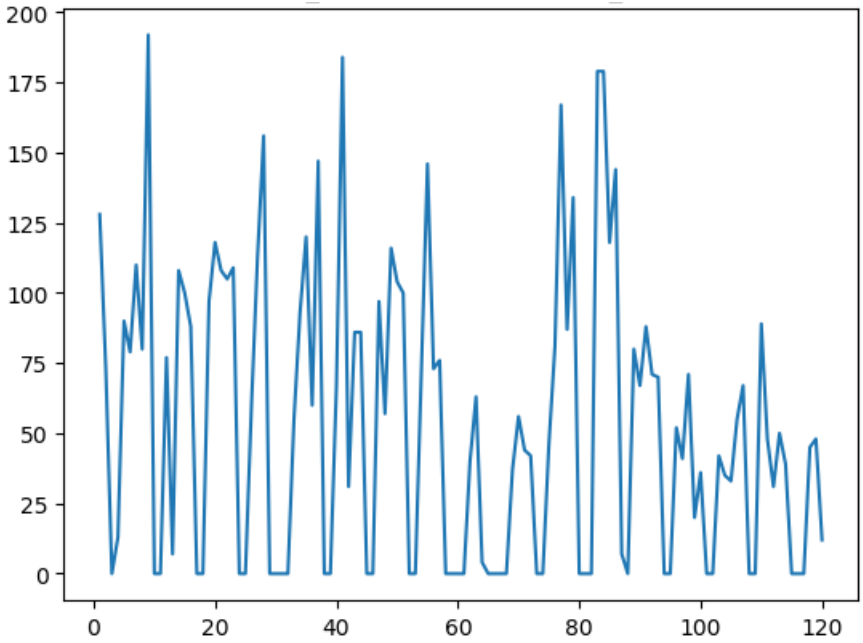

Figure 3.2: Frequency of use of an article in a location for 120 days.

A small example of this is figure 3.2 which presents the demand for a different article in the same location as the demand presented in figure 3.1.

The reason why SVR (Support Vector Regressor) and LSTM (Long-Short-Term-Memory Neural Network) were chosen as the models to drive the present work and to be subject of comparison, is due to the nature of our problem, but also because they have proved a high performance on different types of problems including those using time-series data. Regarding SVR and apart from time series data, there is another characteristic which motivates its use in this work. As it was previously mentioned and also presented in [26], "instead of minimizing the observed training error, SVR attempts to minimize the generalized error bound so as to achieve generalized performance" [26]. As for LSTM, it possesses many different benefits and exploits of relationships existing between the time steps in a way few other models can do it. A very clear review of its

benefits is presented by [34]. However, the motivation to use it in this project is based on its success in more complex problems, and intend to replicate its good performance on less complex problems.

3.1 Support Vector Regressor

One of the first steps to start with the process of building the SVR model is reviewing the issues or risks that can be faced by implementing the model. As it was mentioned previously in the paper, due to the impact of the hyperparameter ϵ in the model performance, overfitting was identified as a risk by [30], when assigning a small value to it. Similarly, it was also mentioned that this risk becomes bigger when the dimensionality in the dataset used is big. However, in this case, the dimensionality of the datasets used during the current work was very low, which made it possible to use a small ϵ without risk of overfitting the model.

Furthermore, the process of building the SVR model has been tightly related to the process of improvements applied to the generation of the dataset, as it is clear that based on the shape and type of data contained in the dataset the performance of the model could improve or worsen.

However, a very important step in the process of using an ML model is understanding its insights and how it works. Following, a brief review of the technical functionality of SVM is presented.

3.1.1 Technical Review

SVR is a technique based on SVM for solving regression problems. As most of the ML techniques and algorithms, SVR and in general SVM have different implementations and variations, which are created for improving the performance on specific cases. However, below, the most basic version of SVR will be presented so it can be used to understand how the built model works and how can the prediction be done.

First, SVR's goal is defined by [1] as finding a function $f(x)$ such that errors lower than ϵ will be ignored, but will not accept deviations larger than that. Based on this, the linear version of the function becomes

$$f(x) = w \cdot x + b \quad \text{with } b \in \mathbb{R}$$

where x is the input predictor, w is the normal vector to the hyperplane and b is the bias.

At this point the optimization problem can be written as

$$\text{minimize} \quad \frac{1}{2}||w||$$

$$\text{subject to} \begin{cases} y_i - (w \cdot x) - b \leq \epsilon \\ (w \cdot x) + b - y_i \leq \epsilon \end{cases}$$

However, for this version, infeasible cases might be present, for which slack variables are included in the function, resulting in

$$\text{minimize} \quad \frac{1}{2}||w|| + C\sum_{i=1}^{\lambda}(\xi + \xi^*)$$

$$\text{subject to} \begin{cases} y_i - (w \cdot x) - b \leq \epsilon + \xi \\ (w \cdot x) + b - y_i \leq \epsilon + \xi^* \\ \xi, \xi^* \geq 0 \end{cases}$$

This, where C determines the penalty term controlling the flatness of the function; and ξ, ξ^* are the slack variables.

The prior is the most complete of the simple versions of the linear problem formulation.

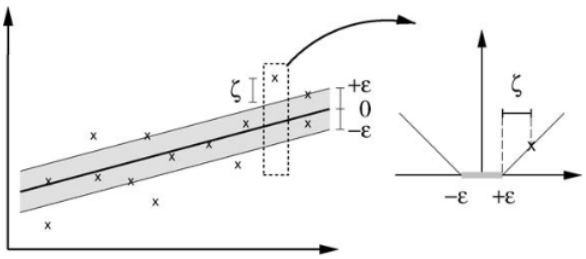

Figure 3.3: Soft margin loss setting for a linear SVM [1]

In figure 3.3, a representation of SVR over sample data can be appreciated. In figure, there is a straight line marked with a zero. This line is the hyperplane and the surrounding area is the margin denoted by ϵ. The data points falling inside the margin

will be ignored, and those falling outside the margin will contribute to the cost of the function, which is intended to be minimized.

However, a more complex and therefore powerful version of SVR can be achieved by applying the so-called kernel trick. The kernel trick is a methodology to use SVR with non-linear problems. The application of the kernel trick is presented in [1] after the formulation of SVM into *support vector expansion*; which is a more complex version of SVM, that includes Lagrange multipliers. However, these steps will not be presented for being considered out of scope. More information about the step by step transformation can be found in [1] and [26].

The formulation for the kernel trick, is as follows:

$$f(x) = \sum_{i=1}^{\lambda} (\alpha_i - \alpha_i^*) k(x_i, x) + b$$

$$\text{maximize} \begin{cases} -\frac{1}{2} \sum_{i,j=1}^{\lambda} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) k(x_i, x_j) \\ -\epsilon \sum_{i=1}^{\lambda} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{\lambda} y_i (\alpha_i - \alpha_i^*) \end{cases}$$

$$\text{subject to} \sum_{i=1}^{\lambda} (\alpha_i - \alpha_i^*) = 0 \quad \text{and} \quad \alpha_i, \alpha_i^* \in [0, C]$$

where α_i, α_i^* are the Lagrange multipliers, and the function $k(x, x)$ denotes the kernel function.

The denoted kernel function must comply given rules and theorems which will not be detailed here, as they are not considered relevant for the scope of this work. However, the library used for SVR implementation provides embedded kernel functions to be used, as hyperparameters.

3.1.2 Scikit SVM Hyperparameters

The implementation was performed in python version 3.5 by using the SVM module of the scikit-learn library.

A total of 179 executions were performed, considering different changes in the tunable hyperparameters of the module, together with modifications on the generation and shaping of the data as well as the predictor features and the output target used.

As hyperparameters in the model, we can modify the kernel used, which can be linear, polynomial, RBF (Radial Basis Function) or sigmoid.

Other hyperparameters tuned during the model building were the C constant, the ϵ margin, the r coefficient and the degree d. The γ value was modified automatically depending on the number of predictor features used, as follows: $\gamma = 1/n$ where n is the number of predictor features.

Apart from the kernel which for obvious reasons play a fundamental role in every run, the most important parameters to set are the constant C, which "allows us to define the trade-off" [22] between how the error of a data point can be and the size of the margin. This means the lower the value of C, the flatter; i.e. the model will not penalize much the outliers. However, if C has a high value, our model will try to avoid having data points out of the ϵ margin.

Another important hyperparameter is ϵ for the same reason stated before, as it will tell the model how close or far away can the model be from the data points.

On the other hand, the coefficient r and degree d are necessary when using the polynomial kernel, whereas γ is needed when using the RBF kernel.

Tuning Considerations

As explained before, ϵ could cause the model to overfit when the dimensionality of the dataset is significantly big. In terms of ML, a dataset is considered to be high-dimensional when it contains a large number of features or predictors. During the current work, as mentioned before, the final dataset contains 29 features, which will not all be used at the same time. For this reason, the dimensionality of the dataset can be considered to be low and thus, a low value of ϵ will not transform in risk of overfitting the SVR model. However, this is not the only way that the dimensionality of the data can affect the performance of a model since a large number of features can cause the performance time to be very long, if it is necessary for the model to process every feature of a data sample. This is also known as the curse of dimensionality, for which [47] provides a deeper look.

Tuning the Model

In order to select the correct parameters for the model, an initial approach was taken, by using the "GridSearchCV" class existent inside the model selection module in the scikit-learn library. However, because of the lack of high-power computational resources, the training and predictions were performed by grouping the dataset by the type of article, which causes that when running the GridSearchCV class for every type

of article, a different optimal configuration was found for every model run. Due to this, it was discussed and decided that as a definitive approach, the tuning of the parameters would be performed manually by executing the models many times with different combinations of parameters until the best overall performance was achieved.

3.1.3 Process

On the first place, the selection of python as the programming language for this work was based on two main factors; as it is the previous experience with that programming language, and its recognition based on its utility, support, tools, and libraries existent for ML algorithms implementation. In addition to this, there exist many books with a practical approach to ML, which use python as the main programming language or present it as a common option to start with [22, 48–51].

Secondly, the selection of scikit-learn as the tool to use for python was based on its simplicity when implementing the SVR model. Other libraries such as tensorflow and theano, which are widely recognized in the industry, seem to be developed to work more often with neural networks and deep learning algorithms. A more in-depth review is provided in [52], where a detailed review of the underlying technologies as well as the code design and few comparisons with other libraries' performance is presented.

The evolution of the implementation, started by using the module SVM inside the scikit-learn library. This module contains the class SVR.

SVR was used over SVC due to the nature of the problem to be solved, which was identified as a regression problem. Considering that the class SVR is intended to be used for regression problems and not for classification as SVC, the former was selected to be used in the implementation. More information about the module and the available implementations can be found in the scikit-learn official documentation.

Due to the fact that the implementation first contained mostly the characteristics of the assets as features, and considering that some of these were not numerical, an encoding solution was implemented. The encoding was provided by the scikit-learn library as well, with the use of the class "LabelEncoder", found in the "preprocessing" module. The LabelEncoder class enabled the capacity of using any desired feature in the dataset. One example of this type of features are the locations, which contained the name of the location instead of its id or another numerical identifier; and the type of the article, as it does not exist as an entity itself in the DTS system, but as a string value.

Once the encoding was performed, the implementation continued with the generation of the model itself, for which the default and recommended values were used as an initial test. This initial implementation was created with the following line of code:

```
SVR(C=1.0, epsilon=0.1, cache_size=300)
```

The prior line of code is very clear, as it defines the values to use as penalty value and ϵ. In addition to this, the kernel, as it is not specified, takes the default value of "rbf" which stands for Radial Basis Function. Similarly, a cache size of 300MB was specified, in an attempt to improve the performance of the model.

At this point, the evolution of the model was focused on the configuration of the hyperparameters. Another tool inside the scikit-learn library was used, which is an estimator called "GridSearchCV". The prior based on the fact that "scikit-learn can evaluate an estimator's performance or select parameters using cross-validation" [52], which is possible with the use of the previously mentioned estimator.

In a similar way, such a solution was mentioned in different works. One such example is [53], where it was defined as the method for the authors to "find which combination of RBF kernels returns the better result, namely, which value of μ to adopt" [53].

Therefore, a solution using "GridSearchCV" was implemented, with the objective of finding the values that would help the SVR model achieving the best possible result. However, due to the fact that the dataset was being further divided in smaller datasets based on the article type, this implementation produced different recommended configurations for each one of the datasets; which was not the main goal of the project. Therefore, a single execution with the whole dataset was performed producing two main conclusions. First, due to the computational power limitations present, the time of execution was higher than desired; and secondly, the result of the "optimal" configuration suggested by the estimator was outperformed by previous tests which used divided datasets.

Based on the prior, the use of an estimator was discarded, and a decision was made to perform all the tests manually; thus, modifying the values of the hyperparameters based on previous results in order to find configurations producing better ones. This evolution of the model was set to be performed in parallel with the evolution of the dataset itself, based on the variations of the length of time considered by each data sample and the features used.

The evolution of the model, however, was performed by introducing the customization of different hyperparameters in different stages of the present work. The prior means that, as the previously presented line of code showed, no variations on the kernels neither in coefficient were included at first. Nevertheless, these variations were included in later stages of the model optimization, and the final line of code for creating the SVR model is the following:

```
SVR(C=MODEL_PENALTY, kernel=KERNEL, epsilon=MODEL_EPSILON, coef0=MODEL_COEFFICIENT,
    cache_size=300, degree=DEGREE)
```

In this final line of code, as opposed to the initial one, the values assigned to the hyperparameters of the model are not static values in the code, but global variables instead. The prior, as a result of including the option of parametrization in the python script, so the values can be defined at the execution time dynamically, avoiding the need of modifying the script's code prior to every test.

Similarly, in this final line of code, we can see different hyperparameters which were not considered initially, such as the kernel, the coefficient, and the degree.

3.2 LSTM - Recurrent Neural Network

As it was mentioned prior in this thesis, LSTM networks have two issues found and placed in the center of attention by [3], which should be taken into consideration when creating an LSTM model. However, luckily these issues have already been addressed by other researchers in the field, and the library used for implementation in python already contained a solution for both, exploding and vanishing gradients.

On the other hand, in a similar way as experienced with SVR, the process of building an LSTM model evolved hand-in-hand with the process of the generation of the dataset. Now we will review the technical functionality of LSTM along with the model building process, and describe all the variables adjusted during the process and give a brief explanation on what their impact on the model performance could be.

3.2.1 Technical Review

In order to understand how LSTM RNN work, it is important to detail what its composition is. As any other neural network, an RNN is composed of units of code called

neurons. According to [51], these units can be classified into three groups based on their function. The units intended to receive the data in the network are called **input units**. The **output units** are the ones in charge of presenting the result of the information processing performed by the network. Finally, the **hidden units** are the neurons in charge of performing the data processing, and represent "latent variables" [51]; as well as the difference between RNN and other networks.

The first characteristic of the RNN that makes it different from the vast majority, is the recurrence present in the hidden units. Regular NNs are called feed-forward (FFNN). This, because "each neuron in a particular layer is connected with all neurons in the next layer" [2], therefore the information only flows forward in the structure of the network, and due to this the information is never processed twice by the same node.

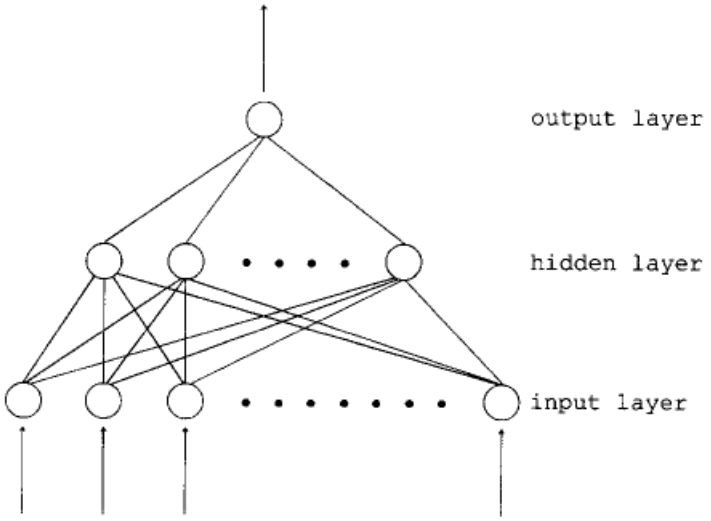

Figure 3.4: Feed-Forward Neural Network [2].

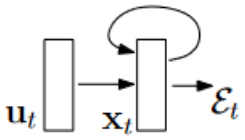

Figure 3.5: Schematic of a Recurrent Neural Network [3].

In RNN, on the contrary, the hidden units have a recurrent connection to itself; this being a key factor on this type of networks being successful when processing sequential

data, as explained by [54] when stating that RNNs "contain cyclic connections that make them a more powerful tool to model such sequence data than feed-forward neural networks". Figure 3.4 presents a visual representation of a regular FFNN, whereas a basic representation of an RNN can be seen on 3.5.

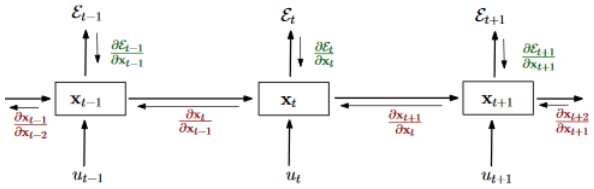

Figure 3.6: Unrolling recurrent neural networks [3].

Figure 3.6 is a representation of the concept of "unrolling" a RNN, used by [3] to present the concept of recursion in a more clear way. This concept might be presented by some authors as "unrolling" [3] [55], and as "unfolding" by others [56] [57].

Moreover, LSTM is a special type of RNN, in which the hidden units contain a more complex algorithm to process the data and fire the connections to the next units.

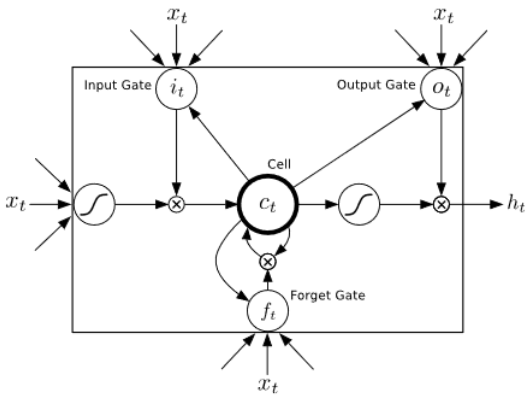

Figure 3.7: Long Short-term Memory Cell [4].

Whereas other networks' units contain only one function to process the data and fire connections, the radiography of an LSTM unit is much more complex than that. In figure 3.7, a graphical representation of the skeleton of an LSTM unit is presented. An LSTM cell consists of four components for processing information. These four components are:

- Forget Gate

- Input Gate
- Output Gate
- Cell State

Forget Gate

This gate controls what information is removed from the cell state. It applies a sigmoid function to the input of the current unit and the output of the previous unit. The formula for the forget gate f_t according to [4] is

$$f_t = \sigma(W_{xf} \cdot x_t + W_{hf} \cdot h_{t-1} + W_{cf} \cdot C_{t-1} + b_f)$$

where h_{t-1} is the output of the previous unit, x_t is the input of the current unit, W represents the connection weight matrices from cell to the gate and b_f is the bias vector. In figure 3.7, the forget gate can be seen at the bottom.

Input Gate

This gate controls what information will be written into the cell state. It consists of a sigmoid operation for deciding which information to update. A hyperbolic tangent (tanh) operation is part of this gate. The corresponding formula is

$$i_t = \sigma(W_{xi} \cdot x_t + W_{hi} \cdot h_{t-1} + W_{ci} \cdot C_{t-1} + b_i)$$

In figure 3.7, the gate can be seen at the top left.

Output Gate

The output gate is, as its name suggests, in charge of producing the output of the unit. It uses a sigmoid function on the input and the previous unit output, which is combined with the hyperbolic tangent of the cell state. The final result is then prompted and transferred to the next unit in the network.

$$o_t = \sigma(W_{xo} \cdot X_t + W_{ho} \cdot h_{t-1} + W_{co} \cdot C_t + b_o)$$

$$h_t = o_t * tanh(C_t)$$

In figure 3.7, the gate can be seen at the top right part of the diagram.

Cell State

The cell state plays the role of an internal memory shared between the nodes in the network. This is the characteristic that allows LSTM to remember information through time steps. In figure 3.7, the cell state is located in the center of the representation, labelled as C_t. The authors in [4] defines the cell state updates as follows:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot tanh(W_{xc} \cdot X_t + W_{hc} \cdot h_{t-1} + b_c)$$

3.2.2 LSTM Structure

The structure of the built network might have bigger or smaller impacts on the prediction's performance depending on the dataset used, as well as its shape and what the output should be.

The structure of the network can vary in two ways, which are the depth and width.

On the one hand, the term "depth" is used to refer to the number of layers stacked. The minimum number of layers that a network could have is 2, one for each, input and output values. This type of network is called "simple perceptron". However, with LSTM it is required to use a multi-layer perceptron (MLP), which consists of 3 or more layers. The layers between the input and output layers are known as hidden layers.

So, all things considered, a minimum of 3 layers must be used; this being the lowest limit. However, there is no such thing as an upper limit, though it is not very common to find networks as deep as having tens of hidden layers.

However, It is worth mentioning that by introducing a second hidden layer, thus having a total of 4 layers; the network would be classified as a deep neural network (DNN).

On the other hand, when talking about width; it has been a topic of discussion whether the width of a network could have as much impact on a network as its depth. Research as the one conducted in [58], concluded that "whenever we replace a finite width layer by a layer of infinite width, this results in drastic improvement in performance". However, the authors went ahead to state that using a single layer reduces significantly the number of hyperparameters that need to be tuned. Thus, the time invested on grid searches is reduced as well. Similarly, in another research work, the authors argue the role of the width when defining the structure of a NN, by concluding that "the depth may determine the abstraction level but the width may influence the

loss of information in the forwarding pass" [59].

However, the width cannot be set to the NN as a whole, but to every layer that makes part of it, by defining the sum of neurons each one will contain.

For this reason, the width of the output layer will be set to one, due to the fact that only one output value is required in the prediction. In a similar way, the input layer will have a width according to the number of input values or features that each of our datasets has.

Likewise, due to the fact that the predictors used for every prediction will be dynamically selected, the width of the input layer will be also defined dynamically.

As there is no indication on what should be the width of the hidden layer(s), those were also let to be defined dynamically, so different structures could be tested with same parameters to establish the structure providing better performance to continue using it.

3.2.3 LSTM Hyperparameters

There are many hyperparameters that can be tuned in an LSTM when building a model.

The implementation was done by using keras for python 3.5. Some of the parameters reviewed below are based on the parameters presented in the keras documentation web page for LSTM layers. It is important to highlight that not all of the hyperparameters were tuned, and their default values were used instead.

Number of Units

As mentioned before, the number of units in a layer defines its width; and the width was set to be dynamically assigned for every layer.

Activation function

The activation function is the function to be performed by the neuron to the input. This function will produce a result and thus, the output of the neuron. On account of the prior, choosing the correct activation function is key when building a NN.

Across the performed tests, four different activation functions were used. These are sigmoid, softplus, softmax and tanh.

The characteristics of these will not be discussed for being considered to be out of the scope of the current work.

Loss function

Is the function whose output value should be minimized, in order to achieve better performance during the execution of the LSTM model. In other words, is the function that measures the error or the distance between a calculated and a target value.

Only the loss functions embedded in the keras API were used, as the objective of the work is to find an "off-the-shelf" solution that can be provided by already existent libraries.

The functions used were cosine_proximity, hinge, logcosh, mae, mape, mse, msle, poisson.

Optimizer function

An optimizer function is that which updates the variables considered by a hypothesis function (activation function in networks) to fit certain data. Better said, it is the function doing the learning in a network.

In a similar way as with the loss function, only optimizers embedded in the keras API were considered during the implementation.

As optimizers were used adadelta, adagrad, adam, adamax, nadam, rmsprop and sgd.

Furthermore, each optimizer can be parameterized, which increases the number of possibilities when building an LSTM model with no other than embedded solutions. This talks about the complexity that building such a model for real-life problems implies.

Learning Rate

The learning rate is one of the most important parameters that can be tuned on the optimizer itself. The learning rate defines how big or small are the changes to the variables affecting the activation function during the process of learning.

When a high value is assigned to the learning rate, the "steps" or changes will be bigger. The same happens in the opposite situation of having a small learning rate, in which case, the changes are smaller.

The value to assign depends on the data, and neither a very big or a very small value can ensure a better performance. Reason for which the tuning of the learning

rate must be done manually by executing the model many times with different values and then choosing the best performing learning rate.

However, there exist an automatic way of finding the most suitable learning rate that is embedded into some of the optimizers, which consists on increasing or decreasing the learning rate depending on the results obtained in the process of training. However, this requires an additional parameter to control how much or how few the learning rate would change. In order to avoid overcomplicating the presentation of results and the model built, this solution was not used and instead, manual executions were used to test different learning rates, during the tuning process.

Epochs

An "epoch" is the term used to refer to the pass of all the training data through the network. In that sense, when in a model a number of epochs are defined, it is the number of times the training data will be passed through the network to update the weights on each feature, and therefore perform the learning process.

A high number of epochs implies a high number of training passes, which would also imply better learning of the model on the data, but also a larger time spent during training and more computational resources required. A high number might also be the cause for overfitting; whereas a low number, might produce underfitting, although it would make the learning process faster to execute.

Batch Size

The learning process is not executed by processing sample per sample, due to the possibility of many computers to process more than one sample at a time; also with the support of parallel programming. Therefore, it is common to pass a number of samples at the same time for the model to process them. This number of samples passed as a group is called the batch size.

Together, the epochs and the batch size define how fast or slow the process of learning is performed; and together with the rest of the hyperparameters, affect directly the performance of the model.

3.2.4 *Process*

The process used to build the final LSTM model was developed in a similar way as with the SVR model.

However, as a first approach, an initial and very simple neural network was implemented without using any libraries. This implementation was used as a hint of how could a NN be implemented. Afterward, research on Python libraries providing the capability of implementing LSTMs was performed. Consequently, theano and tensorflow were found both as powerful libraries to implement the model. However, due to previous experiences and performance reliability, tensorflow was selected. In a similar way, the keras library was found to be able to run over both tensorflow and theano, besides providing an "easy-to-use" interface that would simplify the implementation of the model.

Nevertheless, keras also provides different options that can be used in the construction process of a NN. One of these options is the basic model, which can be built using the class "Sequential" or the class "Model". After a brief review of the keras' documentation, using the "Sequential" class was found to be simpler. Additional support on the use of the class was found on [60].

```
model = Sequential()
model.add(LSTM(1, input_shape=(X_train.shape[1], X_train.shape[2]))
model.add(Dense(1, kernel_initializer='uniform', activation='linear'))
```

The previous sample code was the first approach used for the LSTM model. It consists of two layers with one node each. The definition of the first layer creates an LSTM layer with one node, which works also as the input layer; thus, the input shape must be specified. The second layer is a Dense layer.

Given that some of the hyperparameters are defined during the compilation of the model and some others during the training; below, the initial approach for model compilation and training is presented.

```
# Compile model
model.compile(loss=mae, optimizer=adam, metrics=['accuracy'])

# Fit the model
hist = model.fit(X_train, y_train, epochs=EPOCHS_NUM, batch_size=BATCH_SZE, verbose
    =2, validation_data=(X_test, y_test))
```

The previous code uses the built-in implementation and default configuration for the MAE (Mean Absolute Error) function, as well as the adam optimizer. The epochs and batch size for the training process are defined through parameters in the python script.

With this architecture, around sixty initial tests were performed, for which the accuracy percentage achieved varied between 50% and 80%. This variation was due to the changes in the data sets, mainly in its features, the time length for each data point, as well as the hyperparameters. However, most of these results included data sets which contained too many zero-valued data points, which caused their predictions to be zero, being considered not reliable and therefore discarded on later stages of the work.

Following, a second LSTM approach was created, which made it possible to build a model including multiple hidden layers. This, so that better results could be achieved by increasing the processing of information by the neurons.

```
model = Sequential()
model.add(LSTM(1, input_shape=(X_train.shape[1], X_train.shape[2]), return_seq=True))
model.add(LSTM(1))
model.add(Dense(1, kernel_initializer='uniform', activation='linear'))
```

The prior code includes the "return_seq" parameter set to true for the input layer; which is required when stacking multiple LSTM layers. This makes it possible to include a second LSTM layer containing also one node. The number of nodes was modified accordingly for later tests. The prior changes, however, did not represent a considerable impact on the results obtained with the initial architecture, even though different changes were also applied to the datasets used and the hyperparameters.

Nevertheless, in order to break any limitations and be able to construct dynamically deep networks, the following snippet of code was implemented:

```
# create model
  model = Sequential()
  if len(LSTM_UNITS) == 1:
    model.add(LSTM(LSTM_UNITS[0], input_shape=(X_train.shape[1], X_train.shape[2]),
        activation=LSTM_ACTIVATION, kernel_initializer=init))
  else:
    lastLayer = len(LSTM_UNITS) - 1
    for units in LSTM_UNITS:
      if lastLayer == 0:
        model.add(LSTM(units, activation=LSTM_ACTIVATION, kernel_initializer=init))
```

```
    elif lastLayer == len(LSTM_UNITS) - 1:
      model.add(LSTM(units, input_shape=(X_train.shape[1], X_train.shape[2]),
          activation=LSTM_ACTIVATION, kernel_initializer=init, recurrent_activation
          ='hard_sigmoid', return_sequences=True))
    else:
      model.add(LSTM(units, activation=LSTM_ACTIVATION, kernel_initializer=init,
          recurrent_activation='hard_sigmoid', return_sequences=True))
    lastLayer-=1
  if ADD_DENSE == True:
    model.add(Dense(1, kernel_initializer=init))
```

The previous code snippet provides the opportunity of defining the number of LSTM neurons per layer, by using the array LSTM_UNITS. Each number in the array represents the number of neurons in a new LSTM layer, therefore being possible to create dynamically width and deep models. However, due to the computational limitations, most of the executions contained three layers, even though two tests using five layers were performed as well.

Similarly, at the end of the snippet, a boolean validation controlling the inclusion of a single-neuron "Dense" layer was implemented. This based on the practical implementations on [60] [61].

Likewise, the model compilation process also suffered modifications, which increased, even more, the sum of possible configurations to use when testing the models. The modification consisted of stop using the default values for the optimizers as it had been done until the moment, and implement a mechanism to assign the respective hyperparameters by using python script parameters.

```
def getOptimizer():
  if OPTIMIZER == 'sgd':
    optimizer = optimizers.SGD(lr=LEARNING_RATE, momentum=0.9, decay=0.1, nesterov=
        True)
  elif OPTIMIZER == 'rmsprop':
    optimizer = optimizers.RMSprop(lr=LEARNING_RATE, clipvalue=0.5, rho=0.9, epsilon=
        None, decay=0.0)
  elif OPTIMIZER == 'adagrad':
    optimizer = optimizers.Adagrad(lr=LEARNING_RATE, clipvalue=0.5, epsilon=None,
        decay=0.0)
  elif OPTIMIZER == 'adadelta':
    optimizer = optimizers.Adadelta(lr=LEARNING_RATE, clipvalue=0.5, rho=0.95,
        epsilon=None, decay=0.0)
  elif OPTIMIZER == 'adam':
    optimizer = optimizers.Adam(lr=LEARNING_RATE, beta_1=0.9, beta_2=0.999, epsilon=1
        e-08, decay=0.0, amsgrad=False)
  elif OPTIMIZER == 'adamax':
    optimizer = optimizers.Adamax(lr=LEARNING_RATE, clipvalue=0.5, beta_1=0.9, beta_2
```

```
        =0.999, epsilon=None, decay=0.0)
  elif OPTIMIZER == 'nadam':
    optimizer = optimizers.Nadam(lr=LEARNING_RATE, clipvalue=0.5, beta_1=0.9, beta_2
        =0.999, epsilon=None, schedule_decay=0.004)

  return optimizer

# Get Optimizer's instance
optim = getOptimizer()

# Compile model
model.compile(loss=LOSS_FUNC, optimizer=optim, metrics=['accuracy'])
```

The method *getOptimizer* is in charge of returning an optimizer based on a script parameter. However, an important aspect that had been ignored until that moment was the use of the learning rate as hyperparameter. The learning rate, as it was previously mentioned, plays a very important role in the process of training the NN. Unfortunately, when using keras, there is no simpler way of defining a custom value for the learning rate, than creating a custom object for the desired optimizer, which brings with it the possibility of defining other common and optimizer-specific hyperparameters. Nevertheless, due to the evident overwhelming number of different configurations available when building the model, it was decided to leave all the other hyperparameters for the optimizers with their respective default values in order to limit the scope of the current work, and only tune the learning rate through a script parameter.

3.3 Accuracy Margin

The term accuracy has been mentioned throughout the paper as the driver of the development of the proposed models. However, it is necessary to define what accuracy is and how it is measured to evaluate the predictions generated.

According to the definition given by [62], "accuracy can be defined as the degree of approximation to a certain expected value". A definition closer to ML defines accuracy as a performance measure that needs to be defined to evaluate a performed task [22].

Based on these two definitions, it is possible to set a standard by which the predictions will be judged as accurate or inaccurate.

At the early stages of the implementations, only predictions that were exactly equal as the target value were considered accurate. However, the percentages of accuracy achieved with this definition were not good enough. Because of that, the necessity of becoming more flexible with the definition of what can be considered accurate or

not rose, and was tackled by defining **an initial standard margin of accuracy which considers as accurate values as far as +2 or -2**, from the target value.

This definition of accuracy margin improved the performance of the executed tests. After reviewing some of the weekly predictions results obtained, both SVR and LSTM models were performing very good approximations to the target values; nonetheless, these were not good enough to be inside the +/- 2 accuracy margin previously defined. For this reason, expanding the accuracy margin in function of the length of the period considered for every data sample was proposed; yielding much better percentages of accuracy.

The accuracy margin was expanded by considering the +/-2 initial margin, good enough as a per-day margin. Therefore, increasing by two with every day added to the period's length. As a result of this, **an accuracy margin of 14 was considered for weekly predictions whereas daily predictions continued using a margin of 2**.

4 RESULTS

In this chapter, the achieved results will be presented. However, due to the significant number of tests and different results, only the best results will be detailed.

The definition of the initial accuracy margin boosted the performance of the initial predictions. However, there were mainly two reasons that did not allow the obtained results to be completely satisfactory:

- Length of Periods of Time per Sample: some predictions were executed by having samples containing per-day data, whereas some other datasets were built containing per-week data samples. One-day predictions outperformed weekly predictions, as the target values were generally smaller and, therefore, easier to approach to by the models. However, these one-day predictions would force the institutions to react in a very short period of time for every type of article in every location available. For this reason, predictions based on longer periods of time were preferred. Therefore, even though daily predictions achieved better overall results, it had much more value to be able to achieve average/good results with weekly predictions.

- Zero Demand Datasets: some predictions achieved a high accuracy percentage based on the fact that the article had never been used in a certain location, which causes the datasets for these cases to contain only zeros as demand for every time step. Furthermore, the environment from which the data was extracted to perform this work, contained a large number of these cases. Therefore, due to the fact that a clear majority of articles have no-demand in some locations, all of their predictions are zero, which were correctly predicted by the model with no-effort, thus, producing high percentages of accuracy. However, due to the nature of the datasets used to perform the prediction, it was not possible to infer that the model had been able to interpret the pattern on the data in the correct way, or that it would be able to generalize well enough for future predictions and, therefore,

produce meaningful results. For that reason, those results were discarded.

However, not only by discarding these unsatisfactory results a better performance could be achieved. All the datasets containing less than 60% of samples with non-zero demand values were considered not to provide enough information to perform a meaningful prediction on them. Thus, fewer predictions were executed, making it more difficult to achieve an overall high accuracy percentage.

This modification, together with the definition of the accuracy margin with a +/- 2 margin per day regarding the lapse of time for every data sample, improved the performance of the models to desired levels.

Four different results will be presented, two of which belong to every algorithm (SVR and LSTM). Furthermore, for every model two results will be presented, based on the length of the period predicted (one or seven days).

4.1 SVR

4.1.1 Daily Prediction

For daily predictions, i.e. samples containing daily data, the SVR achieved good performance with the configuration detailed in table 4.1.

Table 4.1: Configuration for Daily Predictions with SVR

Attribute	Value
Features	T-5_Frequency,T-4_Frequency,T-3_Frequency,T-2_Frequency,T-1_Frequency
Output	Frequency
C	1
ϵ	0.0003
γ	$1/5$
Coeff_	0.0003
Kernel	poly
Degree	1
One Sample Prediction	True
Execution Group	Type
Accuracy Margin	2

In figure 4.1, it is possible to visualize the target value plot of the data set, together with the line that the fitting model creates based on the data set.

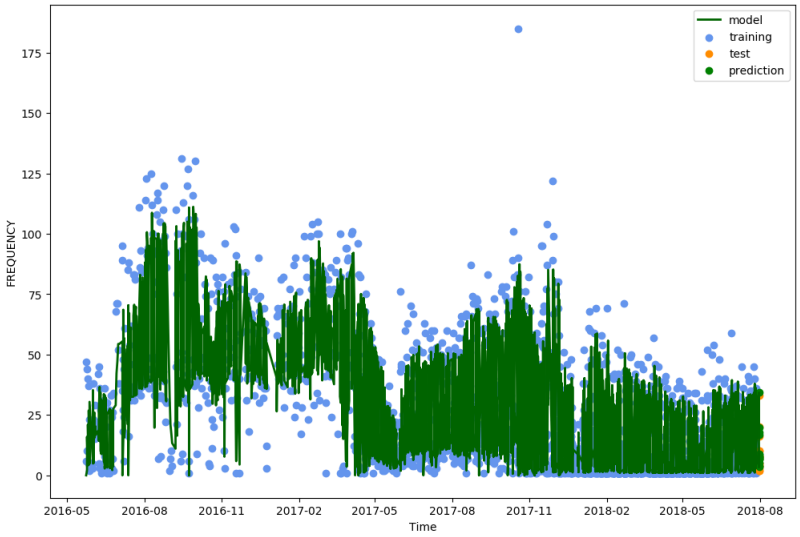

Figure 4.1: SVR daily prediction.

As is clear, the dataset was separated into groups of sub-datasets according to the article type. In this case, it is possible that more than one prediction will be performed per sub-dataset, due to the same article type being used, or having demand in different locations. Therefore, a prediction per location should be performed, producing a view that might not be very clear due to having more than one value in the Y-axis per value in the X-axis.

In the predictions corresponding to the sub-dataset that was presented previously, a total of fifteen predictions were performed, which are reviewed in table 4.2.

By virtue of the length of the period used in the datasets, i.e. per-day samples, the margin of accuracy defined for the execution was +/-2. Taking a closer look on the previous comparison between predicted and target values, it is noticeable that out of these fifteen predictions, nine were inside the set accuracy margin, which means this execution had a 60% accuracy, whereas the **overall performance of this model yielded a 67.57% accuracy**.

4.1.2 Weekly Prediction

The model selected as the best performing weekly-based SVR was built with the configuration presented in table 4.3.

For this execution, a polynomial kernel with degree one was used, which yielded

Table 4.2: Results for Daily Predictions with SVR

Predicted Value	Target Value
8.46170806	5
7.67133433	2
6.50799877	3
17.70121258	16
16.83663561	10
3.39958202	4
3.9134206	4
34.21526691	33
7.01166385	2
7.81330795	8
8.79315287	8
7.65261056	10
6.98821792	5
5.05507247	7
19.8726975	20

Table 4.3: Configuration for Weekly Predictions with SVR

Attribute	Value
Features	T-5_Frequency,T-4_Frequency,T-3_Frequency,T-2_Frequency,T-1_Frequency,
Output	Frequency
C	1
ϵ	0.0003
γ	$1/5$
Coeff_	0.0003
Kernel	poly
Degree	1
One Sample Prediction	True
Execution Group	Type and Location
Accuracy Margin	14

three different predicted values for the three different codes of the articles that are part of this type of article. A comparison between the three predicted values with its respective target value is presented in table 4.4.

As can be seen, the difference between the predicted and the target values is, in every case, smaller than the accuracy margin set for weekly predictions. For this reason, the prediction for this article type yielded 100% accuracy. However, in the **overall performance**, this configuration achieved **90.89%**.

Table 4.4: Results for Weekly Predictions with SVR

Predicted Value	Target Value
91.89805443	87
41.91669914	45
174.13818893	180

4.2 LSTM

4.2.1 Daily Prediction

The LSTM model that showed a better performance for a daily prediction used the configuration detailed in table 4.5.

Table 4.5: Configuration for Daily Predictions with LSTM

Attribute	Value
Features	Point in time, Location, Type, Code, Time to expiration, T-1_Frequency
Output	Frequency
Learning Rate	0.01
Depth	1 LSTM Layers and 1 Dense Layer
Width	8 - 1
Activation	softplus
Loss	mae
Optimizer	adam
Epochs	500
Batch Size	25
One Sample Prediction	True
Execution Group	Type
Accuracy Margin	2

As it is possible to infer, this configuration creates groups of sub-datasets based on the type of the article. However, during the execution of the model, more than one prediction can be performed due to the fact that every article type should be predicted a demand value in every location in which it has previously had a demand. In this case, a total of twelve predictions were performed, due to this article being used in twelve different locations.

A brief comparison between the predicted and the target values are reviewed in table 4.6.

Table 4.6: Results for Daily Predictions with LSTM

Predicted Value	Target Value
6.279979	9
18.479515	15
2.8509958	3
14.708367	26
27.36775	30
1.0042635	1
0.99853075	1
1.0191547	4
1.502041	2
0.99853075	1
1.0015465	2
1.4995872	1

Having a deeper look into the summary of the results, it is easy to see that seven out of twelve predictions fell into the accuracy margin set for daily predictions; which yields an accuracy percentage of 58.33%; whereas **the overall performance of the model was able to achieve a 58.39%**.

Figure 4.2 provide an insight into the dataset used for the previously examined execution, together with the fitted line by the model.

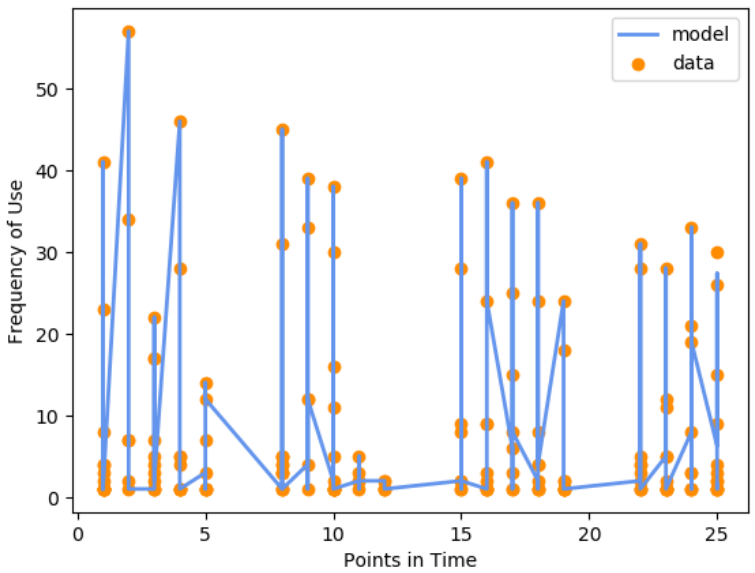

Figure 4.2: LSTM daily prediction.

On the other hand, figure 4.3 presents the evolution of the accuracy value during the process of training and validation for every epoch the model was configured to run; whereas figure 4.4 provides us a visual insight into the behavior of the loss function's value throughout the training and validation of every epoch.

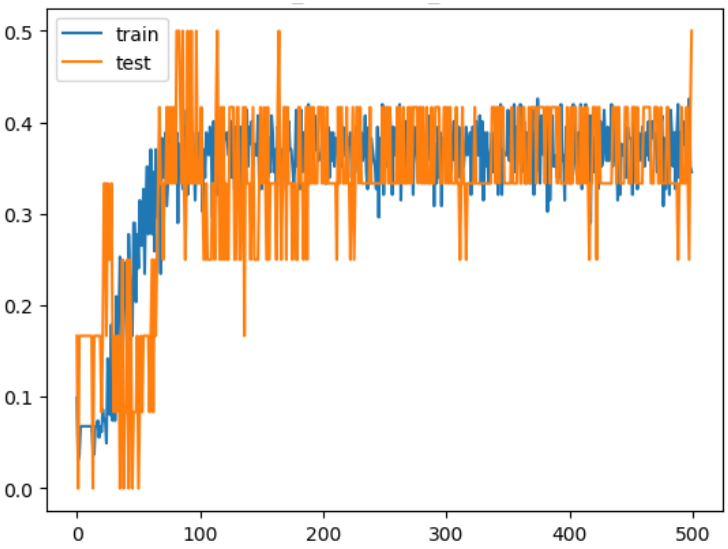

Figure 4.3: LSTM daily prediction accuracy.

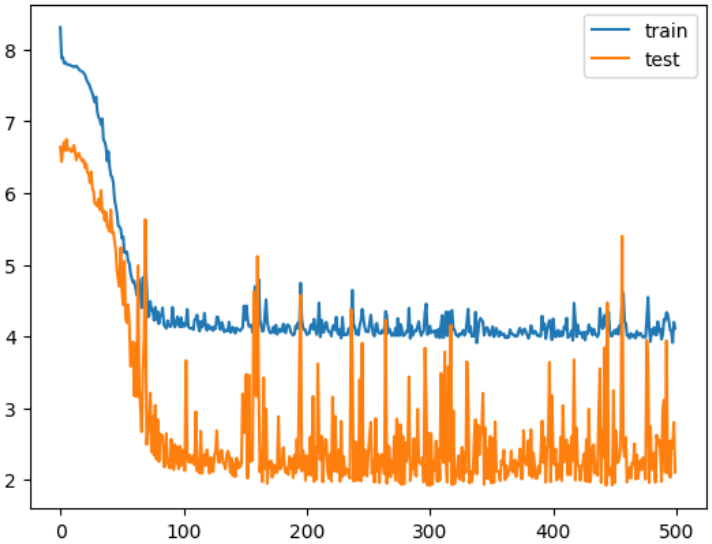

Figure 4.4: LSTM daily prediction loss.

4.2.2 Weekly Prediction

The best prediction for LSTM using weekly samples was achieved with the configuration presented in table 4.7.

Table 4.7: Configuration for Weekly Predictions with LSTM

Attribute	Value
Features	T-1_Frequency
Output	Frequency
Learning Rate	0.007
Depth	2 LSTM Layers, 1 Dense Layer
Width	2 - 1 - 1
Activation	tanh
Loss	mae
Optimizer	adam
Epochs	500
Batch Size	20
One Sample Prediction	True
Execution Group	Type and Location
Accuracy Margin	14

In figure 4.5 we can visualize the frequency of use for an article in a given location (neither of those pieces of information will be clarified in comply of the confidential information policy agreed with LM-Instruments). Similarly, we can appreciate the predicted as well as the target values. As presented in table 4.8, three predictions were performed, due to this article type having three different codes and every code presenting a different demand in the same location. The predicted values were the same for the three codes, $y' = 41.468395$, whereas the target values were $y_1 = 87$, $y_2 = 45$ and $y_3 = 180$. This yield an accuracy of 33% for this prediction; whereas the overall model configuration yielded an **overall accuracy percentage of 73.44%**.

Table 4.8: Results for Weekly Predictions with LSTM

Predicted Value	Target Value
41.468395	87
41.468395	45
41.468395	180

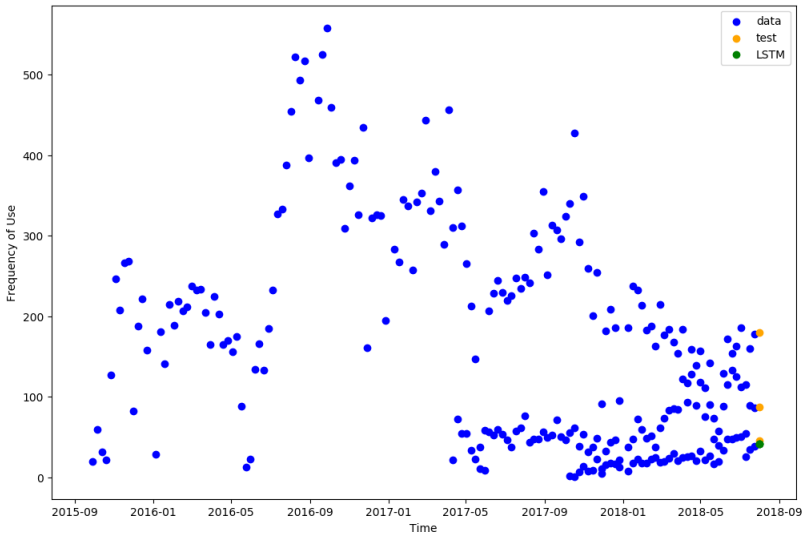

Figure 4.5: LSTM weekly prediction.

Figure 4.6 present the variation of the internal accuracy of the model during the training and the test process for each one of the epochs. It is important to note that this accuracy only considers the exact value as an accurate prediction. However, the closer the predicted value is to the target value, the bigger the accuracy. In the same way, it is also important to note that this accuracy can take values only between zero and one.

Apart from this, figure 4.7 present the variations in the value of the loss function, which in this case was MAE (Mean Absolute Error); throughout the process of training and validation for each one of the epochs run.

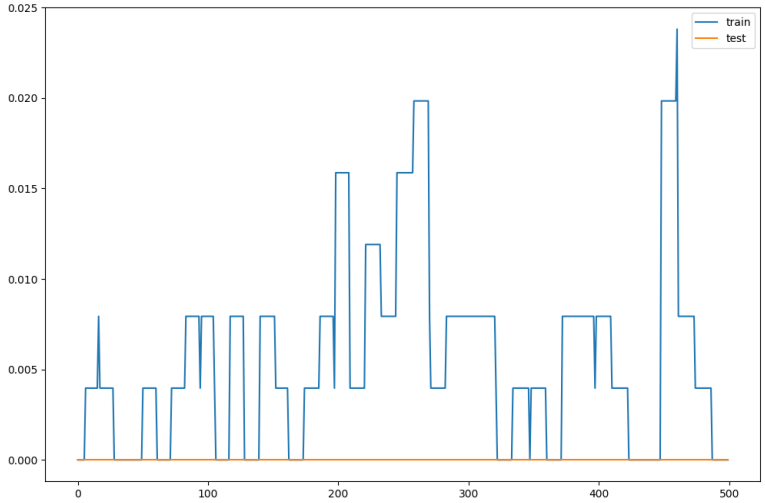

Figure 4.6: LSTM weekly prediction accuracy.

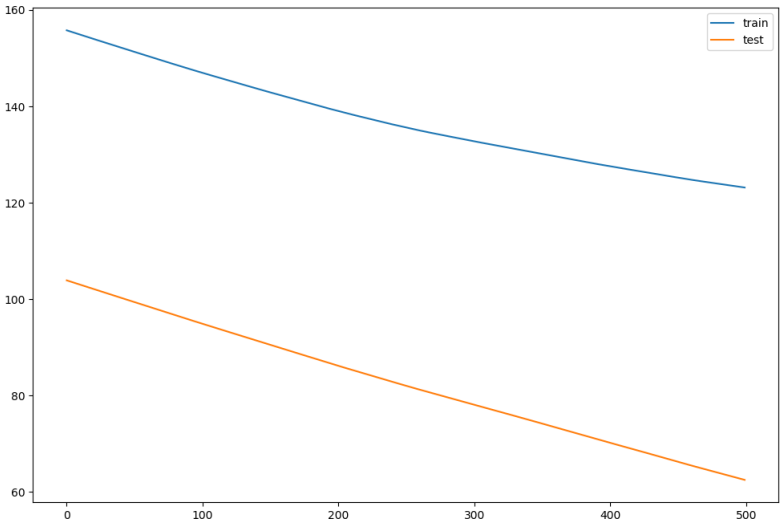

Figure 4.7: LSTM weekly prediction loss.

4.3 Analysis

In order to be able to analyze the four prior selected results, it is useful to do a summary of them. This summary is presented in table 4.9.

Table 4.9: Comparison of SVR & LSTM Results

Model	Overall Result
SVR daily-based	67.92%
SVR weekly-based	90.89%
LSTM daily-based	58.39%
LSTM weekly-based	73.44%

It is worth to mention also that the article selected for the examples, is one of the most commonly used articles in the institution. However, according to what was experienced during the execution of these tests, it was clear that no direct relation exists between the result of the predictions presented for this specific article and the overall result.

A first thing to notice is that in general, weekly predictions have outperformed daily predictions, as it is visible in table 4.9. Nonetheless, this is due to the final definition of the accuracy margin based on the number of days the samples of the dataset contain.

Moreover, the results for week-based predictions were in general outperformed by the daily-based predictions when both of these considered the same initial accuracy margin.

However, one of the most striking conclusions that can be drawn, is that SVR seemed to perform better than LSTM. When having a look at the executions using Frequency as the output feature for both of the techniques, SVR always seems to approach closer to the target values than LSTM. However, it is extremely important to highlight that this does not mean that SVR is better than LSTM. As it was mentioned in the scope of the current work, the objective is to select a model configuration able to successfully predict the demand by using libraries containing off-the-shelf implementations. However, as we already had a glimpse when reviewing how to construct the model, LSTM is highly customizable and therefore, very complex to tune. Nonetheless, a more in-depth tuning of the network using other optimizing techniques could cause LSTM to outperform the results obtained during this work.

A factor that could influence the better performance of SVR over LSTM is the complexity of using each model. Even though SVR is recognized as a complex model to tune and therefore utilize, due to the number of "basic" hyperparameters available and the ones that might appear due to the selection of specific values of the first ones; LSTM and in general all of the Deep Learning (DL) models have a higher level of complexity. This, because of the power of these models as well as their capacity to implement or be used together with different optimization techniques, or even use slightly different versions of a standardized model. The aforementioned complexity plays a toll on the user intending to implement the models, increasing the probabilities of misconfiguration or the use of incorrect values for the hyperparameters available in a model; therefore, decreasing the probabilities of finding the optimal configuration of a model.

Besides the prior, an additional reason that can explain the difference in the performance of the two models is the amount of training data. As it was mentioned by [40], NNs generally require a large amount of training data to converge. On the other hand, SVR during the training process does not make use of all the data points due to its cost function discarding the data points that fall inside the defined ϵ margin, as explained by [26].

A solution attempted to tackle this issue, was the use of a larger number of epochs during the training process. However, it was perceived that the impact of this in the

overall result was negative, as not only the runtime increase significantly but also, the percentage of accuracy did not improve and in most of the cases it even decreased. The prior might be due to overfitting the training data, which causes the model to fit too well the training data but perform poorly with new or unknown data.

Similarly, when comparing the selected kernel in SVR and the activation functions in LSTM, as well as their learning process; it is possible to infer that the way the weights are updated in SVR is much more direct and simpler than in LSTM, as the latter makes use of the algorithm known as Back Propagation Through Time (BPTT) and must basically traverse the network backward to do this update; whereas SVR by using the adam optimizer, applies the update in a much more direct way. Moreover, the calculation of the objective function is also simpler in SVR, in this case by using a polynomial function of degree one, than it is in LSTM, by traversing the network using the activation functions to find the final result. The prior might benefit the performance of SVR when handling basic data types as the current implementation does. However, when handling more complex structures of data such as images, audio or even video recordings, the power of LSTM will safely place it as a good solution to use.

Moreover, when selecting the models which performed the best, only those in which the dataset used the Frequency as output feature, were considered due to the fact that, when using the frequency variance, the obtained results seemed to be mostly driven by the fact that most of the target values were zero. Furthermore, it was observed that most of the predicted values were very close to zero; hence, they were also inside the accuracy margin set for daily-based predictions. However, when the target values were far enough from zero, i.e. more than 2; the predicted values did not seem to be able to approach them, and continued predicting very small values.

In the light of that behavior, datasets using frequency over the variance were preferred.

Incidentally, accordingly to what was expected, time series shaped datasets seemed to have boosted the performance of both models. It catches the attention the fact that for weekly predictions, the best results were achieved by using only time-series data features; whereas for daily predictions, even though it was also benefited from time series data, its impact was less tangible, therefore being necessary to support it with non-time-series features. This is visible when reviewing the daily predictions for LSTM.

Unfortunately, as mentioned in the definition of the scope, the results presented are exclusive for the implementation and optimization of the models. None of these results belong to real-life DTS environments, as LM-Instruments wants to include the

forecast demand feature into the software at a slow pace in a dedicated environment, that allows them to analyze the period-to-period performance.

5 CHALLENGES

In order to ensure that the present paper produces as much knowledge as possible, and with the purpose that future researchers can find the present thesis to be a good support to clearly identify the good and not-so-good practices, the biggest challenges that were faced during the development of the current work will be reviewed in this chapter.

Based on the fact that the whole process is composed by various different stages or phases, the challenges will be described in the section corresponding to the stage they emerged in.

5.1 Data Generation and DataSet Creation

The main challenge faced during the execution of this work was the treatment and preparation of the data. It must not be a secret for people with knowledge and experience of the field of ML that, the data is key when implementing any sort of ML model. This, considering that the models perform an analysis on the training data in order to be able to process new, so-called, unknown data. Therefore, a bad definition or preparation of the data could translate into algorithms not performing as well as expected or achieving confusing results that are difficult to analyze. A good insight into the importance of the data preparation is the fact that, as stated by [42], the data cleaning and preparation process represents 80% of the effort in a data science project.

The big question during the data set generation process was, what data would provide the models with more optimal information, and which structure it should have; which presented two different more fine-grained problems.

On the one hand, the question of whether to consider or discard a data point when an article had a frequency equal to zero in a given location. This, because in some datasets, the zero demand periods translated the dataset into an intermittent one, i.e. with some periods of demand different than zero, and some others equal to zero; however, for some other datasets there were only zero-demand periods, which mean that an art-

icle had never been used in that location, having no other "prediction" to do than zero. For this reason, some predictions achieved an accuracy as high as 97%, only because the number of articles not being used in specific locations was considerably higher than the number of articles that were used. As a result, the predictions performed well, even though most of them lacked in usefulness; hence, they were discarded.

The solution adopted for this was to be able to dynamically remove or not, zero-demand data points during the generation of the datasets. However, it is also possible to dynamically select whether to remove zero-valued data points or not, right before the execution of the prediction, including an extra validation of the number of non-zero data points, so at least 60% of the total number of data points happen to be non-zero.

The decision of enabling the use of a parameter to dynamically select whether or not to discard zero-demand data points, was affected by the availability of two different output features such as the frequency and the frequency variance. When the former is used, zero-demand data points tell the algorithm that the article was not used whatsoever, and the impact of ignoring or discarding those data points might be beneficial. However, when the latter is used, zeros might have a more important role as they reflect no-variance on the demand between two periods of time.

On the other hand, the selection on what pieces of data could have a positive impact on the performance of the models and therefore be used as predictors was also difficult, for the reason that some of the attributes of an article have no historical data. This means that some potential features do not store old values, but only the current ones. This was opposite to what was desired, as the frequency is supposed to vary in terms of the time, based on some characteristics or features of the type of article; hence, old values are required to be used as training samples, whereas the current values would be used as the predictors for the upcoming frequency. However, for potential features such as the cycle count or the expiration dates, there was no way to find their values at previous periods of time, as the data base stores only the value for the current period.

Even though this was the case, a workaround was implemented in order to be able to use values reflecting the state of each data point. This workaround varies from feature to feature, but for the sake of simplicity and having an example, the solution implemented for the cycle count and time to expiration will be presented below.

The cycle count attribute is a simple counter of how many times an article has traversed through the process flow of articles in the institution. In other words, how many cycles has an article completed in the process. In order to be able to have different

values per data sample, and considering that the articles would be grouped by type, the cycle count of each article that had non-zero demand in a given location was added to the "final" cycle count of the data sample. In a more detailed way, this means that, if for a single day d, two different articles x and y of the same type t were both used in a location l; given that $x_{cyclecount} = 3$ and $y_{cyclecount} = 6$, then the sample s will have a cycle count $s_{cyclecount} = x_{cyclecount} + y_{cyclecount}/n$, where n is the number of articles that were added; in this case two.

Thus, every sample would not only reflect how high or low the demand was for the type of article, but also how much or little use the articles involved had in the demand.

The time to expiration, in the same way as the cycle count, contains a value that can be different among all the articles that belong to the same article type. The fact that its data type is date, and not integer or any numerical value, made it slightly more complicated to find a way to turn these values into meaningful data for the algorithms. The time to expiration is, then, calculated for a sample by first finding the difference in days between the beginning of the period of the sample and the date in which the article expires. For this reason, big numbers would mean that the article was used long before its expiration date, whereas a small number would mean that the article was close to expiring when it was used. However, this number is not left as it is, and afterwards it is divided by the length of the period, therefore being divided by one (daily predictions) or by seven (weekly predictions). The previous operation is performed for every unique article that had non-zero demand during a given period of time in a specific location. Following, these values are grouped based on the type of article, thus, finding the mean for the results of the unique articles belonging to the same type.

5.2 Building and Tuning the Models

In this phase of the implementation, one of the biggest challenges was the fact that the data available might not be enough, especially for LSTM-RNN. According to some researchers, for NN "in many cases the amount of training data required for convergence is large" [40]. This issue arises, as explained by the author, due to the models trying to find the mathematical function that relates the predictors to the target value, achieving a certain level of accuracy; all of this with a limited amount of training data. As was already mentioned in a previous chapter, at the early stages we faced a limitation in the amount of data available, even to the point that the database had to be modified

in order to be able to use a reasonable number of transactions, to be able to train our models. However, due to the fact that some of the data points are discarded for having a demand equal to zero, most of the datasets (when the main dataset is grouped by the type of article), become up to 40% shorter in data points.

This is a challenge that cannot be solved at this point, being the only solution to wait for the environment to generate more data. This might have been one of the reasons for SVR producing better results than LSTM-RNN.

Similarly, an additional challenge encountered in this process is the complexity of configuring and tuning the LSTM-RNN, mainly caused by the overwhelming number of hyperparameters that can be set when working with LSTM in Keras. Even though Keras simplifies and provides an easy-to-use interface when building Deep Learning (DL) models, it also considers all (or most of) the existing hyperparameters and, thus, it makes them available for a user to set. This, together with the data set generation process, entails an almost infinite number of different configurations, increasing the possibility of not testing the optimal models for our case study.

6 CONCLUSION

With the present work, it was possible to find the relationship between an industry's business goals and state-of-the-art technology as it is machine learning, and how it can support a company in the journey to improve its profitability.

Different ML models were reviewed in order to support the Dental Tracking System (DTS), developed by LM-Instruments; in the implementation of a demand forecasting tool that will place even more distance between this software and its competitors, by being able to provide its customers and users with tools to optimize their management of instruments and dental assets, lowering the operational costs and keeping good quality standard levels of service.

Through this thesis, the reader is presented a glimpse of what artificial intelligence is and how it is implemented in a real-life project. Furthermore, with the work performed during the implementation of this project, the software team and other business planners of LM-Instruments were introduced to machine learning and the benefits it can bring at a larger scale for the company. In a similar way, a contribution is made to the use of machine learning tools for the implementation of SCM practices in the health care sector.

In regards to the implementation, the suggested tools to use are Python 3.5 as the programming language, together with the scikit-learning library.

From the two proposed models, Support Vector Regressor (SVR) and Long-Short-Term Memory Neural Networks (LSTM); the first one proved to suit more this specific scenario, for many reasons.

As a first and probably most influential reason; the type and amount of data provided might not have been enough to train the LSTNM model in a proper way, as [40] highlighted. On the other hand, SVR does not need a large amount of data, as according to the set parameters, it discards some or most of the data to create the support vectors based on which the model performs.

Additionally, the periodicity of the data set proved to have a big impact on the

performance of the models, as it affects the patterns present in the data and the way the models interpret them. Two different periodicities were used for the implementation and tuning of the models, which are daily (per-day predictions) and weekly (per-seven-days predictions). These two periodicities were selected as the first solution approach. The performance comparison for these two periodicities has a tight dependence on the accuracy margin considered. For this reason, when both periodicities used the same accuracy margin, the daily predictions outperformed the weekly predictions. The prior, due to the daily demand being considerably smaller (closer to zero) than the one obtained on a weekly basis for an article. This caused the models to produce predicted demands that approximate better to the target demands. However, for weekly predictions even though the results were very close, the higher the target demand was, the smaller the error percentage should be in order to consider a prediction as accurate. Once a more fair accuracy margin was established, weekly predictions outperformed by a significant percentage the daily predictions.

Furthermore, SVR proved to be simpler to implement than LSTM, even though because of the number of hyperparameters available, SVR is considered to be complex to tune and optimize. However, considering the lack of prior experience of LM-Instruments on ML tools and their implementations; SVR will result in an easier technique to understand than LSTM; thus, the built model will be easier to tune further. Moreover, regarding the performance, SVR not only showed better results than LSTM but also required less time of execution than its counterpart. This is a high importance matter for LM-Instruments, as the plans for DTS are to not only be available for customers as software in the cloud; but also to be delivered as a local installation in customers' facilities. In response to this requirement, LM-Instruments plans to enable a ready-to-use portable installation of the software, which enforces the application to be as light as possible, consuming the least amount of computing resources. For this reason, SVR becomes not only a good fit for LM-Instruments plans but also an optimal solution.

Based on the prior findings, the suggested solution for LM to implement in the DTS software is the Support Vector Regressor model for weekly predictions, previously presented and detailed.

BIBLIOGRAPHY

[1] Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, (14):199–222, 2004.

[2] Daniel Svozil, Vladimír Kvasnicka, and Jiri Pospichal. Introduction to multilayer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, 39(1):43 – 62, 1997.

[3] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *In Proc. 30th International Conference on Machine Learning*, page 1310–1318, 2013.

[4] A. Graves, A. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, 2013. doi:10.1109/ICASSP.2013.6638947.

[5] World Health Organization. Preamble to the Constitution of WHO as adopted by the International Health Conference. https://www.who.int/suggestions/faq/en/. [Online; accessed 03-December-2018].

[6] Anthony Ledesma, Connor McCulloh, Haden Wieck, and Mason Yang. Health care sector overview. *Washington State University*, 2014.

[7] OECD/EU. Health at a glance: Europe 2016 – state of health in the eu cycle. *OECD Publishing, Paris*, 2016. doi:http://dx.doi.org/10.1787/9789264265592-en.

[8] Douglas M. Lambert, Martha C. Cooper, and Janus D. Pagh. Supply chain management: Implementation issues and research opportunities. *The International Journal of Logistics Management*, 9(2):1–20, 1998. doi:https://doi.org/10.1108/09574099810805807.

[9] Keely L. Croxton, Sebastián J. García-Dastugue, Douglas M. Lamber, and Dale S. Rogers. The supply chain management processes. *The International Journal of Logistics Management*, 12(2):13–36, 2001. doi:https://doi.org/10.1108/09574099810805807.

[10] Fred R. Ricker and Ravi Kalakota. Order fulfillment: The hidden key to e-commerce success. *Supply Chain Management Review*, 11(3):60–70, 1999.

[11] Amy C. Edmondson and Ingrid M. Nembhard. Product development and learning in project teams: The challenges are the benefits. *The journal of product innovation management*, 26:123–138, 2009.

[12] Carlos Callender & Scott E. Grasman. Barriers and best practices for material management in the healthcare sector. *Engineering Management Journal*, 4(22):11–19, 2010. doi:https://doi.org/10.1080/10429247.2010.11431875.

[13] Amjad Hussain Dimitris Bertsimas, Nathan Kallus. Inventory management in the era of big data. *Production and Operation Management*, 25(12):2006–2009, 2016.

[14] Deepesh Singh and Ajay Verma. Inventory management in supply chain. *Materials Today: Proceedings*, 5:3867–3872, 2018. doi:https://doi.org/10.1016/j.matpr.2017.11.641.

[15] Edward A. Silver. Inventory management: An overview, canadian publications, practical applications and suggestions for future research. *INFOR Information Systems and Operational Research*, 2018. doi:10.3138/infor.46.1.15.

[16] Prem Vrat. *Materials Management: an integrated systems approach*. Springer, 2014.

[17] Geoff Relph and Catherine Milner. *Inventory Management: Advanced methods for managing inventory within business systems*. Kogan Page, 2016.

[18] El Naqa I. and Murphy M.J. What is machine learning? *Machine Learning in Radiation Oncology*, pages 3–11, 2015. Springer, Cham. doi:https://doi.org/10.1007/978-3-319-18305-3_1.

[19] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning From Theory to Algorithms*. Cambridge University Press, 2014.

[20] Tom M. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.

[21] Ethem Alpaydın. *Introduction to Machine Learning*. The MIT Press, 2010.

[22] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly, 2017.

[23] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

[24] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[25] Vladimir Vapnik, Harris Drucker, Chris J.C. Burges, Linda Kaufman, and Alex Smola. Support Vector Regression Machines. *Advances in Neural Information Processing Systems*, 9:155 – 161, 1997.

[26] Debasish Basak, Srimanta Pal, and Dipak Chandra Patranabis. Support vector regression review. *Neural Information Processing - Letters and Reviews*, 11(10):203–224, 2007.

[27] Karin Kandananond. A comparison of various forecasting methods for autocorrelated time series. *International Journal of Engineering Business Management*, 2012. doi:https://doi.org/10.5772/51088.

[28] Yukun Bao, Wen Wang, and Jinlong Zhang. Forecasting Intermittent Demand by SVMs Regression. *2004 IEEE International Conference on Systems, Man and Cybernetics*, 2004.

[29] Karin Kandananond. Consumer product demand forecasting based on artificial neural network and support vector machine. *International Journal of Economics and Management Engineering*, 6(3), 2012.

[30] S. Mukherjee, E. Osuna, and F. Girosi. Nonlinear prediction of chaotic time series using support vector machines. In *Neural Networks for Signal Processing VII. Proceedings of the 1997 IEEE Signal Processing Society Workshop*, pages 511–520, Sep 1997. doi:10.1109/NNSP.1997.622433.

[31] Martin Casdagli. Nonlinear prediction of chaotic time series. *Physica D: Nonlinear Phenomena*, 35(3):335 – 356, 1989. URL: http://www.sciencedirect.com/science/article/pii/0167278989900742, doi:https://doi.org/10.1016/0167-2789(89)90074-2.

[32] Warren S. Sarle. Neural networks and statistical models, 1994.

[33] J. J. Hopfield. Artificial neural networks. *IEEE Circuits and Devices Magazine*, 4(5):3–10, Sept 1988. doi:10.1109/101.8118.

[34] Zachary C. Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning, 2015. arXiv preprint arXiv:1506.00019.

[35] Rustam Vahidov Real Carbonneau, Kevin Laframboise. Application of machine learning techniques for supply chain demand forecasting. *European Journal of Operational Research*, (184):1140–1154, 2008.

[36] Noorfa Haszlinna Mustaffa and Andrew Potter. Healthcare supply chain management in malaysia: a case study. *Supply Chain Management: An International Journal*, 14(3):234 – 243, 2009. doi:https://doi.org/10.1108/13598540910954575.

[37] Diana Cordes Feibert, Peter Jacobsen, and Michael Wallin. *Improving Healthcare Logistics Processes*. PhD thesis, 2017.

[38] Murugan Anandarajan Fariborz Y. Partovi. Classifying inventory using an artificial neural network approach. *Computers & Industrial Engineering*, (41):389–404, 2002.

[39] Andrea Bacchetti and Nicola Saccani. Spare parts classification and demand forecasting for stock control: Investigating the gap between research and practice. *Omega: The international journal of management science*, 40(6):722 – 737, 2012. doi:https://doi.org/10.1016/j.omega.2011.06.008.

[40] Somnath Mukhopadhyay Rafael S. Gutierrez, Adriano O. Solis. Lumpy demand forecasting using neural networks. *International journal of production economics*, 111(2):409 – 420, 2008. doi:https://doi.org/10.1016/j.ijpe.2007.01.007.

[41] Henry F.Schwarz Thomas R.Willemain, Charles N.Smart. A new approach to forecasting intermittent demand for service parts inventories. *International Journal of Forecasting*, 20(3):375 – 387, 2004. doi:https://doi.org/10.1016/S0169-2070(03)00013-X.

[42] Shichao Zhang, Chengqi Zhang, and Qiang Yang. Data preparation for data mining. *Applied Artificial Intelligence*, 5-6(17):375–381, 2003.

[43] Huan Liu and Hiroshi Motoda. Feature transformation and subset selection. *IEEE Intelligent Systems*, 13(2), 1998. doi:http://doi.ieeecomputersociety.org/10.1109/MIS.1998.671088.

[44] Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting*. Springer, 2002.

[45] P. Newbold and C. W. J. Granger. Experience with forecasting univariate time series and the combination of forecasts. *Journal of the Royal Statistical Society. Series A (General)*, 137(2):131 – 165, 1974.

[46] Qinghua Hu, Daren Yu, and Zongxia Xie. Information-preserving hybrid data reduction based on fuzzy-rough techniques. *Pattern Recognition Letters*, 27(5), 2006. doi:https://doi.org/10.1016/j.patrec.2005.09.004.

[47] David L. Donoho. High-dimensional data analysis: The curses and blessings of dimensionality. *Aide-Memoire*, 2000.

[48] Stephen Marsland. *MACHINE LEARNING An Algorithmic Perspective*. CRC Press, 2015.

[49] Peter Harrington. *Machine Learning in Action*. Manning Publications, 2012.

[50] Sunila Gollapudi. *Practical Machine Learning*. Packt Publishing, 2016.

[51] Gavin Hackeling. *Mastering Machine Learning with scikit-learn*. Packt Publishing, 2014.

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

[53] Luca Lorenzi, Grégoire Mercier, and Farid Melgani. Support vector regression with kernel combination for missing data reconstruction. *IEEE GEOSCIENCE AND REMOTE SENSING LETTERS*, 10(2):367–371, 2013.

[54] Hasim Sak, Andrew Senior, and Francoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. *Interspeech*, 2014.

[55] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *Cornell University*, 2015.

[56] Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[57] George Saon, Hagen Soltau, Ahmad Emami, and Michael Picheny. Unfolded recurrent neural networks for speech recognition. *Interspeech*, 2014.

[58] Gaurav Pandey and Ambedkar Dukkipati. To go deep or wide in learning?, 2014.

[59] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width, 2014.

[60] Antonio Gulli and Sujit Pal. *Deep Learning with Keras*. Packt Publishing, 2017.

[61] Kai Chen, Yi Zhou, and Fangyan Dai. A lstm-based method for stock returns prediction - a case study of china stock market. *IEEE International Conference on Big Data*, 2015.

[62] Mathias Hofer, Gero Strauss, Kirill Koulechov, and Andreas Dietz. Definition of accuracy and precision — evaluating cas -systems. *International Congress Series*, 1281:548 – 552, 2005.