University of Padova

Department of Mathematics Tullio Levi-Civita

Master Thesis in Data Science

Machine and Deep Learning Applications for Inventory Replenishment Optimization

Supervisor
Prof. Mariangela Guidolin
University of Padova

Master Candidate
Massimo Andreetta
2038499

Accademic Year

2022-2023

Abstract

Inventory replenishment is the process of obtaining the items, components, and raw materials required to make and sell products. It guarantees that items and resources are acquired and delivered in an efficient and timely manner. Poorly managed inventory replenishment can have a severe influence on customers and the overall health of a business, which may result in lost revenue, reduced profits and damaged reputation. Implementing the correct inventory replenishment helps manufacturers and sellers in avoiding major issues such as stock-outs, delayed deliveries and overstocking. Accuracy of forecasting is therefore crucial to retailers' profitability. Fashion businesses need precise and accurate sales forecasting tools to prevent stock-outs and maintain a high inventory fill rate. This thesis navigates the complex landscape of fashion retail forecasting, addressing the challenges posed by intermittent time series data and stock management. Advanced forecasting models have been implemented to account for the intermittent nature of fashion product demand, resulting in predictions more accurate and reliable.The study extends also to stock replenishment strategies, emphasizing the importance of the reorder point, the Cycle Service Level and the safety stock. Lastly, it culminates in the development of a replenishment algorithm aimed at reducing stock-outs, which is a modified version of the Periodic Review Policy: Order-Up-To-Level, now tailored to the sporadic nature of intermittent demand.

Contents

Listing of figures

Listing of tables

Listing of acronyms

SKU	Stock Keeping Unit
LGBM	Light Gradient Boosting Machine
LSTM	Long Short-Term Memory
NBEATS	Neural Basis Expansion Analysis for Interpretable Time Series
ML	Machine Learning
ARIMA	Auto-Regressive Integrated Moving Average
NN	Neural Network
DNN	Deep Neural Network
OHE	One Hot Encoding
SES	Simple Exponential Smoothing
TSB	Teunter-Syntetos-Babai
RNN	Recurrent Neural Network
MAE	Mean Absolute Error
RMSE	Root Mean Squared Error
CFE	Cumulative Forecast Error
MSE	Mean Squared Error
MASE	Mean Absolute Scaled Error
MAPE	Mean Absolute Percentage Error
CSL	Cycle Service Level

1 Introduction

Fashion retail is a highly competitive market where inventory replenishment plays a key role in the profitability of companies. Inventory replenishment planning is not only an important component of supply chain management, but it is also a key factor of business logistics. Inventory replenishment is the process of ordering and obtaining from suppliers the necessary stock to meet customer demand while avoiding stock-outs and overstocks [1], i.e., stock shortages and stockpiling unnecessary stock. Accurate sales forecasting is therefore fundamental, as poor forecasting might lead to stock-out or overstock situations, obsolescence, low service level, rush orders, inefficient resource utilization and bottlenecks, all of which can have a direct and immediate impact on the company's profitability [2]. An inefficient forecasting system can also have an impact on customer service quality. For example, if a customer encounters a stock-out situation, they may decide to shop at a different retailer [3].

For these reasons, demand forecasting has become a popular research topic and many models for forecasting fashion products have been proposed in the literature over the past few decades. However, different business and industry decisions can lead to different methods to calculate forecasts. For example, in the fashion industry, products are usually characterized by long replenishment lead times, short selling seasons and nearly unpredictable demand and therefore, inaccurate forecasts [4]. All these features make the issue of forecasting demand particularly challenging.

I had the opportunity to work within a company headquartered in Triveneto, a prominent player in the footwear and clothing industry. My involvement with this company allowed me to explore, analyze, and finally develop a thesis on inventory replenishment optimization, a critical aspect of modern supply chain management. This thesis is centered around the study and implementation of various forecasting models to be used as basis for the Periodic Review Policy: Order-Up-To-Level [5] predictive replenishment method. This task was prompted by the company's pursuit of a better approach to inventory management.

By predicting future stock requirements with greater accuracy, the company aimed to not only optimize its resource allocation but also ensure a continuous supply of products to fulfill the ever-changing demands of its customers.

This thesis, therefore, represents a comprehensive exploration of the challenges, strategies, and innovations to enhance predictive replenishment within a company in the fashion industry. By the end of this research, we aim to provide valuable insights and solutions that can benefit not only the company itself but also other organizations seeking to refine their replenishment processes and stay at the forefront of their respective industries.

1.1 Operative Context

In recent years, the fashion industry has experienced a significant transformation due to a convergence of various factors [6], including:

- **Increase in customer requirements and expectations:** Customer demands have evolved, driven by the industry's shift from mass fashion to segmented fashion [7]. Additionally, consumer preferences exhibit a remarkable diversity within the fashion industry [8] and expect businesses to satisfy their needs in terms of both products and services. Lastly, one of the key criteria for customer satisfaction is service level quality [9], i.e., the company's ability to meet customer expectations while optimizing its supply chain processes.

- **Dynamic Pricing Strategies and Inventory Management:** Dynamic pricing strategies and inventory management have emerged as pivotal elements of change within the fashion industry [10]. These strategies allow businesses to adjust pricing based on real-time factors such as demand, supply, and market conditions. Effective dynamic pricing and inventory management play a crucial role in meeting customer demands while optimizing inventory levels and ultimately affecting a company's profitability and competitive positioning.

- **New technologies:** The fashion industry has been greatly influenced by the rapid dissemination of information and technology. Customers' capacity to instantly learn about new trends and brands has contributed to raising their expectations. On the flip side, manufacturers, distributors, and retailers have gained access to advanced tools for data gathering, which, in turn, has empowered them to make more informed and strategic business decisions [11].

As a result, the industry acknowledged the crucial necessity to quickly adapt to changing trends while responding to the complex and specialized needs of its customers [12].

As pointed out in [13], products in this industry tend to exhibit the following characteristics:

- **Short life cycles:** The fashion industry is characterized by short product life cycles, meaning that products have a limited period of relevance and marketability. This transitory nature is inherent to the industry as fashion items are designed to capture and express the prevailing trends, styles, and consumer preferences of a specific moment. These products are often created to mirror the spirit of the times, and as a result, their marketable lifespan is notably brief [14]. When the relevance of a fashion item fades due to shifting tastes and preferences, it quickly becomes less desirable.

- **Short selling season:** To maintain their competitiveness and consumer interest, many fashion retailers adopt a strategy of increasing the number of "seasons." Each season represents a distinct collection of products, often reflecting the latest fashion trends or themes. By doing so, retailers can frequently refresh their product lines and respond to changing consumer preferences. This approach accelerates the rate of change and product turnover within the fashion industry, thus posing a challenge for supply chain management.

On the other hand, the characteristics of the demand are:

- **High volatility:** The demand for this type of products is rarely stable or linear. Moreover, there are many potential sources of uncertainty that can have a direct impact on the sales such as the weather conditions, holidays, marketing actions, promotions, fashion trends and even the current economic situation [15]. For these reasons, it is very challenging to make precise forecasts due to the volatility of demand.

- **Impulse purchasing:** Additionally, the decision to purchase these products is often made at the moment of sale. This means that the customers are more encouraged to purchase an article if presented with it, which is why availability is so important.

- **Large variance in demand and product variety:** Demand is now more segmented and consumers are more selective regarding choice and quality. Fashion collections are also composed of an extremely large number of different products in many different models, colors and sizes, corresponding to many different stock keeping units (SKUs) [16]. As a result, the sales volume per SKU is quite low [17] and demand for SKUs within the same product line might vary considerably [18].

In addition to the aforementioned characteristics, it is worth highlighting that demand patterns in the fashion retail industry often exhibit an intermittent behavior. Intermittent demand, which is characterized by irregular, sporadic, and unpredictable sales patterns, is a well-documented phenomenon in this industry [19]. The nature of fashion products, heavily influenced by factors such as seasonality, trends, and consumer preferences, leads to instances where certain items experience surges in demand, followed by extended periods of relative quietness. This intermittency is further exacerbated by factors like limited editions, unique designs, and the constant emergence of new collections. As a result of these irregular demand patterns, fashion retailers must contend with the difficulty of efficiently managing stocks and optimizing replenishment strategies.

In the context of this thesis and the related research project, the company and its products also show this kind of intermittent behavior, especially regarding the high volatility and large variance of its SKUs. The company's various articles could be divided into three main categories:

1. **Shoes**: As a cornerstone of the business, footwear plays a pivotal role in the company's product lineup.
2. **Clothing**: In parallel with its footwear offerings, the company also boasts an extensive range of clothing items, maintaining a strong presence in this sector.
3. **Accessories**: In addition to shoes and clothing, the company's product line also includes a variety of accessories.

Moreover, it's important to note that this research was conducted focusing only on the company-owned stores, which represent a carefully selected and more limited portion of locations where their articles are available.

1.2 Thesis Outline

This thesis is divided as follows:

- In Chapter 2 we introduce the literature regarding demand forecasting, highlighting the challenges of intermittent time series and the advantages and disadvantages of various forecasting methodologies.

- In Chapter 3 we introduce the dataset provided by the hosting company, exploring its characteristics and properties and discussing the pre-processing techniques utilized.

- In Chapter 4 are described the forecasting models implemented for this project, namely Naïve, Croston's method, Light Gradient Boosting Machine (LGBM), Long Short-Term Memory (LSTM) network, and Neural Basis Expansion Analysis for Interpretable Time Series (NBEATS). Training and evaluation procedures are also discussed.

- Chapter 5 is centered instead on inventory replenishment, presenting the theory behind the chosen replenishment algorithm, its implementation and the simulations used to evaluate its efficacy in reducing stock-outs.

- In Chapter 6 we present and discuss the results obtained by the forecasting models and by the simulations of the replenishment algorithm.

- Lastly, Chapter 7 concludes this thesis discussing future research and works.

2
Demand Forecasting

Demand forecasting is one of the biggest challenges for retailers, wholesalers, and manufacturers in any industry, and this subject has drawn a lot of interest from both researchers and practitioners. Accurate predictions of future demand are essential for efficient inventory management, meeting customer expectations, and ultimately, ensuring business success. However, the dynamic nature of fashion, characterized by short product life cycles and unpredictable demand patterns, makes forecasting particularly complex. The main question is then whether the fashion industry can leverage forecasting methods effectively.

2.1. Intermittent Demand Forecasting

Intermittent demand series, also known as sporadic or count series, are a particular kind of time series. In this case, the time gaps between the arrival of successive demands are often quite lengthy and frequent, while the quantities demanded tend to be relatively low [20], as we can notice from the intermittent time series shown in Figure 2.1. On the other hand, traditional demand time series, often described as "smooth", tend to be more stable, exhibiting only very occasional gaps between demand occurrences. Typical products described via an intermittent time series are spare parts, heavy machinery, electronics [21] and the aforementioned fashion articles.

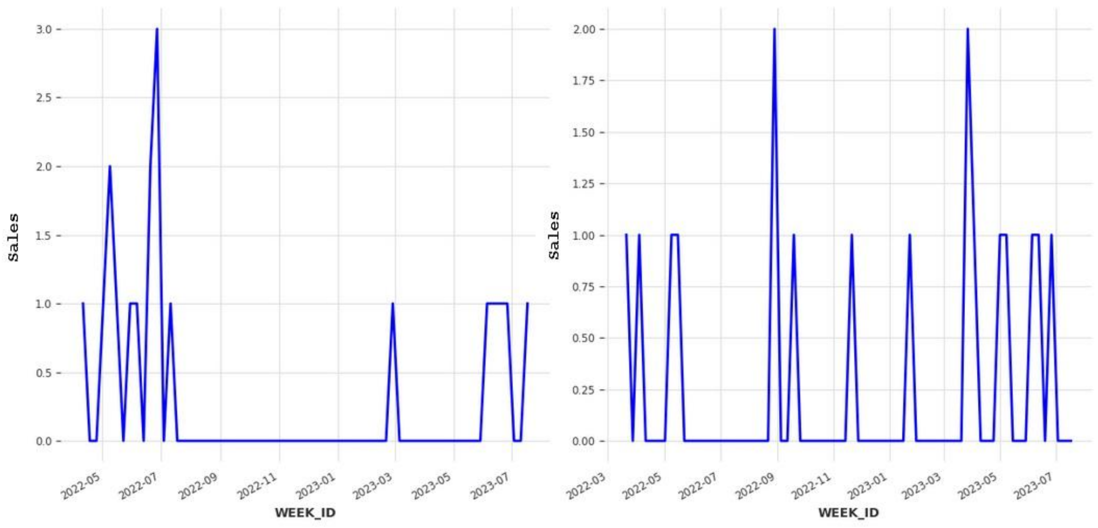


Figure 2.1: Examples of intermittent time series.

Forecasting intermittent time series is often recognized as one of the most difficult forecasting challenges [22]. As pointed out in [23], this is largely because intermittent demand time series forecasting introduces two key issues:

1. **Determining the timing of demand**: In this specific context, the main challenge lies in determining when the next demand period will occur. This issue does not exist in traditional time series forecasting, where demand tends to follow a more constant and stable pattern.
2. **Estimating the demand volume**: When intermittent demand appears, the objective is to correctly forecast its volume. This aspect is usually more complex to handle, primarily due to the inherent unpredictability of intermittent time series. First of all, prolonged periods of zero demand often separate active demand periods from each other, making it generally more challenging to learn how these periods affect each other. Secondly, the sporadic nature of demand arrivals further intensifies the complexity compared to traditional scenarios.

2.2. Forecasting Methodologies

A number of forecasting methods have been developed and employed in the fashion retail industry over the past few years. Current forecasting techniques are generally divided into two groups: classical methods based on mathematical and statistical models, and modern heuristic methods using machine learning (ML) and deep learning techniques.

2.2.1 Statistical Models

Statistical models have been widely used for forecasting both short-term and long-term sales and demand [24]. Among these models, the most commonly employed are Auto-Regressive Integrated Moving Average (ARIMA) [25], exponential smoothing [26], regression [27], Box & Jenkins [28] and Holt Winters [29]. However, these traditional forecasting methods are designed for scenarios characterized by smooth, high-volume demand. For this reason, they face limitations when applied to intermittent time series data, which is prevalent in the fashion industry, as we saw previously. These methods struggle to transform qualitative data features into quantitative ones, especially as sales patterns in the fashion retail industry exhibit significant variation [30].

Linear forecast models encounter substantial constraints in capturing the nonlinear relationships between exogenous variables, addressing outliers, handling missing data, and accommodating nonlinear components frequently found in real-world time series data [31]. They also fall short in capturing features like occasional outlying observations and asymmetric cycles, which are common in fashion demand data.

Furthermore, the complexity of the relationship between time-series data and the variables affecting sales performance demands more than just high precision for accurate forecasting. It requires reliable historical data for model structure identification and parameter tuning, posing a challenge when dealing with the unique characteristics of the fashion industry [32]. As demonstrated also in [33], conventional time-series approaches may not consistently capture nonlinear patterns in data, necessitating the exploration of more complex systems like Neural Networks (NNs) as an alternative to overcome these limitations.

2.2.2 Machine Learning Models

Because machine learning (ML) methods are data-driven, they are more generic and easier to adapt to forecasting series of different characteristics [34]. However, categorizing forecasting methods into statistical and machine learning is not simple, as various criteria can be used to perform this task [35]. In this regard, methods based on unstructured, non-linear regression algorithms, such as NNs, Decision Trees, and Support Vector Machines, are recognized as part of the ML domain [36].

NNs, inspired by the functioning of biological neurons [37], stand out for their proficiency in modeling complex, non-stationary, and nonlinear datasets. This distinct capability, bolstered by increased computational prowess, has positioned NNs at the forefront of research across diverse scientific domains [24]. They typically consist of three distinct layers: input, hidden and output layer. These layers are composed of nodes, also called neurons, which are the fundamental components of the NN and are responsible for processing inputs and producing the corresponding outputs. Each node of a layer has connections called weights with all the nodes, or a subset, of the next layer.

The input layer, found at the network's beginning, receives the data that the network is designed to process. At the opposite end lies the output layer, containing nodes that correspond to the response variables, which in the context of this study represent forecasting results. Between these two layers, there may be one or even multiple hidden layers, thus resulting in a Deep Neural Network (DNN). The nodes within these hidden layers represent the computational core of a NN and determine the model's capacity to capture complexity in the data, enabling it to fit intricate patterns and relationships. Figure 2.2 illustrates the multi-layer structure of a DNN.

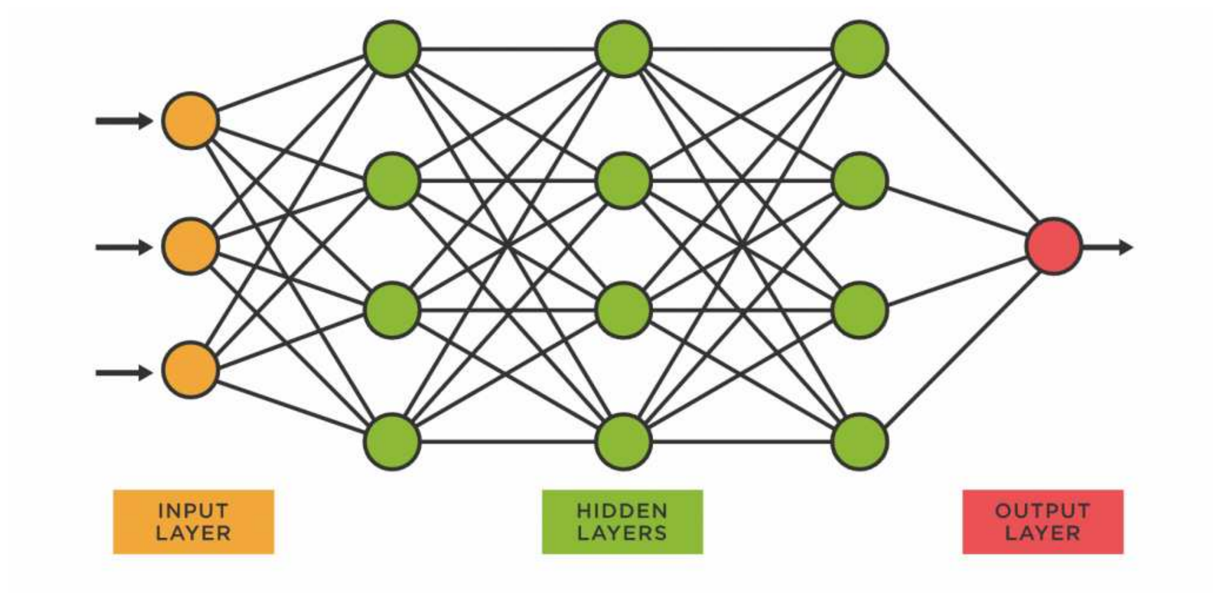


Figure 2.2: Architecture of a deep (multi-layer) neural network.

The output of neurons is computed through a series of operations, involving the dot product of input and weight vectors, the subtraction of bias values, and transformation via an activation function, typically a non-linear function like the sigmoid or ReLU. This process is repeated until the information reaches the output layer. Here, the output of the networks is used to compute the model's error through the loss function, which calculates the difference between the actual real value and the one predicted by the model. The loss function is generally selected based on the nature of the forecasting problem.

The result is then used to adjust the network's weights and biases in order to reduce the model's loss. The most common procedure is the back-propagation of errors. This is commonly achieved via stochastic gradient descent, which computes the gradients of the model's parameters with respect to the cost function. This is accomplished by applying the chain rule, beginning at the output layer and working backward through the network. After computing the gradients, the parameters' values are incrementally adjusted to bring the predictions closer to their true values.

Despite the efficacy of ML approaches, they have inherent limitations. To begin with, sufficient data is essential for ML algorithms to fully exploit their capabilities [36]. Moreover, computational time may become a critical issue, particularly when forecasting several series at weekly and daily frequencies [38]. To address these constraints, ML methods can be deployed in a cross-learning (CL) fashion rather than series-by-series [39], allowing models to collectively learn from multiple time series to enhance their capacity to forecast individual series effectively.

The underlying principle of CL hinges on the idea that shared patterns can emerge across different time series, especially when data is hierarchically organized and provided with categorical and exogenous/explanatory variables [40]. The CL technique has a number of advantages. First of all, it substantially reduces computational time, as a single model can simultaneously forecast numerous time series [41]. Moreover, data limitations are mitigated, and significant information can be accessed at the global level, allowing to capture patterns shared among the series, including seasonal cycles [42]. Additionally, models trained on one dataset can be employed to forecast time series from different datasets exhibiting similar characteristics [43]. Many researchers have achieved great results utilizing NN, and among these there are some interesting uses related to fashion demand [44, 45]. Other state-of-the-art implementations include LSTM [46], Temporal Convolutional Networks [47], and Temporal Fusion Transformers [48].

3
Data Exploration and Pre-Processing

This chapter introduces the dataset used and its properties. This real-world data collection was provided by the hosting company and consists of sales data encompassing various store-SKU combinations from diverse locations across Europe. In the context of this thesis, we identify with "SKU" a specific size of a distinct article.

3.1 Data Selection

In accordance with the company, a 2-year time horizon from 19/7/2021 to 17/7/2023 was deemed appropriate in order to avoid post-Covid-19 influences and also guarantee sufficient data to work with. After defining the scope of the data, it was agreed to aggregate the sales weekly and to select only a small subset of data from the dataset to ensure that the selected SKUs had relatively stable demand while also having the necessary data for performance assessment.

More specifically, it was decided to consider only items that existed in the store's product portfolio for at least 52 weeks, i.e.,1 year. Moreover, in order to select time series with a relatively stable demand, we first calculated the number of weeks with non-null sales for each SKU of each store. These were then divided by the corresponding total number of weeks. We then filtered out the SKUs with a ratio lower than 0.2, thus selecting the store-SKU combinations with at least 20% of weeks with actual sales. The reasons for this filtering are due to the fact that the majority of the store-SKU combinations, even after the aggregation to weekly data, present a demand pattern similar to the one shown in Figure 3.1.

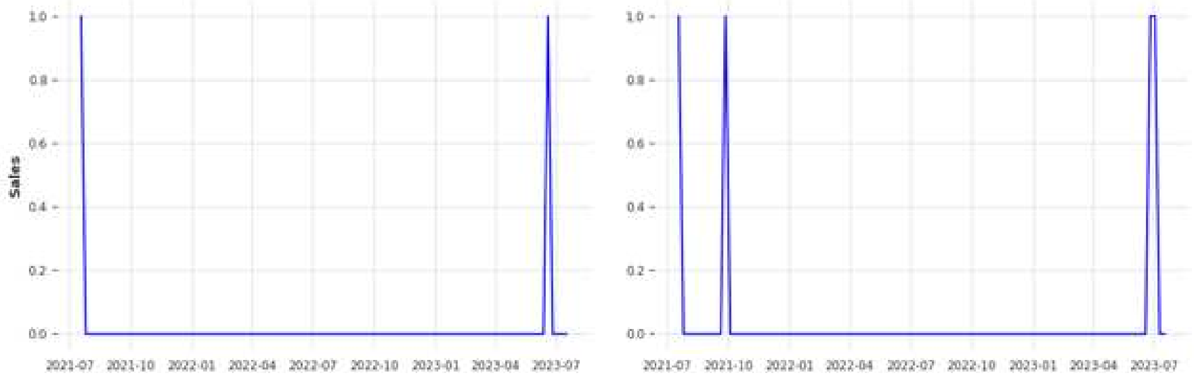


Figure 3.1: Examples of extremely intermittent time series that are very common in the full dataset.

This level of sporadicity is extremely difficult to forecast accurately. But the main reason to avoid such series is that they would not really benefit by the predictive replenishment method that will be discussed in Chapter 5, as it would simply resend the article it has just been sold, something that clearly does not require an elaborate method.

3.2 Data Exploration

The resulting subset of weekly sales data consists in 2077 unique store-SKU combinations divided into 141 stores and 311 distinct SKUs represented by 15 subcategories, which identify the type of article regardless of its size. These store-SKU time series have different lengths, ranging from 1 year to 2 years of sales history. Working with time series data of different lengths is generally not recommended, as many models and forecasting methods assume that data points are observed at consistent intervals.

To overcome this obstacle, we used the *Darts* forecasting library [49], which distinguishes itself by its ability to handle time series of various lengths. Moreover, it provides a powerful toolkit for implementing various forecasting methodologies, including classical statistical models and modern machine learning techniques. As we can see in Figure 3.2, the majority of the store-SKU combinations are centered between 20% and 25% of weeks with actual sales, while the rest is distributed between 25% and 45%, with one even passing 50%.

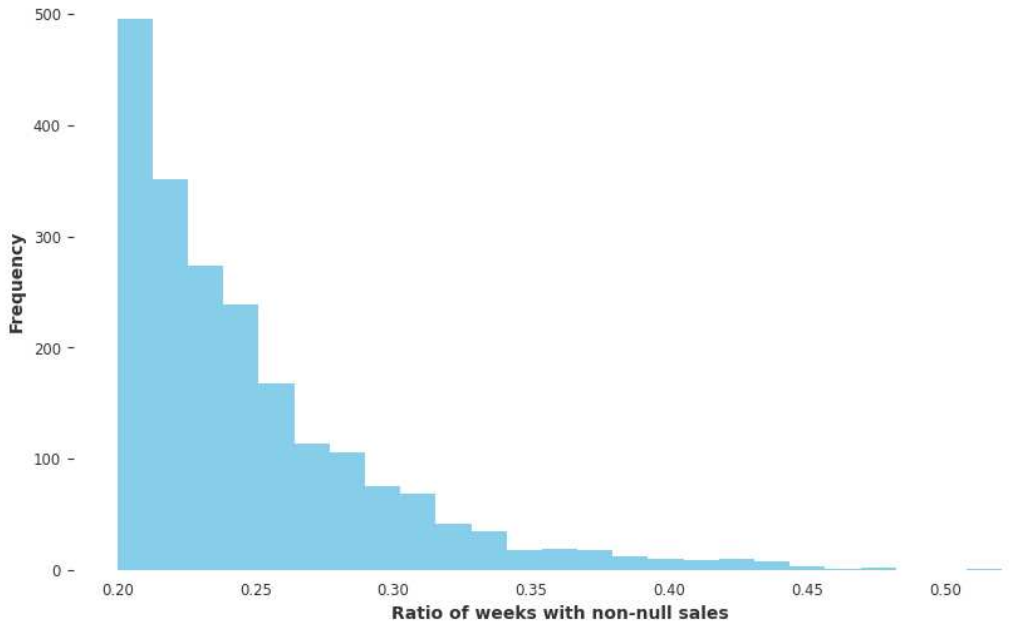


Figure 3.2: Distribution of the ratio of weeks with non-null sales.

However, it is worth mentioning that, even after selecting this more stable subset of products, we are still dealing with intermittent time series. Figure 3.3 confirms that the majority of weeks still have no sales, while the overall volume remains very low.

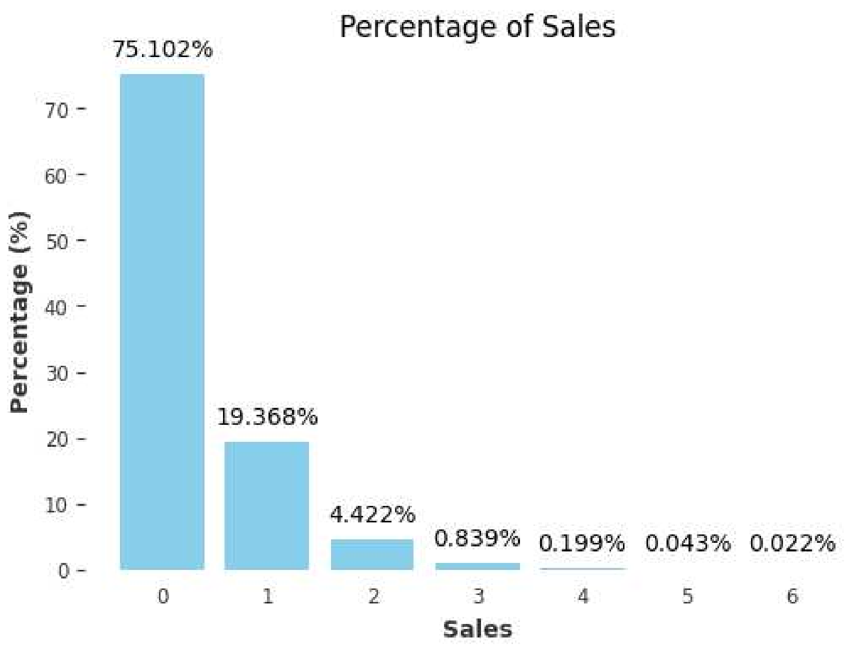


Figure 3.3: Distribution of the weekly sales of all the 2077 store-SKUs. The x-axis represents the number of weekly sales, while the y-axis represents the percentage of weeks with the corresponding sales.

Each of the 2077 combinations has both static and temporal features divided into two categories: Forecasting features and Replenishment features. The first category includes the features used in the forecasting models that will be described in Chapter 4. The store ID, SKU ID and subcategory ID are static features, i.e., features that remain constant over time. When dealing with multiple time series, this time independent information can help models identify the nature of the underlying series and improve forecasts. The temporal features are instead the weekly sales of each store-SKU and store-subcategory combination, as well as meteorological data such as the average temperature and rainfall of the locations where the stores are situated. A quick summary of these features can be found in Table 3.1.

Forecasting Features		
Feature Name	**Feature Type**	**Description**
Store ID	Static	Unique identifier of each store
SKU ID	Static	Unique identifier of each SKU, i.e., each article-size combination
Subcategory ID	Static	Unique identifier of each article independently of its size
Weekly Sales	Temporal	Weekly sales of each store-SKU combination
Subcategory Weekly Sales	Temporal	Weekly sales of each store-subcategory combination
Average Temperature	Temporal	Average weekly temperature registered in the store's location
Average Rainfall	Temporal	Average weekly rainfall registered in the store's location

Table 3.1: Description of the static and temporal features used for forecasting.

It was also proposed to use the "foot flow", i.e., the number of customers that entered a store each week, but unfortunately the data gathered by the company revealed several instances of missing values, too many to fill with regular data imputation techniques. Therefore, this feature was discarded since the performances of the forecasting models were affected negatively when we used it.

Other exogenous features added are the so-called Time-related features. These features are highly beneficial for time series forecasting as they can help forecasting models identify correlations between fluctuations in demand and time. Incorporating time-related features, such as the number of the week or the month, can help the models find patterns relating to seasonality and trends, improving the accuracy of the models. The time-related features added to the dataset are year, month, week of the year and the relative position with respect to the forecasting time. How these features and all the previous ones are encoded and pre-processed will be described in Section 3.3.

Lastly, replenishment features are instead the ones used in the implementation of the replenishment algorithm and during the simulation of its operation. Here are listed only the input features and not the ones that will be derived from the forecasts of the best model, as the implementation of the algorithm and its simulation are both going to be described more in detail in Chapter 5.

The static features include store ID, SKU ID, subcategory ID and logistic information such the average lead time and the "quantity to display", i.e., the minimum quantity of a SKU to have in store. Temporal features are instead the daily sales of the selected 2077 combinations, their quantity of stocks occupied in the stores each week and the forecasts produced by the best model. Table 3.2 contains a short description of each of these features.

Replenishment Features		
Feature Name	**Feature Type**	**Description**
Store ID	Static	Unique identifier of each store
SKU ID	Static	Unique identifier of each SKU, i.e., each article-size combination
Subcategory ID	Static	Unique identifier of each article independently of its size
Lead Time	Static	Average time required to send the new stock from the warehouses to a specific store
Quantity to display	Static	Minimum quantity of each SKU to always have in store
Forecasted Sales	Temporal	Weekly forecasted sales generated by the best model for each store-SKU combination
Daily Sales	Temporal	Daily sales of each store-SKU combination during the forecasted period
Stock Inventory	Temporal	Quantity of stock of each SKU stored in a store at the end of each week

Table 3.2: Description of part of the static and temporal features used for the replenishment algorithm and during its simulation.

3.2.1 Distribution

A chi-square test was performed to assess if the dataset follows the Poisson distribution, something that is important to know for the correct implementation of the replenishment algorithm in Chapter 5. Intermittent series are often associated with Poisson patterns because of their random and sporadic characteristics. However, it is important to note that not all intermittent series are necessarily well described by a Poisson distribution. This distribution is usually used to model rare and independent events in a fixed interval of time or space, whereas intermittent series can have a variety of different behaviors

The *Scipy* library was used to perform the test and it was used a confidence of 95% to reject the null hypothesis. In the chi-squared test, the null hypothesis is a statement that indicates there is no significant difference between the observed data and the expected Poisson distribution, meaning that the data is likely to follow this distribution. The alternative hypothesis is the opposite of the null hypothesis. It suggests that there is a significant difference between the observed data and the expected distribution, indicating that the data does not follow a Poisson distribution.

The p-value is a probability value that quantifies the evidence against the null hypothesis. Since we are using a 95% confidence level, if the p-value is lower than 0.05, it suggests that the observed data significantly deviates from the expected Poisson distribution. In this case, we may reject the null hypothesis and conclude that the data does not follow this distribution. If instead the p-value is larger or equal to 0.05, it suggests that the observed data is consistent with the expected Poisson distribution.

p-value >= 0.05	**p-value < 0.05**
1915	162

Table 3.3: Results of the chi-square test on the 2077 store-SKU time series

The results of the chi-square shown in Table 3.3 indicate that the majority of the time series follow a Poisson distribution.

3.2.2 Target and Covariates

Darts differentiate between four data types: target, past covariates, future covariates and static covariates. All except the static covariates are data in the form of time series. Figure 3.4 visualizes the main differences between the time series data types. Covariates are series that we do not want to forecast, but which can provide helpful additional information to improve the prediction of the target series. The target feature in this thesis is the Weekly Sales, which is the weekly number of units sold for a particular SKU of a particular store. The forecasting will therefore yield the predicted continuation of the target series.

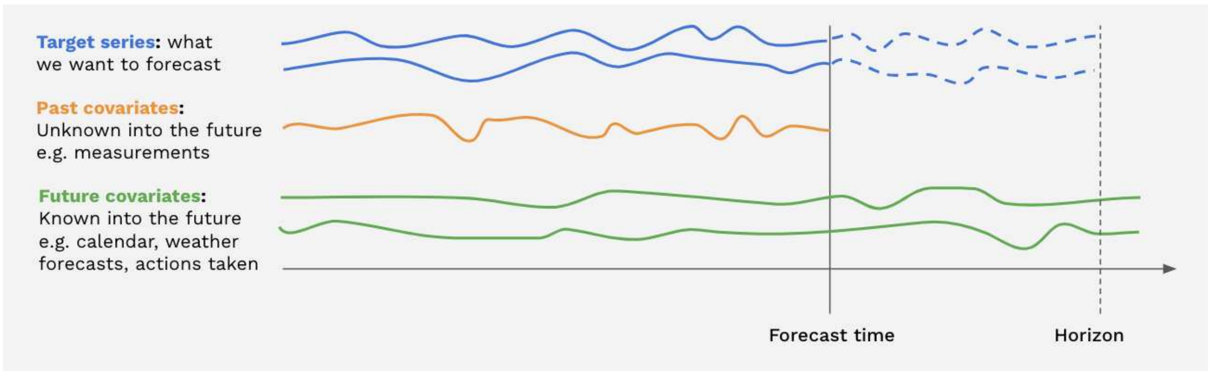

Target series: what we want to forecast
Past covariates: Unknown into the future e.g. measurements
Future covariates: Known into the future e.g. calendar, weather forecasts, actions taken
Forecast time
Horizon

Figure 3.4: Figure shows how different covariates are used for forecasting relative to the forecast time [49].

Past covariate are time series with information that can be known only into the past (e.g., measurements) up to the time of the prediction and not beyond. Using information from these past covariate time series beyond the prediction time would mean that the forecasting would use information about the future that it shouldn't be aware of at that moment. The subcategory sales, for example, is going to be used as a past covariate, since normally we do not know the future amount of sales past the prediction point.

On the other hand, future covariate series are time series whose future values are known or can be easily inferred. For example, the time-related features and the weather data are going to be used as future covariate, since information for future time steps can be easily obtained also after the prediction point. It is worth mentioning that Darts' forecasting models have different support modes for covariates. Some support all of them, others support only past or future covariates while some do not support covariates at all.

3.3 Pre-processing

This section describes the techniques used to pre-process the data as well as the reasoning behind the methods chosen for various types of features. In the case of numerical features, the values are scaled to make ML models easier to train, improve their performance, and make them more robust. Darts provides a *Scaler* package that implements MinMaxScaler to scale the values between 0 and 1. Scaling the data ensures that the features are on the same scale, thus preventing the models from favoring features based on their scale and making the models less sensitive to outliers in the data. Furthermore, scaling the data helps the ML models to converge faster, thus speeding up the training process.

Moreover, we utilized Dart's *MovingAverageFilter* to incorporate four additional past covariates. This was achieved by creating sliding windows spanning 4 and 12 weeks for the Weekly Sales and Weekly Subcategory Sales data, enabling us to examine monthly and quarterly trends as well as seasonal patterns. In this method, a window of specified length n moves over the data, sample by sample. The output for each input sample is the average over the window of the current sample and the n - 1 previous samples. This process aids in recognizing long-term trends, seasonality, and cyclic patterns that may be concealed in the original data, helping to understand and forecast seasonal patterns more accurately.

For the categorical features like the store ID, SKU ID and subcategory ID, it was used Dart's *StaticCovariatesTransformer* to implement a technique called One Hot Encoding (OHE) [50]. In OHE, each value of a feature is transformed into a binary vector representation of that value, enabling machine learning algorithms to work with these features effectively while preserving the categorical information and avoiding potential biases. Moreover, it preserves the categorical nature of the data by not introducing any ordinal relationship among categories. This is important because in our case, the categories have no inherent order, and treating them as numerical values might lead to incorrect conclusions.

For the time-related features instead, Darts provides a built-in functionality, called "encoders", to easily integrate such covariates into forecasting models. The year and the relative position w.r.t the forecasting time have been encoded numerically, while for the month and the week of the year it was used a different encoding. The reason is that the first and last month of a year would be encoded as being far away even though they are not. Using a cyclic encoding instead, allows to express proximity in cycles, thus encoding the last and first values in numerical ranges to be close to each other. This cyclic encoding is performed by transforming the values of a feature into two dimensions using a sine and cosine transformation shown here:

$$Xsin = sin\left(\frac{2\pi x}{max(x)}\right) \tag{3.1}$$

$$Xcos = cos\left(\frac{2\pi x}{max(x)}\right) \tag{3.2}$$

where x is the unique value, e.g., the numerical value of the month or the week, and max(x) is the highest value of x. Lastly, all the four time-related features are scaled using the MinMaxScaler Technique.

4 Forecasting

For the replenishment algorithm to work properly, it needs to have accurate forecasts for the next two weeks each week. This chapter offers the theory needed to understand the methods utilized and the results obtained. It begins by discussing the forecasting models used in this thesis, namely Naïve [51], Croston's method [52], LGBM [53], LSTM network [46], and NBEATS [54].

When it comes to intermittent demand forecasting, different models offer unique advantages and drawbacks. The Naïve method, while simple and computationally efficient, tends to be limited in capturing complex patterns. Croston's method excels in handling intermittent demand, but as we discussed in Section 2.2.1, conventional time-series approaches may not be the optimal choice, especially when compared with the more complex ML models. One of these models is LGBM, which is known for its speed, efficiency and its robustness, but may require careful tuning to obtain optimal results. Going in the realm of deep-learning, LSTM and its ability to capture temporal dependencies allows it to efficiently handle sequential data like time series. However, its complex network may be computationally demanding.

Lastly, NBEATS is a powerful non-recurrent neural network designed for time-series forecasting, able to capture intricate patterns within the data but may require substantial training data and be computationally expensive like LSTM. All the information regarding the implementation and training procedures of these models are provided in the following sections, while in the last one, we discuss the various evaluation methodologies.

4.1 Naïve

In the context of time series forecasting, a "naïve" forecast is one where the prediction for a future data point is simply set to be equal to the most recent observation in the time series. Naïve models are characterized by their simplicity, as they involve minimal computational complexity and do not incorporate any elaborate modeling techniques. They are frequently used as a baseline or benchmark against which the performance of more sophisticated forecasting methods is compared. This benchmarking approach helps assess whether the adoption of more complex forecasting models is justified by the potential improvement in accuracy [51].

Notably, when dealing with time series data that exhibit a *random walk* behavior, naïve forecasts are found to be optimal. A random walk is a stochastic process where each data point is a random step away from the previous point, making future values unpredictable based on historical data alone. Consequently, in situations characterized by high volatility and randomness, like the intermittent demand domain, naïve forecasts can be challenging to outperform, thus playing a valuable role in setting the performance standards for more advanced forecasting methods.

4.2 Croston's Method

Intermittent demand, characterized by sporadic periods of demand with extended periods of little to no demand, presents unique challenges for forecasting. Croston's method is a forecasting approach specifically designed to address this issue. Mathematically, Croston's method is based on two fundamental components described by two exponential smoothing models [23]:

- **Demand size:** The first component is the demand size of the periods with non-zero values. A Simple Exponential Smoothing (SES) method is used to predict the values for the days when there is demand as shown here:

  $$a_{t+1} = \alpha * y_t + (1 - \alpha) * a_t \tag{4.1}$$

  where y is the true demand size, a is the estimated value and α is the smoothing factor, typically set between 0 and 1.

  For the days with no demand, the predicted value a_{t+1} is instead equal to the previous one.

- **Inter-demand interval:** The second step is to use a different SES model to predict the intervals between days with sales. This step is important because it helps us to predict the number of days until the next sale occurs. When there is demand the formula for the Inter-demand interval is:

  $$p_{t+1} = \alpha * q + (1 - \alpha) * p_t \tag{4.2}$$

  where p is the estimated time between occurrences and q is the time elapsed since the previous demand occurrence.

  For the days with no demand, $p_{t+1}$is instead equal to the previous one.

The final step is to divide the first forecast (sales value) by the second forecast (number of days until the next sale) as follows:

$$y'_{t+1} = a_t / p_t \tag{4.3}$$

This gives us an estimate of the demand for each day. An improved variant of this model is the Teunter-Syntetos-Babai (TSB) Model. This method is inspired by the Croston method but instead of using the interval between value occurrences, it uses the probability of a value occurring. Like in Croston, the first step in the TSB Model is to use the SES method to predict the values of non-zero days. The SES is fit to a series of values that exclude the zeros, then it forecasts the next step.

The second step instead, introduces a β Exponential Smoothing parameter that modifies the p parameter . When there is demand we have:

$$p_{t+1} = 1 * \beta + (1 - \beta) * p_t = \beta + (1 - \beta) * p_t \tag{4.4}$$

On the other hand, when there is zero demand we have:

$$p_{t+1} = 0 * \beta + (1 - \beta) * p_t = (1 - \beta) * p_t \tag{4.5}$$

where 1 means there was a non-zero value while 0 means there was not. This step helps us predict the probability that there will be demand on any given day.

To get the final forecast, we instead multiply the first forecast (sales value) by the second forecast (probability of non-zero value) as follows:

$$y_{t+1} = p_{t+1} * a_{t+1} \tag{4.6}$$

4.3 LGBM

Unlike the previous two statistical models, LGBM is a gradient-boosting framework that uses tree-based learning algorithms. Gradient boosting is a machine learning technique for regression, classification and forecasting, and consists in iteratively optimizing the predictive performance by minimizing errors [55]. This iterative process involves the creation of a sequence of decision trees, with each successive tree aimed at correcting the errors of its predecessor, leading to the gradual improvement of predictive accuracy. Given a dataset with N samples, M trees in the ensemble, and a prediction function $F(x)$, the prediction process can be described as follows:

For the m-th tree, the model learns a function $f_m(x)$ that minimizes a differentiable loss function $L(y,\ f_m(x))$. The learning process typically employs a gradient descent algorithm. The final prediction $F(x)$ is a weighted sum of the individual tree predictions:

$$F(x) = \sum_{m=1}^{M} w_m \cdot f_m(x) \tag{4.7}$$

Where $f_m(x)$ denotes the prediction of the m-th tree and w_m signifies the weight assigned to the m-th tree's prediction. These weights are determined during the training process to minimize the overall model loss, with the specifics depending on the problem at hand.

The "Light" in LGBM's name emphasizes its efficiency, speed, and lower memory usage, making it particularly suitable for handling large datasets and real-time applications. LGBM adopts a leaf-wise growth strategy when constructing decision trees. Unlike conventional depth-wise growth, which incrementally expands the tree level by level, leaf-wise growth prioritizes nodes that offer the most significant reduction in loss. This strategy enables LGBM to construct deeper trees, facilitating the capture of complex patterns within the data. Additionally, LGBM incorporates histogram-based learning for data binning, enhancing training efficiency by organizing data into histograms and operating on these histograms to make informed splits. Moreover, LGBM excels in handling categorical features, ensuring that it can effectively incorporate pertinent categorical information into its predictions. However, it is worth mentioning that LGBM requires careful hyperparameter tuning to obtain optimal results.

4.4 LSTM

The ability to model and predict demand patterns characterized by temporal dependencies is critical in the field of intermittent sales forecasting. Recurrent Neural Networks (RNNs) are specifically designed to handle sequential data, i.e., data where values exhibit interdependence over time [56]. This inherent capability makes RNNs particularly well-suited for tasks such as natural language processing, video/audio processing and, more importantly, demand forecasting, setting them apart from conventional ANNs.

The recurrent layers, which contain neural units sequentially interconnected, are the core of a RNN, as they allow the network to process data sequences over time. This temporal learning capacity is particularly crucial when dealing with intermittent demand patterns, in which periods of minimal or no demand are sporadically interrupted by instances of demand. Nonetheless, while RNNs have significant advantages, they also have drawbacks. They can be computationally demanding, limiting their applicability in some situations. Additionally, RNNs are vulnerable to issues such as exploding and vanishing gradients, which become more pronounced with longer sequences [57].

The integration of LSTM units represents a step forward in overcoming these obstacles. LSTMs are equipped with memory cells capable of selectively retaining or passing information between units in recurrent layers through a gating mechanism. As described in [58], each cell processes a single step of a sequence at a time, treating each observation of the time series individually. At the same time, it maintains an internal state that serves as the network's memory, as it contains information about previous time steps. Each data point within a sequence is composed of the value at that specific time step and an internal representation derived from previous observations.

Therefore, the LSTM consists of three main components: the internal state, the hidden representation, and the raw values of the time series. At the start of the sequence, both the internal state and the hidden representation are initialized to zero. When it receives data from a sequence, the initial step in the LSTM process involves determining which information should be discarded from the existing internal state. This decision is made by a "forget gate," a multiplication operation applied to the input data using a matrix transformed by a sigmoid function. In the subsequent step, the LSTM updates its internal state with new information. Like the forget gate, an "input gate" comes into play, dictating which information should be added to the internal state.

Following this, the LSTM merges information from the current internal state with the newly acquired information from the input series, thereby generating a refreshed internal state. In the final step, the LSTM employs an "output gate" to create the hidden representation to be passed to the next step based on the current internal state. Figure 4.1 better visualizes the internal structure of an LSTM cell. This iterative process involving the hidden representation, the updated internal state, and the next time step of the time series permits the LSTM to effectively capture and learn from temporal dependencies and patterns within the data.

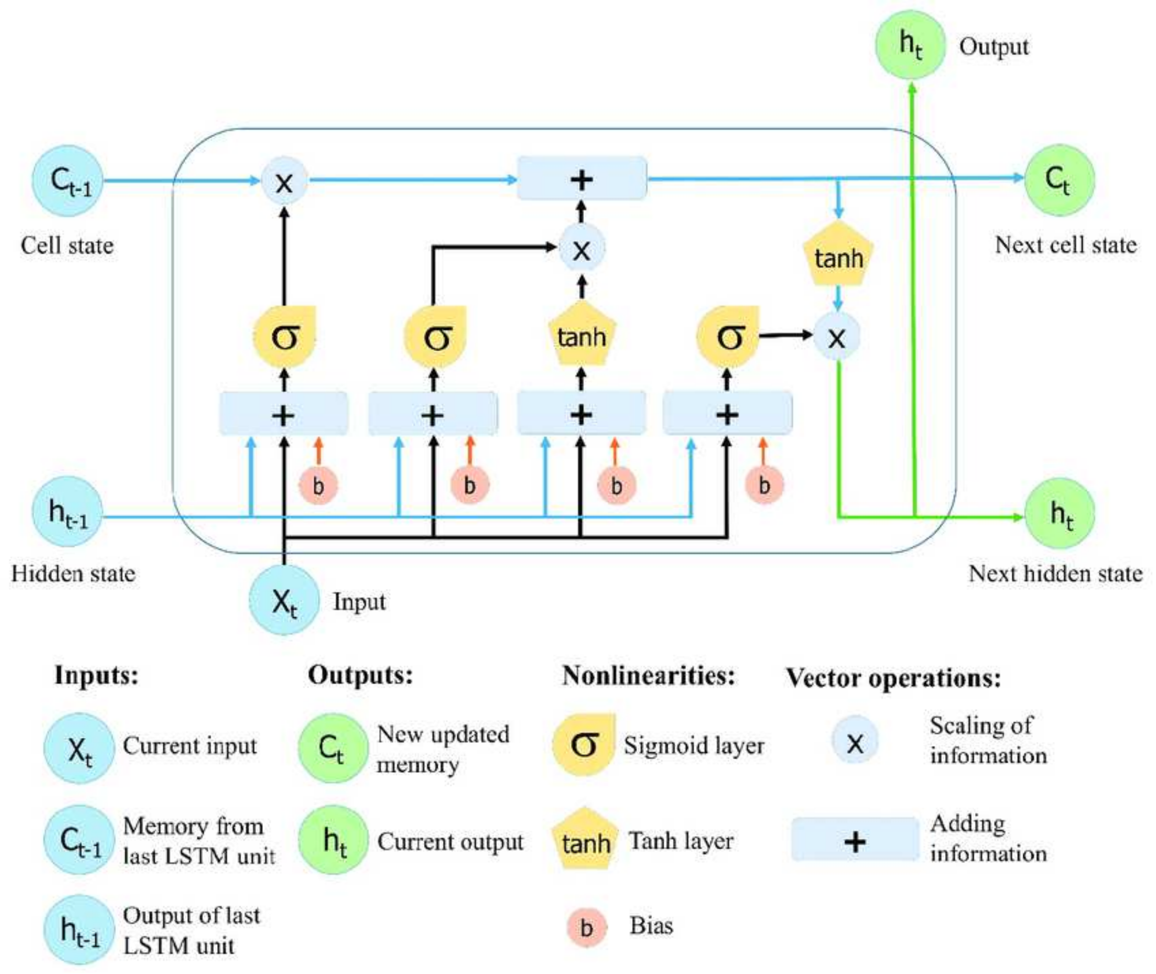


Figure 4.1: Internal structure and processes of an LSTM cell [59].

Therefore, the LSTM components are described by the following formulas:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{4.8}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{4.9}$$

$$c_t = f_t * c_{t-1} + i_t * tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{4.10}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{4.11}$$

$$h_t = o_t * tanh(c_t) \tag{4.12}$$

where:

- f_t, i_t, and o_t represents the forget gate, the input gate and the output gate respectively.
- W_f, W_i, W_c and W_o represent the weight matrices of the respective gates.
- b_f, b_i, b_c, and b_o are bias vectors associated with the gates.

4.5 NBEATS

N-BEATS recently emerged as a cutting-edge deep-learning architecture designed exclusively for time-series forecasting. It distinguishes itself as one of the most successful non-recurrent neural network architectures, capable of forecasting multiple time-series data without reliance on intricate time-series-specific feature engineering. This exceptional adaptability positions N-BEATS as an invaluable tool for a wide array of forecasting problems.

At its core, N-BEATS comprises a series of basic building blocks, each designed to receive an input and produce two outputs: one for forecasting (forward prediction) and the other for backcasting (backward estimation) [60]. The full architecture is constructed by stacking these blocks in a hierarchical structure, as it can be seen in Figure 4.2. These building blocks encompass two fully connected layers arranged in a fork architecture, and they come into play after an initial stack of fully connected layers processes the input data. The forecast layers focus on generating precise predictions for future data points, while the backcast layers work on estimating the input values themselves, staying within the constraints of the functional space available for signal approximation.

These networks produce forward and backward expansion coefficients that are crucial for the prediction process. In N-BEATS, the forecast and backcast outputs are generated using a weighted combination of these coefficients. This design allows for modeling both future data points and the past, giving N-BEATS its forecasting and backcasting capabilities. The forward prediction, capturing the future trend, is computed using a weighted combination of the forward expansion coefficients:

$$F_i(x) = \sum_{i=1}^{m} w_i^f \cdot g_i(x) \tag{4.13}$$

The backward estimation, which focuses on reconstructing the input data, is calculated by a weighted combination of the backward expansion coefficients:

$$B_i(x) = \sum_{i=1}^{m} w_i^b \cdot h_i(x) \tag{4.14}$$

Here, $F_i(x)$ and $B_i(x)$ represent the forward forecast and the backward estimate for the i-th data point, respectively. The weights w_i^f and w_i^b are learned during training, determining the significance of each forecast and backcast output in the final prediction. The functions $g_i(x)$ and $h_i(x)$ represent the fully connected networks in the forward and backward branches.

The stacking of these basic building blocks is orchestrated using a technique known as "doubly residual stacking" [54], creating a hierarchical structure with residual connections across different layers. This approach enhances interpretability and fosters a transparent network structure.

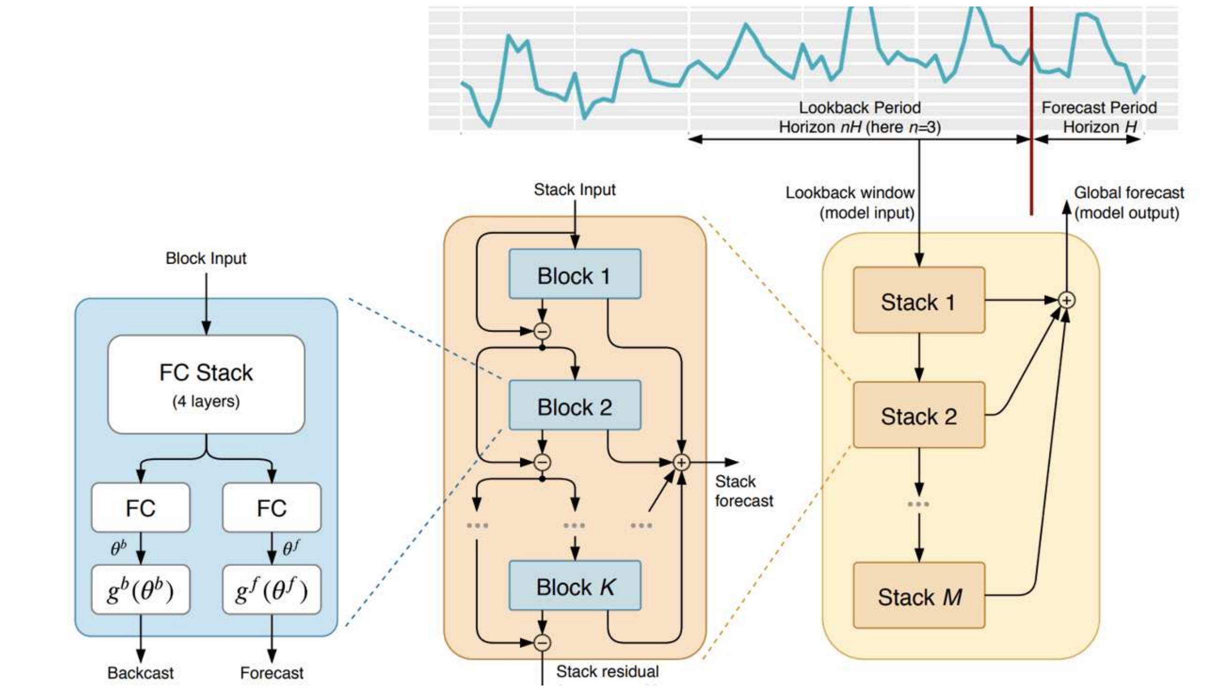


Figure 4.2: Internal architecture of NBEATS [54].

Each block's input and output are linked by a residual branch, which subtracts the backcast output of the previous block from its input, thus crafting a continuous processing chain for the input signal. This meticulous approach ensures that each block concentrates on specific aspects of the data, ultimately culminating in a more accurate and efficient prediction. The partial forecasts, created by individual blocks, each capture different patterns and components of the input data. As we move through the hierarchical structure, these partial forecasts are combined to form the final output. This creates a hierarchical decomposition of the forecasting process, where forecasts from the basic building blocks are combined to form the overall prediction.

4.6 Implementation

All models described previously were implemented using *Darts*. As mentioned in the previous chapter, *Darts'* forecasting models have different support modes for covariates. Naïve, for example, does not support covariates, while Croston's method supports future covariates but only through *Darts'* "encoder" function. Therefore, it can only use time-related covariates. LGBM can support all types of covariates, while LSTM and NBEATS only support past covariates. For this reason, the future covariates will be converted to past covariates when used on these two models.

Additionally, some of *Darts'* forecasting models, called "global" models, can be trained on multiple time series at the same time. The advantage of training on multiple series is that a single model can be exposed to more patterns occurring across all series in the training dataset, allowing to deploy models in a CL fashion. The global models used in this thesis are LGBM, LSTM and NBEATS. Therefore, Naïve and Croston's method will have to be trained on one time series at the time instead. A final comparison between the models highlighting pros and cons of each model can be seen in Table 4.1.

Model	**Pros**	**Cons**
Naïve	- Simple and computationally efficient. - Serves as a baseline for benchmarking.	- Limited ability to capture complex patterns. - Can only handle univariate series. - Statistical model
Croston's Method (TCB)	- Designed for intermittent patterns. - Effective for handling sporadic demand. - Supports future temporal-features.	- Sensitivity to data characteristics may affect performance. - Can only handle univariate series. - Statistical model
LGBM	- Global model allows CL. - Supports static, past and future covariates. - Fast and efficient. - Robust handling of categorical features. - Efficient histogram-based learning for data binning.	- Leaf-wise growth strategy may lead to overfitting in small or noisy datasets. - Requires careful hyperparameter tuning for optimal results. - Less interpretable.
LSTM	- Global model allows CL. - Supports static and past covariates. - Memory cells enable selective information retention and passing. - Effective in capturing intricate patterns, temporal dependencies and non-linear relationships in sequential data.	- Computationally demanding. - Potential for overfitting. - Requires extensive training data. - Challenging interpretation.
NBEATS	- Global model allows CL. - Supports static and past covariates. - Deep-learning architecture designed for time-series forecasting. - Non-recurrent neural network capable of capturing intricate patterns. - Produces forward and backward forecasts for enhanced interpretability. - Hierarchical structure allows capturing diverse patterns at different levels.	- May require substantial training data for optimal performance. - Computationally demanding. - Challenging interpretation.

Table 4.1: Comparison of the pros and cons of the chosen models.

4.6.1 Training

In order to train the global models, Darts first creates a dataset of inputs/outputs examples from the provided time series. There are several ways this can be done. For example, LSTM and NBEATS will build all the consecutive pairs of input/output sub-sequences (of lengths input_chunk_length and output_chunk_length) existing in the series. Figure 4.3 and Figure 4.4 visualize how this process works with one series or multiple series, respectively.

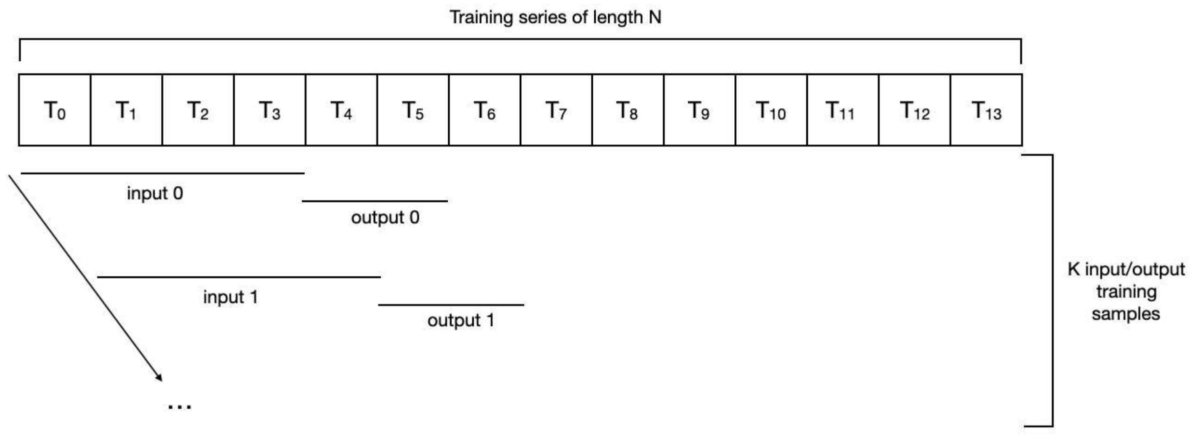

Training series of length N
T0 T1 T2 T3 T4 T5 T6 T7 T8 T9 T10 T11 T12 T13
input 0
output 0
input 1
output 1
...
K input/output training samples

Figure 4.3: Series of length N with input_chunk_length=4 and output_chunk_length=2. For such a dataset, a series of length N would result in a "training set" of K samples, where K = N - input_chunk_length - output_chunk_length + 1 [49].

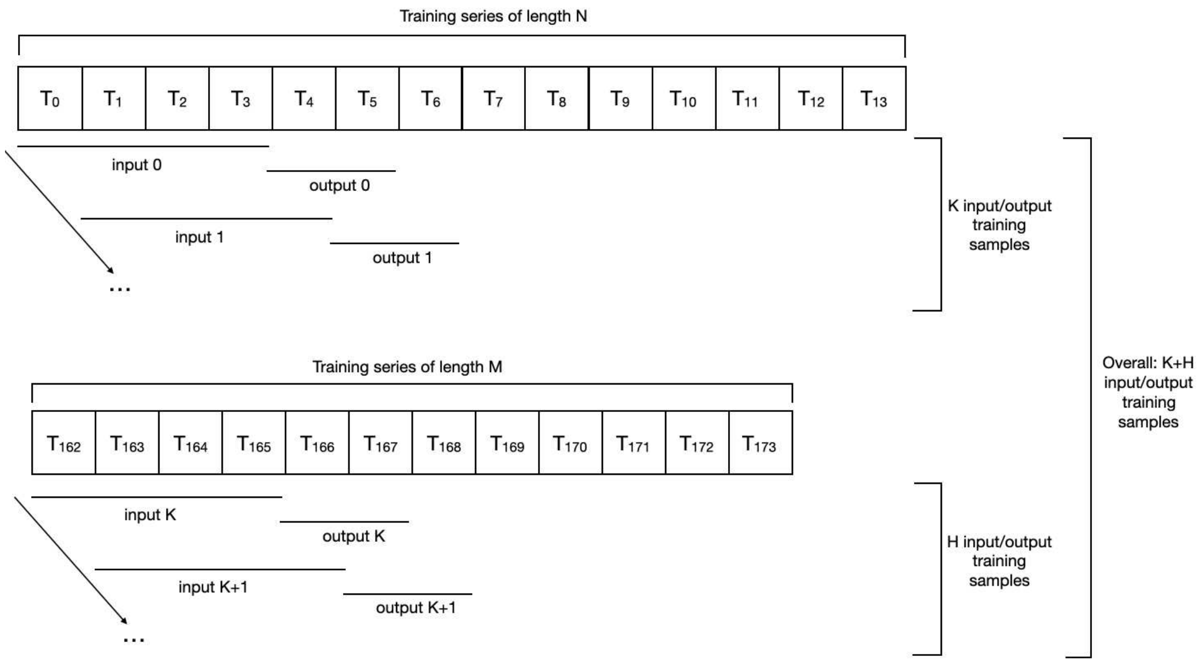

Training series of length N
T0 T1 T2 T3 T4 T5 T6 T7 T8 T9 T10 T11 T12 T13
input 0
output 0
input 1
output 1
...
K input/output training samples
Training series of length M
T162 T163 T164 T165 T166 T167 T168 T169 T170 T171 T172 T173
input K
output K
input K+1
output K+1
...
H input/output training samples
Overall: K+H input/output training samples

Figure 4.4: Dataset with two series of lengths N and M, input_chunk_length = 4 and output_chunk_length = 2. The total number of samples in the training dataset will be the union of all the training samples contained in each series [49].

On the other hand, LGBM allows to indicate which and how many lags to use as a substitute for input_chunk_length. With lags we indicate the number of past values on which we base our predictions. Moreover, we can indicate the number of future covariate and past covariate values as well.

The hyperparameter selection for all the models, except Naïve, were performed using *Optuna* [61]. This Python library simplifies the process of finding the best hyperparameter settings for a model by treating it as a Bayesian optimization problem. After defining the hyperparameter search space, it uses various algorithms to efficiently search for the optimal configuration. Moreover, it offers built-in search algorithms and supports early stopping and other popular machine learning libraries.

To implement Croston's TSB variant, we need to specify the parameters αp and αd, which are the smoothing parameters to apply on probability and demand, respectively. The optimal values found with *Optuna* are $\alpha_p = 0.2$ and $\alpha_d = 0.15$. Since they are more complex, the hyperparameters used for LGBM, LSTM and NBEATS can be found in Table 4.2, 4.3 and 4.4. However, it is important to mention that Darts' implementation of LGBM only allows to tune the number of lags to consider. Lastly, an early stopping criterion was used to avoid overfitting when training the two deep learning models, thus stopping the training when the loss has not changed by more than 0.000000005 for the past five epochs, which means that the model has stabilized.

LGBM	
Hyperparameters	**Value**
lags	[-12, -8,-4, -3, -2, -1]
lags_past_covariates	[-12,-8, -4, -3, -2]
lags_future_covariates	[-12,-8, -4,-3,-2,-1]
output_chunk_length	2

Table 4.2: Hyperparameter selection for LGBM. The number of lags indicates which past values are used for the predictions.

LSTM	
Hyperparameters	**Value**
input_chunk_length	12
output_chunk_length	2
n_rnn_layers	4
hidden_dim	25
lr	0.0006706078509496326
batch_size	1024
dropout	0.04037786017766179

Table 4.3: Hyperparameter selection for LSTM: **n_rnn_layers** is the number of layers in the RNN module, **hidden_dim** is the size for feature maps for each hidden RNN layer, **lr** is the learning rate, **batch_size** is the number of time series used in each training pass, **dropout** is the fraction of neurons affected by the regularization technique Dropout.

NBEATS	
Hyperparameters	**Value**
input_chunk_length	12
output_chunk_length	2
num_layers	6
layer_widths	512
num_blocks	1
num_stacks	30
lr	0.0002697739877476035
batch_size	1024
dropout	0.1709873191774702

Table 4.4: Hyperparameter selection for NBEATS: **num_layers** is the number of fully connected layers preceding the final forking layers in each block of every stack, **layer_widths** is the number of neurons that make up each fully connected layer in each block of every stack, **num_blocks** is the number of blocks making up every stack, **num_stacks** is the number of stacks that make up the whole model. **lr** is the learning rate, **batch_size** is the number of time series used in each training pass, **dropout** is the fraction of neurons affected by the regularization technique Dropout.

4.7 Evaluation

The dataset was split by a chosen timestamp so that each time series is split into two sets. The first set of time series is reserved for training, while the second set is used to evaluate the models. The 1/05/2023 was chosen as the timestamp to split the time series in order to ensure that the time series of the training set contained at least 70% of the data. The evaluation methods for the models primarily focus on assessing the accuracy of sales predictions for the upcoming 12 weeks remaining in the evaluation set.

4.7.1 Evaluation metrics

Three distinct evaluation metrics will be employed to assess the accuracy of our forecasts. These metrics, namely the Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Cumulative Forecast Error (CFE), have been endorsed by research in intermittent demand forecasting [62, 63], and they are defined as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i' - y_i| \tag{4.15}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i' - y_i)^2} \tag{4.16}$$

$$CFE = \sum_{i=1}^{n} (y_i' - y_i) \tag{4.17}$$

where y_i' is the forecasted value, y_i is the true value and n is the total number of values in the validation set.

MAE is defined as the average of absolute difference between forecasted values and true values. It is an unsigned, non-squared error that tells us how big of an error we can expect from the forecast on average. The lower the MAE value, the better the model is. However, since MAE has shown an affinity to zero forecast [64], it is preferable to use MSE based error metrics.

MSE provides the assurance to get an unbiased forecast and is defined as the average of squares of the error. RMSE is an extension of MSE and is defined instead as the square root of mean square error. The reason to use this version is that the RMSE value is expressed in the same units as the forecasted values, making it easier to understand compared to MSE. Because of the square term, this metric penalizes more large errors. RMSE can also be compared to MAE to determine whether the forecast contains large but infrequent errors. The larger the difference between RMSE and MAE the more inconsistent the error size is.

Lastly, the CFE is one of the key metrics used to evaluate the performance of models for intermittent time series forecasting. It's a metric that quantifies the cumulative difference between forecasted and actual values over a specified time. It provides a measure of the overall forecasting accuracy, reflecting both overestimations and underestimations, accumulated across multiple time points. In our example, we simply take the difference between our forecasted sales and the actual sales that took place. For instance, if our model predicts 3 units sold in a given week, but we sell only 2 units, the error for that week is 1 unit. This process is repeated for all weeks, and the cumulative forecast error is the sum of these individual errors. In practice, it doesn't usually matter if we will sell the product this week or next week, as long as we know how many units we will sell in the two weeks.

Another often recommended metric for intermittent time series forecasting is the Mean Absolute Scaled Error (MASE), but it was excluded because when the time series are essentially on the same scale (like in our case), the use of a scale-based metric like MASE becomes unnecessary [65]. We've also excluded commonly employed time series evaluation metrics, such as percentage-based metrics like the Mean Absolute Percentage Error (MAPE). The reason comes from their formulation, which involves dividing by the true value, resulting in division by zero due to true values often being zero.

5
Inventory Replenishment

One of the biggest challenges in modern retail and manufacturing is stock management. Overstocking on inventory by replenishing stock too early or without regard to changes in demand or seasonality can cause dead stock, which increases carrying costs by having unsellable inventory sit on shelves for too long. On the other hand, if not enough inventory is available, stock-outs may occur, resulting in missed potential sales and customer dissatisfaction. Inventory management is therefore a financial trade-off between inventory costs and stock-out costs. Most businesses use one of the following stock replenishment strategies [1]:

- **Demand replenishment:** The simplest and most straightforward restocking method. As its name suggests, this method bases replenishment on the forecasted demand, meaning that we just reorder enough stock to meet the expected demand. This means that unexpected sales will be lost.
- **Reorder point replenishment:** This method maintains consistent inventory levels to avoid stockout issues, ensuring you have enough stock to meet demand. When the stock level of an item falls below the reorder point, an automatic replenishment order is triggered to return it to the reorder point quantity. However, we may have too much stock or not enough when demand or supply chain shifts.
- **Top-off replenishment:** Using this strategy, inventory levels for a particular product are "topped off" in their respective storage locations during slower periods or down time, so that you can maintain a high inventory turnover rate without encountering stock-outs, ensuring that popular products are always available during peak times. In practice, the top-off method is susceptible to supply chain disruption and subject to vendor efficiency.
- **Periodic inventory replenishment:** When handling a large portfolio of items with different replenishment cycle lengths, using a continuous review inventory policy may cause a higher number of restock orders compared to a periodic one [5], resulting in higher shipment costs and inefficiencies. When using this strategy, inventory is restocked at fixed intervals, such as every 6 weeks or every 3 months, regardless of seasonality or how low stock levels have dropped in the meantime. For this reason, it is generally recommended to businesses with predictable customer demand and large warehouse capacity.

For most retailers, inventory management systems take a fixed, rule-based approach to forecast and replenishment orders management, which leave them open to unexpected fluctuations in the demand or in the supply chain. The objective of this thesis is to build a dynamic replenishment policy that will minimize stock-outs, while also trying to avoid unnecessary overstocks. For this reason, the advantages of these strategies have been combined into the Periodic order-up-to policy described in [5], which was modified to account for intermittent sales data. In the following sections we will discuss the main components of this replenishment method.

5.1 Reorder Point

As discussed before, this method encourages retailers to store a minimum amount of inventory, below which a replenishment order is triggered. This ensures that we have enough stock to meet demand during lead time. However, since we are using a Periodic review policy, we also have to consider the review period in addition to the lead time. This period indicates how often we check the inventory to see if we have to trigger a replenishment order. In our case this period will be 7 days long. If there were no uncertainty, i.e., if future demand was perfectly known and supply was perfectly reliable, the reorder point would simply be equal to the average demand during the lead time and review period. However, because of uncertainties, in practice we have:

$$\begin{aligned} Reorder\ point &= \mu_{LD+R} + Safety\ Stock \\ &= \mu_{day} * (LD + R) + Safety\ Stock \end{aligned} \tag{5.1}$$

In this equation:

- μ_{LD+R} is the average demand during the lead time and review period.
- μ_{day} is the average daily demand, i.e., how many units of a product are sold per day.
- LD is the average lead time, which is the number of days a store must wait to receive new stock after placing a replenishment order.
- R is the weekly review period.

The reorder point can be described as the point at which you need to order a product before you start using your safety stock [66]. Safety stock is simply extra inventory held by a retailer or a manufacturer in case demand increases unexpectedly. This means it's additional stock above the desired inventory level that you would usually hold for day-to-day operations.

One of the reasons that retailers and manufacturers implement a safety stock strategy is to protect against three external factors over which you have little to no control:

- Changes in consumer demand
- Incorrect forecasts
- Variability in lead times

Moreover, when the lead times to restock stores are large, it can act as a buffer, avoiding stock-outs while we wait for the new stock to physically arrive. Even if the forecasts are 100% accurate, if you can't send the required stock in time you will inevitably lose some of the forecasted sales. Since we are going to use a Periodic review policy, this is a crucial issue. However, it can be addressed by preparing the correct amount of safety stock, which will be described in the next section.

5.2 Cycle Service Level

Before starting, it is important to remember that stock-outs will always occur, no matter how much you want to prevent them. The safety stock formula is there to prevent the majority of stock-outs, but can't prevent all of them, as this would lead inevitably to overstocking. The Cycle Service Level (CSL) plays a key factor when calculating safety stock. It is the probability that the amount of inventory on hand during the lead time is sufficient to meet the expected demand, i.e., is the probability that there will not be a stock-out within a replenishment cycle [66]. This is frequently used as a performance metric where the inventory policy is designed to minimize cost to achieve an expected service level, and is described as:

$$CSL = 1 - P[Stockout] = 1 - P[X > k] = P[X \leq k] \tag{5.2}$$

where X is the distribution of the customer demand and k is the "service factor".

In the context of inventory management, the service factor is often expressed as a safety stock multiplier. It determines the number of standard deviations needed to achieve the desired service level. To calculate k, it is commonly used the inverse cumulative normal distribution to obtain the Z-score corresponding to the service level desired. For example, If we are trying to maintain a service level of 95%, the service factor k will be 1.64. However, as mentioned in Section 3.2.1, the distribution of our time series follows a Poisson distribution. Hence, we will use the corresponding inverse cumulative distribution function to obtain the appropriate service factor.

The specific formula for safety stock calculation based on the service factor can vary depending on the method used, but it generally involves multiplying the standard deviation of the demand by the service factor k. In our case, the safety stock will be calculated as:

$$\begin{aligned} Safety\ Stock &= k * \sigma_{LD+R} \\ &= k * \sigma_{day} * \sqrt{LD + R} \end{aligned} \tag{5.3}$$

where σ_{LD+R} is the standard deviation of the demand during the lead time and review period and σ_{day} is the daily standard deviation of the demand. Hence, the reorder point is equal to:

$$Reorder\ point = \mu_{LD+R} + k * \sigma_{LD+R} \tag{5.4}$$

Having zero safety stock would result in a service level of 50% [67], while a 100% service level would mean you always have stock, but it is usually not the best solution, as shown in Figure 5.1.

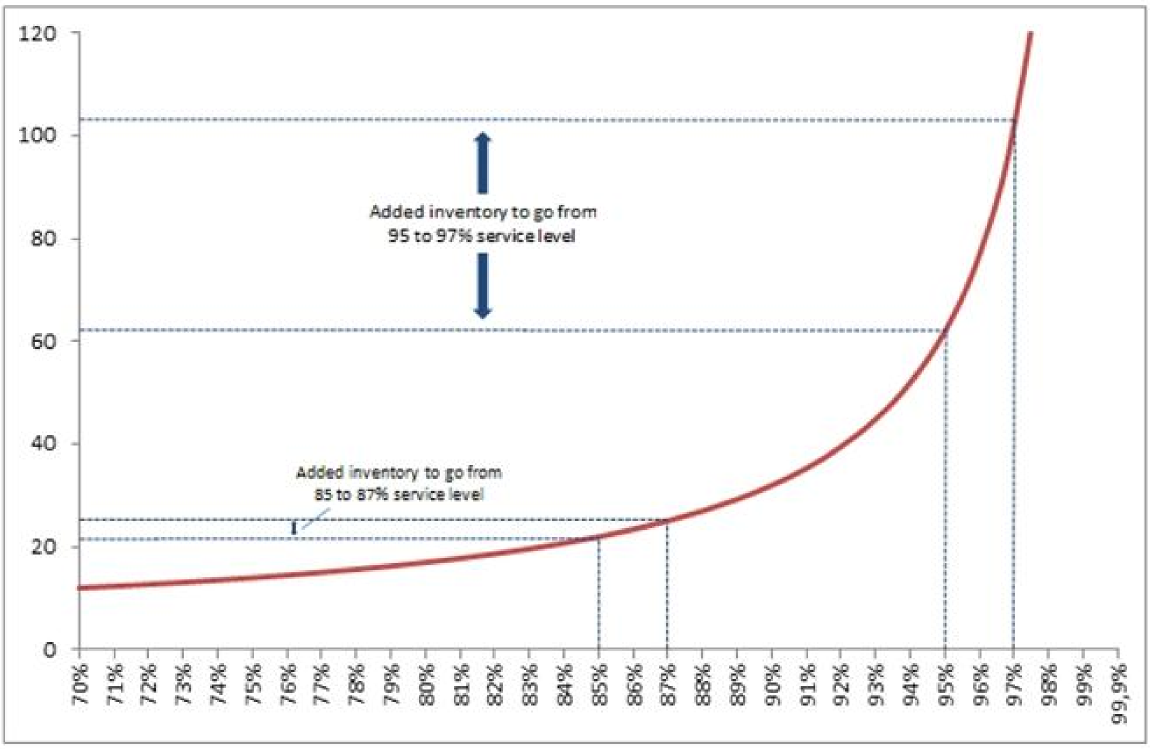


Figure 5.1: Relationship between the service level and the inventory level [68].

The key consists in finding the right balance between the cost of holding inventory and a service level that avoids most of the stock-outs. Targeting high service levels, typically above 95%, is the norm in most retail sectors [68]. However, achieving higher service levels is a classic case of diminishing returns, where each extra marginal effort, i.e., extra inventory, yields lower returns, i.e., a smaller fraction of stock-outs being eliminated.

5.3 Implementation

In this section we describe the implementation of the Periodic Review Policy: Order-Up-To-Level (s, Q), where s is the reorder point and Q is the quantity to order. At each review period R, the level of the inventory on hand IOH and in transit IT is reviewed. We need to consider the inventory in transit as well because the lead time of our dataset varies between 5 and 9 days, meaning that some restock orders might still be active when we reach the next review period. This is done to avoid resending stock that is about to arrive at the store in the next few days. Therefore, if at the review period the reorder point is equal or lower than the sum of IOH and IT, we don't reorder anything. Otherwise, the quantity to reorder Q will be:

$$Q = s - (IOH + IT) \tag{5.5}$$

The default implementation expects a static reorder point that never changes. This is inapplicable in an intermittent demand scenario, as we risk overstocking during periods with no sales and understocking during active periods. Therefore, it was modified to change the reorder point every week based on the forecasts of the next two weeks. To obtain the reorder point of the current week, we first calculate the average daily demand of each store-SKU combination by dividing the sum of the next two weekly forecasts by 14. We then calculate for each combination the corresponding service factor using the inverse cumulative distribution function of the Poisson provided by the *Scipy* library. Since we are using this distribution, we can also assume that the average daily demand is equal to the daily variance.

Therefore, the μ_{LD+R} and σ_{LD+R} are calculated as follows:

$$\mu_{LD+R} = \mu_{day} * (LD + R) \tag{5.6}$$

$$\sigma_{LD+R} = \sqrt{\mu_{day}} * \sqrt{LD + R} = \sigma_{day} * \sqrt{LD + R} \tag{5.7}$$

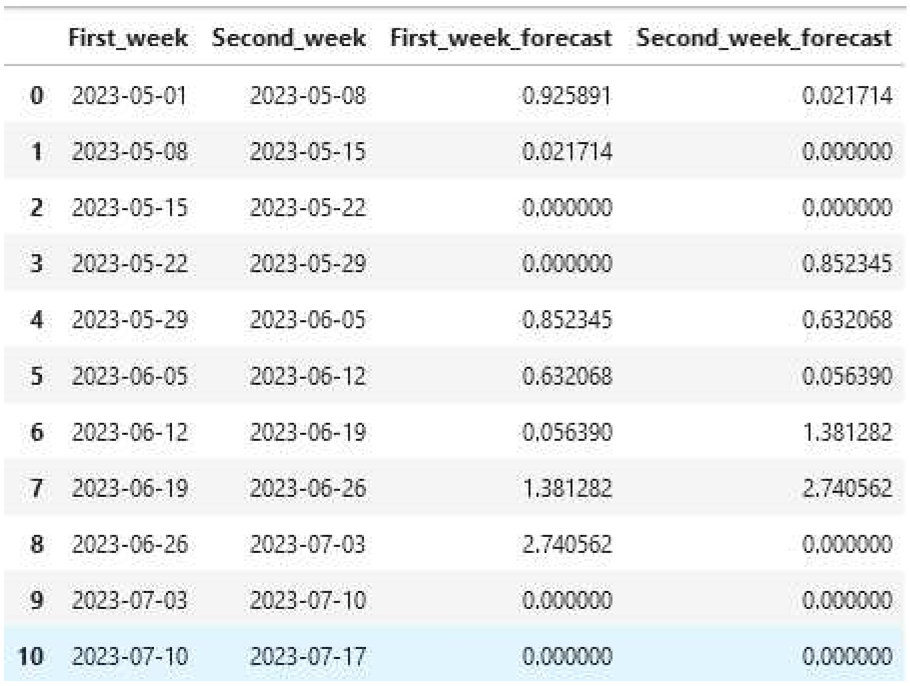
The reorder point for each SKU of the current week is then:

$$s = \mu_{LD+R} + k * \sigma_{LD+R} \tag{5.8}$$

As requested by the hosting company, when the reorder point calculated is lower than the quantity to display, i.e, the minimum quantity to always have in store, we set the reorder point to be equal to the quantity to display instead. Therefore, when the reorder point would be 0 because we expect no sales in the following two weeks, we set this minimum quantity which is equal to 1 or 2 depending on the SKU.

5.3.1 Simulations

In order to evaluate the effectiveness of this algorithm and the best CSL to use, we implemented 5 simulations using a CSL of 95%, 97%, 98%, 99% and 99.9%, respectively. The simulations last for 77 days using the 12 weekly values forecasted by the best model among the ones described in the previous Chapter. For each store-SKU combination, these weekly forecasts were rearranged consecutively as shown in Figure 5.2, which was necessary to calculate the reorder point of each week.



	First_week	Second_week	First_week_forecast	Second_week_forecast
0	2023-05-01	2023-05-08	0.925891	0.021714
1	2023-05-08	2023-05-15	0.021714	0.000000
2	2023-05-15	2023-05-22	0.000000	0.000000
3	2023-05-22	2023-05-29	0.000000	0.852345
4	2023-05-29	2023-06-05	0.852345	0.632068
5	2023-06-05	2023-06-12	0.632068	0.056390
6	2023-06-12	2023-06-19	0.056390	1.381282
7	2023-06-19	2023-06-26	1.381282	2.740562
8	2023-06-26	2023-07-03	2.740562	0.000000
9	2023-07-03	2023-07-10	0.000000	0.000000
10	2023-07-10	2023-07-17	0.000000	0.000000

Figure 5.2: Rearrangement of the forecasted values to simulate a bi-weekly forecast every week for each store-SKU combination.

After that, we calculate all the components of the 11 reorder points for every store-SKU combination as described previously. As mentioned before in Section 3.2, other variables used for the simulation include the daily sales of each store-SKU combination and the quantity of stock of each SKU stored in a store at the end of each week. The first will be used to simulate how the sales reduce the inventory of a given SKU during the simulation period, while the latter will be used to calculate the average inventory, described in [69] as the mean between the inventory level at the start point and at the end point of a chosen interval.

The historical average inventory will be compared with the ones obtained during the simulations to assess the risk of overstocking. The simulations will automatically handle the quantity to order each week for each store-SKU combination and the variations in the inventory due to sales and restock orders arrivals. To simulate we are already running at full capacity from the beginning, the initial stock of each combination is set to half of the initial reorder point, rounded up. It will be also handled the case in which two restock orders are active at once for stores with long lead times. Lastly, we will record the number of days a SKU was in stock-out and the number of sales we would have lost because of it. The results of these simulations can be seen in the following chapter.

6 Results

This chapter presents how the forecasting models perform and their results. Next, we will use the predictions of the best model to implement the simulations of the replenishment algorithm to assess the optimal CSL to use.

6.1 Forecasting Models

This section presents how the 5 chosen models perform compared to each other. Figures 6.1, 6.2, 6.3, 6.4 and 6.5 show the corresponding MAE, RMSE and CFE scores for the predictions of the 2077 store-SKU combinations. All the models were trained with the parameter setups presented in Section 4.6.1 and evaluated on the 2077 time series using the data splitting method discussed in Section 4.7.

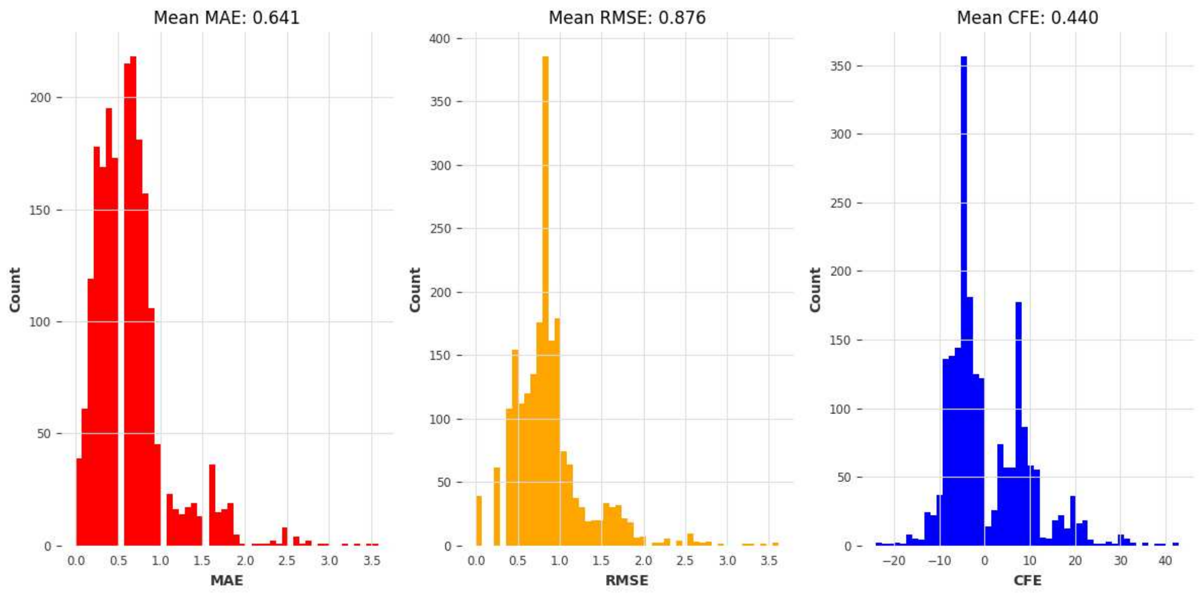


Figure 6.1: MAE, RMSE and CFE scores of the Naïve model used as baseline.

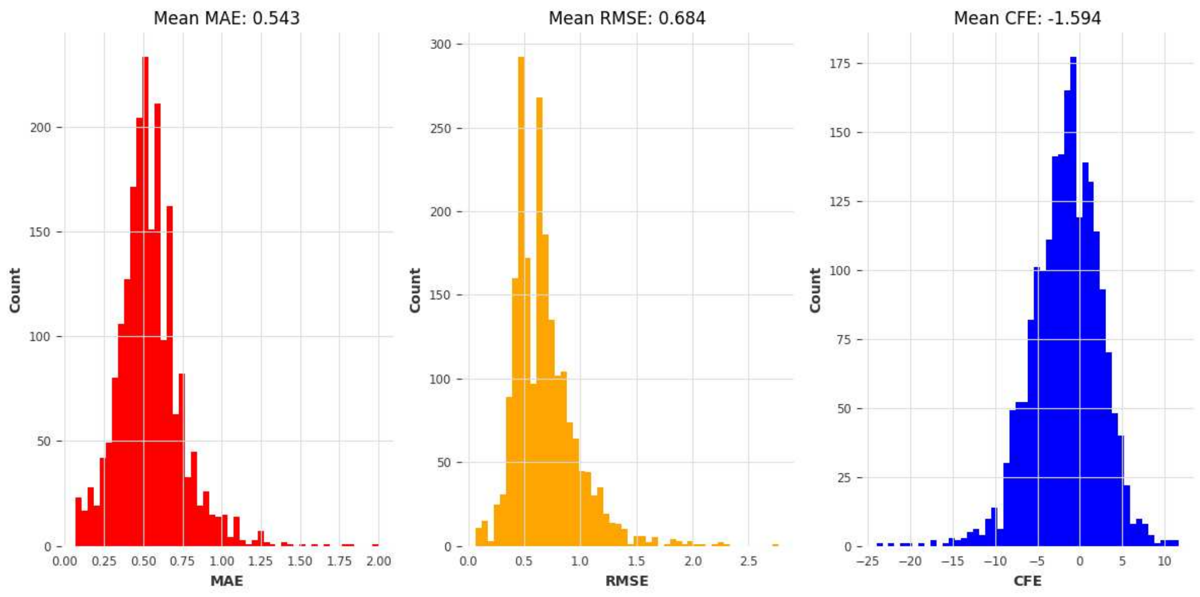


Figure 6.2: MAE, RMSE and CFE scores of Croston's method.

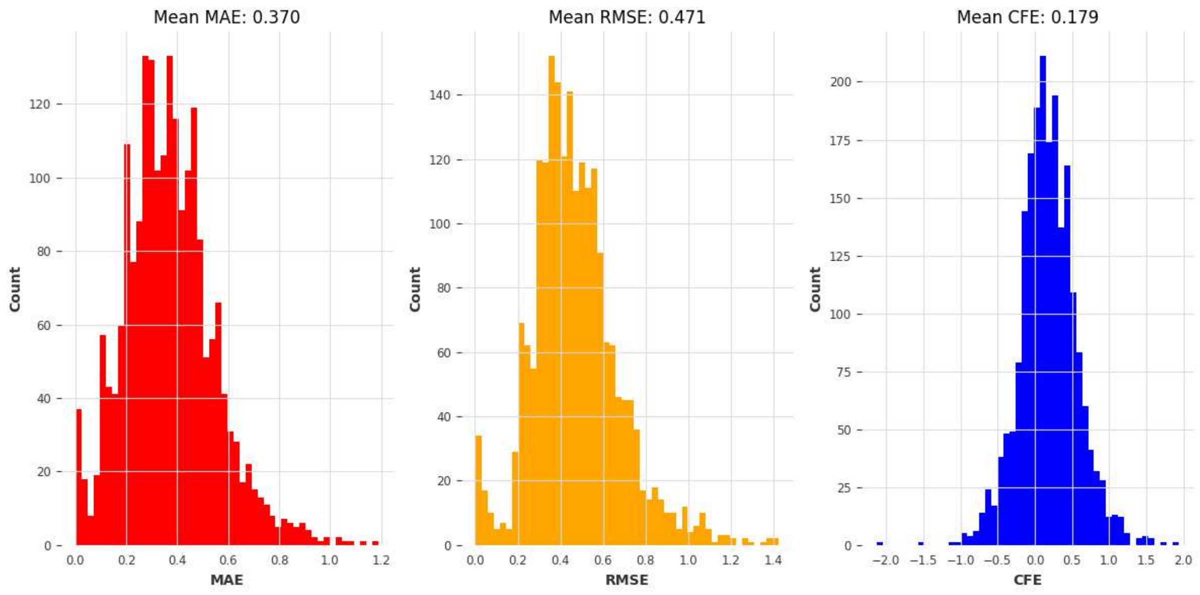


Figure 6.3: MAE, RMSE and CFE scores of the LGBM model.

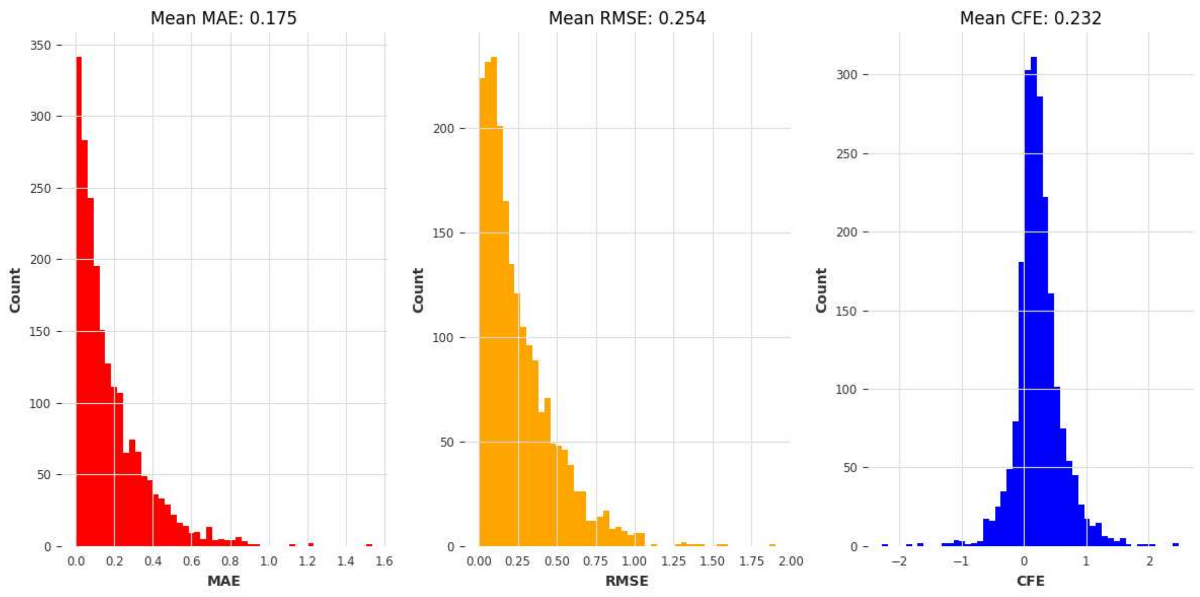


Figure 6.4: MAE, RMSE and CFE scores of the LSTM model.

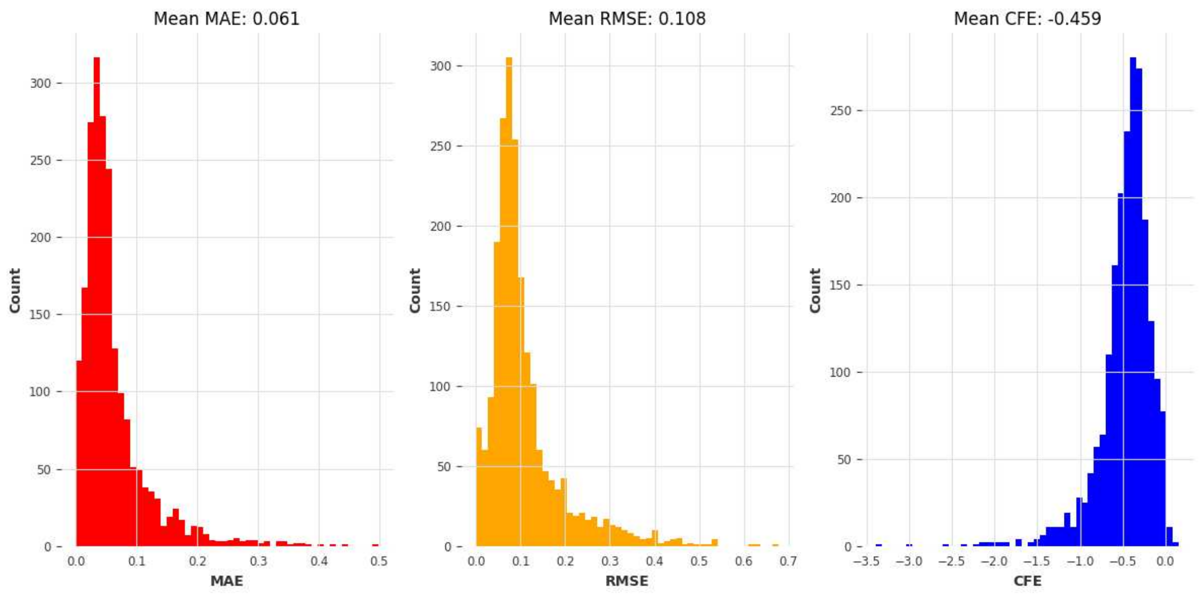


Figure 6.5: MAE, RMSE and CFE scores of the NBEATS model.

Table 6.1 shows a summary of the results obtained by the models. As we can see, NBEATS clearly outperforms the other methods on the MAE and RMSE, with an improvement of 90.48% and 87.67% compared to the baseline. Instead, the best CFE overall was obtained by LGBM, with an improvement of 59.31% compared to the baseline. Additionally, all models except NBEATS and Croston's method have positive CFE scores, indicating that these three models generally have a positive bias. Since NBEATS' average CFE score is similar to the one of the baseline, we can declare it the best model. However, because its CFE is negative, the model tends to slightly underestimate the number of sales. Therefore, we must keep this in mind when choosing the best CSL for the replenishment algorithm.

Model	MAE	RMSE	CFE
Naïve	0.641	0.876	0.440
Croston's Method	0.543 (+15.28%)	0.684 (+21.91%)	-1.594 (-262.27%)
LGBM	0.370 (+42.27%)	0.471 (+46.23%)	**0.179 (+59.31%)**
LSTM	0.175 (+72.69%)	0.254 (+71%)	0.232 (+47.27%)
NBEATS	**0.061 (+90.48%)**	**0.108 (+87.67%)**	-0.459 (-4.31%)

Table 6.1: Results of the models evaluated on the validation set. The best result for each metric is marked in bold. In parenthesis there are the percentage improvements compared to the Naïve model.

Figures 6.6, 6.7, 6.8, 6.9, and 6.10 show two examples of how each model forecasts.

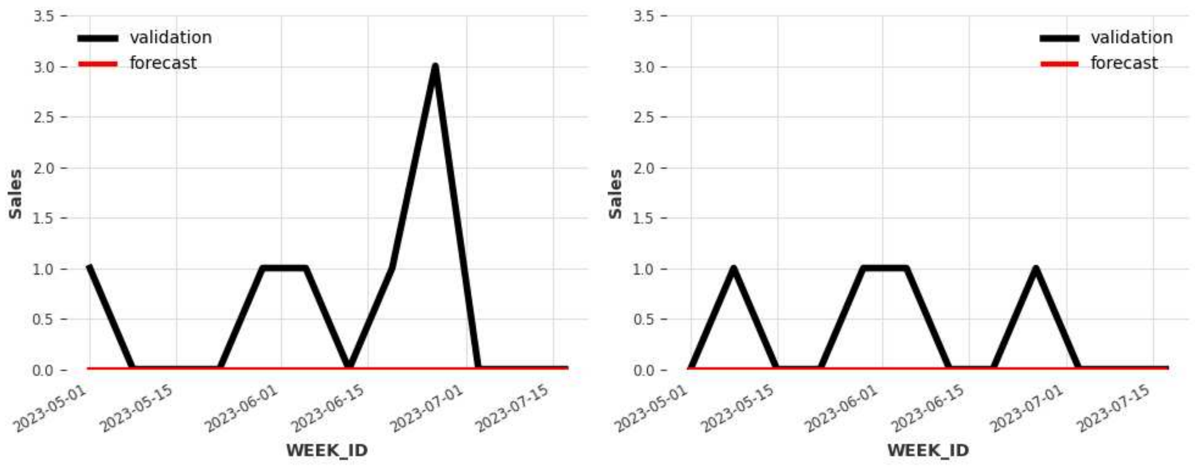


Figure 6.6: Naïve (MAE=0.641, RMSE=0.876, CFE=0.440).

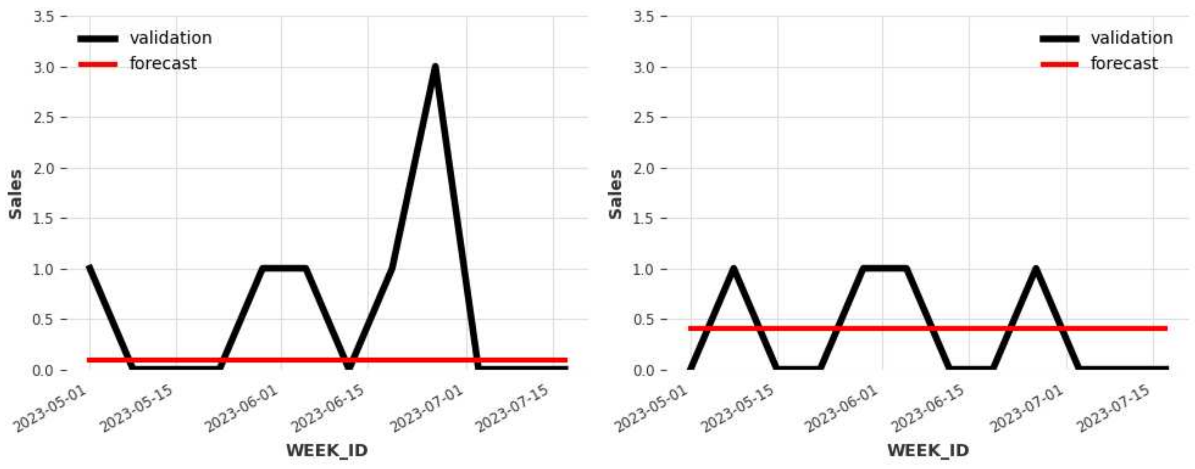


Figure 6.7: Croston's method (MAE=0.543, RMSE=0.684, CFE=-1.594).

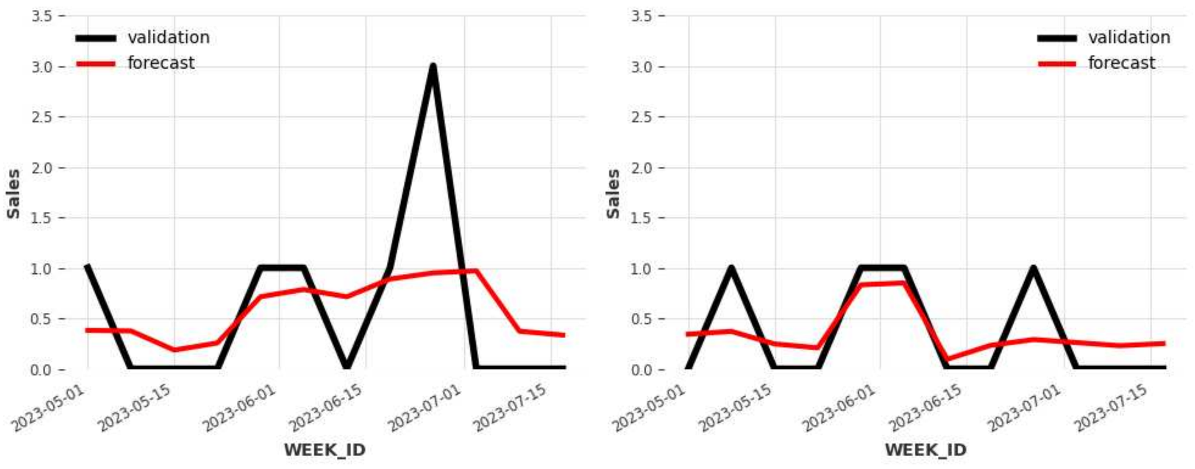


Figure 6.8: LGBM (MAE=0.370, RMSE=0.471, CFE=0.179).

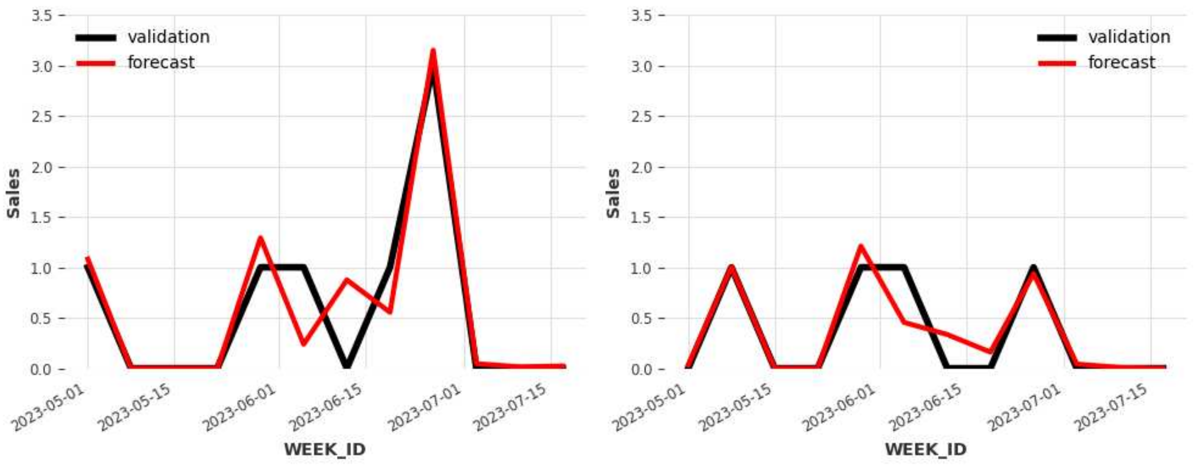


Figure 6.9: LSTM (MAE=0.175, RMSE=0.254, CFE=0.232).

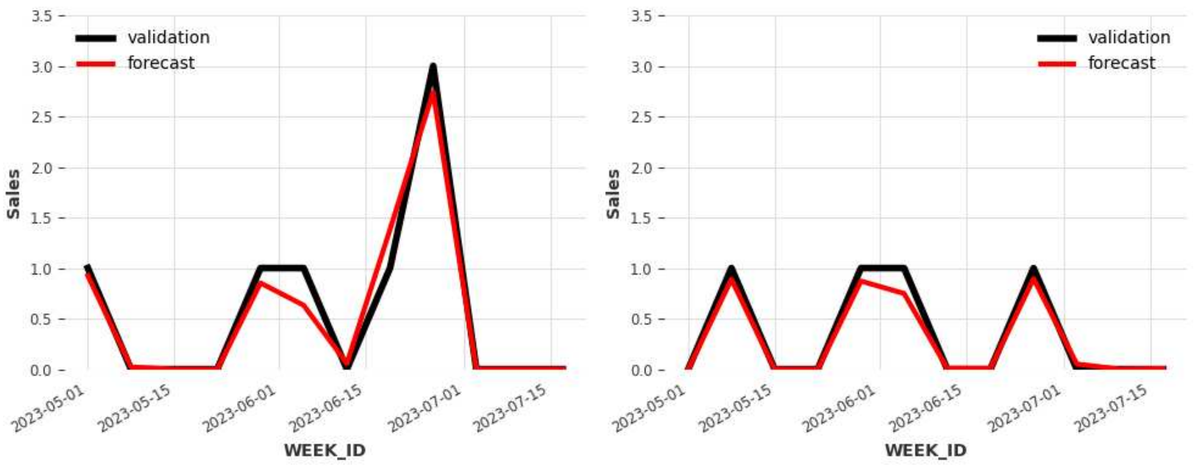


Figure 6.10: NBEATS (MAE=0.061, RMSE=0.108, CFE=-0.459).

As expected, Naïve continuously predicts zero because the last data point of the training set was zero as well. Croston's method instead tends to highly underestimate the sales by predicting close to zero, failing to produce consistent forecasts. Even though it has the best CFE score, LGBM also struggles to follow the sales, predicting correctly only occasionally. It is only with the deep learning models that we start to have reliable predictions. However, we can see that LTMS is prone to overestimation whereas NBEATS tends to slightly underestimate. Despite this, its predictions are more consistent overall, making it rightfully the best model.

6.2 Replenishment Simulations

This section presents instead how the replenishment algorithm, implemented using the forecasts of the NBEATS model, performs using 5 different CSLs, which directly affect the size of the safety stock. In Figure 6.11, 6.12, 6.13, 6.14 and 6.15 we can see an example of how the algorithm reacts to sales and handles inventory and restock orders using a CSL of 95%, 97%, 98%, 99% and 99.9% respectively.

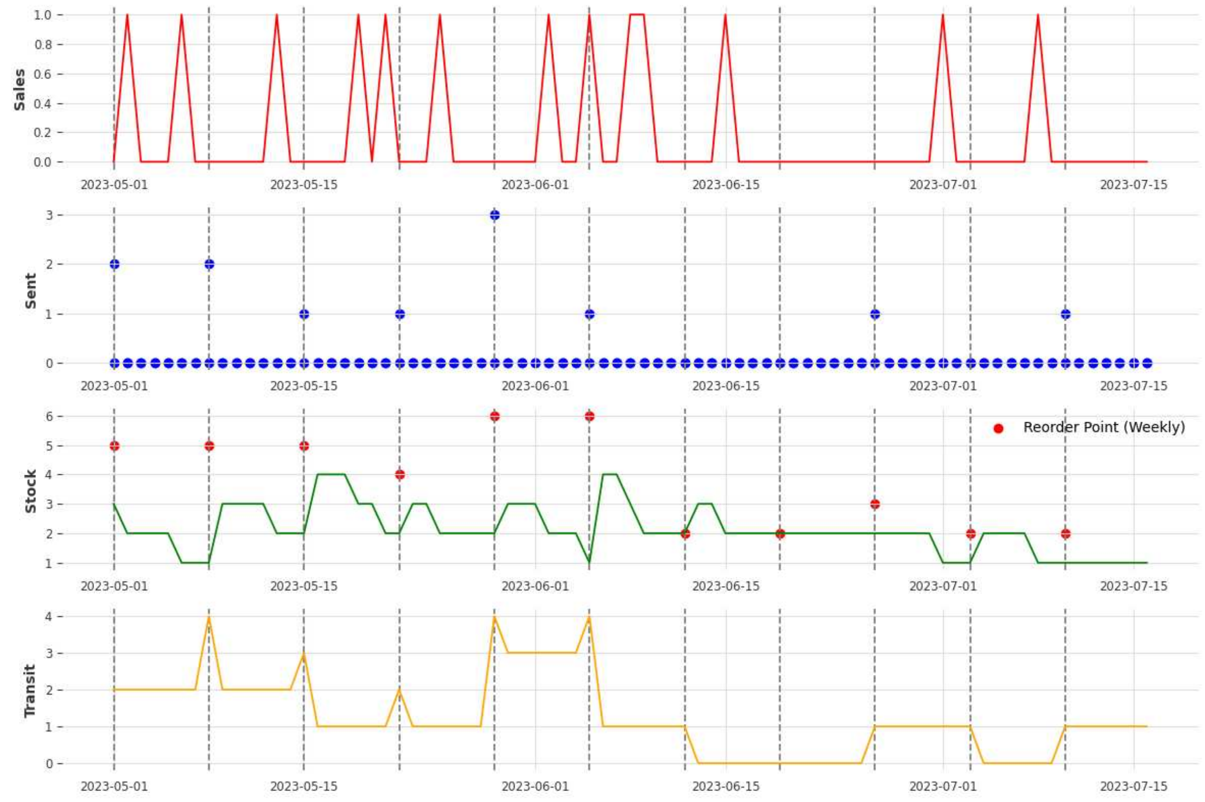


Figure 6.11: Simulation using a 95% CSL. In red we have the daily sales from 1/05/2023 to 17/07/2023 of a specific store-SKU. In blue we have the quantity ordered at each review period. In green we have the daily inventory on hand while the red dots are the weekly reorder points. Lastly, in yellow we have the stock currently in transit while the vertical lines indicate the review period interval.

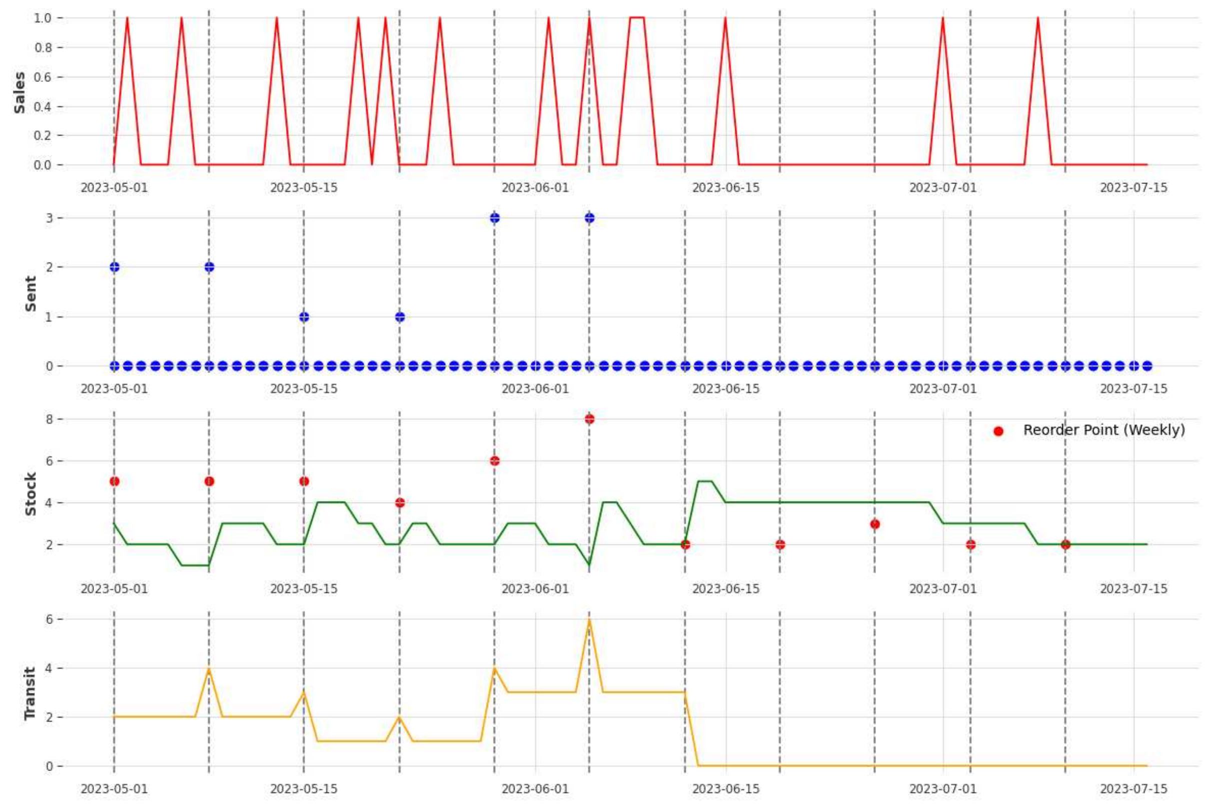


Figure 6.12: Simulation using a 97% CSL.

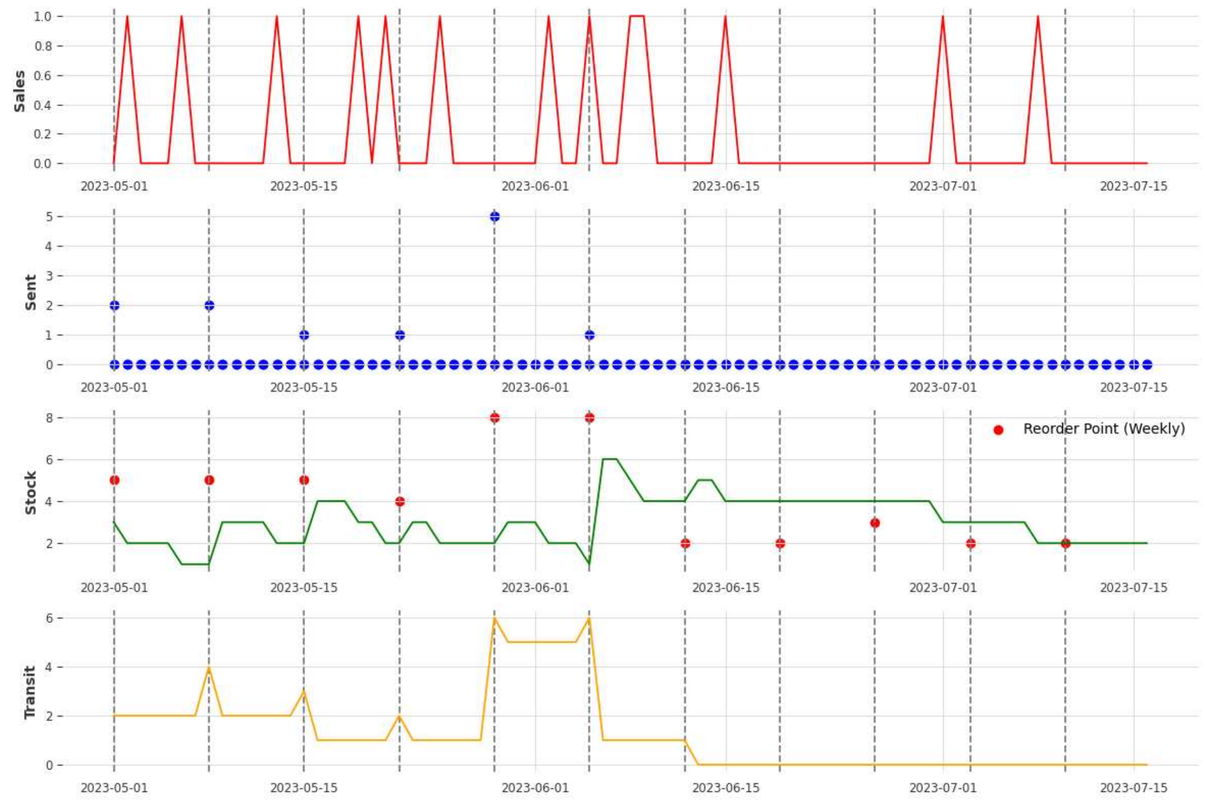


Figure 6.13: Simulation using a 98% CSL.

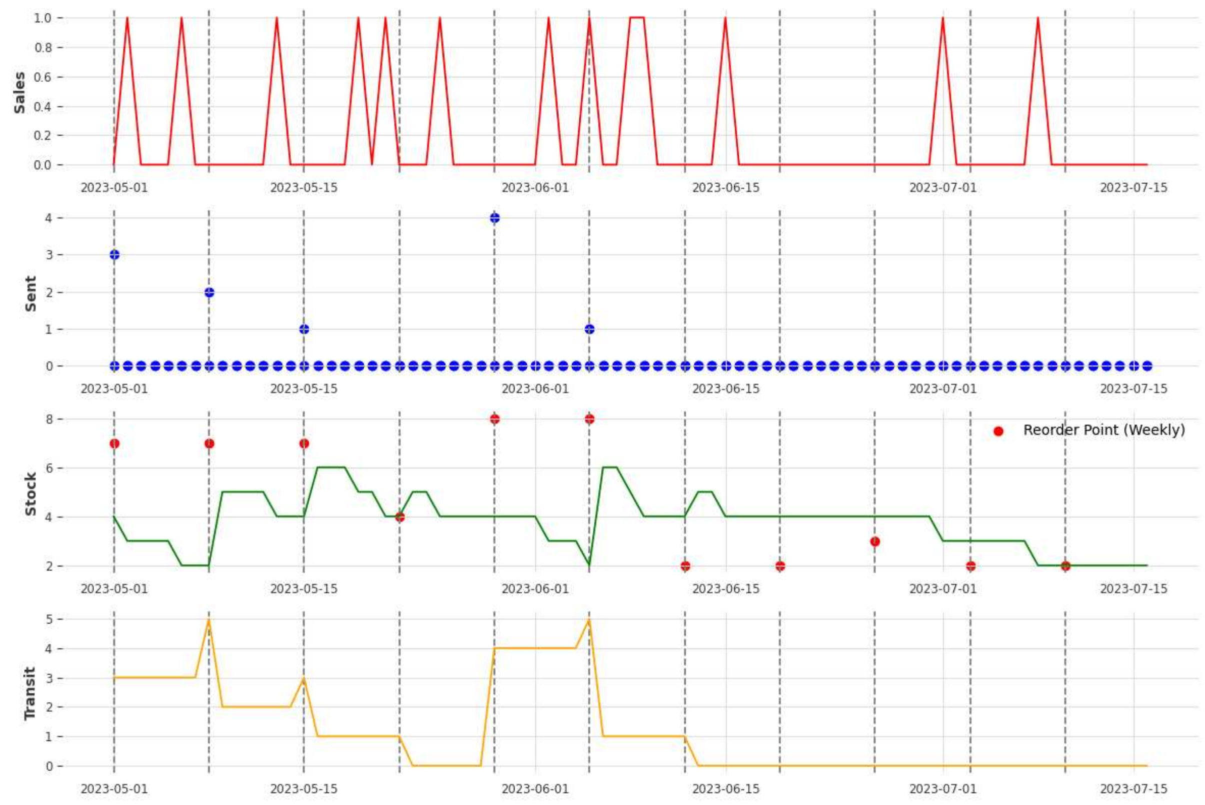


Figure 6.14: Simulation using a 99% CSL.

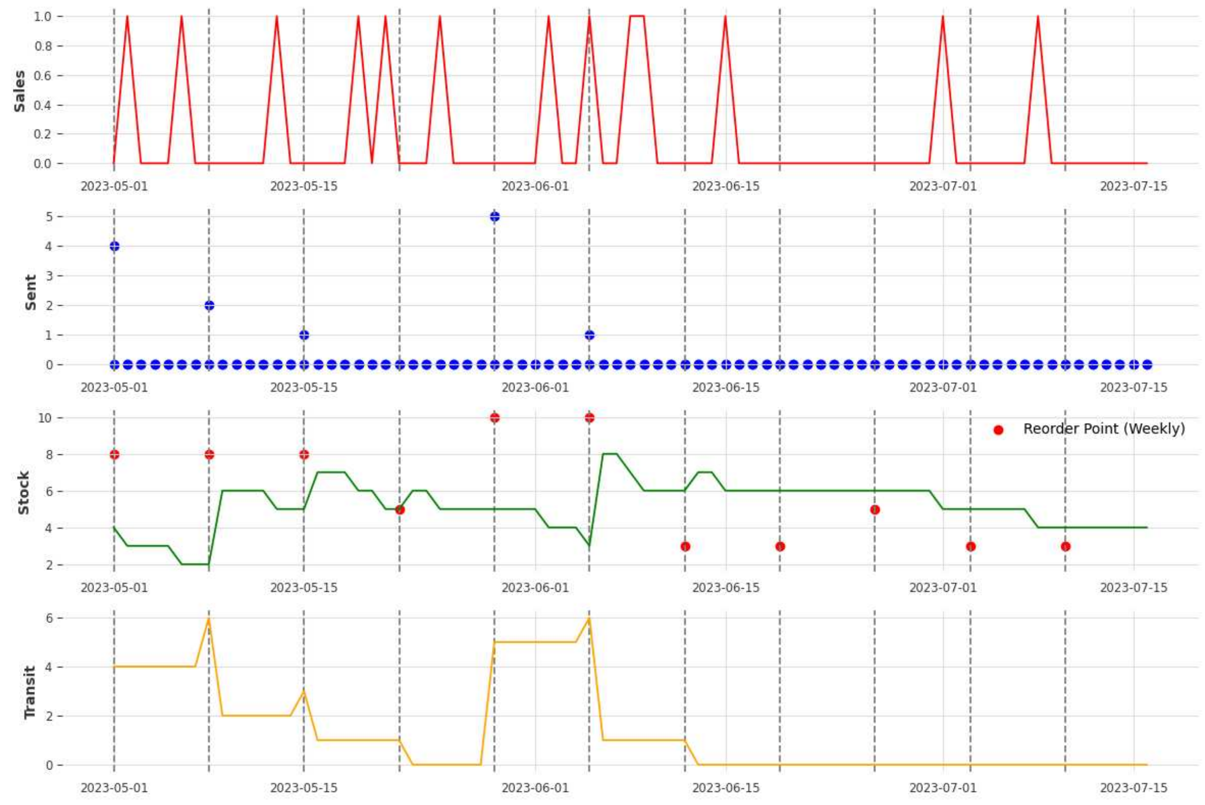


Figure 6.15: Simulation using a 99.9% CSL.

Table 6.2 shows the results obtained by the simulations regarding the number of days with stock-outs, the number of sales lost and the average inventory during the first half (from 1/05/2023 to 5/06/2023), the second half (from 5/06/2023 to 17/07/2023) and the full simulation period.

	CSL 95%	**CSL 97%**	**CSL 98%**	**CSL 99%**	**CSL 99.9%**
Days without stock	5072	4757	4695	4486	868
Days without stock per store-SKU	2.441	2.290	2.260	2.159	0.417
Sales lost	43	37	36	33	8
Sales lost per store-SKU	0.020	0.017	0.017	0.015	0.003
Average Inventory (First half)	2474	2517	2552	2722	4155
Average Inventory (Second half)	2607	2743	2843	3260	5437
Average Inventory (Full period)	2543	2642	2718	3000	4592
Average Inventory per store (First half)	17.546	17.851	18.102	19.304	29.471
Average Inventory per store (Second half)	18.492	19.457	20.166	23.124	38.563
Average Inventory per store (Full period)	18.039	18.741	19.276	21.280	32.567

Table 6.2: Summary of the results obtained by the simulations using a CSL of 95%, 97%, 98%, 99% and 99.9%.

For comparison, the actual average inventories during these periods were:

- Average Inventory (First half) = 2992
- Average Inventory (Second half) = 3520
- Average Inventory (Full period) = 3445
- Average Inventory per store (First half) = 21.219
- Average Inventory per store (Second half) = 24.964
- Average Inventory per store (Full period) = 24.432

The results obtained are optimal since the number of days without stock per store-SKU are, on average, below 3 days, regardless of the CSL used. Moreover, the number of sales lost is also quite low compared to the number of articles we have considered for this thesis. As expected, the greatest reduction of stock-outs is achieved using the highest CSL, but at the cost of severely overstocking each store, as we can see in Figure 6.15 and by considering the fact it greatly surpasses even the historical inventory levels reached by the hosting company. Therefore, it is better to avoid using it.

We can also safely avoid the 95% as it is the one that provides the least reduction to stock-outs. However, the difference between 97% and 98% on stock-outs is minimal, but since the NBEATS model we are using tends to slightly underestimate the forecasts, the extra inventory of the 98% might actually be beneficial, considering we are still below the historical inventory levels. However, as shown in Figure 6.16, it is not worth it reaching 99%, as we risk ending up with too much stock on hand right before a period with no sales starts, thus creating dead stock. Therefore, we could assume that a CSL of 98% is the optimal solution to minimize stock-outs while also avoiding unnecessary overstocking.

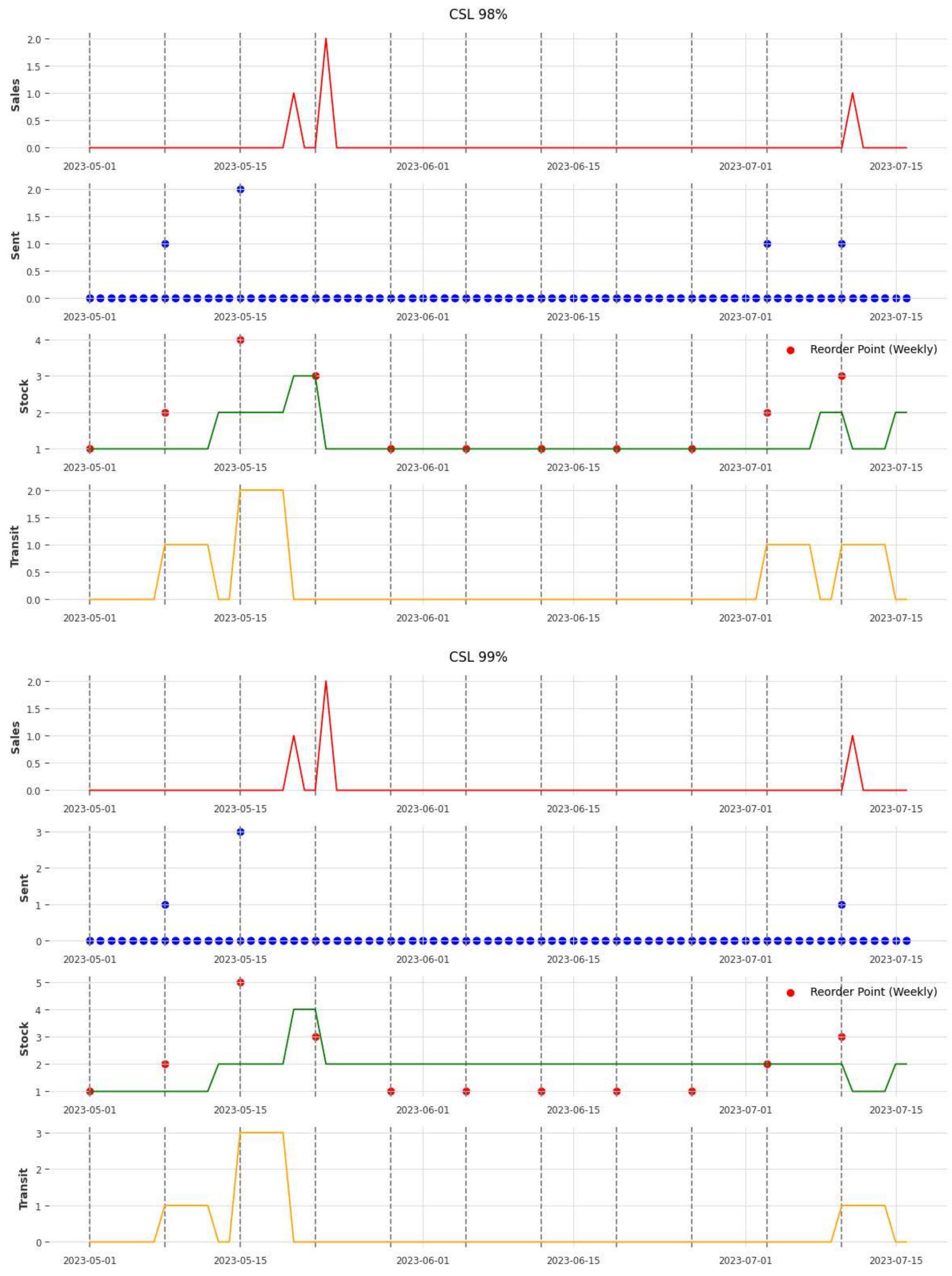


Figure 6.16: Comparison between 98% and 99% CSL. We can notice that using a CSL of 99% leads to storing an extra stock that remains unutilized for over a month. As discussed in Section 5.3, during empty periods the reorder point is simply equal to the quantity to display.

7 Conclusion

In conclusion, this thesis has delved into the intricate realm of fashion retail forecasting, addressing the challenges posed by intermittent time series data and by stock management, resulting in a replenishment algorithm that minimizes stock-outs. We discussed the evolution of forecasting methodologies, from classical methods based on mathematical and statistical models to modern heuristic methods using ML and deep learning techniques, highlighting the advantages that these techniques provide compared to statistical methods. Through the application of advanced forecasting models designed to accommodate the intermittent nature of the time series data, such as Croston's method, LGBM, LSTM and NBEATS, this thesis attempted to improve the accuracy and reliability of predictions of the volatile fashion products, thereby increasing the efficacy of decision-making processes in the dynamic fashion retail sector. Despite being designed to handle intermittent time series, Croston's method doesn't come even close to the performances of the other three models, showing the limits of statistical models compared to ML and deep learning techniques. However, LGBM showed unremarkable results as well, probably due to *Darts* not allowing the customisation of the hyperparameter necessary for this model to shine. LSTM and NBEATS showed similar results, with the latter having a little edge thanks to its structure designed for time series forecasting, allowing it to predict with great accuracy even intermittent time series.

We then discussed the various stock replenishment strategies used in this field, the concept of reorder point and CSL, and the advantages of having a safety stock to protect against the unexpected fluctuation in the demand or in the lead time. To implement the replenishment algorithm, we have then used the forecasts of NBEATS to power the Periodic Review Policy: Order-Up-To-Level, which was modified to account for the sporadic nature of intermittent demand. Lastly, through the implemented simulations, we have determined 98% to be the optimal CSL to minimize stock-outs while also avoiding unnecessary overstocking.

Various improvements could still be made in the future. New covariates like price, holidays, and promotions could be added to expand the information used by the forecasting models. Different pre-processing methods like clustering encoding could be also introduced. Lastly, to address NBEATS' forecast underestimation, future research could be conducted into alternative forecasting models like Temporal Convolutional Networks [47], Temporal Fusion Transformers [48] and other transformer-based models that have recently emerged like Channel-Aligned Robust Dual Transformer [70].

References

[1] J. Liliedahl, Inventory Replenishment: Methods and Policies to Increase Your Inventory's Productivity, Inventory Planner, 2022, [Online]. Available: https://www.inventory-planner.com/what-is-inventory-replenishment/

[2] Z.-L. Sun, T.-M. Choi, K.-F. Au, Y. Yu, Sales forecasting using extreme learning machine with applications in fashion retailing, Decision Support Systems 46 (1), 411–419, 2008. https://doi.org/10.1016/j.dss.2008.07.009

[3] D. Corsten, T.W. Gruen, Stock-outs cause walkouts, Harvard Business Review 82 (5) 26–28, 2004.

[4] S. Minner, G.P. Kiesmuller, Dynamic Product Aquisition in Closed Loop Supply Chains, International Journal of Production Research, 50, pp. 2836-2851, 2012.

[5] Samir Saci, Inventory Management for Retail — Periodic Review Policy, Jan 25, 2022, [Online]. Available: https://www.samirsaci.com/inventory-management-for-retail-periodic-review-policy/

[6] G.S. Frings, Fashion: From Concept to Consumer, Pearson Education, 2005.

[7] T. Sekozawa , H. Mitsuashi, Y. Ozawa, One-to-One recommendation system in apparel online shopping. Electronics and Communications in Japan, 94 (1), pp. 51-60. 2011.

[8] M. Marufuzzaman, K.B. Ahsan, K. Xing, Supplier selection and evaluation method using Analytical Hierarchy Process (AHP): a case study on an apparel manufacturing organization. Int. J. Value Chain Management, 3 (2), pp.224–240. 2009.

[9] D. Rayman, D.J. Burns, C.N. Nelson, Apparel product quality: its nature and measurement. Journal of Global Academy of Marketing Science, 21 (1), pp.66-75. 2011.

[10] C. Giri, Y. Chen, Deep Learning for Demand Forecasting in the Fashion and Apparel Retail Industry. Forecasting, 4(2):565-581, 2022. https://doi.org/10.3390/forecast4020031

[11] S. D'Amico, L. Giustiniano, M.E. Nenni, L. Pirolo, Product Lifecycle Management as a tool to create value in the fashion system. International Journal of Engineering Business Management, 2013.

[12] F. De Felice, A. Petrillo, C. Autorino, Key success factors for organizational innovation in the fashion industry. International Journal of Engineering Business Management, 2013.

[13] M. E. Nenni, L. Giustiniano, L. Pirolo, Demand Forecasting in the Fashion Industry: A Review, Intech, International Journal of Engineering Business Management Special Issue on Innovations in Fashion Industry, 2013.

[14] T.-M. Choi, C.-L. Hui, S.-F. Ng, Y. Yu, Color trend forecasting of fashionable products with very few historical data, IEEE Transactions on Systems, Man, and Cybernetics Part C: Applications and Reviews 42 (6), 1003–1010, 2012.

[15] S. Thomassey, A. Fiordaliso, A hybrid sales forecasting system based on clustering and decision trees, Decision Support Systems 42 (1) 408–421, 2006. https://doi.org/10.1016/j.dss.2005.01.008

[16] N. Liu, S. Ren, T.-M. Choi, C.-L. Hui, S.-F. Ng, Sales forecasting for fashion retailing service industry: a review, Mathematical Problems in Engineering, 738675, 2013. https://doi.org/10.1155/2013/738675.

[17] Y. Gutgeld, D. Beyer, Are you going out of fashion? The McKinsey Quarterly 3, pp. 55–65, 1995.

[18] F.H. Abernathy, J.T Dunlop, J.H. Hammond, D. Weil, Retailing and Supply Chains in the information age. Technology in society, 22, pp. 5-31, 2000.

[19] V. Varghese , M.D. Rossetti, A Parametric Bootstrapping Approach to Forecast Intermittent Demand. Industrial Engineering Research Conference Proceedings,Vancouver, Canada, May 17-21, 2008.

[20] K. Nemati-Amirkolaii, A. Baboli, M. Shahzad, and R. Tonadre, Demand forecasting for irregular demands in business aircraft spare parts supply chains by using artificial intelligence (ai), IFAC-PapersOnLine, vol. 50, pp. 15 221– 15 226, Jul. 2017.

[21] N. Kourentzes, Intermittent demand forecasts with neural networks, International Journal of Production Economics, vol. 143, no. 1, pp. 198–206, 2013.

[22] S. Mukhopadhyay, A. O. Solis, and R. S. Gutierrez, The accuracy of non-traditional versus traditional methods of forecasting lumpy demand, Journal of Forecasting, vol. 31, no. 8, pp. 721–735, 2012.

[23] O. Helgesson, N. Laszlo, Sparse Time Series Demand Forecasting for Intermittent Availability. A deep learning solution using Temporal Fusion Transformers for marked-down perishable products with a limited shelf life, Chalmers University of Technology, University of Gothenburg, 2023.
https://odr.chalmers.se/server/api/core/bitstreams/fad55a55-3a2d-49d1-a1fb-9e99e5a57681/content

[24] J.G. de Gooijer, R.J. Hyndman, 25 Years of Time Series Forecasting; Elsevier: Amsterdam, The Netherlands, 2006. https://www.sciencedirect.com/science/article/pii/S0169207006000021

[25] E. Stellwagen, L. Tashman, ARIMA: The Models of Box and Jenkins. Foresight: Int. J. Appl. Forecast. 28-33, 2013.
https://www.researchgate.net/publication/285902264_ARIMA_The_Models_of_Box_and_Jenkins

[26] R.G. Brown, Smoothing, Forecasting and Prediction of Discrete Time Series, Prentice-Hall, Englewood Cliffs, 1962.

[27] A.D. Papalexopoulos, T.C. Hesterberg, A regression-based approach to short-term system load forecasting, IEEE Transactions on Power Systems 5 (4), 1535–1547, 1990.

[28] G.E.P. Box, G.M. Jenkins, Time Series Analysis: Forecasting and Control, HoldenDay, 1976.

[29] P.R. Winters, Forecasting sales by exponentially weighted moving averages, Management Science 6 (3), 324–342, 1960.

[30] P.C.L Hui, T.-M Choi, Using artificial neural networks to improve decision making in apparel supply chain systems. In Information Systems for the Fashion and Apparel Industry; Elsevier: pp. 97–107, Amsterdam, The Netherlands, 2016.

[31] S. Makridakis, S. Wheelwright, R. Hyndman, Forecasting Methods and Applications; John Wiley & Sons: New York, NY, USA, 1998.

[32] M. Xia, W.K. Wong, A seasonal discrete grey forecasting model for fashion retailing, Knowledge-Based Systems, Volume 57, Pages 119-126, ISSN 0950-7051, 2014. https://doi.org/10.1016/j.knosys.2013.12.014

[33] R. S. Gutierrez , A. Solis , S. Mukhopadhyay, Lumpy Demand Forecasting Using Neural Networks. International Journal of Production Economics, 111, pp. 409-420, 2008.

[34] E. Spiliotis, A. Kouloumos, V. Assimakopoulos, S. Makridakis, Are forecasting competitions data representative of the reality?, International Journal of Forecasting, Volume 36, Issue 1,Pages 37-53, 2020. https://www.sciencedirect.com/science/article/pii/S0169207019300159

[35] T. Januschowski, J. Gasthaus, Y. Wang, D. Salinas, V. Flunkert, M.Bohlke-Schneider, L. Callot, Criteria for classifying forecasting methods, International Journal of Forecasting, Volume 36, Issue 1, Pages 167-177, 2020. https://www.sciencedirect.com/science/article/pii/S0169207019301529

[36] T. Makridakis , C. Agiropoulos , S. Karkalakos Growth, Electric Power Consumption Externalities and Patterns of Localisation of Emissions. Journal of Economics, Management and Trade, 21(12), 1–15. 2018. https://doi.org/10.9734/JEMT/2018/44455

[37] F. Petropoulos, D. Apiletti, V. Assimakopoulos, G. Sermpinis et al., Forecasting: theory and practice, International Journal of Forecasting, Volume 38, Issue 3, Pages 744, 2022. https://www.sciencedirect.com/science/article/pii/S0169207021001758

[38] B. Seaman, Considerations of a retail forecasting practitioner, International Journal of Forecasting, Volume 34, Issue 4, Pages 822-829, 2018. https://www.sciencedirect.com/science/article/pii/S0169207018300293

[39] S. Makridakis, E. Spiliotis, V. Assimakopoulos, The M4 Competition: 100,000 time series and 61 forecasting methods, International Journal of Forecasting, Volume 36, Issue 1, Pages 54-74, 2020. https://www.sciencedirect.com/science/article/pii/S0169207019301128

[40] C. Fry, M. Brundage, The M4 forecasting competition – A practitioner's view, International Journal of Forecasting, Volume 36, Issue 1, Pages 156-160, 2020. https://www.sciencedirect.com/science/article/pii/S0169207019301189

[41] A. Semenoglou, E. Spiliotis, S. Makridakis, V. Assimakopoulos, Investigating the accuracy of cross-learning time series forecasting methods, International Journal of Forecasting, Volume 37, Issue 3, Pages 1072-1084, 2021.
https://www.sciencedirect.com/science/article/pii/S0169207020301850

[42] M. Dekker, K.l van Donselaar, P. Ouwehand, How to use aggregation and combined forecasting to improve seasonal demand forecasts, International Journal of Production Economics, Volume 90, Issue 2, Pages 151-167, 2004.
https://www.sciencedirect.com/science/article/pii/S0925527304000398

[43] B. N. Oreshkin, D. Carpov, N. Chapados, Y. Bengio, Meta-learning framework with applications to zero-shot time-series forecasting, 2020.
https://arxiv.org/abs/2002.02887

[44] W. Du, S. Y. S. Leung, C. K. Kwong, A multiobjective optimization-based neural network model for short-term replenishment forecasting in fashion industry, Neurocomputing, Volume 151, Part 1, Pages 342-353, 2015.
https://www.sciencedirect.com/science/article/pii/S0925231214012132

[45] A.L.D. Loureiro, V.L. Miguéis, L. F.M. da Silva, Exploring the use of deep neural networks for sales forecasting in fashion retail, Decision Support Systems, Volume 114, Pages 81-93, 2018.
https://www.sciencedirect.com/science/article/pii/S0167923618301398

[46] S. Smyl, A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting, International Journal of Forecasting, Volume 36, Issue 1, Pages 75-85, 2020.
https://www.sciencedirect.com/science/article/pii/S0169207019301153

[47] S. Bai, J. Z. Kolter, V. Koltun, An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling, 2018. https://arxiv.org/abs/1803.01271

[48] B. Lim, S. O. Arik, N. Loeff, T.s Pfister, Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting, 2020. https://arxiv.org/abs/1912.09363

[49] J. Herzen et al., "Darts: User-friendly modern machine learning for time series," Journal of Machine Learning Research, vol. 23, no. 124, pp. 1–6, 2022.
http://jmlr.org/papers/v23/21-1177.html.

[50] J. Brownlee, Why one-hot encode data in machine learning? Jul. 28, 2017, [Online]. Available: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/

[51] P. Goodwin, Using Naïve Forecasts to Assess Limits to Forecast Accuracy and the Quality of Fit of Forecasts to Time Series Data, 2014. http://dx.doi.org/10.2139/ssrn.2515072

[52] J. D. Croston, Forecasting and Stock Control for Intermittent Demands, Journal of the Operational Research Society, 23:3, 289-303, 1972. DOI: 10.1057/jors.1972.50

[53] G. Ke, Q. Meng, T. Finley, T. Wang, W.i Chen, W. Ma, Q. Ye, T. Liu, LightGBM: A Highly Efficient Gradient Boosting Decision Tree, Advances in Neural Information Processing Systems 30, 2017. https://papers.nips.cc/paper_files/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

[54] B. N. Oreshkin, D. Carpov, N. Chapados, Y. Bengio, N-BEATS: Neural basis expansion analysis for interpretable time series forecasting, International Conference on Learning Representations, 2020. https://openreview.net/forum?id=r1ecqn4YwB

[55] M. Pace, LightGBM for TimeSeries forecasting, Published in Data Reply IT | DataTech, Jan 19, 2022, [Online]. Available: https://medium.com/data-reply-it-datatech/lightgbm-for-timeseries-forecasting-408971289a12

[56] Z. C. Lipton, J. Berkowitz, and C. Elkan, A critical review of recurrent neural networks for sequence learning, 2015. https://arxiv.org/abs/1506.00019

[57] A. Vaswani et al., Attention is all you need, Advances in neural information processing systems, vol. 30, 2017.

[58] M. Filho, Multiple Time Series Forecasting With LSTM, In Python | Forecastegy, 2023, [Online]. Available: https://forecastegy.com/posts/multiple-time-series-forecasting-with-lstm-in-python/

[59] L. X. Hien, H. Hung, L. Giha, J. Sungho, Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting - Scientific Figure on ResearchGate, 2019, Available from: https://www.researchgate.net/figure/The-structure-of-the-Long-Short-Term-Memory-LSTM-neural-network-Reproduced-from-Yan_fig8_334268507

[60] M. Filho, Multiple Time Series Forecasting With N-BEATS, In Python | Forecastegy, 2023, [Online]. Available: https://forecastegy.com/posts/multiple-time-series-forecasting-nbeats-python/

[61] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, Optuna: A Next-generation Hyperparameter Optimization Framework, In KDD. 2019. https://arxiv.org/abs/1907.10902

[62] P. Wallström, Evaluation of forecasting techniques and forecast errors: With focus on intermittent demand, Ph.D. dissertation, Luleå tekniska universitet, 2009.

[63] R. J. Hyndman, Another look at forecast-accuracy metrics for intermittent demand, Foresight: The International Journal of Applied Forecasting, vol. 4, no. 4, pp. 43–46, 2006.

[64] M. Joseph, Forecast Error Measures: Intermittent Demand, 2020, [Online]. Available: https://deep-and-shallow.com/2020/10/07/forecast-error-measures-intermittent-demand/

[65] R. J. Hyndman, A. B. Koehler, Another look at measures of forecast accuracy, International Journal of Forecasting, vol. 22, no. 4, pp. 679–688, 2006.
https://doi.org/10.1016/j.ijforecast.2006.03.001

[66] Safety Stock Formula: how to calculate and use, Linnworks, 2020, [Online]. Available: https://www.linnworks.com/blog/safety-stock-formula/?skuvault=true

[67] J. Vermorel, Safety Stock, Lokad, 2012, [Online]. Available: https://www.lokad.com/calculate-safety-stocks-with-sales-forecasting/

[68] S. Schalit, J. Vermorel, Service Level (Supply Chain), Lokad, 2014, [Online]. Available: https://www.lokad.com/service-level-definition/

[69] J. Liliedahl, The Ultimate Guide to Inventory Forecasting, Inventory Planner, 2022, [Online]. Available:
https://www.inventory-planner.com/ultimate-guide-to-inventory-forecasting/

[70] W. Xue, T. Zhou, Q.g Wen, J. Gao, B. Ding, R. Jin, Make Transformer Great Again for Time Series Forecasting: Channel Aligned Robust Dual Transformer, 2023.
https://arxiv.org/abs/2305.12095