FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Using data science methods for forecasting and managing the inventory of community pharmacies

Gabriel Fraga Outeiro

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Professor Mário Amorim Lopes

June 30, 2023

Resumo

A previsão exacta da procura é um aspeto crucial de uma gestão de inventário eficaz. Envolve a previsão da procura de produtos através da análise de dados históricos de vendas, padrões sazonais e outros factores de influência. Ao obter previsões precisas da procura, as empresas podem otimizar os seus níveis de inventário para satisfazer a procura dos clientes sem se depararem com rupturas de stock ou inventário excessivo.

Na indústria farmacêutica, a distinção entre vendas ao fim de semana e durante a semana tem um significado especial devido às variações na procura de diferentes categorias de produtos. No entanto, os distribuidores farmacêuticos enfrentam frequentemente limitações na entrega de produtos durante os fins-de-semana, uma vez que as suas operações e pessoal não estão normalmente disponíveis. Consequentemente, as farmácias têm de antecipar a procura ao fim de semana com bastante antecedência e fazer as encomendas em conformidade. Caso contrário, os distribuidores recorrem a visitas frequentes às farmácias ao longo do dia, o que leva a um desperdício de tempo e contribui para a poluição ambiental.

Para enfrentar estes desafios, propomos uma arquitetura inovadora que aproveita exclusivamente os dados de vendas ao fim de semana e agrupa-os com base no ingrediente ativo comum (CNPEM) dentro de cada *sku* (*Stock Keeping Unit*). Esta abordagem permite uma previsão mais eficiente da procura, considerando apenas os padrões de fim de semana relevantes e agregando dados dentro dos grupos. Ao concentrar-se em grupos de *sku* que partilham o mesmo ingrediente ativo, a arquitetura atenua o impacto do comportamento errático de cada *sku* e permite uma previsão mais informada.

Abstract

Accurate demand forecasting is a crucial aspect of effective inventory management. It involves predicting product demand by analyzing historical sales data, seasonal patterns, and other influencing factors. By achieving precise demand forecasts, businesses can optimize their inventory levels to meet customer demand without facing stockouts or excessive inventory.

In the pharmaceutical industry, the distinction between weekend and weekday sales holds particular significance due to variations in demand for different product categories. However, pharmaceutical distributors often face limitations in delivering products during weekends, as their operations and staff are typically not available. Consequently, pharmacies must anticipate weekend demand well in advance and place orders accordingly. Alternatively, distributors resort to frequent visits to pharmacies throughout the day, leading to time wastage and contributing to environmental pollution.

To address these challenges, we propose an innovative architecture that leverages exclusively weekend sales data and clusters it based on the common active ingredient (CNPEM) within each stock keeping unit (*sku*). This approach allows for more efficient demand forecasting by considering only relevant weekend patterns and aggregating data within clusters. By focusing on clusters of *sku* sharing the same active ingredient, the architecture mitigates the impact of erratic individual *sku* behavior and enables more informed forecasting.

Agradecimentos

Gostaria de expressar um agradecimento especial ao Professor Mário Amorim Lopes pelas constantes conversas, apoio, disponibilidade e motivação demonstradas ao longo do desenvolvimento da dissertação.

Quero agradecer do fundo do coração aos meus pais por tudo o que fazem por mim, por todo o amor e apoio que me dão sempre. Sem eles, nada seria possível.

Ao meu irmão, agradeço todo o amor e apoio incondicionais.

A todos os meus amigos, agradeço por todas as conversas e amizade, e por me mostrarem que o mundo pode ser visto de diferentes perspetivas.

À minha namorada, agradeço pelo amor, motivação e apoio incondicionais.

Agradeço também a todas as pessoas que me ajudaram a chegar até aqui.

Gabriel Fraga Outeiro

"The darkest hour of the night comes just before the dawn"

Paulo Coelho

Contents

List of Figures

List of Tables

Abreviaturas e Símbolos

ANN	Artificial Neural Network
FTS	Fuzzy Time Series
ICEEMDAN	Improved Complete Ensemble Empirical Mode Decomposition with Adaptive Noise
MODA	Multi-Objective Dragonfly Algorithm (MODA)
LSTM NN	Long-Short-Term-Memory Neural Network
MAE	Mean Absolute Error
MSE	Mean Squared Error
RSE	Relative Squared Error
MAPE	Mean Absolute Percentage Error
ARIMA	Auto Regression Integrated Moving Average
GRU	Gated Recurrent Unit
GAT	Graph Attention Network
MQ-RNN	Multi-Quantile Recurrent Neural Network
MQ-CNN.	Multi-Quantile Convulucional Neural Network
SVR	Support Vector Regression
RMSE	Root Mean Squared Error
MAAPE	Mean Arctangent Absolute Percentage Error
NN	Neural Networks
ETS	Exponential Smoothing
LR	Linear Regression
RF	Rolling Forecast

Chapter 1

Introduction

1.1 Context

Inventory management is a crucial aspect of any business, including pharmaceutical distributors and community pharmacies. Effective inventory management ensures that the right products are available when customers need them while avoiding overstocking that can lead to waste and losses, like capital tied up in inventory and the environmental impact caused by the constant trips between the pharmaceutical distributor and the community pharmacies for the supply of depleted products.

One of the most important factors in inventory management is demand forecasting. This involves predicting the demand for products based on historical sales data, seasonal trends, and other factors that may affect sales. By accurately forecasting demand, businesses can optimize their inventory levels to ensure they have enough stock to meet customer demand without overstocking or having stockouts.

However, obtaining good accuracy in this type of product is still a significant challenge [7]. Other aspect that is needed to be taken into account, and is a main goal, is the performance when obtaining the results. The time that the process takes to be completed is an important aspect for the business planning is these situations, as it is considered to decide when to start the process.

1.2 Problems

Inventory management models assume a stationary demand and start from products with fairly regular consumption patterns. However, the reality is much more complex. Among the problems associated with low accuracy in demand forecasting, only a few will be studied throughout this dissertation and will be presented below.

The main issue is that each product variant, *sku* (Stock Keeping Unit), has a different demand trend, which makes it hard to come up with methods to handle the different irregularities in the analyzed data over time. The demand for pharmaceutical products can fluctuate significantly based on the day of the week or time of the year, which can create inventory management challenges.

Weekend and weekday sales are particularly relevant in the pharmaceutical industry, as the demand for certain products may vary depending on the day of the week. For instance, pharmacies may experience increased demand on weekends for over-the-counter medications and prescription refills, while the demand for products related to workplace injuries or illnesses may be higher on weekdays.

However, the capacity of pharmaceutical distributors to deliver products on weekends may be limited, as many of them may not have operations or staff working during weekends. This can create a challenge for pharmacies to meet the increased demand on weekends, as they may need to order products well in advance to ensure they have enough inventory on hand. Otherwise, pharmaceutical distributors usually need to go to each pharmacy more than once a day, which leads to a lot of wasted time and contributes to environmental pollution.

To overcome these challenges, pharmaceutical distributors and pharmacies must leverage effective demand forecasting models that take into account the varying demand patterns associated with weekends and weekdays. They must also implement robust inventory management systems to optimize inventory levels and ensure that they have enough stock on hand to meet customer demand.

In the context of product SKUs on the pharmaceutical industry, each SKU is associated with a characteristic molecule known as the active ingredient (CNPEM). This active ingredient is responsible for giving specific predefined attributes to the SKU. One area of investigation that warrants attention is the practice of forecasting demand by SKU group (SKUs containing the same active ingredient), as opposed to individual SKUs forecasting. This approach is particularly pertinent in situations where generic drugs hold a substantial share of the market, making it challenging to ascertain the exact percentage of sales attributable to each product. To mitigate this issue, it will be estimated the demand for collections of drugs containing the same active ingredient. By doing so, it becomes feasible to determine the corresponding market shares for each product. Moreover, this method can be valuable for predicting the demand for newly introduced pharmaceutical products since it leverages data from all SKUs to generate a comprehensive forecast.

1.3 Methodology

The methodology used in this study is essential for guiding the research process and achieving the study's goals. It outlines how the work was conducted and the overall approach followed, rather than focusing on specific methods employed.

This thesis follows a practical and data-driven approach, using techniques from data science to answer research questions. By using this approach, the study aims to find meaningful insights and draw reliable conclusions from the available data. The quantitative approach ensures a systematic analysis based on numbers and statistics to uncover patterns, trends, and relationships.

A key part of the methodology is the analysis of real-time series data. This type of data, which captures information over time, is valuable for studying dynamic phenomena and understanding

patterns in a specific field. By using real-time series analysis, the study aims to identify and improve the best methods for addressing the research objectives.

To ensure accurate results, the methodology incorporates various data science techniques. These include exploring the data, preparing it for analysis, creating useful features, building models, and evaluating their performance. By applying these techniques thoughtfully, the study aims to extract valuable insights, validate hypotheses, and make accurate predictions or classifications based on the available data.

Moreover, the methodology takes into account existing theories, literature, and industry practices to provide a broader context for the research. By considering these factors, the study ensures that the methodology aligns with established knowledge and addresses gaps in current understanding.

Additionally, the methodology acknowledges that the research process may need adjustments as new insights are gained and analyses are conducted. This flexibility allows for adapting to emerging trends, addressing challenges, and refining the research direction.

In summary, this master's thesis adopts a practical and data-driven approach, using data science methods to analyze real-time series data. By integrating various techniques and considering the context, this methodology aims to improve the methods used to achieve the research goals, ultimately contributing to a comprehensive and insightful study.

Chapter 2

Literature Review

The upcoming chapter will be structured into two segments. The first part will entail a contextual overview, wherein pivotal notions will be elucidated to facilitate comprehension of the methodologies employed in the study. The succeeding section will entail a presentation of the state of the art, wherein approaches advanced for resolving analogous issues to the current dissertation will be examined, providing a comprehensive survey of these methods and expounding on their applicability to our research. This division will be partitioned into two subsections, based on the two principal predicaments to be addressed. Therefore, the initial portion will concentrate on resolving dilemmas akin to the unpredictability of demand during weekends, while the latter component will be devoted to tackling the challenge of forecasting demand for hierarchical clusters of retail products.

2.1 Time series forecasting

Time series is a series of data points indexed in time order. Most commonly, a time series is a sequence taken at successive equally spaced points in time.

Taking into account the number of variables varying over time, there are two types of time series:

> **Uni-variant time series**: only have one variable varying over time. An example of this is the humidity captured from a sensor of a dehumidifier over time
>
> **Multi-variant time series**: have more than one variable, varying over-time

Forecasting processes available data and predicts future events to improve and help the decision process. This process can be qualitative or quantitative. Qualitative forecasting is based on people's opinions with excellent knowledge about the subject and the factors that may influence the target forecasting process. Usually, this type of forecasting is carried out when no data is available. As human decisions prevail, it is very likely to find predictions influenced by preconceived ideas. Quantitative forecasting is based on data obtained a priori and their analysis through computer algorithms [1].

2.1.1 Time Series Components

The characteristics of the time series can usually be divided into four components: seasonality, trend, cycle and noise (residuals). Typically, trend and cyclical components are combined into a single component, called trend, for simplicity. Thus, a time series, when decomposed, contains three main components: seasonality, trend, and a remainder component, the latter being what remains of the characterization of the time series, in addition to the other two components [1].

Trend: component that shows the growth trend of the time series. This trend can be increasing, decreasing or flat (no trend).

Seasonality: component that shows the repetition of a pattern periodically. In each cycle, the time series exhibits similar behaviors over a fixed period of time (such as summer/winter, etc.)

Remainder component: what remains after extracting the previous components. It is completely irregular

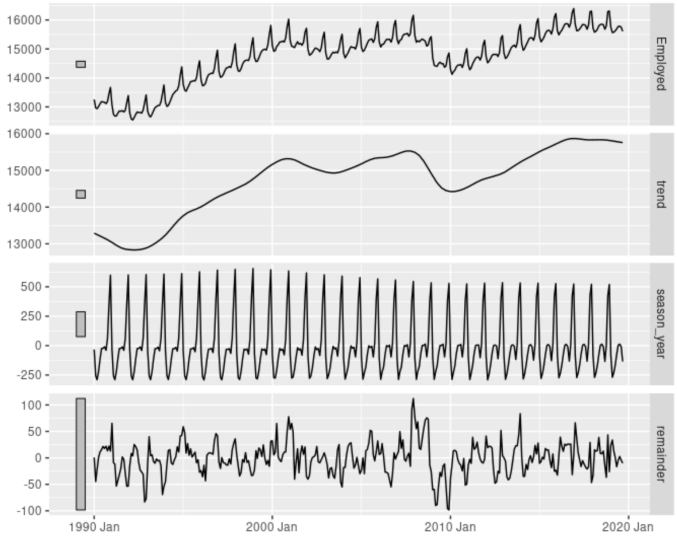

Figure 2.1: Time series components in [1]

In figure 2.1, it is possible to see the employment in the US retail sector. The bottom three panels present the three components separately. By combining these components, the data displayed in the top panel can be reconstructed. It is important to note that the seasonal component changes over time, resulting in similar patterns for consecutive years but potentially different seasonal patterns for years that are further apart.

The bottom panel illustrates the remainder component, which represents what remains after subtracting the seasonal and trend-cycle components from the data.

To provide a visual comparison of the component scales, grey bars are shown on the left side of each panel. Although each grey bar has the same length, their sizes vary due to the different

scales of the plots. In particular, the large grey bar in the bottom panel indicates that the variation in the remainder component is relatively smaller compared to the overall variation in the data.

If we were to resize the bottom three panels until their bars matched the size of the bar in the data panel, all panels would be presented on the same scale.

2.2 Demand Classification

Several research studies try to classify items based on demand criteria other than the usual ABC classification to guide which forecasting methodologies to utilize. The work of [2], which categorizes items into four classes (erratic, lumpy, intermittent, and smooth), is based on the average demand interval and the coefficient of demand variation and is the most well-known and widely used method.

Their research focuses on understanding the characteristics of different demand patterns and developing appropriate forecasting and inventory management strategies for each type. They proposed a framework for classifying demand patterns into four categories:

Intermittent demand: This type of demand occurs sporadically, with periods of no demand followed by sudden spikes. For example, some spare parts or repair services may exhibit intermittent demand patterns. The intermittent demand pattern is difficult to forecast because it lacks a clear pattern.

Lumpy demand: This type of demand occurs in large, irregular batches. For example, a retailer may receive a large order for a particular product, which may not be repeated for some time. The lumpy demand pattern creates challenges for supply chain management, as it may be difficult to predict when demand will occur and how much inventory to keep on hand.

Erratic demand: This type of demand varies significantly from one period to the next, with no clear pattern or trend. For example, fashion items or seasonal products may exhibit erratic demand patterns. The erratic demand pattern makes it challenging to plan production and inventory levels effectively.

Smooth demand: This type of demand is relatively consistent and predictable over time. For example, everyday consumer goods such as toothpaste or milk may exhibit smooth demand patterns. The smooth demand pattern makes it easier to plan and optimize production and inventory levels.

The theoretical rule presented earlier is formulated using two key parameters: the squared coefficient of variation (CV^2) and the average inter-demand interval, and these were used to define the four quadrants (Figure 2.2).

Syntetos and Boylan also developed different forecasting methods and inventory management strategies for each type of demand pattern. For example, for intermittent demand, they suggested using specialized forecasting techniques such as Croston's method, while for lumpy demand, they recommended using dynamic safety stock models. This classification helps choose the best model to use in forecasting the different types of SKU through an algorithm computed with this purpose and helps to increase the efficiency of the predictions [8].

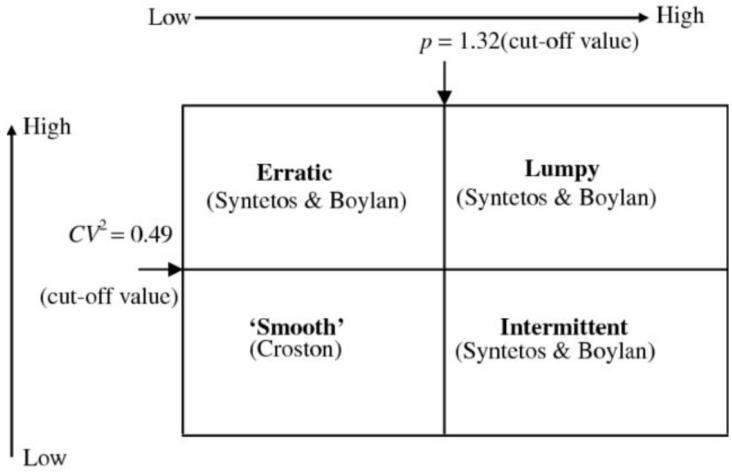

Figure 2.2: Demand classification and cutoff values in [2]

2.3 Forecasting

The study of forecasting methods in the literature still does not allow us to state which is the most efficient method for all types of SKU, but there are promising data regarding the performance of machine learning methods [9].

It is essential to clarify the differences between two types of methods: statistical methods are based on mathematical approaches. These studies may aim to infer something from the relationships between data or make predictions based on them. There are a lot of statistical methods that can make predictions. However, they have some problems modeling nonlinear data; Machine learning meters have varying degrees of interpretability, from very high levels to levels where none exists, the so-called black boxes, such as NN. In the latter case, interpretability is sacrificed for predictability, mainly due to the ability to deal with nonlinearity in the data [10]. Regarding the methods that will be presented, LSTM, XGBoost, Holt-winters and ARIMA will be presented, as they have demonstrated a lot of potential in performance (XGBoost) [11] and time series interpretability (LSTM) [4], and have been used as a common practice (Holt-Winters and ARIMA) in the previous investigation regarding this project.

2.3.1 Forecasting machine learning methods

2.3.1.1 LSTM

Long short-term memory (LSTM) networks are a type of recurrent neural network (RNN) that are able to capture long-term dependencies in data [12]. They are specifically designed to avoid the vanishing gradient problem, which is a common issue in traditional RNNs [13], having the ability to propagate useful gradient information from the output end of the model back to the layers near the input end of the model.

The architecture of an LSTM network consists of a series of "memory cells," which are small modules that can store information for an extended period of time. These memory cells are controlled by "gates," which decide what information should be stored, what information should be thrown away, and what information should be outputted [14]. The gates are implemented using logistic regression models and are trained using backpropagation [15].

In addition to the memory cells and gates, LSTM networks also have input and output layers, which are used to receive input data and produce output, respectively. The input layer is often followed by one or more hidden layers, which are used to transform the input data using a set of weights and biases that are learned during the training process [16]

One of the key features of LSTM networks is the ability to selectively remember or forget information, which allows them to model long-term dependencies in data [12] [14].

LSTM networks excel in capturing long-term dependencies; Compared with Conventional RNNs, which encounter challenges such as vanishing and exploding gradients when trained on lengthy sequences, LSTM networks address this issue by integrating a gating mechanism that selectively preserves or discards information. The utilization of LSTM enables the model to effectively grasp and retain essential context, even in cases where there exists a considerable temporal interval between significant events within the sequence [17] [12]

LSTM networks are more computationally demanding compared to simpler architectures like feed-forward neural networks. As a result, their scalability can be limited when dealing with large-scale datasets or constrained environments [17].

Training LSTM networks can take longer compared to simpler models due to their computational complexity. This means that training LSTMs often requires more data and extended training times to achieve high performance [17] [12].

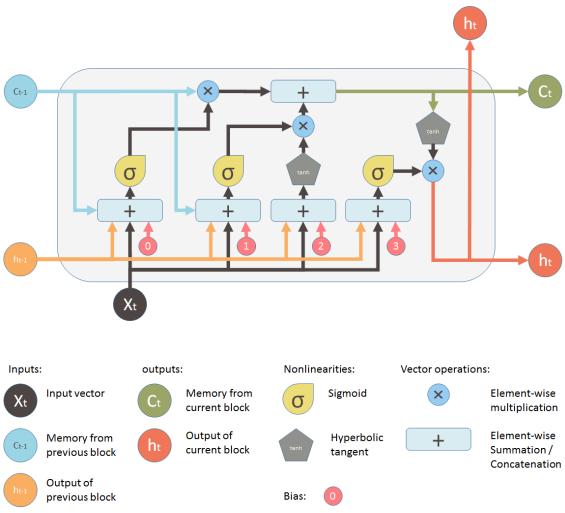

Figure 2.3: Architecture of a LSTM block

There are several variations of the LSTM architecture, including the "peephole LSTM," which allows the memory cells to access their own past states [18], and the "coupled LSTM," which

allows the cells to interact with each other in more complex ways [19].

Model explanation

As mentioned before, this method is composed by three types of gates: Input, Forget and Output gates [12]. LSTM gates employ sigmoid activation functions, meaning they produce a value that ranges between 0 and 1. Typically, this value tends to lean towards either 0 or 1 in the majority of instances.

$$sig(t) = \frac{1}{1+\varepsilon^{-t}} \tag{2.1}$$

It uses "*sigmoid function for gates because, we want a gate to give only positive values and should be able to give us a clear cut answer whether, we need to keep a particular feature or we need to discard that feature*" [12].

So, zero means that the gate is blocking everything and 1 means that the gate is allowing everything to go through.

Input gate equation

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \tag{2.2}$$

This one informs us about the forthcoming data that will be stored in the cell state.

Forget gate equation

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f) \tag{2.3}$$

The forget gate, as the second component, determines the information that ought to be discarded from the cell state.

Output gate equation

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o) \tag{2.4}$$

For the third element, we have the output gate, which delivers the activation signal to the ultimate output of the LSTM block at timestamp 't'.

i_t, f_t and o_t represent input, forget and output gates respectively; σ represents the sigmoid function; w_x represents the weight for the respective gate(x) neurons; h_{t-1} represents the output of the previous lstm block (at t-1); x_t is the input at the current timestamp; and finally b_x represents the biases for the respective gates(x).

Cell state, candidate cell state and the final output

$$\tilde{c_t} = \tanh\left(w_c\left[h_{t-1}, x_t\right] + b_c\right) \tag{2.5}$$

$$c_t = f_t c_{t-1} + i_t \tilde{c_t} \tag{2.6}$$

Where c_t is the cell state memory at timestamp t and $\tilde{c_t}$ represents the candidate or cell state at timestamp t.

To obtain the memory vector c_t for the present timestamp, the computation of the candidate is performed. By analyzing the aforementioned equation, it becomes evident that the cell state at any given timestamp possesses knowledge regarding what it should discard from the previous state (i.e., f_t * c_{t-1}) and what it should prioritize from the current timestamp (i.e., i_t * $\tilde{c_t}$).

Finally, the cell state undergoes a filtering process before being fed into an activation function, which determines the portion that will serve as the output of the current LSTM unit at timestamp 't'.

2.3.1.2 XGBoost

XGBoost, short for "Extreme Gradient Boosting," is a highly effective machine learning algorithm that has gained significant popularity due to its exceptional performance in various tasks such as classification, regression, forecasting [20] [11]. From a top-down perspective, XGBoost is a sub-class of supervised machine learning. And, as its name suggests, XGBoost is an advanced variant of boosting Machine, which is a sub-class of tree-based ensemble algorithm. The algorithm combines multiple weak models, typically decision trees, to create a strong predictive model.

XGBoost incorporates several key features that contribute to its success. It employs regularization techniques, including L1 and L2 regularization, to prevent overfitting and enhance generalization [20]. The algorithm also supports parallel processing, making it faster than many other boosting algorithms [11]. It can handle missing values in datasets, reducing the need for extensive data preprocessing [20]. It also offers early stopping, which prevents overfitting and improves training efficiency [20].

The advantages of XGBoost lie in its high performance, scalability, and robustness [20]. XGBoost is optimized for speed and scalability, allowing it to handle large-scale datasets and leverage parallel processing [20] [11]. The algorithm incorporates regularization and pruning techniques, enhancing its robustness against overfitting and improving generalization [20]. Its exceptional accuracy has made it a popular choice in machine learning competitions, including several Kaggle competitions [20].

Model Explanation

This method extends the idea of gradient boosting in the sense that it also uses the second derivative (Hessian: Curvature) of the objective function in addition to its first derivative (Gradient) to further optimize its learning process.

After obtained the regularized objective described in [20], and since "the tree ensemble model in includes functions as parameters and cannot be optimized using traditional optimization methods in Euclidean space", it was obtained the following objective function that is trained in an additive manner:

$$\mathscr{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t\left(\mathbf{x}_i\right)\right) + \Omega\left(f_t\right) \tag{2.7}$$

Using second order Taylor approximation as described in [20], it is obtained the following:

$$\mathscr{L}^{(t)} \simeq \sum_{i=1}^{n} \left[l\left(y_i, \hat{y}^{(t-1)}\right) + g_i f_t\left(\mathbf{x}_i\right) + \frac{1}{2} h_i f_t^2\left(\mathbf{x}_i\right) \right] + \Omega\left(f_t\right) \tag{2.8}$$

Second order approximation:

$$f(x) \approx f(a) + f'(a)(x-a) + \frac{1}{2} f''(a)(x-a)^2 \tag{2.9}$$

where

$$g_i = \partial_{\hat{y}(t-1)} l\left(y_i, \hat{y}^{(t-1)}\right) \tag{2.10}$$

$$h_i = \partial^2_{\hat{y}(t-1)} l\left(y_i, \hat{y}^{(t-1)}\right) \tag{2.11}$$

Finally, if we remove the constant parts, we have the following simplified objective to minimize at step t:

$$\tilde{\mathscr{L}}^{(t)} = \sum_{i=1}^{n} \left[g_i f_t\left(\mathbf{x}_i\right) + \frac{1}{2} h_i f_t^2\left(\mathbf{x}_i\right) \right] + \Omega\left(f_t\right) \tag{2.12}$$

The expression mentioned above represents the summation of multiple simple quadratic functions in a single variable. It can be effectively minimized through the utilization of established techniques. Consequently, our subsequent objective pertains to the identification of a learner capable of minimizing the loss function during iteration t.

At iteration t, the goal is to develop a learner that minimizes the loss function to the maximum extent possible. This objective raises certain questions, including the feasibility of identifying the optimal next learner and determining the loss reduction achieved by adding a specific learner. Fortunately, the authors propose a scoring function for assessing the quality of a tree structure q, which aligns with the aforementioned solution based on simple quadratic functions. The score function is the following:

$$\tilde{\mathscr{L}}^{(t)}(q) = -\frac{1}{2} \sum_{j=1}^{T} \frac{\left(\sum_{i \in I_j} g_i\right)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T \tag{2.13}$$

The introduction of XGBoost brought forth two notable innovations, namely the sparsity-aware algorithm and weighted quantile sketch [20]. The sparsity-aware algorithm incorporates a

built-in feature known as default direction, which captures the pattern of sparse data structures and determines the split direction at each node based on this pattern. There were identified three common causes for sparsity, including the presence of missing values, frequent zero entries in statistics, and artifacts resulting from feature engineering, such as one-hot encoding.

By virtue of its sparsity-aware algorithm, XGBoost becomes capable of handling missing data without requiring imputation by the user. While default direction addresses the split direction, the weighted quantile sketch contributes by proposing candidate split points. In [20], it is provided a concise summary of this approach, describing it as a novel distributed weighted quantile sketch algorithm capable of handling weighted data with provable theoretical guarantees. The underlying idea involves the utilization of a data structure that supports merge and prune operations, with each operation demonstrating a certain level of accuracy maintenance.

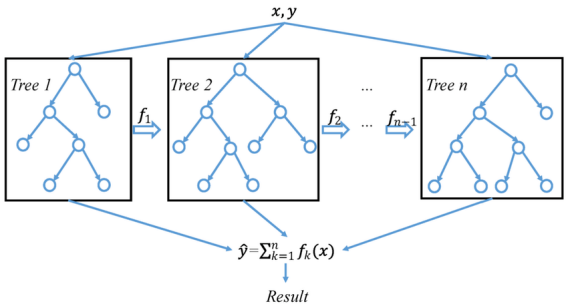

Figure 2.4: Architecture of XGBoost algorithm in [3]

2.3.2 Forecasting statistical methods

2.3.2.1 Holt-Winters

Holt-Winters method is a popular forecasting technique that incorporates trend, seasonality, and level components in time series data [21] [22]. It offers several advantages, but it also has some limitations.

The method uses historical data to estimate the current level, trend, and seasonality of the time series and then projects these components into the future.

Breaking down its three main components: the level component represents the average value of the time series over time. It is updated using a smoothing factor (alpha) that controls the weight of the current observation on the estimated level [23]. A smaller alpha value puts more weight on past observations, while a larger alpha value gives more importance to recent observations; the trend component captures the overall direction of the time series. It is updated using a smoothing factor (beta) that determines the weight of the current estimated trend [23]. Similar to the level component, a smaller beta value emphasizes past trend values, while a larger beta value focuses on recent trend values; the seasonality component represents the repetitive patterns or cycles in the time series. It is updated using a smoothing factor (gamma) that controls the weight of the current

seasonality estimate [23]. Like the level and trend components, a smaller gamma value prioritizes past seasonality values, while a larger gamma value emphasizes recent seasonality values.

Holt-Winters method includes two variations, depending on whether the seasonality is additive or multiplicative. The preference for the additive method arises when the seasonal variations remain relatively consistent throughout the time series, whereas the multiplicative method is favored when the seasonal variations are proportional to the level of the series. In the additive method, the seasonal component is represented in absolute terms, aligning with the scale of the observed series. In the level equation, the series is adjusted for seasonality by subtracting the estimated seasonal component [23]

To forecast future values using Holt-Winters, the method projects the level, trend, and seasonality components into the future based on their estimated values. The combination of these three components provides a more accurate forecast by capturing the underlying patterns in the time series [23]

Advantages of the Holt-Winters method include its ability to handle time series data with trend and seasonality, making it suitable for forecasting in scenarios where these patterns are present [23]. The method can capture and project both short-term and long-term trends, allowing for more accurate forecasts. It also takes into account the seasonal fluctuations, which can be useful in industries where demand varies cyclically.

However, the Holt-Winters method has some limitations and considerations. First, it assumes that the patterns observed in historical data will continue into the future, which may not always be the case if there are sudden changes or external factors affecting the time series. Additionally, the method requires a sufficient amount of historical data to estimate the model parameters accurately. If the time series is too short or has irregular patterns, the forecasts may be less reliable.

The Holt-Winters method is commonly used in various scenarios, including demand forecasting, sales forecasting, inventory management, and financial forecasting [24]. Its ability to capture both trend and seasonality makes it particularly suitable for industries with predictable cyclical patterns, such as retail, hospitality, and seasonal goods. Additionally, the method can be applied to various time series data, including hourly, daily, weekly, or monthly data.

Model explanation

Starting with the simple exponential smoothing model:

$$\hat{y}_{t+1} = \alpha y_t + (1-\alpha)\hat{y}_{t-1} \tag{2.14}$$

Where $\hat{y}_{t+1}$ is the value we are forecasting, y_t is the most recent observed values, $\hat{y}_{t-1}$ is our previous forecast and α is the smoothing factor ($0 <= \alpha <= 1$).

Overal equation:

$$\hat{y}_{t+h} = l_t \tag{2.15}$$

Level component:

$$l_t = \alpha y_t + (1-\alpha) l_{t-1} \tag{2.16}$$

Here h is the time step we are forecasting and $l_t = \hat{y}_{t+1}$ to explicitly state that this is the level component of the model.

Holt's Linear Trend Method

In [21] was introduced an extension to simple exponential smoothing, which enables the forecasting of time series data containing a trend component. This method involves the utilization of three main equations: a forecast equation, as well as two smoothing equations dedicated to estimating the level and trend components, respectively.

Overall equation:

$$\hat{y}_{t+h} = l_t + hb_t \tag{2.17}$$

Level equation:

$$l_t = \alpha(y_t) + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{2.18}$$

Trend equation:

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta) b_{t-1} \tag{2.19}$$

Here b_t is the forecasted trend, b_{t-1} is the previous forecasted trend and β is the trend smoothing factor (0 <= β <= 1).

Holt-Winters additive method

In [22] is extended the Holt's linear trend method [21] by adding seasonality to the forecast. The addition of seasonality gives rise to two different Holt Winters' model, additive and multiplicative. The difference between the two models is the size of the seasonality fluctuations. For an additive model the seasonality fluctuations are mostly constant. However, for multiplicative model the fluctuations are proportional to the value of the time series at that given time [23].

Overall equation:

$$\hat{y}_{t+h} = l_t + hb_t + s_{t+h-m} \tag{2.20}$$

Level equation:

$$l_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{2.21}$$

Trend equation:

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta) b_{t-1} \tag{2.22}$$

Seasonality equation:

$$s_t = \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m} \tag{2.23}$$

Where m is the seasonality of the time series, s_t is the seasonal forecast component, s_{t-m} is the forecast for the previous season and γ is the seasonal component smoothing factor (0 <= γ <= 1α).

Holt-Winters multiplicative method

Overall equation:

$$\hat{y}_{t+h} = l_t + hb_t + s_{t+h-m} \tag{2.24}$$

Level equation:

$$l_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{2.25}$$

Trend equation:

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \tag{2.26}$$

Seasonality equation:

$$s_t = \gamma\frac{y_t}{l_{t-1} + b_{t-1}} + (1-\gamma)s_{t-m} \tag{2.27}$$

2.3.2.2 ARIMA

The effectiveness of the ARIMA (p, d, q) mathematical model has been well-established in the literature. This model combines autoregressive (AR) and moving average (MA) components, with lag orders of p and q, respectively. The integrated (I) component plays a crucial role in achieving stationarity in the time series by transforming the raw observations. Specifically, it involves taking the differences between consecutive values, replacing the original data points with these differences. By applying this approach, ARIMA (p, d, q) effectively transforms non-stationary time series into stationary ones. The mathematical formulation of ARIMA (p, d, q) can be expressed through equations below, which outline the specific calculations involved in the model [23].

$$\varphi(L)(1-L)^d yt = \theta(L)\varepsilon t \tag{2.28}$$

$$\left(1 - \sum_{i=0}^{p} \varphi \mathrm{i} L^i\right)(1-L)^d \mathrm{y}_t = \left(1 + \sum_{j=1}^{q} \theta \mathrm{j} L^j\right)\varepsilon \mathrm{t} \tag{2.29}$$

$$\mathrm{Y}_t = \phi_0 + \phi_1 \mathrm{y}_{t-1} + \phi_2 \mathrm{y}_{t-2} + \cdots + \phi_p \mathrm{y}_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \cdots - \theta_p \varepsilon_{t-p} \tag{2.30}$$

εt is the error an time t, y_t is the actual value. The selection of the parameters p, d, and q in the ARIMA model involves positive integer values, representing the order of the autoregressive, integrated, and moving average components, respectively. In practice, a commonly employed method for determining suitable values for p and q is through the analysis of the autocorrelation function (ACF) and partial autocorrelation function (PACF) plots. By inspecting these plots, insights can be gained regarding the correlation structure of the data and the maximum order of the autoregressive component (p) that can effectively capture the non-stationarity of the time series. The PACF plot plays a particularly valuable role in this process, aiding in the decision-making regarding the appropriate order of the autoregressive component. θ and α are model parameters [23] [25].

ARIMA has been used for some applications like electricity load pattern [26], electricity consumption profile [25], and in both investigations it was surpassed, by the proposed method and ANN, respectively.

2.3.3 Intermittent demand forecasting methods

Another matter that needs to be taken into account is the fact that there are types of time series that are very difficult to model, as the intermittent and lumpy ones, and were developed specifically models to address this scenarios:

The Croston method is a forecasting technique used specifically for intermittent demand data [27], where there are long periods of zero demand, and sporadic periods of non-zero demand [28].

Croston optimized, works like Croston, but this model optimizes the simple exponential smoothing for both the non-zero demand size and the inter-demand intervals. The intermittent demand rate is calculated as the ratio of the number of non-zero demand periods to the total number of periods. This is used to estimate the probability that a non-zero demand period will occur in the next period. The forecast for the next period is then calculated as the product of the intermittent demand rate and the average of the non-zero demand periods. The average is calculated using the exponential smoothing method, where the previous forecast and the actual demand are used to update the forecast.

The Croston method has been shown to be effective in forecasting intermittent demand data, especially when there are a large number of zero demand periods. Croston's method is robustly superior to exponential smoothing and could provide tangible benefits to manufacturers forecasting intermittent demand [28] However, it has some limitations, such as its assumption of independence between demand periods and its sensitivity to outliers. Therefore, it is important to carefully consider the characteristics of the data and the specific context in which it is being used before applying the Croston method. [29] [28].

Model explanation

The essence of this method can be distilled into a concise three-step process [27]: Assessing the mean level of demand during instances of demand occurrence. Determining the mean duration between two consecutive demand occurrences. Predicting the demand by multiplying the demand level (during occurrence) with the probability of an occurrence. The level estimate is denoted

as "a" (similar to exponential models), and the observed demand is denoted as "d." The level estimate "a" is updated only when an actual observation is available. Similar to various exponential smoothing models, a learning parameter alpha ($0 < \alpha < 1$) is employed to determine the relative significance assigned to recent observations compared to historical ones.

If $d_t > 0$, then:

$$a_{t+1} = \alpha d_t + (1-\alpha)a_t \tag{2.31}$$

If $d_t = 0$, then:

$$a_{t+1} = a_t \tag{2.32}$$

The estimation of the time between two demand occurrences is denoted as "p" (for periodicity), while the time elapsed since the previous demand occurrence is denoted as "q." The update for "p" occurs exclusively when a demand occurrence is observed. It is worth noting that we utilize the same learning parameter, "alpha," for both the estimation of the demand level and the estimation of the time between demand occurrences.

If $d_t > 0$, then:

$$p_{t+1} = \alpha q + (1-\alpha)p_t \tag{2.33}$$

If $d_t = 0$, then:

$$p_{t+1} = p_t \tag{2.34}$$

The forecast can be determined straightforwardly by dividing the demand level (denoted as "a") by the periodicity (denoted as "p").

Algorithm overview:

If $d_t > 0$:

$$\begin{cases} a_{t+1} = \alpha d_t + (1-\alpha)a_t \\ p_{t+1} = \alpha q + (1-\alpha)p_t \\ f_{t+1} = \frac{a_t}{p_t} \end{cases} \tag{2.35}$$

If $d_t = 0$:

$$\begin{cases} a_{t+1} = a_t \\ p_{t+1} = p_t \\ f_{t+1} = f_t \end{cases} \tag{2.36}$$

The TSB (Teunter-Syntetos-Babai) forecasting method is an approach designed for intermittent demand forecasting and it was presented in [30]. It represents a forecasting approach addressing both obsolescence and inventory decisions. Unlike the Croston method, which updates the demand interval, this method updates the demand probability and the risk of obsolescence.

Although this distinction may seem minor, as the demand interval is the reciprocal of the underlying probability of demand occurrence, updating the demand probability offers greater flexibility. While the demand interval can only be updated following a positive demand, the demand probability can be updated in each period. Additionally, it is demonstrated that utilizing the demand probability instead of the demand interval eliminates forecasting bias when considering any arbitrary point in time [30] [31].

Model explanation

The estimation of the level remains unchanged compared to the regular Croston model. However, a significant modification is introduced in the determination of periodicity. The new approach involves expressing the periodicity "p" as the probability of a demand occurrence. Consequently, "p" represents a frequency or probability ranging from 0 (indicating no demand occurrence) to 1 (indicating demand occurs at every period). Notably, the periodicity will now be updated at each period, even in the absence of a demand occurrence. Under this revised framework, the periodicity will exhibit the following characteristics: It will decrease when there is no demand occurrence, reflecting a reduction in the probability of demand happening. This decrease follows an exponential pattern, similar to other exponential smoothing models. Conversely, it will increase when a demand occurrence is observed, reflecting an augmentation in the probability of demand occurring.

If $d_t > 0$, then:

$$p_{t+1} = \beta + (1-\beta)p_t \tag{2.37}$$

If $d_t = 0$, then:

$$p_{t+1} = (1-\beta)p_t \tag{2.38}$$

If $d_t > 0$:

$$\begin{cases} a_{t+1} = \alpha d_t + (1-\alpha)a_t \\ p_{t+1} = \beta + (1-\beta)p_t \\ f_{t+1} = a_{t+1}p_{t+1} \end{cases} \tag{2.39}$$

If d_t:

$$\begin{cases} a_{t+1} = a_t \\ p_{t+1} = (1-\beta)p_t \\ f_{t+1} = a_{t+1}p_{t+1} \end{cases} \tag{2.40}$$

In [32], it was proposed an alternative approach to handle intermittent demand, known as the aggregate–disaggregate intermittent demand approach (ADIDA). This approach utilizes equally sized time buckets for non-overlapping temporal aggregation, aiming to address the issue of demand intermittence. By appropriately aggregating the data and removing intermittence, ADIDA

focuses solely on forecasting the non-zero demands in an aggregated form. The effectiveness of the ADIDA framework can be attributed to the reduction or elimination of variance observed in the intervals. However, it is important to note that if the framework does not consider a mechanism for disaggregation in the final stage, ADIDA primarily answers the question of "how many SKUs will be sold over a predetermined lead time?" [32] [33]

The algorithm has shown significant improvements in accuracy and outperformed other popular methods such as Croston's method and simple exponential smoothing [32] [34] [33].

iMAPA (Intermittent Multiple Aggregation Prediction Algorithm) is another way for implementing temporal aggregation in demand forecasting. However, in contrast to ADIDA that considers a single aggregation level, iMAPA considers multiple ones, aiming at capturing different dynamics of the data. Thus, iMAPA proceeds by averaging the derived point forecasts, generated using SES [35]. IMAPA has been applied in various studies within the field of demand forecasting, particularly for intermittent demand [29] [36].

Temporal aggregation has certain drawbacks associated with it. These drawbacks include the problem of excessive data smoothing, and the loss of information due to a reduced number of observations. This loss of information is especially significant when dealing with intermittent demand data, which often consists of short series [37] [38].

2.3.4 Hybrid Forecasts

This type of prediction is based on using the outputs of two or more other prediction methods so that, through an algorithm such as weighted average, a more precise result is specified. As seen previously, the effectiveness of the forecast for each SKU depends on the method used. It is possible to obtain better results for a particular type of SKU with specific techniques. Thus, using a combination of ways, it is possible to improve accuracy, sharing knowledge between the algorithms [10].

2.3.5 Uncertainty associated with patterns in the calendar

This section presents the most feasible approaches found in the literature to solve the weekend demand uncertainty problem.

In [6], the approach was to divide the dataset into working days and weekends for short-term load forecasting in the field of energy and electricity system development. The method developed is based on a hybrid algorithm, composed of the following steps: Noise elimination procedure (ICEEMDAN) [39]; Hidden feature mining through FTS approach; Artificial neural network optimized by a multi-objective optimization algorithm (MODA).

The work developed in [40] aimed to improve the bicycle rental forecasts, and also took the initiative to split the dataset into two parts, working days and weekends, because working days are much more numerous compared to weekends and this discrepancy would affect the forecasts. The

new model developed has as a first approach the application of a fuzzy c-means based unsupervised genetic algorithm to pre-classify the data into groups, which then serve as input to a BPN (back propagation neural network). This last element calculates the demand for bicycle rental after being trained as all historical data.

The performance of the developed architecture was evaluated using the RMSE and MAE metrics, and the results show that using FCM-based GA before BPN improves the predictions. It was also proven through the tests that the predictions were highly related to the weather and the different lifestyles that people adopt on weekends and on workdays.

In a future use of this type of architecture a possible improvement could be to increase the number of layers of the neural network that is used in the calculation of the predictions [40].

There were two parts to the experimentation of the method: the first using point forecasts for working days and non-working days, and the second using interval forecasts for the same type of data. The results presented by the new methodology are auspicious, surpassing the results presented by more conventional methods, such as ARIMA. However, the continuous updating of the parameters involves many optimization algorithms, resulting in a high computational cost, being possible to use historical data for updating this parameter based on the day of the week (working days or non-working days).

In order to assess the relevance of each of the above elements, the reduced relative error (RRE) of the MAPE metric was calculated, and the findings were that the data cleaning (ICEEMDAN) and data mining (FTS) algorithms largely contributed to better results.

It is also possible to solve the problem of weekend demand uncertainty through multi-step-ahead forecasting, which allows forecasting demand over seven days.

This type of forecasting approach can be divided into the following strategies [41]:

Recursive Strategy: iterates a one-step-ahead forecasting model H times to get the H forecasts. Following the future series value estimation, it is used as an input for the forecast that follows. [41].

Direct Strategy: calculates a set of H forecasting methods, each of which returns a prediction for the ith value (i ε 1, . . . , H).

DirRec: incorporates elements from both the Direct and Recursive techniques. In other words, each stage employs a distinct model, but estimates from previous steps are incorporated into the input set. [41].

Multi-Input Multi-Output: "returns a vector of future values in a single step" [41] in order to maintain the stochastic dependence, which characterizes time series, between the predicted values.

DIRMO: points to protect the maximum engaging angles of both the DIRect and Mimo techniques. This methodology points to discovering a trade-off between the property of protecting the stochastic dependency between the forecasted values and the adaptability of the modeling method [41].

Each of the strategies mentioned above requires the incorporation of a predictive or learning model to calculate the function representing the temporal stochastic dependencies [41]. In this case, lazy learning algorithms were used, namely: single output lazy learning algorithm with Leave-One-Out (LOO) error associated, to estimate the generalization ability of the algorithm; Multiple output lazy learning algorithm with associated LOO error or an auto-correlation measure (MIMO-ACFLIN), the latter being an alternative to LOO, evaluating the stochastic discrepancy between the predicted values the time series used to train the algorithm.

Since the accuracy of each prediction method is dependent on multiple factors, each time series was subjected to the same pré-processing, following the next steps [41]:

Gaps Removal

Deseasonalization (Yes or No)

Embedding Dimension Selection

Input selection (Yes or No)

Model Selection (WINNER, COMB or WCOMB) [41]

SMAPE error, Friedman's test, and posthoc test evaluated the methods.

The results showed that the best method is MIMO-ACFLIN, with input selection, de-seasonalization, and equal weight combination (COMB). The best strategies are MIMO and DIRMO (Multi-Output). Deseasonalization is a critical factor that improves the forecasts, and it is an influence to consider for future work. The problem of multi-step ahead forecasting is still significant because of the uncertainty created as the required forecast time window advances [41].

The case of multi-step forward prediction can be addressed by applying LSTM neural networks [4] [42], which are a type of recurrent neural network. This type of methodology is naturally valuable for studying time series, as it can capture the temporal relationships between the various points analyzed and can store information as it runs through the data [4].

In [4], the case of demand forecasting in retail (e-grocery), the preprocessing of the dataset was based on two steps:

To each record was added the value relative to the sales of the previous record;

Time-series normalization was performed because when training with many time series, the range of values between them is likely to differ.

Regarding deseasonalization, in the case of LSTMs, this preprocessing step is not necessary due to their ability to learn oscillation behavior (e.g., cycles or seasonality) [43].

The architecture of the LSTM used is based on Figure 2.5. This type of structure brings a high probability of overfitting in insufficient data. This paper used two regularization methods, early stopping and random dropout, during the algorithm's training phase to prevent this type of event.

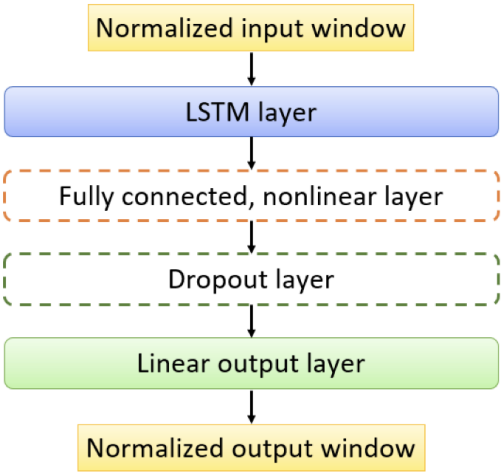

Figure 2.5: LSTM Architecture in [4]

For evaluation purposes, two metrics were used in work addressed before MAE and mMAPE. The results show that, for univariate time series, if we count only the mMAPE, this LSTM architecture outperformed all the benchmark methods, such as ETS, LR, RF. However, if the metric used is the MAE, this methodology obtained the second-worst result. It can be a consequence of the immense size of the group of products. However, using multivariate time series with the previous demand as a feature, the LSTM architecture surpassed all the benchmark methods. Then, since beverages have a higher price elasticity and the demand forecast performance for food and beverage is different, the tests were conducted individually for these products. The combination of previous demand, known orders, day of the week, whether tomorrow/the day after tomorrow the store is open and whether tomorrow/the day after tomorrow is a public holiday, as features were shown to have the best results for both product groups. The last four contribute the most to the predictive efficiency of the approach [4].

This approach has some points that require attention, of which the accuracy of the predictions decreases if small-time series are used, and the high computational cost that may be associated with the training of this architecture.

The research conducted in [5] have developed a new strategy to study hierarchical time series forecasting on tourism demand, and the framework is presented in Figure 2.6.

Tier 1 extracts time patterns and compares them to existing data. The algorithm considers regular and floating patterns (e.g., public holidays), and the records with similar temporal patterns are retrieved and sent into the next layer. Tier 2 establishes numerous periods to identify the temporal pattern of the current date. The two nearest neighbors for each temporal frame are generated using the K-NN algorithm, which detects patterns using the data obtained in the preceding layer. Tier 3 combines each element of the two nearest neighbors linearly to produce just one predicted value. Following that, all of the values of the other time frames are combined to give the final prediction

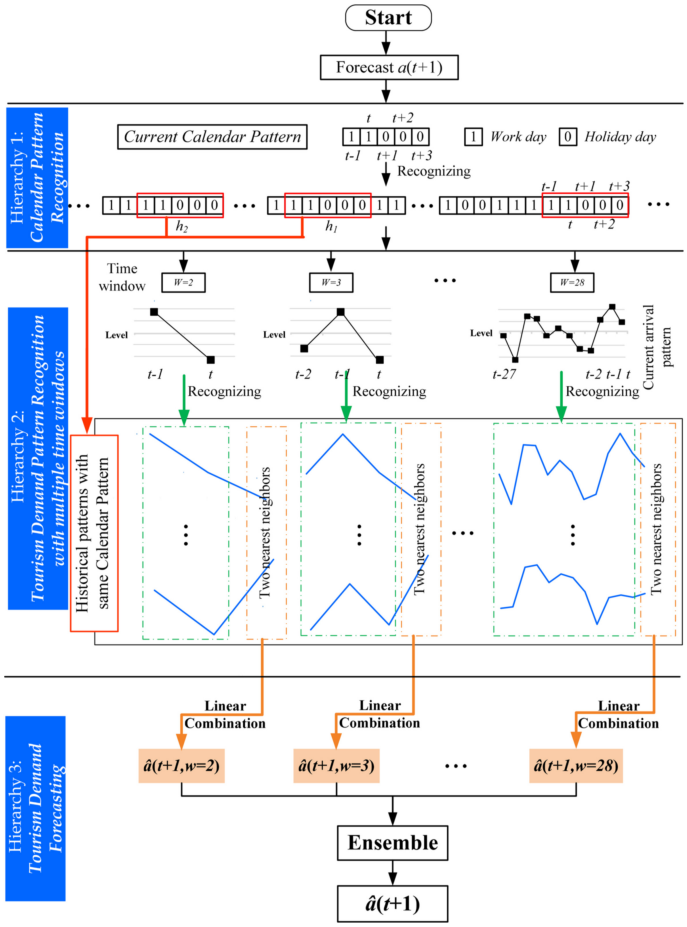

Figure 2.6: Hierarchical Diagram in [5]

value.

The approach gives reasonable indications for solving the demand uncertainty at weekends since regular calendar patterns, such as the weekend, are identified so that the forecasting process can be carried out most accurately.

The method used to retrieve the calendar patterns is based on a pre-processing technique, where workdays are denoted as a '1' and the holidays (weekends and floating holidays)are denoted as a '0'.

Since the calendar pattern on the current date is always compared with historical data, identification of floating vacations is also possible. In this way, the premise of seasonalization, involving events occurring at equal intervals, can be circumvented. It is possible to analyze the overlapping effects of multiple levels of seasonality using multiple time windows. To prove the relevance of the newly developed architecture, it was compared with reference methods according to the MAPE and MASE metrics. The reference methods include the seasonal naïve model, SARIMA model, ETS, exponential space smoothing state model with Box-Cox transformation, ARMA errors, trend, and seasonal components (TBATS). The other two are a time series model with intervention (SARIMA with explanatory variable, SARIMAX) and a pattern recognition model (simple

k-NN).

The tests were performed for three different time series, and the results show that the developed architecture outperforms all reference methods from one-day-ahead to 14-day-ahead forecasting in all the tests. For the first case, for example, according to the MAPE metric, the average results varied between 0.3265 and 0.4837 for the reference methods, while the average results, according to the same metric, for the newly developed methodology was 0.285. According to the MASE metric, the results varied between 1.2588 and 2.8689 for the reference methods and averaged 1.0093 for the new methodology. In the second and third cases, the new methodology performs better than the reference methods up to a forecast of at least nine steps ahead.

2.3.6 Hierarchical Forecasting

Predicting the search on groups of drugs with common characteristics (e.g., SKU active ingredient) improves inventory management capabilities and allows finding correlations and dependencies among the various products [44]. As mentioned earlier, LSTMs can predict demand for many items in parallel, which allows one to understand and model complementary and substitute effects between different products [4]. It is a good starting point to this section's approaches.

In [44], a new methodology is proposed that captures the interdependencies among products and nonlinearities, as opposed to benchmark methods such as ARIMA. The idea is based on an aggregation algorithm to segment items with the help of a product structure tree, like in Figure 2.8.

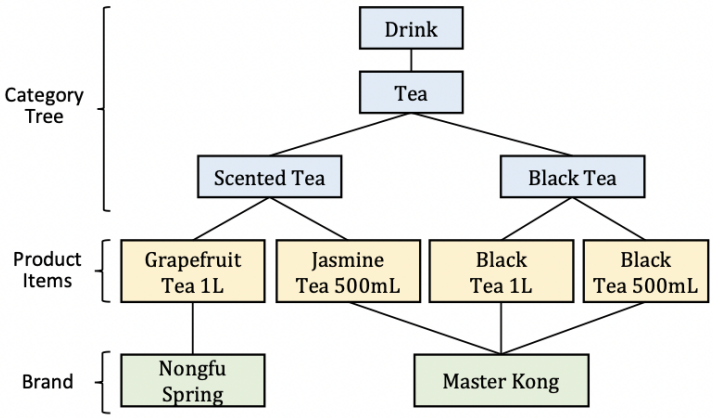

Figure 2.7: Example of a product tree in [6]

This approach uses a graph attention network to capture the structural interdependencies among the various time series and a gated recurrent unit to capture the temporal patterns. After these two steps, a Variable-Wise Temporal Attention component is incorporated to capture temporal dynamic patterns since the previous components do not deal with this characteristic. An auto-regressive component is also incorporated throughout this process to capture the local demand trend for each product. The architecture is shown in Figure 2.8.

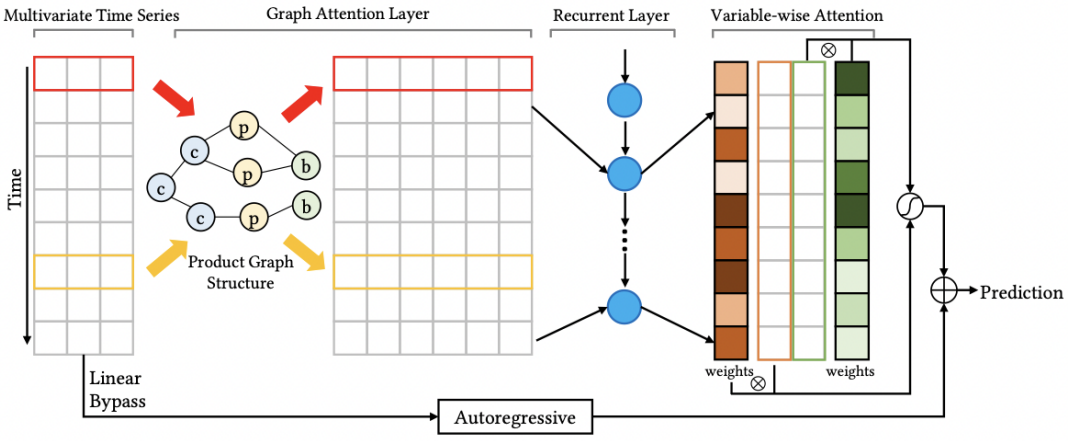

Figure 2.8: STANet Architecture in [6]

The main advantages of this architecture are that it can handle time series with different periodicities without making assumptions about them, capture non-linear relationships and interdependencies among time series, and adapt the processing depending on the forecasting output without making inferences about the intrinsic relationships.

The metric used for evaluating the methodology is the RSE, and the results show that STENet outperforms the methods used for comparing, and they show good prospects for future use.

The research made in [45] is based on a three-main-step approach: data visualization, a graph-based approach, and application of the prediction model.

The graph-based approach is carried out by calculating the cross-correlation matrix and the adjacency matrix related to the medicines (each medicine is an element of the graph) and the generation of the click sets, which are groups of completely interconnected vertices of the graph. Then, forecasting was conducted through two methods: ARIMA and a hybrid neural network approach, the last being composed of a linear ANN to take care of the linearities and a non-linear ANN to deal with the nonlinearities. The results were evaluated with MSE and MAE metrics, and it could be proven that the outcome was better using the forecast with the records of the medicine and the ones related to it, using the research of the paper.

In a hierarchical forecast, products are organized in a tree diagram, where at the top is the maximum of product aggregation, with the level of aggregation decreasing as one moves down the diagram, as in Figure 2.9. In [1], three hierarchical forecasting strategies are presented:

> **bottom-up approach**: forecasts are made for the elements of the lowest levels of the diagram and are then consecutively summed to obtain the predictions for the higher, more aggregated levels.

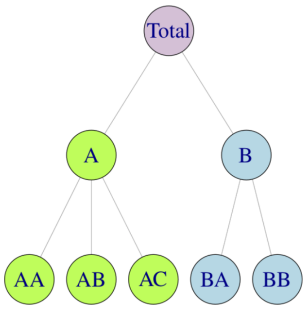

Figure 2.9: Hierarchical Diagram in [1]

top-down approach: forecasts are first made for the top element of the diagram, which represents the total of the time series, and this forecast is then divided by disaggregation factors to calculate the value for the lower elements of the hierarchy (diagram), and these can be calculated using different techniques.

middle-out approach: forecasts are obtained by combining the two methods mentioned above. An intermediate level is chosen, and forecasts are made for time series at this level. Then, for series at higher levels, a bottom-up approach is used to forecast them, and for the levels below the chosen intermediate level, a top-down approach is used to calculate the time series forecast.

$$\begin{bmatrix} y_t \\ y_{\mathrm{A},t} \\ y_{\mathrm{B},t} \\ y_{\mathrm{AA},t} \\ y_{\mathrm{AB},t} \\ y_{\mathrm{AC},t} \\ y_{\mathrm{BA},t} \\ y_{\mathrm{BB},t} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_{\mathrm{AA},t} \\ y_{\mathrm{AB},t} \\ y_{\mathrm{AC},t} \\ y_{\mathrm{BA},t} \\ y_{\mathrm{BB},t} \end{bmatrix}$$

$$y_t = Sb_t \tag{2.41}$$

The matrix S is denoted as the "summing matrix" and represents the aggregation structure, shown in fig.2.9, and b_t represents the time-series matrix of the lower levels.

In [1] are present some approaches on how to deal with top-down methodology and how to calculate the aggregate proportions:

Average historical proportions

This approach calculates the mean historical proportions of the time series $y_{j,t}$ relative to the aggregate time-series y_t in periods t=1, ..., T.

$$p_j = \frac{1}{T}\sum_{t=1}^{T}\frac{y_{j,t}}{y_t} \tag{2.42}$$

Proportions of the historical averages

In this scenario, the proportion p_j is calculated by the mean historical proportions of the time series $y_{j,t}$ relative to the average of the aggregate time-series y_t in periods t=1, ..., T.

$$p_j = \frac{\sum_{t=1}^{T}\frac{y_{j,t}}{T}}{\sum_{t=1}^{T}\frac{y_t}{T}} \tag{2.43}$$

Forecast proportions

Because the preceding techniques only evaluate historical proportions and ignore the reality that they might alter over time, they are more likely to provide less reliable forecasts at the lower-level time series. The projected proportions technique aims to determine the proportions predicted by the hierarchy using predictions from the all-time series. The following is the formula:

$$p_j = \prod_{\ell=0}^{K-1}\frac{\hat{y}_{j,h}^{(\ell)}}{\hat{S}_{j,h}^{(\ell+1)}} \tag{2.44}$$

In the formula, $\hat{y}_{j,h}^{(\ell)}$ corresponds to the initial h-step-ahead forecast of the series below the node that is ℓ levels above node j and $\hat{S}_{j,h}^{(\ell+1)}$ is the sum of the h-step-ahead initial forecasts directly below node j.

Optimal combination approach

Previous techniques had the issue that the total of the predicted time series lacks the consistency of aggregation since it does not sum up according to the hierarchy. This method, described in [1], is based on calculating the base forecasts for all-time series in the hierarchy while ensuring that the sums of the higher-level series match the sums of the lower-level series, then combining all the revised forecasts to create a group of revised forecasts that are as close to independent forecasts as possible, in the following way:

$$\tilde{y}_h = S\left(S'W_h^{-1}S\right)^{-1}S'W_h^{-1}\hat{y}_h \tag{2.45}$$

Where W is the forecast error variance of the h-step-ahead base forecasts. In [1], the methods above were applied to the prediction of the prison population in Australia, with the ETS model being the primary forecasting method. The results show that the best results were obtained through the optimal combination method, evaluated through the MASE metric and the CRPS skill score because the approach can consider information from all the levels of the series diagram.

The novel top-down approach technique proposed in [46] is based on estimating the ratio between the lowest level and highest level series in an interval h steps ahead to acquire the disaggregation proportions. The following formula is used to calculate these disaggregation ratios:

$$\hat{p}_{j,h} = \frac{\widehat{y_{J,h}}}{y_{t,h}} \tag{2.46}$$

In this type of approach, the time series is observed as a function in time rather than as a single mean value, as in the case of average historical proportions and proportions of historical averages. The relationship between low-level and high-level series is preserved since the averaging operation can be misleading when the distribution of the range of values relative to the mean value does not follow a normal distribution. The simulation was conducted with ARIMA as the main forecasting algorithm, and the results obtained in this paper show that the impact of the newly developed method was not significant, being outperformed by the bottom-up approach and the optimal approach. However, the results are very close to the best-performing methods, which can help in possible future use.

The research conducted on [47] led to a novel approach that is based on the Support Vector Regression methodology, which allows nonlinear modeling features from the data with a low risk of over-fitting, applied to hierarchical time series organized as in Figure 2.9, for example.

To implement this procedure, three different approaches were used in predicting tourists visits:

Bottom-Up SVR: first, an SVR algorithm is applied to each node of the lowest layer, calculating the respective forecasts. Then the predictions of the upper layer nodes are computed by aggregating the predictions of the lower layer nodes that are connected to each upper layer node, according to the formula $F_n = \sum_{i \in S_n} F_i$.

Top-Down SVR: an SVR algorithm is applied to the topmost layer node to calculate the prediction relative to the sum of all-time series. Then, this forecast is disaggregated according to the average proportions method, previously discussed and described in [1], to obtain the estimates for the lower layer nodes.

Middle-Out SVR: the initial step is to choose a layer of nodes j between the root node and the lowest layer in the hierarchy. Then an SVR algorithm is applied to all nodes in this layer. A bottom-up approach is followed for all nodes in layers above the one chosen, followed by a top-down approach to computing predictions for all nodes in layers below the one initially chosen.

To study the effectiveness and performance of this new approach, we used the three hierarchical techniques applying the SVR, ARIMA, and Holt-Winters algorithms to calculate the forecasts. To evaluate the results, the MAPE metric was used for each node and the average of the MAPE of the nodes for the evaluation of each layer. The analysis indicates that the SVR methods used

consistently outperform the statistical techniques and can be considered suitable alternatives since they achieve a good trade-off between model fit and regularization, reducing the risk of overfitting. It is possible to obtain excellent and flexible regressions through the kernel functions.

2.4 Error metrics

The use of metrics to assess the results of forecasts is of great importance, so it is crucial to understand the meaning behind some of the most used metrics.

RMSE: Root Mean Squared Error is the square root of Mean Squared Error (MSE). The meaning behind this metric is the difference between predicted results and actual results.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{1}{n} (Yactual - Ypredicted)^2} \tag{2.47}$$

MAPE: Mean Absolute Percentage Error represents how reliable a prediction is. This one is better for comparisons because the obtained value is put in relation to the actual sales/demand [48].

$$MAPE = \frac{1}{n} \sum_{i=1}^{n} \frac{|Yactual - Ypredicted|}{Yactual} \tag{2.48}$$

The MAPE have the disadvantage that they put a heavier penalty on positive errors than on negative errors [23].

MAD: Mean Absolute Deviation represents how big is the error, in average, along all predictions. The fact that it is an average value, in units, it isn't very good for comparisons [48].

$$MAD = \frac{1}{n} \sum_{i=1}^{n} |Yactual - Ypredicted| \tag{2.49}$$

SPEC: Stands for Stock-keeping-oriented Prediction Error Costs, explained in [49], and can be calculated with the formula:

$$\begin{aligned} \text{SPEC}_{\alpha_1,\alpha_2} = \frac{1}{n} \sum_{t=1}^{n} \sum_{i=1}^{t} \Bigg(& \max \left[0; \min \left[y_i; \sum_{k=1}^{i} y_k - \sum_{j=1}^{t} f_j \right] \right. \\ & \left. \cdot \alpha_1; \min \left[f_i; \sum_{k=1}^{i} f_k - \sum_{j=1}^{t} y_j \right] \cdot \alpha_2 \right] \cdot (t - i + 1) \Bigg) \end{aligned} \tag{2.50}$$

In the above formula n represents the length of the time series, y_t represents the actual demand verified at time t and f_t represents the calculated forecasts. The opportunity cost is represented by α_1 and the stock-keeping cost by α_2. It is recommended that their sum is 1, namely that their ratio is α_1 = 1-α_2.

MAAPE: Mean Arctangent Absolute Percentage Error is a variation of the Mean Absolute Percentage Error (MAPE) that uses the arctangent function to avoid the undefined values that can occur when the actual value is zero. MAAPE is calculated by taking the average of the arctangent of the absolute percentage difference between the predicted and actual values [50]:

$$MAAPE = \frac{1}{N}\sum_{t=1}^{N} \text{AAPE}_\text{t} = \frac{1}{N}\sum_{t=1}^{N} \arctan\left(\left|\frac{y_t - f_t}{y_t}\right|\right) \tag{2.51}$$

One advantage of MAAPE is that it provides a measure of the prediction error that is bounded between zero and infinity, unlike MAPE, which can produce large values when the actual value is close to zero. This makes MAAPE more robust than MAPE for applications where the actual values can be small. Another advantage of MAAPE is that it is less sensitive to outliers than other error metrics such as MSE or RMSE. This is because the arctangent function compresses large errors, reducing their impact on the overall error metric. However, MAAPE also has some disadvantages. One disadvantage is that it can be difficult to interpret, as it is not in the same units as the predicted or actual values. This makes it less suitable for applications where interpretability is important. Another disadvantage of MAAPE is that it can be more computationally expensive to calculate than other error metrics, as it involves calculating the arctangent function for each observation. This can be a concern for large datasets or real-time applications where speed is important [51] [50].

Chapter 3

Methods

3.1 Introduction

In the context of this project, the ultimate goals were to increase the forecasting effectiveness of the products sold over the weekend; to optimize the forecasting of products with some similar characteristics, like the active ingredient; to increase the forecasting performance in terms of time.

3.2 Pre-processing architecture

To accomplish the aforementioned objectives, a technique was developed, which involves aggregating SKUs sharing the same active ingredient. Consequently, clusters are formed, containing data from SKUs exhibiting similar characteristics (see Figure 3.1). This approach serves the purpose of enhancing the accuracy and efficiency of both individual and cluster-based forecasting, while also improving the time performance of value predictions for each pharmacy.

By aggregating time series data and conducting forecasts solely for the aggregated group, considerable time savings can be achieved. Rather than fitting and training algorithms for the complete set of time series data for all SKUs, focusing solely on the final aggregated time series enables more streamlined processes.

Each cluster within the dataset contains information pertaining to medications commonly utilized for similar purposes. For instance, a cluster may encompass medications such as *Benuron* and its corresponding generic alternatives.

At the end, we ended up with data frames composed accordingly the example Figure 3.1.

The aggregation process involved following a series of predefined steps based on established principles. It was imperative to avoid merging similar SKUs with varying lengths of data. Any missing values within the dataset were uniformly assigned a value of zero, as per the company's assurance that all units sold were consistently registered. Consequently, if null values were present, they were treated as zeros starting from the designated initial date.

To ensure the utmost accuracy of data for training and feature learning purposes, the decision was made to solely consider the most recent two years of available SKUs' dataframes. Over

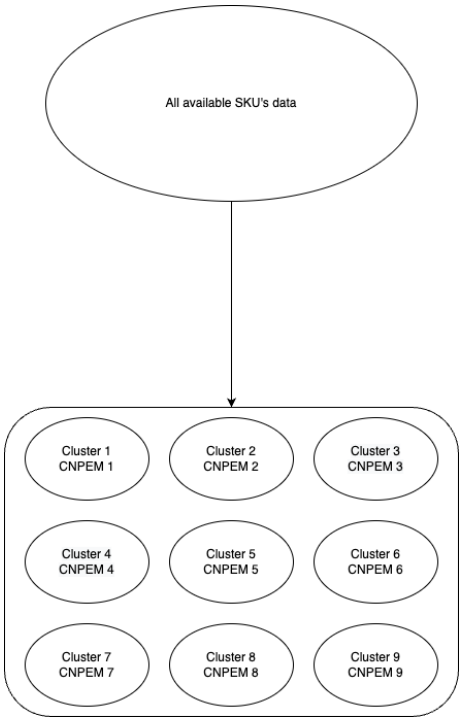

Figure 3.1: Architecture of the aggregation algorithm

time, the sales trends of these medications have undergone changes, and older data can exert a substantial and unfavorable influence on the results.

In order to have more ability do forecast future weekend values, those time series were splited in week and weekend days, so it was possible to deal only with the weekend values, which is the main goal of the investigation (Figure 3.2 and Figure 3.3).

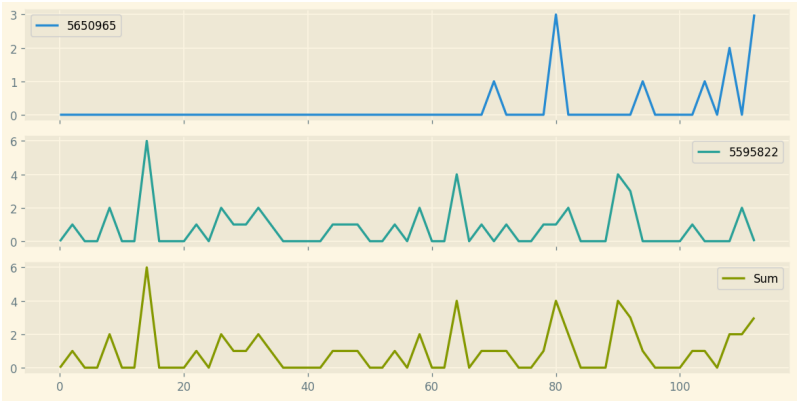

Figure 3.2: Weekends sales history example

After this, a module was developed that would calculate a new time series from the sum of all the time series of the previous clusters. This resultant time serie is the base of our forecasting

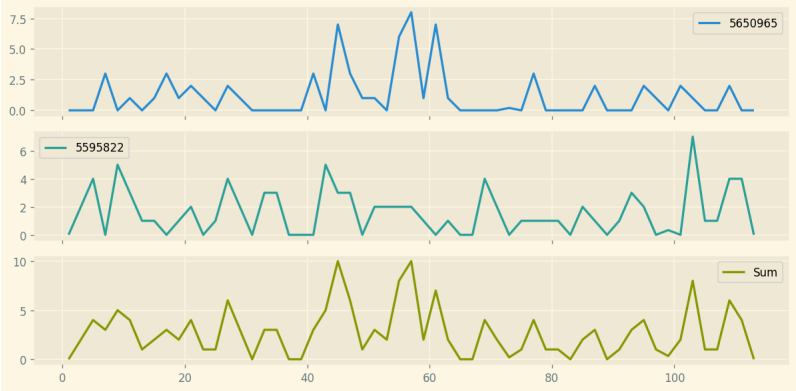

Figure 3.3: Weekdays sales history example

algorithm and it is from these values that the prediction methods will learn. It is possible to see it in the Figure 3.4.

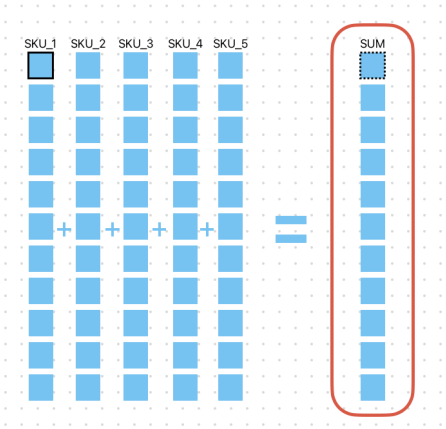

Figure 3.4: Final time serie calculation

It was also implemented the normalization of the values between an interval. The interval chosen for this scope was [-1;1], as it normally helps algorithms, like long-short term memory networks, to work faster and effectively [52].

Once all these pre-processing processes have been implemented, the data are ready to serve as input to the prediction algorithms developed.

Another parallel step of the pre-processing stage was to calculate a 'new' time serie of the predicted "market percentages" for each of the SKUs. This has the aim of taking into account the importance of each product.

To calculate the percentages, the number of units sold for each product was divided by the total amount of units sold per cluster, per unit of time (day, week, or weekend) Figure 3.5.

The next step of this process was to calculate the average of the percentages for each SKU time series, based on the algorithm "Average historical proportions" [46].

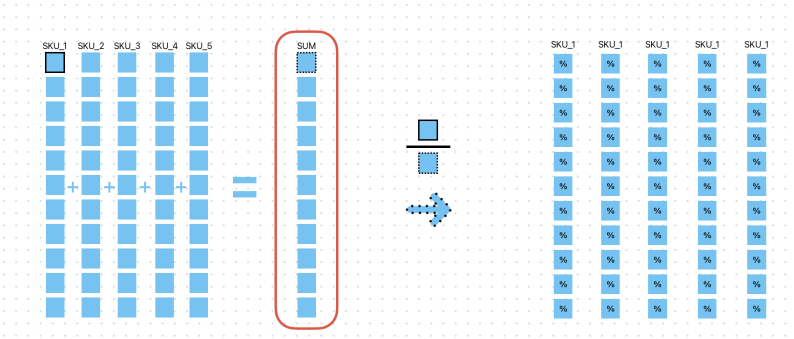

Figure 3.5: Percentages calculation

This approach fastens the way forecasts are made, as it decreases the number of times that the algorithm needs to fit to the data and compute new results. Instead of making predictions to all SKUs of each pharmacy (5317), we only make forecasts for clusters (2829), and then use matrix theory process to calculate every subsequent batch of values. It makes the process more efficient.

3.3 Forecasting Algorithms

3.3.1 Parameter tuning

Parameters are vital components of a model that are predetermined prior to training, influencing the model's behavior. Tuning these parameters is crucial since their optimal values are often unknown, and different values can result in varying model performances.

In the context of neural networks, an inadequate or excessive number of neurons can lead to underfitting or overfitting, respectively, negatively impacting model performance. Striking the right balance in the number of neurons becomes essential to optimize performance.

When a model involves multiple parameters, exploring a multidimensional space becomes necessary to identify the most suitable combination of values. This process, known as parameter tuning, is complex and time-consuming [53]

Grid search is a widely-used and straightforward algorithm for hyperparameter tuning. It systematically tests all possible combinations of hyperparameter values within a predefined range, using cross-validation to evaluate performance metrics. By exhaustively exploring the parameter space, grid search aims to identify the combination of values that maximizes average performance. However, it can be time-consuming and resource-intensive due to its evaluation of all combinations.

In order to efficiently evaluate the developed architecture and its algorithms, some algorithms widely used for forecasting purposes served as start point on this project. This work focuses on one-step-ahead forecasts as it is the most relevant for the application.

3.3.2 Machine learning

In this study, it was employed a combination of LSTM and XGBoost algorithms with a look-back window size of four observations. During the training phase, these models were trained to make predictions for each observation, taking into account the four most recent observations as their reference (which corresponds to approximately one month for weekly aggregated data). Subsequently, we utilized the same window size to forecast out-of-sample values at each time step. The hyper-parameters tuning was not in focus during the research, as the main goal was to understand the effectiveness of developed architecture.

3.3.2.1 XGBoost

Several key parameters were explored to control overfitting and optimize model performance.

The maximum depth of a tree was utilized to regulate overfitting. Higher depths allowed the model to capture specific relationships tailored to individual samples, but caution was required to prevent excessive complexity and overfitting. Typical values ranged from three to ten, striking a balance between complexity and overfitting.

The parameter *min_child_weight* determined the minimum sum of weights required for observations in a child node. It played a crucial role in controlling overfitting by constraining relationships that were too specific to individual samples. Careful consideration was needed to avoid setting excessively high values that could lead to underfitting. Excessively high values can lead to underfitting.

The subsample parameter determined the fraction of observations randomly sampled for each tree, serving as a preventive measure against overfitting. By introducing diversity and reducing dependence on individual instances, subsampling helped improve model generalization. Typical values ranged from 0.5 to 1, finding a balance between reducing variance and maintaining data representation.

Listing 3.1: XGBoost python code structure

```
# split data into train and test
train, test = prepare_data(list(data))

#define hyper parameters
hyperparameters = {
    "n_estimators" : 100,
    "max_depth" : 6,
    "subsample" : 0.5,
    "min_child_weight" : 1
}

#define model
model = XGBRegressor(
    n_estimators=hyperparameters["n_estimators"],
    max_depth=hyperparameters["max_depth"],
```

```
    subsample=hyperparameters["subsample"],
    min_child_weight=hyperparameters["min_child_weight"],
    objective="reg:squarederror",
    tree_method="hist"
)

#define input train data
X, y = train[:, 0:n_lag], train[:, n_lag:]

# fit model
trained_model = MultiOutputRegressor(model).fit(X, y)

# make forecasts
forecasts = make_forecasts_xgboost(trained_model, test)
```

To choose the hyperparameters an iterative process of choice was applied until it was realised that there was no overffiting in the model and that the model was managing to process the data correctly. Also it was taken into account the time performance when adjusting the hyperparameters.

3.3.2.2 Long-short term memory networks

For the LSTM implementation it was used the *Keras* library, and it was followed the implementation below. The model is a *simple LSTM*, with only two layers, which the goal is just to prove some points regarding accuracy. In order to address the issue of overfitting, it is recommended to incorporate a Dropout layer alongside each LSTM layer. The purpose of this additional layer is to mitigate overfitting by selectively deactivating random neurons during the training process. Consequently, this technique reduces the dependence on individual neuron weights and promotes a more robust model performance. A commonly employed compromise for the dropout rate is 20%, striking a balance between preserving model accuracy and mitigating overfitting tendencies [54]. Then number of epochs and neurons in each layer need to take into account a good treshold between maximum learning, complexity, performance and overfitting. To choose the hyperparameters an iterative process of choice was applied until it was realised that there was no overffiting in the model and that the model was managing to process the data correctly. Also it was taken into account the time performance when adjusting the hyperparameters.

Listing 3.2: LSTM python code structure

```
n_neurons_l1 = 3
n_neurons_l2 = 1
n_epochs = 5
n_batch = 1


# prepare data
scaler, train, test = prepare_data(data, n_test,
                        n_lag, n_period)
```

```
10
11
12 # reshape training into [samples, timesteps, features]
13 X, y = train[:, 0:n_lag], train[:, n_lag:]
14 X = X.reshape(X.shape[0], 1, X.shape[1])
15
16 # design network
17 model = Sequential()
18
19 #first layer
20 model.add(LSTM(n_neurons_l1,
21                batch_input_shape=(n_batch,
22                                   X.shape[1],
23                                   X.shape[2]
24                                   ),
25                return_sequences=True
26          )
27
28 #second layer
29 model.add(LSTM(n_neurons_l2, activation='relu'))
30
31 #droput layer
32 model.add(Dropout(0.2))
33 model.add(Dense(y.shape[1]))
34 model.compile(loss='mean_squared_error', optimizer='adam')
35
36 # fit network
37 model.fit(X, y, epochs=n_epochs)
38
39
40 # make forecasts
41 X, y = test[i, 0:n_lag], test[i, n_lag:]
42
43 # make forecast
44 forecast = forecast_lstm(model, X)
```

3.3.3 Statistical Algorithms

The Holt-Winters method employed in investigation uses all available previous data at the time of forecasting for calibration. This calibration process is iterative, where the current observation is incorporated into the training data as forecasts are generated. Subsequently, the model is recalibrated to predict the next step. One advantage of this approach is that each forecast is produced out of sample, ensuring that the model is continuously updated with previous observations, enhancing its accuracy and reliability.

3.3.3.1 Holt Winters

The most basic exponential smoothing model is (funnily) simple exponentially smoothing also knows as single exponential smoothing. This model just forecasts the level of the time series and does not take into account trend or seasonality [23].

The next step from this simple model is Holt's linear trend method, which is also known as double exponential smoothing. Like its name suggests, this model incorporates the trend as well as the level.

Finally, the next step from Holt's method is to find a way to to include seasonality in the exponential smoothing model. This is where Holt Winters (triple exponential smoothing) comes in!

Below is a demonstration of Holt-Winters implementation on this investigation.

Listing 3.3: Holt-Winters python code structure

```
from statsforecast.models import HoltWinters

# split data into train and test
test_data = prepare_data(list(data))

# define model
model = HoltWinters(season_length=4,
                    error_type = 'A'
                    )

# make forecast
forecasts = model.forecast(y = train_data,
                           h = n_period
                           )
```

For the implementation, it was used the Statsforecast library [55]. The additive method was used as the multiplicative one is inappropriate model for data with a lot of zero values [55]. It was considered a seasonality equal to four to train the algorithm. The hyper parameters were left to the algorithm to calculate the default value (library has that capabilities), as each time serie processed needs a different value of α, β and γ, depending on the characteristics of the time serie. the change in values will affect the weight of the most recent observations in the forecats, and smoothing factor that will be applied in trend and seasonality, respectively.

3.3.3.2 Intermittent demand methods

To implement Croston and TSB methods, it was used the Statsforecast python library. This library enables for fast forecasting processes, enabling to make forecasts with more than one method at the same time [55]. For the parameters on the Croston method, the choice was left for the algorithm to find the best parameter values. For α and β on TSB model, values 0.3 and 0.2 were chosen, respectively. This choice followed an iterative process of value testing to understand which one suited the forecasts for all time series better. Tuning those parameters will determine the relative

significance assigned to recent observations compared to historical ones and how periodicity will be updated at each period.

Listing 3.4: Croston and TSB python code structure

```
from statsforecast import StatsForecast
from statsforecast.models import Croston,TSB

models = [Croston(), TSB(alpha_d=0.3, alpha_p=0.2)]

#prepare intermittent data
train_data, test_data = prepare_data_individual(data)


model = StatsForecast(models=models, freq='W', n_jobs=-1)

model.fit(train_data)
forecasts = model.predict(h=n_period)

errors = evaluate_forecasts_intermittent(test_data,
                                         forecasts,
                                         n_period
                                         )
```

3.4 Post-processing

This section pertains to the last stage of the overall procedure. Its completion involves utilizing the pre-calculated percentage values derived from the pre-processing stage. By multiplying the market share linked to each stock keeping unit (SKU) with the total forecasted time series, we derive the time series corresponding to each product. It is a fast operation as it uses matrix theory to obtain the final results. It should be noted that the same percentage is applied to each forecasted time step. In essence, this step serves to conclude the aforementioned process.

3.5 Bench mark architecture

In the previous implementation, the estimation of future product demand in Portuguese pharmacies used a distinct framework. Notably, the current research differs from the previous approach in some aspects. The data was previously aggregated on a weekly basis during the preprocessing phase to predict future weekly values. Subsequently, the weekend values were derived by calculating the weekend-to-week percentage based on the preceding six-month data, determining the average value for each stock keeping unit (SKU), and finally obtaining the predicted weekend value.

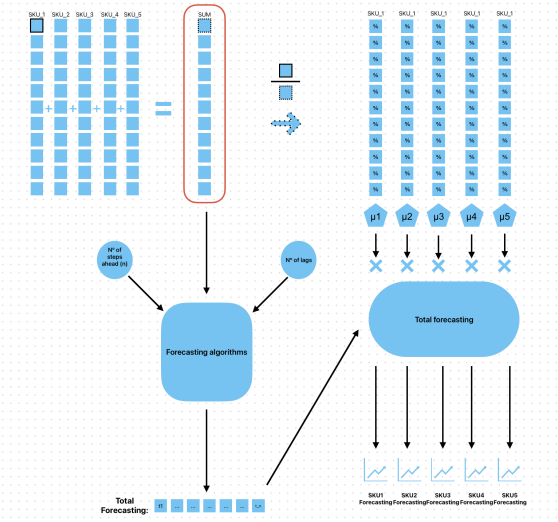

Figure 3.6: Complete forecasting process

Consequently, our novel architecture introduces two notable advancements. Firstly, we extract weekend-specific time series from the original daily dataset. Secondly, the data is subsequently aggregated by clusters, which represent SKUs associated with the same primary molecule.

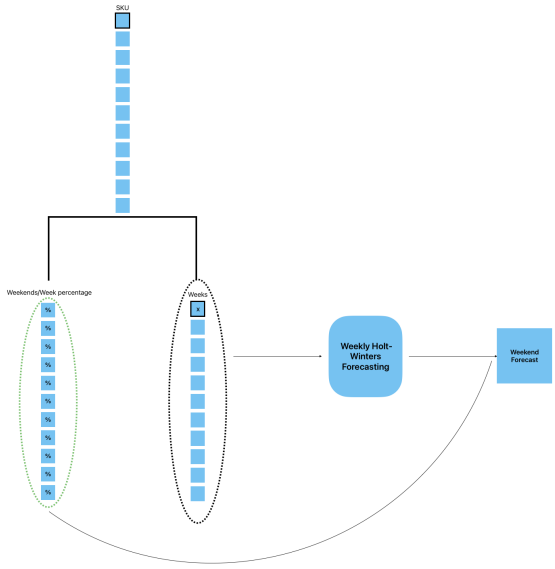

Figure 3.7: Bench mark architecture

3.6 Evaluation

The approach taken to evaluate our methods was to use a rolling origin forecast evaluation. This involves using a training set of historical data to fit a model and then using the model to predict

the next k steps in the forecast sequence. The first k actual values in the forecast sequence are then used to evaluate the accuracy of the forecast. The process is then repeated by updating the training set with the actual values and making new forecasts for the next k steps, until the end of the forecast sequence is reached. This approach allows for the evaluation of the entire forecast sequence, rather than just the initial forecast values.

Then, the average of each step ahead forecast was calculated to see how the models behave in general. To evaluate the results, both Root Mean Squared Error (RMSE) and Mean Arctangent Absolute Percentage Error (MAAPE) were used.

Chapter 4

Results

In the upcoming chapter, the obtained results from both strategies employed in this dissertation will be presented. The primary focus of comparison will be between the application of Holt-Winters in the old and new architectures. Additionally, XGBoost and LSTM will be tested in the new architecture to explore the potential of alternative methods within the new approach. Intermittent characteristics will also be evaluated to compare how the new approach deals with this type of time series.

The evaluation process comprises two main stages: firstly, the assessment of forecasting accuracy using metrics such as mean absolute percentage error (MAAPE) and root mean square error (RMSE); secondly, the evaluation of the computational efficiency, specifically the forecasting time performance exhibited by each algorithm.

Regarding the dataset and SKUs classification, the Figure 4.1 shows where all the SKUs used to validate the new approach are classified taking into account the work of [2]. Comparing it to the Figure 2.2 in Section 2.2, it is clear that a large part of the SKUs fall into the intermittent category. Smooth SKUs have a much smaller representation in the dataset used. This information is very useful to interpret the results achieved with the developed approach.

Looking into mean values for mean value and standard deviation of all SKUs weekend time series, from Table 4.1 and from the distribution of results, Figure 4.2, it is possible to see the framing of the various SKUs in Figure 4.1, proving the greater number of intermittent products in relation to the rest.

Table 4.1: Performance time for weekend forecasts

Algorithm	Value
Mean	0.3354
Standard Deviation	0.5095

The dataset used on this approach comprises daily data from 5317 SKUs, since January 1st of 2021 until the last weekend of May 2023. After pre-processing, where each time series is aggregated in weeks/weekends, and then the resultant weekend time series are aggregated taking

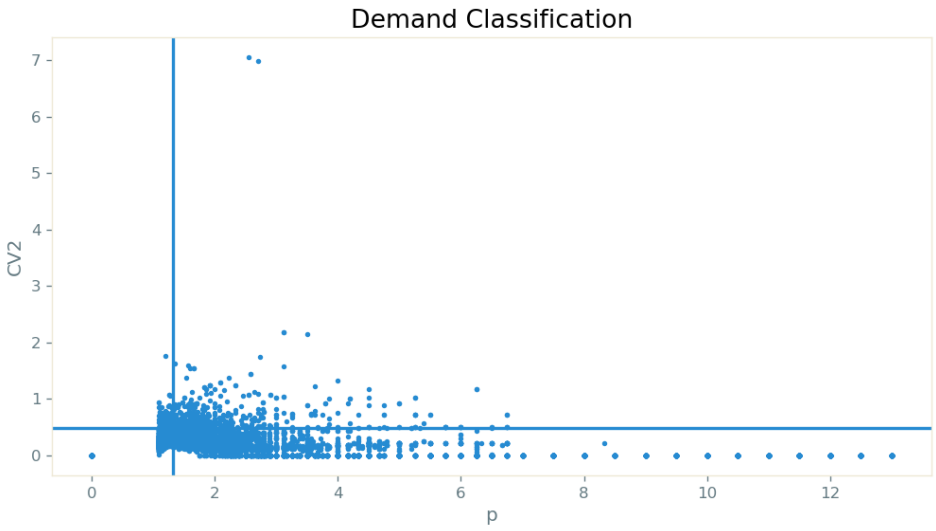

Figure 4.1: Demand classification, where the vertical and horizontal lines illustrate the cut-off points for ADI (1.32) and CV2 (0.49), respectively, on the Syntetos-Boylan classification framework [2]

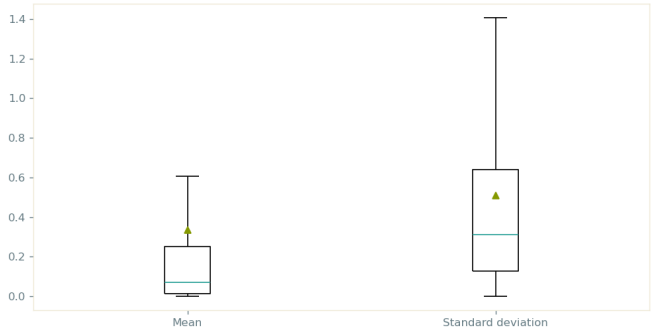

Figure 4.2: Mean values for SKUs time series mean and standard deviation

into account the common molecule (CNPEM), lead to have "only" 2829 time series (number of clusters), with only weekend data points.

The dataset used in this study consists of daily data from 5317 product variants (SKUs) starting from January 1st, 2021 until the last weekend of May 2023. The data was processed by grouping the time series into weekly and weekend intervals, as mentioned before. Then, these weekend intervals were further combined based on a common factor called CNPEM, resulting in a reduced dataset of 2829 time series.

These time series represent different groups, each containing only data points from weekends.

Each CNPEM time serie contains 127 data points corresponding to the sum of the number of weeks/weekends of each year since the mentioned date.

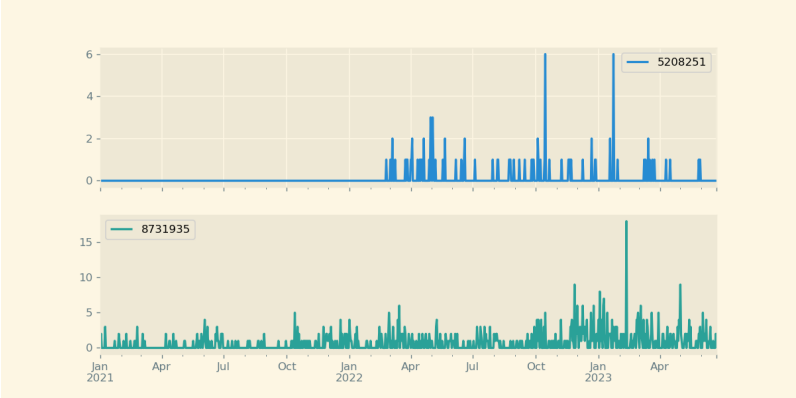

Figure 4.3: Daily time series for all SKUs inside molecule 50005707

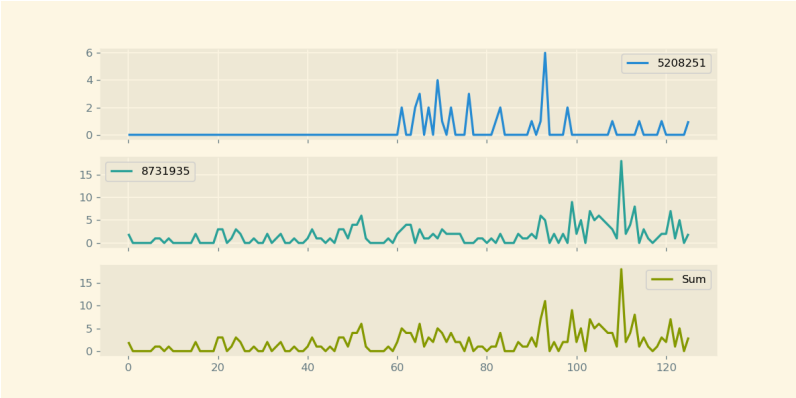

Figure 4.4: Example weekend time series for all SKUs inside molecule 50005707

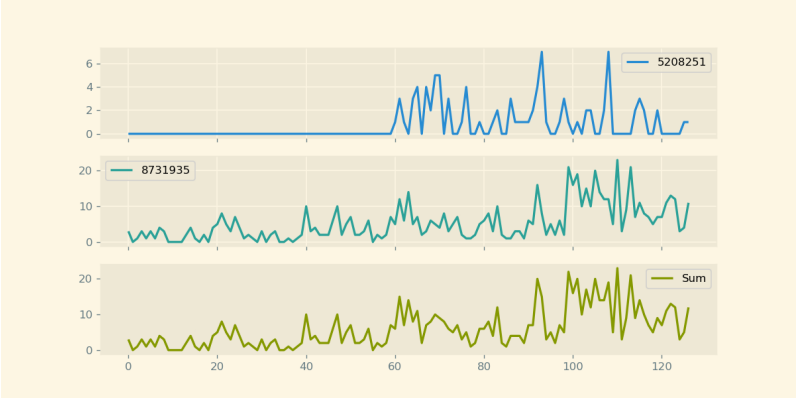

Figure 4.5: Example week time series for all SKUs inside molecule 50005707

To calculate the market share of each SKU to use in the post processing part (Section 3.4), it was only used data of the past six months, starting at January 2023. This choice comes from the fact that demand varies, so it is needed to have the most recent data in order to understand the market needs. Figure 4.4 provides a visual representation of the aforementioned observation.

The graph clearly illustrates the contrasting demand patterns between the past six months and the corresponding period two years ago.

Figure 4.6: Example of market shares time series for all SKUs inside molecule 50005707. The SKU 5208251 has a mean value of 15.26% (market share) and SKU 8731935 has a mean value of 84.73% (market share)

After calculating the market share for each SKU in each data point (Figure 4.6), it is further calculated the average value of each serie, to then use in the last stage of the process, where it is calculated the individual forecasts based on the cluster forecast.

4.1 Forecasting Accuracy

4.1.1 Week forecasts

The comparison between week clustering forecasting and week individual forecasting provides a baseline for showcasing the potential of the developed architecture in the realm of demand forecasting.

Table 4.2 and Figures 4.7 and 4.8 present the results, *a* corresponds to the outcomes obtained using the developed architecture (clustering by CNPEM), while column *b* represents the results for individual forecasts without employing SKU clustering.

A careful analysis of these comparisons reveals substantial improvements brought about by the new methodology.

4.1.2 Weekend forecasts

Taking the example shown above in Figures 4.4, 4.5 and 4.6, the plots below, Figures 4.10 and 4.11, show the results for that specific cluster and data.

The percentages show in Figure 4.6, were used to calculate the forecasts for the respective SKUs (Figure 4.10 and Figure 4.11).

That methodology was then applied for all the items in the data set, and the results are presented in Table 4.3.

Table 4.2: Mean, median and standard deviation on errors for each method and metric over the complete set of week time series forecasts

	MAAPE			RMSE		
	Mean	Median	Std	Mean	Median	Std
xgboost a	0.2292	0.1917	0.1709	0.4347	0.3852	0.3038
xgboost b	0.2950	0.2356	0.2301	0.8534	0.5477	0.6915
holt-winters a	0.1928	0.1570	0.1583	0.3227	0.3162	0.2763
holt-winters b	0.2667	0.1801	0.2123	0.8156	0.4472	0.7773
lstm a	0.2641	0.2356	0.2435	0.5092	0.4479	0.3657
lstm b	0.3199	0.2668	0.2819	1.004	0.6632	0.9129

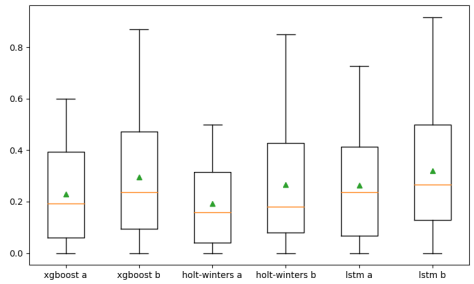

Figure 4.7: Maape data visualization for week forecasts (mean values are plotted with triangles)

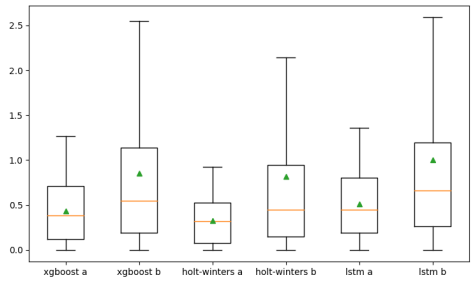

Figure 4.8: Rmse data visualization for week forecasts (mean values are plotted with triangles)

Table 4.3: Mean, median and standard deviation on errors for each method and metric over the complete set of items on weekend forecasts

	MAAPE			RMSE		
	Mean	Median	Std	Mean	Median	Std
xgboost	0.1624	0.07	0.1899	0.3730	0.17	0.5134
holt-winters	0.1499	0.07	0.1803	0.3096	0.14	0.4113
lstm	0.1856	0.09	0.2057	0.3996	0.19	0.5248
bench mark	0.3143	0.252	0.2611	0.8758	0.4472	1.2694

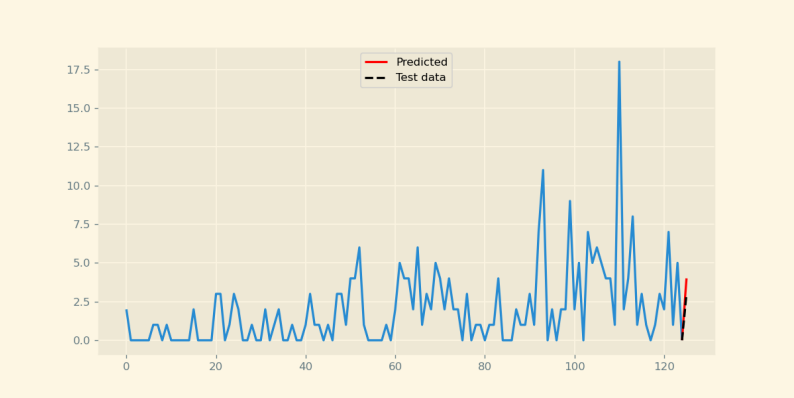

Figure 4.9: Forecast for the sum of SKUs time series inside cluster 50001817

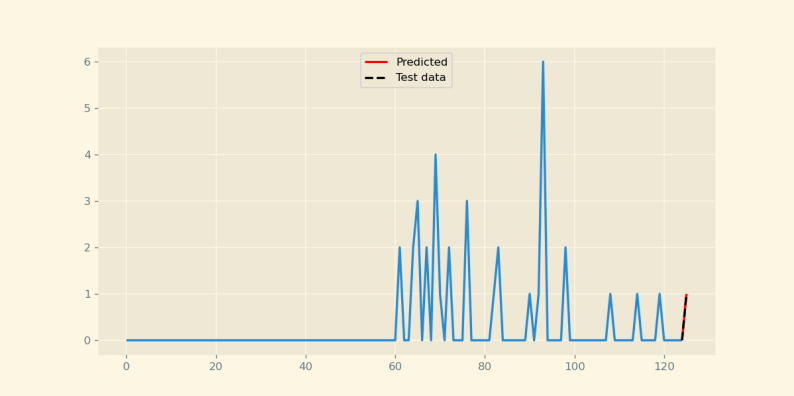

Figure 4.10: Forecast for SKU 2532893 inside 50001817 cluster

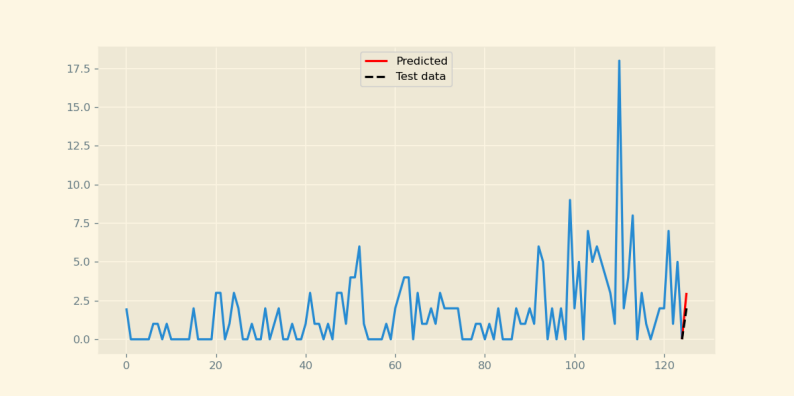

Figure 4.11: Forecast for SKU 9577700 inside 50001817 cluster

Figure 4.12 and Figure 4.13 show a graphic visualization of *MAAPE* and *RMSE* error, respectively, for the complete set of items forecasts.

When comparing the performance of Holt-Winters in both the new and benchmark architectures, notable improvements are observed. Specifically, the new architecture demonstrates a

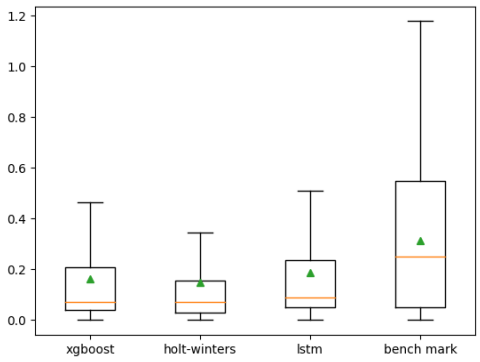

Figure 4.12: Maape data visualization for weekend forecasts (mean values are plotted with triangles)

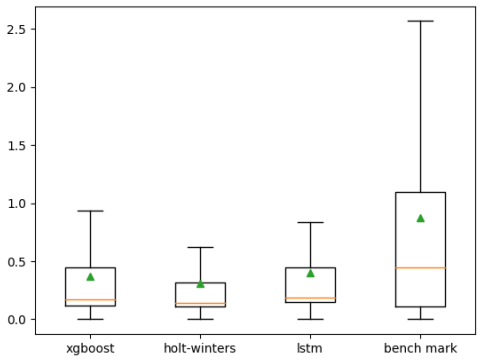

Figure 4.13: Rmse data visualization for weekend forecasts (mean values are plotted with triangles)

substantial enhancement of 50.94% in terms of mean absolute percentage error (MAAPE) and a significant reduction of 64.64% in root mean square error (RMSE), both in average values. Comparing the bench mark architecture with the new architecture in terms of median values, it is possible to see the big decrease, which shows that the new approach is more accurate than the older approach.

Furthermore, it is evident that the clustering forecasts contribute to the attenuation of higher error values (Figure 4.12 and Figure 4.13). Many time series data associated with SKUs exhibit intermittent or erratic patterns, which often result in significant forecasting errors when predicting future values, especially during weekends. However, by aggregating and predicting for the entire cluster, the forecasts consider the **sum** of all SKUs time series that share the same active ingredient, making the resultant time serie smoother.

This approach of aggregating and considering the entire cluster enables a more comprehensive and informed forecasting process. It helps to mitigate the impact of erratic individual SKU

behavior by incorporating information from similar SKUs with the same active ingredient. Also, taking into account only weekends instead of weekly data, enables algorithms to catch only patterns associated with weekends, reducing the amount of unnecessary information regarding week data that needs to be processed.

As a result, the forecasting accuracy is improved, and the errors associated with individual SKUs are mitigated or minimized.

4.1.3 Intermittent demand

One of the topics that we also tested was to check if the clustering technique was effective when forecasting for SKUs with intermittent characteristics.

When aggregating intermittent individual time series based on the CNPEM (3.2), 401 of 1808 cluster time series are not intermittent anymore. It represents a 23% of cluster time series.

Analyzing the error values in Table 4.4, and the results distribution in Figures 4.14 and 4.15 , both methods have similar results when forecasting individually for SKUs time series.

Table 4.4: Mean, median and standard deviation on errors for each method and metric over the complete set of intermittent items

	MAAPE			RMSE		
	Mean	Median	Std	Mean	Median	Std
tsb	0.4555	0.4792	0.4212	0.9112	0.8366	0.8451
croston	0.4642	0.4819	0.4371	0.9205	0.8368	0.8637

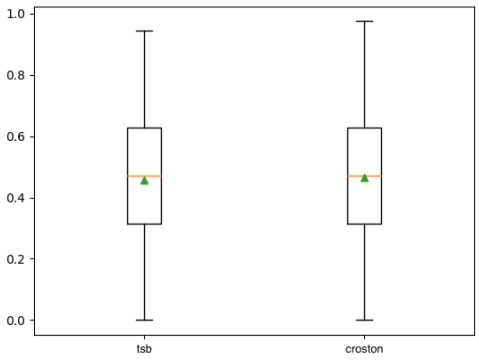

Figure 4.14: Maape data visualization for intermittent week forecasts (mean values are plotted with triangles)

And now, the forecasting results after cluster aggregation method. After aggregation of intermittent SKUs into clusters, as mentioned before, there are 23% clustered time series who started to be smooth. It means than the methods to be applied cannot be the same as for intermittent time series, Croston and TSB. So, it was applied Holt-Winters, which is our bench mark method. The results shown in Table 4.5, whose distribution is visible on Figures 4.16 and 4.17, prove that

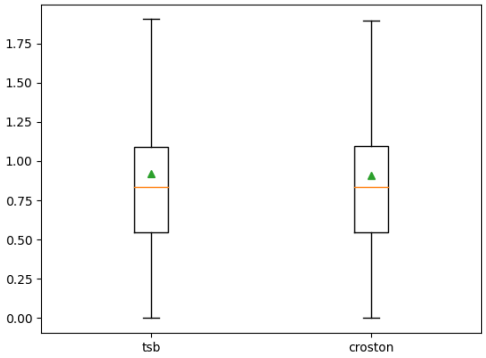

Figure 4.15: Rmse data visualization for intermittent week forecasts (mean values are plotted with triangles)

aggregation time series by molecule improve the way forecasts can be done, as it decreases the number of intermittent time series to process, which are more difficult to understand, and increases the number of smooth time series. The last ones enables algorithms to understand patterns better and achieve better results.

Table 4.5: Mean, median and standard deviation on errors for each method and metric over the complete set of intermittent/smooth items on week forecasts

	MAAPE			RMSE		
	Mean	Median	Std	Mean	Median	Std
holt-winters	0.2717	0.1570	0.2645	0.4195	0.3607	0.3296
tsb	0.3665	0.3712	0.2873	0.5314	0.4472	0.4208
croston	0.3479	0.3212	0.2837	0.5353	0.4483	0.4241

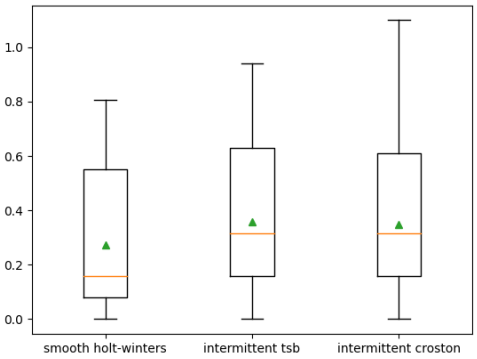

Figure 4.16: Maape data visualization for smooth vs intermittent clustering weeks forecasts (mean values are plotted with triangles)

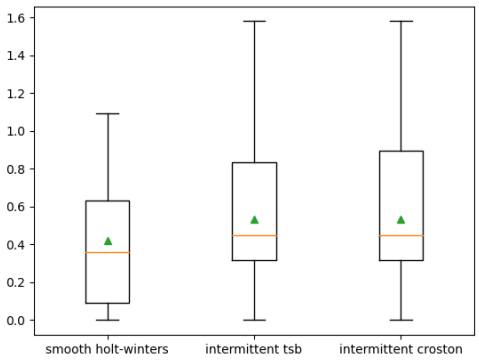

Figure 4.17: Rmse data visualization for smooth vs intermittent clustering weeks forecasts (mean values are plotted with triangles)

4.2 Forecasts time performance

One of the most relevant goals of this investigation was to optimize the performance in which forecasts were being generated. Time for training the algorithms, gather the results and validate them against unknown data needs to be taken into account when choosing the algorithm to generate the results for the following time steps.

For testing purposes, we tested a basic Long-Short term network, Holt-Winters and XGBoost again each other and the bench mark architecture to conclude which one is the best one in terms of time performance.

The Figure 4.18 shows the final results for weekend forecasts for all the clusters (CNPEMs) in a Portuguese pharmacy.

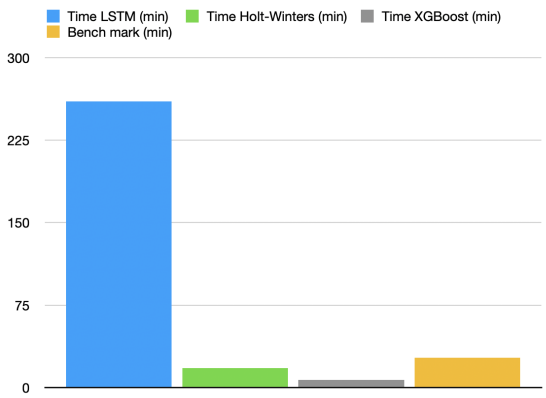

Figure 4.18: Performance times for weekend forecasts

The aforementioned values correspond to the time taken by each algorithm for training with the dataset, predicting future values, and validating them against new data. Notably, XGBoost

Table 4.6: Performance time for weekend forecasts

Algorithm	Time (min)
LSTM	260.23
Holt-Winters	17.45
XGBoost	5.94
Bench mark	27.26

emerges as the clear winner in terms of execution time. This significant improvement in execution time is primarily attributed to the fact that training and testing are performed on only 2829 time series, representing the number of clusters, as opposed to conducting these processes on all 5317 individual SKUs present in our dataset. With the new architecture, it was possible to improve in $32,35\%$ the execution time, with Holt-Winters. When looking to bench mark architecture with Holt-Winters against the new architecture with XGBoost, we see a clear improvement, more precisely $78,18\%$. This is a great sign that the efficiency can be improved a lot. On the other side, LSTM shows a lach of efficiency in terms of time, which is explained by the high complexity of the algorithm. Even with the new architecture, it was not possible to see improvements in time.

This reduction in computational workload is one of the major advantages offered by the new architecture. By leveraging clustering techniques, the number of distinct time series that require training and testing is significantly reduced. Consequently, the computational burden is alleviated, resulting in faster execution times.

The ability to work with a reduced set of time series not only accelerates the training and prediction process but also contributes to improved efficiency in terms of computational resources and memory utilization. This advantage reinforces the effectiveness of the new architecture and further supports its suitability for handling large datasets with a vast number of SKUs.

Chapter 5

Conclusions

In conclusion, this master thesis has successfully demonstrated the effectiveness of the developed architecture in improving week forecasts. Furthermore, the primary objective of accurately forecasting weekend demand has been accomplished, with the developed architecture and the utilization of solely weekend data proving to be more efficient compared to the previously employed approach. The improvements achieved are noteworthy, with a remarkable enhancement of 50.94% in MAAPE and 64.64% in RMSE regarding accuracy. The utilization of clustering forecasts played a crucial role in mitigating higher error values. By aggregating and considering entire clusters, the adverse impact of erratic individual SKU behavior was minimized. This approach facilitated a more comprehensive and informed forecasting process by incorporating information from similar SKUs sharing the same active ingredient. Additionally, by focusing solely on weekends instead of weekly data, the algorithms captured specific patterns associated with weekends, effectively reducing the processing of unnecessary weekly data.

Furthermore, the investigation into intermittent demand, specifically through the application of clustering by molecule (CNPEM), revealed that certain clusters exhibited a shift from intermittent to smoother patterns. The results underscored the benefits of this transformation, as forecasting smooth time series proved to be easier and more accurate than forecasting intermittent ones.

The performance analysis focusing on weekend data, which was also a must have of this research, showcased significant improvements of 78,18%. These advancements can be attributed to the cluster aggregating architecture implemented in the forecasting process.

While the present study has achieved notable results, it also highlights areas for future improvement. One avenue for further exploration involves optimizing the hyper-parameters of the forecasting methods, which could potentially yield even better results. Additionally, considering the incorporation of a moving average approach to calculate the market share of each SKU against the primary cluster could further enhance the forecasting accuracy.

In summary, this master thesis has made significant contributions to the field of demand forecasting by introducing a novel architecture that improves week forecasts and achieves excellent results in forecasting weekend demand. The findings highlight the importance of considering clustering techniques for intermittent demand and underscore the potential for future enhancements in

the hyper parameter tuning and market share calculation methodologies. The outcomes of this research serve as a solid foundation for further advancements in the field of demand forecasting.

References

[1] R.J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice, 3rd edition*. OTexts: Melbourne, Australia, 2021. https://otexts.com/fpp3/[Accessed on 2021-12-28].

[2] M Syntetos, John Boylan, and JD Croston. On the categorization of demand patterns. *Journal of the Operational Research Society*, 56, 05 2005. doi:10.1057/palgrave.jors.2601841.

[3] Yuanchao Wang, Z. Pan, J. Zheng, L. Qian, and Li Mingtao. A hybrid ensemble method for pulsar candidate classification. *Astrophysics and Space Science*, 364, 08 2019. doi:10.1007/s10509-019-3602-4.

[4] Marta Gołąbek, Robin Senge, and Rainer Neumann. Demand forecasting using long short-term memory neural networks. [Accessed Feb. 10, 2022].

[5] Mingming Hu, Richard T.R. Qiu, Doris Chenguang Wu, and Haiyan Song. Hierarchical pattern recognition for tourism demand forecasting. *Tourism Management*, 84, 2021. [Accessed Feb. 3, 2022]. doi:10.1016/j.tourman.2020.104263.

[6] Chen Li. A fuzzy theory-based machine learning method for workdays and weekends short-term load forecasting. *Energy and Buildings*, 245:111072, 8 2021. [Accessed Jan. 21, 2022]. doi:10.1016/J.ENBUILD.2021.111072.

[7] Galina Merkuryeva, Aija Valberga, and Alexander Smirnov. Demand forecasting in pharmaceutical supply chains: A case study. volume 149, 2019. [Accessed Dec. 10, 2021]. doi:10.1016/j.procs.2019.01.100.

[8] Evangelos Spiliotis, Spyros Makridakis, · Artemios-Anargyros Semenoglou, and Vassilios Assimakopoulos. Comparison of statistical and machine learning methods for daily sku demand forecasting. *Operational Research*, 123. [Accessed Jan. 10, 2022]. URL: https://doi.org/10.1007/s12351-020-00605-2, doi:10.1007/s12351-020-00605-2.

[9] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34, 2018. [Accessed Jan. 8, 2022]. doi:10.1016/j.ijforecast.2018.06.001.

[10] Raheel Siddiqui, Muhammad Azmat, Shehzad Ahmed, and Sebastian Kummer. A hybrid demand forecasting model for greater forecasting accuracy: the case of the pharmaceutical industry. *Supply Chain Forum*, 2021. [Accessed Jan. 7, 2022]. doi:10.1080/16258312.2021.1967081.

[11] Jinghui Ma, Zhongqi Yu, Yuanhao Qu, Jianming Xu, and Yu Cao. Application of the xgboost machine learning method in pm2.5 prediction: A case study of shanghai. *Aerosol and Air Quality Research*, 20, 01 2019. doi:10.4209/aaqr.2019.08.0408.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997. URL: https://doi.org/10.1162/neco.1997.9.8.1735, arXiv:https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf, doi:10.1162/neco.1997.9.8.1735.

[13] Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu. Advances in optimizing recurrent networks. 12 2012. URL: http://arxiv.org/abs/1212.0901, doi:10.48550/arxiv.1212.0901.

[14] Felix Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12:2451–71, 10 2000. doi:10.1162/089976600300015015.

[15] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[16] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015. doi:10.1038/nature14539.

[17] aakarsha_chugh. Deep learning | introduction to long short term memory, 2023. URL: https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/.

[18] Felix Gers, Nicol Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of Machine Learning Research*, 3:115–143, 01 2002. doi:10.1162/153244303768966139.

[19] Klaus Greff, Rupesh Srivastava, Jan Koutník, Bas Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28, 03 2015. doi:10.1109/TNNLS.2016.2582924.

[20] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. 3 2016. URL: http://arxiv.org/abs/1603.02754http://dx.doi.org/10.1145/2939672.2939785, doi:10.1145/2939672.2939785.

[21] Charles C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1):5–10, 2004. URL: https://www.sciencedirect.com/science/article/pii/S0169207003001134, doi:https://doi.org/10.1016/j.ijforecast.2003.09.015.

[22] Peter R. Winters. Forecasting sales by exponentially weighted moving averages. *Management Science*, 6(3):324–342, 1960. URL: http://www.jstor.org/stable/2627346.

[23] Rob J. Hyndman and Anne B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006. URL: https://www.sciencedirect.com/science/article/pii/S0169207006000239, doi:https://doi.org/10.1016/j.ijforecast.2006.03.001.

[24] Georgi Boshnakov. Introduction to time series analysis and forecasting, 2nd edition, wiley series in probability and statistics, by douglas c.montgomery, cheryl l.jennings and muratkulahci (eds). published by john wiley and sons, hoboken, nj, usa, 2015. total number of pag: Introduction to time series analysis and forecasting, 2nd edition, wiley series in probability and statistics, by douglas c. montgomery, cheryl l. jen. *Journal of Time Series Analysis*, 37, 08 2016. doi:10.1111/jtsa.12203.

[25] Chafak Tarmanini, Nur Sarma, Cenk Gezegin, and Okan Ozgonenel. Short term load forecasting based on arima and ann approaches. *Energy Reports*, 9:550–557, 2023. 2022 The 3rd International Conference on Power, Energy and Electrical Engineering. URL: https://www.sciencedirect.com/science/article/pii/S2352484723000653, doi:https://doi.org/10.1016/j.egyr.2023.01.060.

[26] Bishnu Nepal, Motoi Yamaha, Aya Yokoe, and Toshiya Yamaji. Electricity load forecasting using clustering and arima model for energy management in buildings. 2019. URL: https://onlinelibrary.wiley.com/doi/full/10.1002/2475-8876.12135, doi:https://doi.org/10.1016/j.egyr.2023.01.060.

[27] J. D. Croston. Forecasting and stock control for intermittent demands. *Operational Research Quarterly (1970-1977)*, 23(3):289–303, 1972. URL: http://www.jstor.org/stable/3007885.

[28] Thomas R. Willemain, Charles N. Smart, Joseph H. Shockor, and Philip A. DeSautels. Forecasting intermittent demand in manufacturing: a comparative evaluation of croston's method. *International Journal of Forecasting*, 10(4):529–538, 1994. URL: https://www.sciencedirect.com/science/article/pii/0169207094900213, doi:https://doi.org/10.1016/0169-2070(94)90021-3.

[29] Aris A. Syntetos and John E. Boylan. The accuracy of intermittent demand estimates. *International Journal of Forecasting*, 21:303–314, 4 2005. [Accessed Jan. 30, 2022]. doi:10.1016/J.IJFORECAST.2004.10.001.

[30] Ruud H. Teunter, Aris A. Syntetos, and M. Zied Babai. Intermittent demand: Linking forecasting to inventory obsolescence. *European Journal of Operational Research*, 214:606–615, 11 2011. doi:10.1016/J.EJOR.2011.05.018.

[31] Mohamed Zied Babai, Aris Syntetos, and Ruud Teunter. Intermittent demand forecasting: An empirical study on accuracy and the risk of obsolescence. *International Journal of Production Economics*, 157:212–219, 11 2014. doi:10.1016/J.IJPE.2014.08.019.

[32] K Nikolopoulos, A A Syntetos, J E Boylan, F Petropoulos, and V Assimakopoulos. An aggregate–disaggregate intermittent demand approach (adida) to forecasting: an empirical proposition and analysis. *Journal of the Operational Research Society*, 62(3):544–554, 2011. URL: https://doi.org/10.1057/jors.2010.32, arXiv:https://doi.org/10.1057/jors.2010.32, doi:10.1057/jors.2010.32.

[33] Fotios Petropoulos, Nikolaos Kourentzes, and Konstantinos Nikolopoulos. Another look at estimators for intermittent demand. *International Journal of Production Economics*, 181:154–161, 11 2016. doi:10.1016/J.IJPE.2016.04.017.

[34] Xiaotian Zhuang, Ying Yu, and Aihui Chen. A combined forecasting method for intermittent demand using the automotive aftermarket data. *Data Science and Management*, 5:43–56, 6 2022. doi:10.1016/J.DSM.2022.04.001.

[35] Fotios Petropoulos and Nikolaos Kourentzes. Improving forecasting via multiple temporal aggregation. *Foresight - The international journal of applied forecasting*, 34:12–17, 08 2014.

[36] Nikolaos Kourentzes, Fotios Petropoulos, and Juan R. Trapero. Improving forecasting by estimating time series structural components across multiple frequencies. *International Journal of Forecasting*, 30:291–302, 4 2014. doi:10.1016/J.IJFORECAST.2013.09.006.

[37] Fotios Petropoulos and Nikolaos Kourentzes. Forecast combinations for intermittent demand. *Journal of the Operational Research Society*, 66, 06 2014. doi:10.1057/jors.2014.62.

[38] G.P. Spithourakis, Fotios Petropoulos, Konstantinos Nikolopoulos, and Vassilis Assimakopoulos. A systemic view of the adida framework. *IMA Journal of Management Mathematics*, 25, 04 2014. doi:10.1093/imaman/dps031.

[39] Marcelo A. Colominas, Gastón Schlotthauer, and María E. Torres. Improved complete ensemble emd: A suitable tool for biomedical signal processing. *Biomedical Signal Processing and Control*, 14:19–29, 11 2014. [Accessed Jan. 18, 2022]. doi:10.1016/J.BSPC.2014.06.009.

[40] Xuehong Gao and Gyu M. Lee. Moment-based rental prediction for bicycle-sharing transportation systems using a hybrid genetic algorithm and machine learning. *Computers and Industrial Engineering*, 128, 2019. [Accessed Feb. 19, 2021]. doi:10.1016/j.cie.2018.12.023.

[41] Souhaib Ben Taieb, Gianluca Bontempi, Amir F. Atiya, and Antti Sorjamaa. A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition. *Expert Systems with Applications*, 39, 2012. [Accessed Feb. 4, 2022]. doi:10.1016/j.eswa.2012.01.039.

[42] Kasun Bandara, Peibei Shi, Christoph Bergmeir, Hansika Hewamalage, Quoc Tran, and Brian Seaman. Sales demand forecast in e-commerce using a long short-term memory neural network methodology. [Accessed Feb. 15, 2022].

[43] Felix A. Gers, Douglas Eck, and Jürgen Schmidhuber. Applying lstm to time series predictable through time-window approaches. volume 2130, 2001. [Accessed Feb. 10, 2022]. doi:10.1007/3-540-44668-0_93.

[44] Shanhe Liao, Xianghong Zhou, and Weixiong Rao. Accurate demand forecasting for retails with deep neural networks. volume 2020-March, 2020. [Accessed Jan. 5, 2022]. doi:10.5441/002/edbt.2020.56.

[45] Neda Khalil Zadeh, Mohammad Mehdi Sepehri, and Hamid Farvaresh. Intelligent sales prediction for pharmaceutical distribution companies: A data mining based approach. *Mathematical Problems in Engineering*, 2014, 2014. [Accessed Feb. 5, 2022]. doi:10.1155/2014/420310.

[46] Dejan Mirčetić, Svetlana Nikoličić, Durdica Stojanović, and Marinko Maslarić. Modified top down approach for hierarchical forecasting in a beverage supply chain. volume 22, 2017. [Accessed Feb. 10, 2021]. doi:10.1016/j.trpro.2017.03.026.

[47] Juan Pablo Karmy and Sebastián Maldonado. Hierarchical time series forecasting via support vector regression in the european travel retail industry. *Expert Systems with Applications*, 137, 2019. [Accessed Feb. 15, 2022]. doi:10.1016/j.eswa.2019.06.060.

[48] Janne Nissi, Johanna Småros, Tommi Ylinen, and Timo Ala-Risku. Measuring forecast accuracy: The complete guide. relex solutions. [Accessed Dec. 28, 2021]. URL: https://www.relexsolutions.com/resources/measuring-forecast-accuracy/.

[49] Dominik Martin, Philipp Spitzer, and Niklas Kühl. A new metric for lumpy and intermittent demand forecasts: Stock-keeping-oriented prediction error costs. volume 2020-January, 2020. [Accessed Jan. 10, 2022]. doi:10.24251/hicss.2020.121.

[50] J.K. Ord and R. Fildes. *Principles of Business Forecasting*. South-Western Cengage Learning, 2013. URL: https://books.google.pt/books?id=SKZzMwEACAAJ.

[51] Spyros Makridakis, S. Wheelwright, and Rob Hyndman. *Forecasting: Methods and Applications*, volume 35. 01 1984. doi:10.2307/2581936.

[52] Bob Rupak Roy II. Multi-step lstm time series forecasting, 2021. URL: https://bobrupakroy.medium.com/multi-step-lstm-time-series-forecasting-6d9e4777883b.

[53] Gianluca Malato. Hyperparameter tuning. grid search and random search, 2021. URL: https://www.yourdatateacher.com/2021/05/19/hyperparameter-tuning-grid-search-and-random-search/.

[54] Karsten Eckhardt. Choosing the right hyperparameters for a simple lstm using keras, 2018. URL: https://towardsdatascience.com/choosing-the-right-hyperparameters-for-a-simple-lstm-using-keras-f8e9ed76f0

[55] Cristian Challú Kin G. Olivares Federico Garza, Max Mergenthaler Canseco. StatsForecast: Lightning fast forecasting with statistical and econometric models. PyCon Salt Lake City, Utah, US 2022, 2022. URL: https://github.com/Nixtla/statsforecast.