University of Padova

Department of Mathematics

Master Thesis in Data Science

Deep Learning and Hierarchical Approaches for Sporadic Demand Forecasting

Supervisor
Professor Mariangela Guidolin
University of Padova

Master Candidate
Gioele Ceccon

Academic Year

2023-2024

To those who have supported, helped, and kept up with me all these years.

Abstract

In the retail industry, demand forecasting can play a pivotal role in successful management, given that accurate predictions of customer purchases are essential for inventory planning, resource allocation, and overall operational efficiency. However, achieving precise forecasts in retail, especially within the fashion domain, presents unique challenges. The fashion industry is indeed characterized and influenced by inherent uncertainties in supply and demand, a lengthy and inflexible supply process, a short life cycle, vast product variety, a high propensity for impulsive buying and other exogenous variables. This work then navigates the complex landscape of fashion retail forecasting, addressing the challenges posed by intermittent time series data and, more in general, by taking into account the forecastability of each time series. Specifically, it is proposed a deep learning solution, represented by the Temporal Fusion Transformer (TFT) model, to a real-world scenario, represented by the hosting company data. Moreover, another approach is built on top of the generated forecasts through hierarchical forecasting, by trying to leverage the company's business logic to enhance accuracy further.

Contents

Listing of figures

Listing of tables

Listing of acronyms

ADI	Average Demand Interval
ARIMA	Auto-Regressive Integrated Moving Average
CFE	Cumulative Forecast Error
CV	Coefficient of Variation
DBA	Dtw Barycenter Averaging
DNN	Deep Neural Network
DTW	Dynamic Time Warping
ELU	Exponential Linear Unit
GLU	Gated Linear Unit
GRN	Gated Residual Network
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MASE	Mean Absolute Scaled Error
MSE	Mean Squared Error
NN	Neural Network
OHE	One-Hot-Encoding
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SKU	Stock Keeping Unit
TFT	Temporal Fusion Transformer

1 Introduction

In the retail industry, demand forecasting is a critical process that involves forecasting how many units of a particular product customers are likely to buy over a specific period. This forecast is based on a comprehensive analysis of historical sales data, market trends, seasonal patterns, and other relevant factors such as economic indicators and consumer behavior. Therefore, advanced statistical techniques and machine learning algorithms are often employed to model these complex relationships and generate accurate forecasts [1]. The goal, then, is to understand the likely demand for specific products or services during a particular period, ranging from a few weeks to several months or even years.

Therefore, for a retail business accurate demand forecasting can provide several benefits that cut across different fields:

- **Inventory Management:** demand forecasting helps retailers determine the optimal amount of stock to maintain. By accurately predicting customer demand, retailers can avoid overstocking, which ties up capital and leads to increased storage costs, and understocking, which can result in lost sales and dissatisfied customers.

- **Marketing and Promotions:** with an understanding of future demand, retailers can design effective marketing campaigns and promotional activities. For instance, if a particular product is forecasted to have high demand, retailers can focus their marketing efforts on that product to boost sales further.

- **Workforce Planning:** demand forecasting also aids in efficient workforce planning. By identifying peak periods and slower seasons, retailers can adjust their staffing levels accordingly. This ensures that there are sufficient employees to handle the demand during busy times while avoiding unnecessary labor costs during slower periods.

- **Financial Planning:** accurate demand forecasts can inform financial planning and budgeting processes. They provide insights into future sales revenues, helping retailers make informed decisions about capital investments, operational expenses, and potential growth strategies.

In a retail context, then, effective forecasting can lead to better customer satisfaction, reduced costs, and improved overall business performance. It is, therefore, a key aspect of successful retail management.

Nevertheless, achieving an accurate forecast in a retail setting can be quite challenging, particularly when dealing with fashion-related data as in the case of this thesis. The fashion industry is indeed characterized by inherent uncertainties in supply and demand, a lengthy and inflexible supply process, a short life cycle, vast product variety (e.g., style, color), and a high propensity for impulsive buying [2]. To further increase this volatility, thus adding another layer of complexity to demand forecasting, there are also other exogenous factors to take into account, such as economic and weather conditions, general market trends, and global events. Given the industry's dynamic and fast-paced nature, with trends changing rapidly and consumer preferences evolving constantly, articles tend to have short-selling cycles, due to which there are frequent demand fluctuations [3]. Therefore, one of the key challenges is dealing with sporadic demand [4]. This refers to demand patterns characterized by numerous periods with zero demand, making forecasting particularly challenging. The fashion industry often experiences sporadic demand: the nature of fashion products, heavily influenced by external variables such as seasonality, trends, and consumer preferences, leads to instances where certain items experience surges in demand, followed by extended periods of relative quietness. Factors like limited editions, unique designs, and the constant emergence of new collections further exacerbate this intermittency. Another significant challenge is dealing with demand censoring [5]. This phenomenon occurs when the actual demand for a product exceeds its availability, causing the total sales to fall short of the total demand. In the fashion industry, this can be due to a variety of factors such as supply chain disruptions, production issues, or logistical challenges. When demand censoring occurs, it can lead to lost sales and customer dissatisfaction. Moreover, it can distort the historical sales data used for forecasting, making it appear as though demand is lower than it is. This can result in inaccurate forecasts, leading to further stock-outs and perpetuating a cycle of lost sales and customer dissatisfaction [6].

In light of these challenges, retailers are increasingly turning to advanced analytics and machine learning techniques to improve the accuracy of their forecasts. As in the case, I had the opportunity to work in this direction during my tenure at a company headquartered in Veneto, a key player in the footwear and clothing industry. The company's data mirrored the challenges mentioned earlier, exhibiting sporadic and highly intermittent behavior in a significant number of instances, particularly concerning the high volatility and large variance of its articles. My role, then, involved developing a solution to enhance the forecast accuracy for both the company's directly owned stores and the franchise stores, which are spread throughout Europe and represent a carefully selected and limited portion of locations where their products are available. Therefore, this work aims to display the system I developed thanks to the application of statistical and machine learning techniques and in collaboration with the business side of the hosting company.

1.1 Operative context and previous work

The groundwork of the devised solution is represented by a previously developed project [7] at the hosting company, whose aim was to optimize inventory replenishment through the use of deep neural network models. The work presented here, then, shares a common point with that project: the data collection used consists of sales data regarding store-SKU combinations from diverse European locations. In the context of this thesis, each Stock Keeping Unit (SKU) identifies a specific size of a distinct article. On the contrary, in the previous project it was selected only a subset of the possible store-SKU combinations, formed from the time series of articles that were part of the store's product portfolio for a minimum of 52 weeks and had at least 20% of weeks with actual sales. This selection criterion was established in order to ensure that the selected SKUs had relatively stable demand while also having the necessary data for performance assessment. However, this selection process significantly limited the number of combinations considered: the total number of unique store-SKU combinations decreased from $1,102,667$ to $2,077$, with these combinations deriving from 141 out of 191 stores and 311 out of $31,869$ distinct SKUs. The primary objective of this study, then, was to investigate a method to overcome the constraints of the previous project. This involved expanding both the quantity of stores considered and the combinations of store-SKU, thereby addressing sporadic and extremely intermittent time series.

1.2 Thesis outline

The thesis is organized as follows:

- Chapter 2 showcases an in-depth analysis of the data and a classification in terms of forecastability of the store-SKU time series;
- Chapter 3 discusses how the data have been selected in collaboration with the hosting company;
- Chapter 4 explains how the data have been treated before the forecasting step and how various explanatory variables have been derived;
- Chapter 5 illustrates the strategy adopted, explaining the architecture of the chosen model and how hierarchical forecasting integrates with the company's business logic;
- Chapter 6 discusses the results obtained by the model and the effect hierarchical forecasting had on them;
- Chapter 7 reports the concluding remarks, summarizing key findings and suggesting potential areas for future research to build upon this work.

2
Data exploration and classification

The original dataset provided by the hosting company consisted of $1,381,701$ time series. Each of these represented the historical sales of a store-SKU combination, obtained from 301 stores scattered across Europe and $47,524$ distinct SKUs. These time series have a weekly frequency, with each observation being recorded on the monday of each week, and have a varying length that covers a 1-year time horizon.

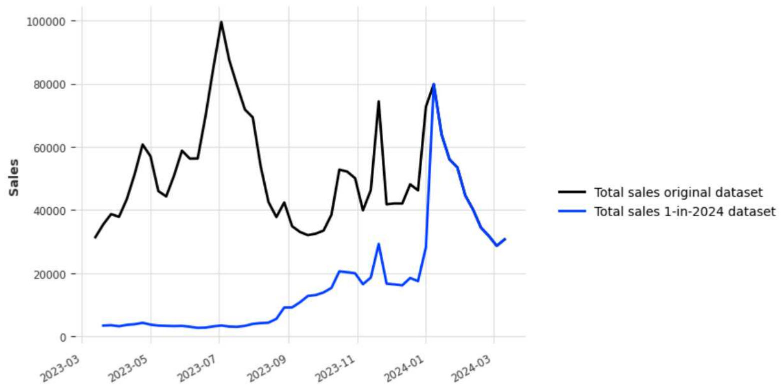

Figure 2.1: Comparison between total sales of the original dataset and of the dataset restricted to those store-SKU combinations with at least one registered sale in 2024.

Moreover, in order to avoid taking into account ”dead” time series, which refer to those store-SKU combinations that are no longer being sold and replenished by the company, only those with at least one week of sales in 2024 were considered, resulting in $357,928$ combinations coming from the same 301 stores and a reduced pool of distinct SKUs, counting $21,376$ elements. The comparison between the total sales of the dataset derived from this restriction, called "1-in-2024", and the original one can be seen in Figure 2.1. As expected given the nature of the articles in the fashion retail industry and the type of selection made, most of the sales in the 1-in-2024 dataset are concentrated in the first weeks of 2024, while in the original dataset the peak is reached in the first week of July 2023.

2.1 Data exploration

The dataset's time series have various lengths, covering a 52 week window spanning from March 13, 2023 to March 11, 2024. Moreover, these time series are continuous (there are no missing observations) and they all culminate at the same final time stamp. A closer look at the lengths' statistics of the dataset shows that:

- The longest time series, obviously, cover the whole 52 weeks of the time window;
- The shortest time series have length 1. They make 5% of the dataset and represent those articles that have just been added to the stores' portfolio. Because of their lack of past data, then, these time series were not considered for the forecasting step;
- The mean length is 16. A further investigation regarding this data highlights that:
  - 55% of the time series cover less than 14 weeks;
  - 26% of the time series have a length between 14 and 26 observations;
  - The rest of the dataset (19%) spans between 26 and 52 weeks.

This data, then, highlights the short life cycle of articles in the retail industry, a topic that was discussed in Chapter 1.

One of the challenges mentioned in Chapter 1 is dealing with sales data in the retail industry, given that they can exhibit sporadic and extremely intermittent behavior. Most of the time series of the dataset seem to fall into the case, as can be seen looking at Figure 2.2.

Particularly, it is easy to notice how the majority (65%) of the samples pile up before the 0.2 mark, meaning that they are characterized by a low ratio of weeks with non-zero sells and, thus, a high number of observations being 0. It is also worth noticing how a small but consistent

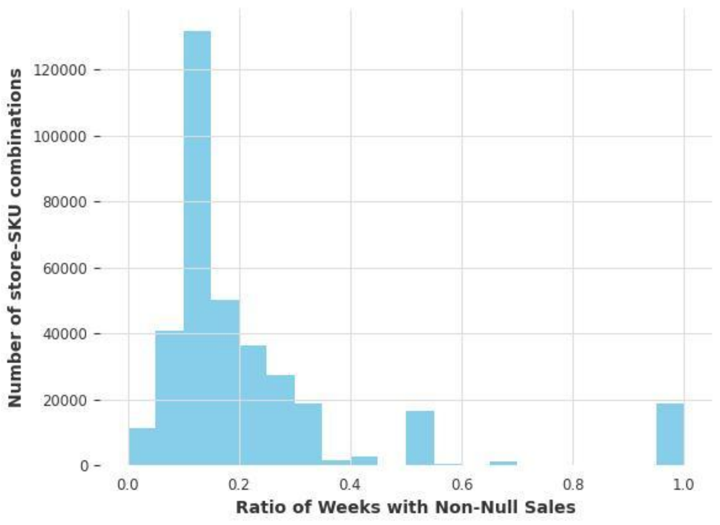

Figure 2.2: Distribution of the ratio of weeks with non-null sales.

(5%) pool of time series falls in the last bin of the histogram. Most of the store-SKUs in this group are articles that have just been added to the stores' portfolio given that their time series are made of just one observation. Nevertheless, the longest time series in this group has a length of 5.

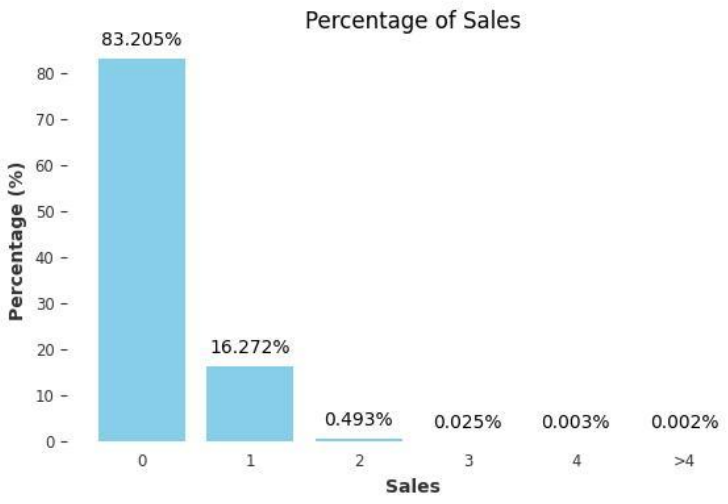

Figure 2.3: Distribution of the weekly sales of all the store-SKU combinations. The x-axis represents the number of weekly sales, while the y-axis represents the percentage of weeks with the corresponding sales.

Furthermore, Figure 2.3 highlights how the majority (83.205%) of the total weeks in the dataset has no sales recorded. Other than further proving the intermittency of the time series of the dataset, Figure 2.3 showcases also a low volume of sales, given that the weeks with more than 1 sale registered amount for nearly over 5% of the total.

2.2 Data classification

Section 2.1 showcased the widespread presence of sporadic behavior among the time series of the dataset. Figure 2.4 helps with further investigating this phenomenon, showcasing the classification of the dataset in terms of forecastability of each time series [8]. This classification is then based on the characteristics of the historical demand, summed up by two coefficients:

- The Average Demand Interval (ADI) measures the demand regularity in time. It does so by computing the average interval between two demands, as can be seen in 2.1, with T representing the length of the time series and D the number of non-zero observations.

$$ADI = \frac{T}{D} \tag{2.1}$$

- The square of the Coefficient of Variation (CV^2), displayed in 2.2, which measures the variation in quantities.

$$CV^2 = \left(\frac{\sigma}{\mu}\right)^2 \tag{2.2}$$

According to the literature [8], it is then possible to classify the demand profiles into four different categories based on these two dimensions,:

- Smooth demand (ADI < 1.32 and CV^2 < 0.49). The demand is very regular in time and in quantity. It is therefore easy to forecast.
- Intermittent demand (ADI ≥ 1.32 and CV^2 < 0.49). The demand history shows very little variation in demand quantity but a high variation in the interval between two demands. Though specific forecasting methods tackle intermittent demands, the forecast error margin is considerably higher.
- Erratic demand (ADI < 1.32 and CV^2 ≥ 0.49). The demand has regular occurrences in time with high quantity variations. This may be reflected in the forecasting accuracy.
- Lumpy demand (ADI ≥ 1.32 and CV^2 ≥ 0.49). The demand is characterized by a large variation in quantity and in time. It is actually very hard in these instances to produce a reliable forecast.

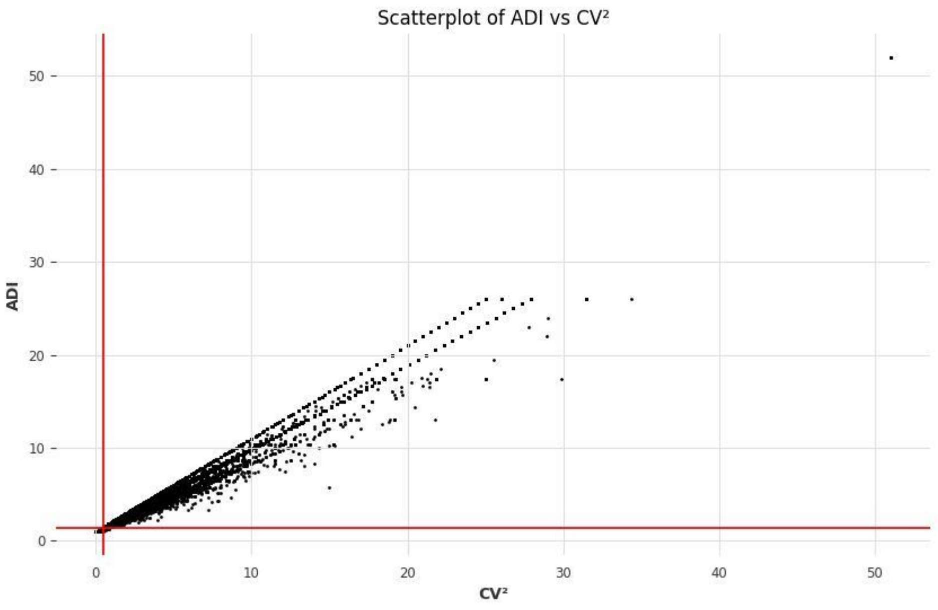

Figure 2.4: Dataset "forecastability" classification based on Squared Coefficient of Variation and Average Demand Interval.

Concerning the dataset provided for this thesis, the classification was carried out over $189,343$ ($52,8\%$) out of the $357,928$ original store-SKU combinations, since it was decided to exclude those time series having only one non-zero observation. This approach was taken to allow for a more comprehensive analysis, which will enhance the understanding of the forecasting results in subsequent chapters. However, to provide a broader view of the overall scenario, Figure 2.4 displays a scatter plot of all data points. At first look, it is easy to notice how the vast majority of the time series belong to the lumpy class. Additionally, a distinct point can be seen in the far-right corner: it actually represents $4,687$ time series that share the same length, 52, and have only one non-zero observation.

Then, the classification performed over the reduced set of store-SKUs provided the following results:

- 715 (0.2%) time series are smooth;
- 127 (0.04%) are intermittent;
- 188498 (52.66%) are lumpy;
- 9 are erratic.

Clearly, these results highlight a complex scenario in terms of forecastability, given the high number of lumpy time series and of time series with just one non-zero observation. Furthermore, Figure 3.1 allows to take a closer look to how a time series of each class looks like and showcases how demand regularity and quantity variations characterize each class.

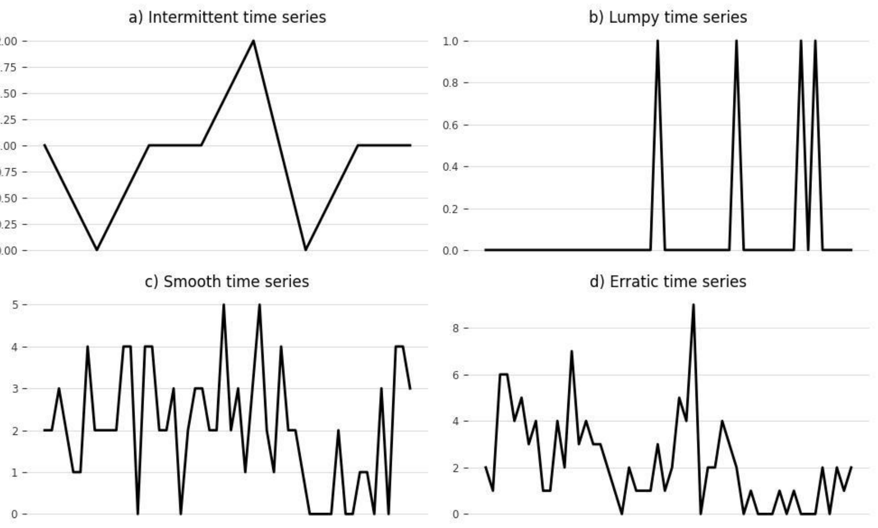

Figure 2.5: From left to right: a) Intermittent time series (ADI=1.33, CV²=0.47), b) Lumpy time series (ADI=13, CV²=12), c) Smooth time series (ADI=1.24, CV²=0.48), Erratic time series (ADI=1.87, CV²=0.83).

3 Data pre-processing

Due to the large size of the dataset, which includes over $300,000$ time series, a decision was made in collaboration with the hosting company to narrow the perimeter of the analysis. However, to preserve the context, the number of combinations was reduced by focusing on a smaller set of stores, selected on the basis of a clustering performed over:

- The historical sales data (of those SKUs selected in Chapter 2) of each store, as explained in Section 3.1.
- A collection of store-related attributes and sales-related statistics, as presented in Section 3.2.

3.1 Dynamic time warping clustering

The first clustering performed exploits a modified version of the k-Means algorithm, tailored to perform over time series data. Firstly, instead of using the Euclidean distance, it uses Dynamic Time Warping (DTW) as similarity measure between time series [9]. Given two time series $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_m)$ of respective lengths n and m, computing the DTW similarity between them is formulated as the optimization problem presented in Eq. 3.1:

$$DTW(x,y) = \min_{\pi} \sqrt{\sum_{(i,j)\in\pi} d(x_i, y_i)^2} \tag{3.1}$$

where d is a distance function (for this work, the Euclidean distance) and $\pi = [\pi_1, \ldots, \pi_k]$ is a path that satisfies the following properties:

- It is a list of index pairs: $\pi_k = (i_k, j_k)$ with $0 \leq i_k < n$ and $0 \leq j_k < m$;
- $\pi_0 = (0,0)$ and $\pi_k = (n-1, m-1)$;
- $\forall k > 0, \pi_k = (i_k, j_k)$ is related to $\pi_{k-1} = (i_{k-1}, j_{k-1})$ as follows:
  - $i_{k-1} \leq i_k \leq i_{k-1} + 1$;
  - $j_{k-1} \leq j_k \leq j_{k-1} + 1$.

Then, a solution is represented by a minimal path, which can be seen as a temporal alignment of time series such that Euclidean distance between the aligned time series is minimal.

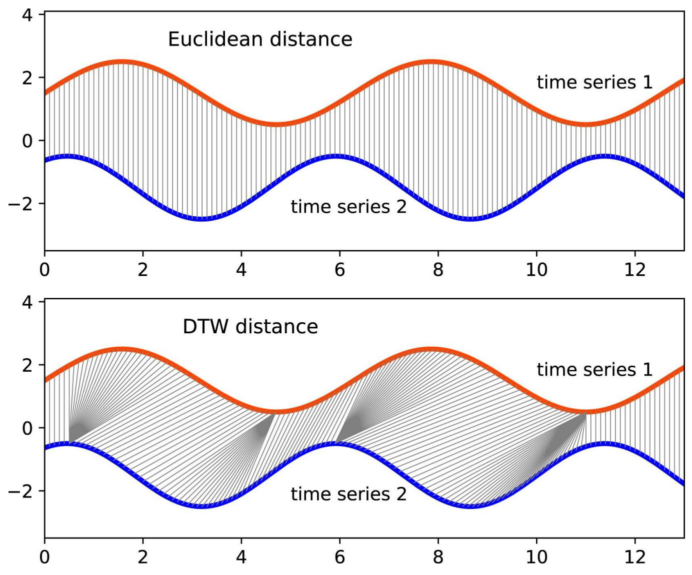

Figure 3.1: Example of Euclidean distance calculation and DTW distance calculation [10].

Therefore, doing so DTW minimizes the effects of shifting and distortion in time by allowing "elastic" transformation of time series in order to detect similar shapes with different phases [11]. Because of this DTW distance computation ignores local and global shifting of the time dimension while Euclidean distance simply calculate the sum of distances between matching points [10], as can been seen in Figure 3.1.

Other than computing the similarity between the time series, this specialized version of k-Means needs to compute at each iteration the centroid of each cluster by averaging all the time series in it. In order to do that, the Dynamic-time-warping Barycenter Averaging (DBA) technique is used [12]. It consists of a heuristic strategy that iteratively refines an initially (potentially arbitrary) average sequence, in order to minimize its squared distance (DTW) to averaged sequences.

Finally, to evaluate the goodness of the obtained clustering it was used the silhouette score:

$$s = \frac{b - a}{max(a, b)} \tag{3.2}$$

where:

- a represents the mean distance between a sample and all other points in the same cluster;
- b represents the mean distance between a sample and all other points in the next nearest cluster.

The score is bounded between -1 for incorrect clustering and $+1$ for highly dense clustering, thus meaning that the score is higher when clusters are dense and well separated.

This kind of clustering was then performed on a dataset made of 301 time series of length 52, representing the weekly historical sales data of each store. Such time series were obtained by aggregating all the store-SKU combinations, having at least a week with a sale in 2024, whithin each store. Furthermore, the clustering was performed giving in input, since the k-Means algorithm requests it, values ranging from 2 to 10, representing the number of clusters to be found. The resulting clusters were then evaluated according to their silhouette score, described by 3.2. The best score, 0.65, was achieved by a solution proposing 2 clusters: cluster 0 (containing 50 stores) and cluster 1 (containing 251 stores). The centroid of each cluster is displayed in Figure 3.2. Looking at them, it is easy to notice their similar shape even though the time series representing the centroid of cluster 0 is made of much larger observations. This happens because the stores in cluster 0 have more sales on average with respect to the ones in cluster 1.

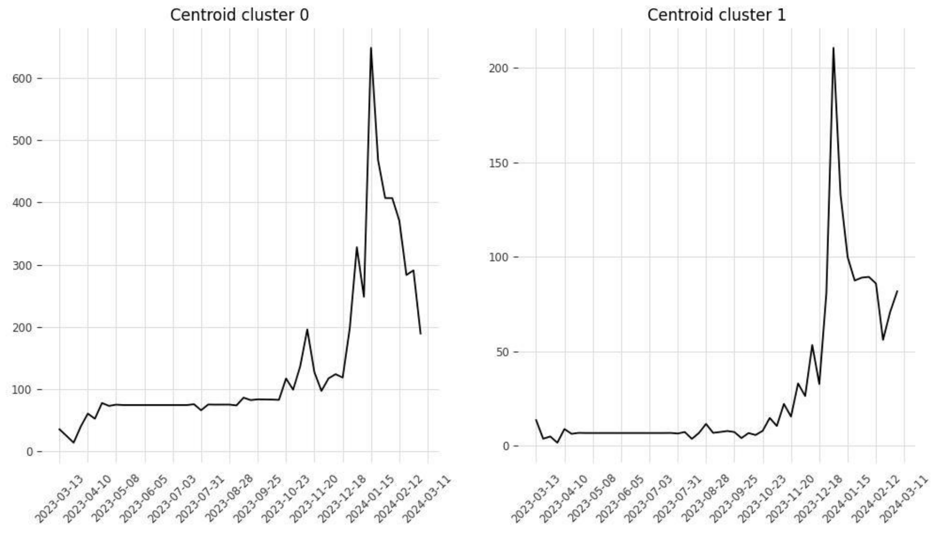

Figure 3.2: Centroids of DTW clustering.

3.2 Stores attributes clustering

Prompted by the hosting company, it was performed a clustering using the k-Means algorithm over a dataset made of 301 data samples, one for each store, built as follows:

- Three categorical values, representing:
  - The country where the store is located;
  - The store's type, which can either be directly owned or a franchise;
  - The company's internal classification, which divide the stores into four groups based on their marketability, determined according to internal business rules.

  All these variables were encoded using the One-Hot-Econding (OHE) technique [13]. In OHE, each value of a feature is transformed into a binary vector representation of that value. Therefore, this approach allows to maintain the categorical information while at the same time prevents biases, especially when categories lack a natural order, like in this case, by not introducing any ordinal relationship among them.

- Four numerical values:
  - The square meters of the store;
  - The average sales of the last 52 weeks;

– The variance of the sales of the last 52 weeks;
– The minimum in terms of sales of the last 52 weeks;
– The maximum in terms of sales of the last 52 weeks.

All these values were then standardized to prevent some features from dominating others due to their different units or scales.

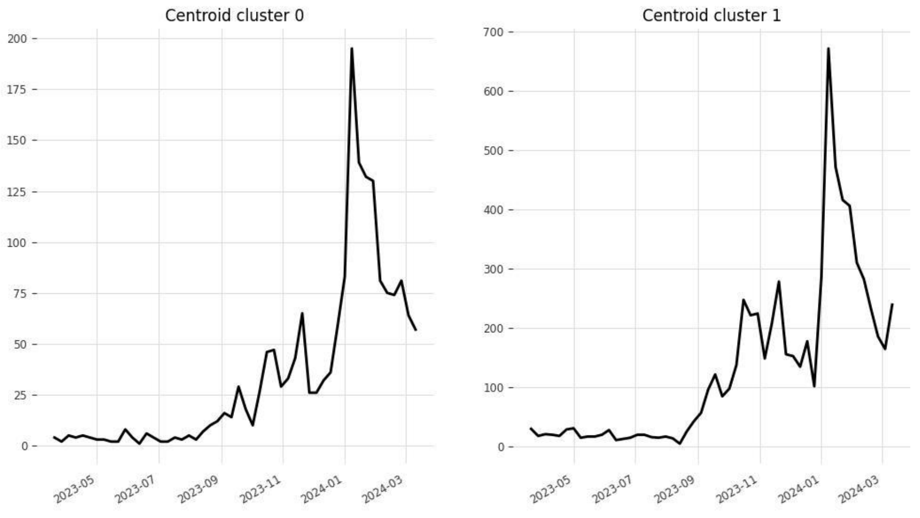

Figure 3.3: Centroids of classical k-Means clustering.

As done in Section 3.1, the algorithm was given in input values ranging from 2 to 10, representing the number of clusters to be found. The best silhouette score, 0.53, was returned by the clustering proposing 2 clusters: cluster 0 (made of 268 stores) and cluster 1 (made of 33 stores). Figure 3.3 showcases their centroids, computed as done in Section 3.1 through DBA, in order to allow a better comparison. Similarly, the centroids resemble each other and the smaller cluster (cluster 1) exhibits a larger mean value of sales across the observed 52 weeks, since it contains the stores that perform better in terms of sales.

3.3 Final perimeter

Both the clustering presented in Section 3.1 and the one discussed in Section 3.2 propose a similar solution, with a smaller cluster containing the most performant stores and a larger cluster containing all the others. Moreover, a further analysis reveals that all the stores (33) of cluster from 1 Section 3.2 are contained in cluster 0 from Section 3.1. This is an expected result, given that in both cases these clusters contain the best selling stores. Nevertheless, as can be seen in Figure 3.4, the centroids obviously resemble each other in shape, even though the centroid of cluster 1 showcases a higher peak, achieved a couple of weeks before the one reached by the centroid of cluster 0. This happens because its stores, while representing a subset of the stores of cluster 0, represent the best selling stores overall.

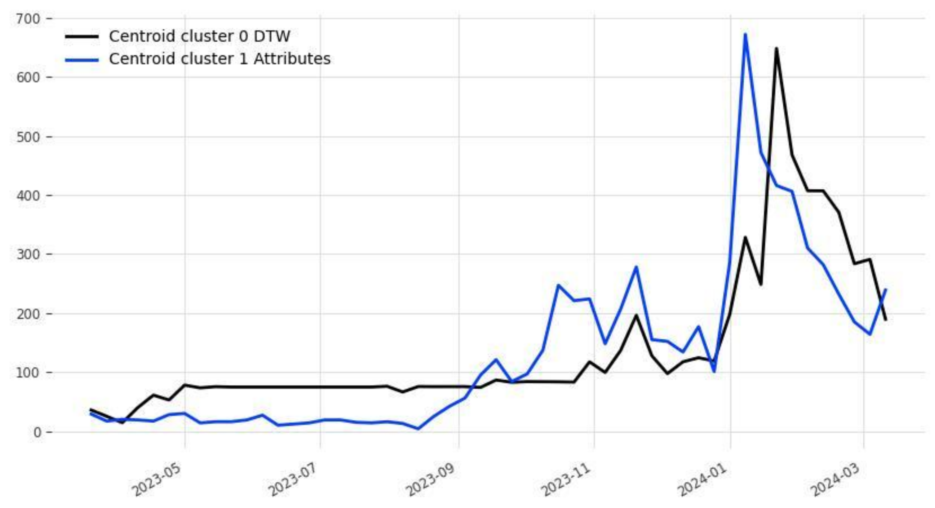

Figure 3.4: Centroids comparison.

It was then decided to keep 10 of the best selling stores, accounting for 28, 729 store-SKU combinations and 11, 067 distinct SKUs. Regarding their classification, these time series are grouped as follows:

- 117 are smooth;
- 17803 are lumpy;
- 27 are intermittent;
- 7 are erratic.

Furthermore, the remaining 10775 combinations have just one week with sales. Regarding their length, the average is 19 weeks, with 13688 (48%) time series with less than 14 observations registered and 8279 time series lasting between 14 and 26 weeks.

4 Explanatory variables

To aid with the forecasting effort, each store-SKU time series has been equipped with a set of explanatory variables. These variables are divided into three categories:

- Static variables, discussed in Section 4.1. They hold time-independent (constant/static) information about the time series to be forecasted. Moreover, they are particularly useful when dealing with multiple time series to forecast, given that the static information they carry can help models identify the nature/environment of the underlying series, thus improving the forecasts [14];

- Past variables, presented in Section 4.2. They contain information about the past (up to the forecast point) and typically represent measurements or temporal attributes;

- Future variables, examined in Section 4.3. They hold information about the future (beyond the forecast point) and typically represent forecasts, temporal attributes or other future-known data.

4.1 Static variables

The following static variables were taken into account:

- *store id*. It represents the unique identifier of the store of each store-SKU time series. Each store might have different characteristics such as location, size, customer demographics, and more, which can significantly influence the sales of a SKU. Including this identifier as a static variable aims at considering this kind of information in the forecasting process.

- *group id*. It represents the business classification of the SKU, schematized in Figure 4.1, which is defined by:

  - The type of customer the article is intended to (such as Men, Women, Teenagers and Babies);
  - The article's category (whether it's a pair of shoes, an accessory, or clothing);
  - The article's subcategory (like sneakers for a pair of shoes).

  Moreover, it must be specified that the mapping between these three components is not complete, as shown in Figure 4.1. For example, an article intended for babies can only be a shoe, while in the case of the teenager's line, it is possible to have both shoes and accessories. Furthermore, this asymmetry is reflected also at the subcategory level, since some subcategories are specific not only for the type of customer but also for their gender: for example, 'ballerina' shoes are internally classified as articles for adult and young women since this type of shoes is not produced for babies.

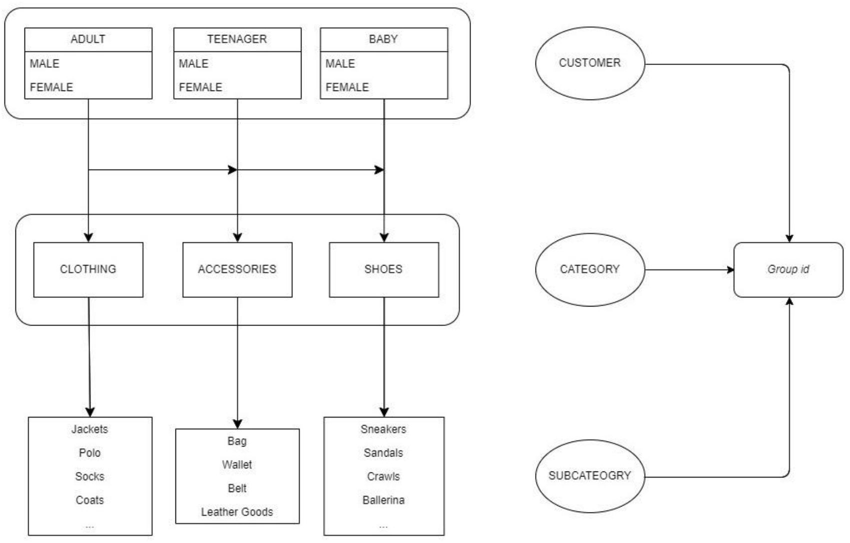

Figure 4.1: Business grouping of an SKU, defined by its *group id*. It depends on the customer the article is intended to, its category and its subcategory.

Then, given that different customer groups may have different purchasing behaviors, the goal in taking into account this covariate is to improve the forecasts by comprehending this kind of trends.

4.2 Past variables

Regarding the past variables, it was decided to consider:

- The time series representing the discount history of each SKU in each store. Discounts can significantly influence customer purchasing behavior: when articles are discounted, customers may be more likely to purchase them, leading to an increase in demand [15];
- Four time series obtained by applying two moving average filters of window 4 and 12 to:
  - The store-SKU time series to be forecasted, in order to highlight the presence of past trends and reduce the impact of long periods without sales;
  - The time series obtained by aggregating the store's sales according to the SKU's *group id*. As explained in Section 4.1, this attribute reflects the business classification of each SKU. Taking into account such covariates can then help capture broader trends within each business group, providing additional context that might be useful for the forecasts.

The filters perform their computations as follows: given a time series $x = (x_1, \dots, x_m)$ of length m and a moving average filter of window n, it computes the average for every observation $x_i \in x$ using x_i and the $\frac{n-1}{2}$ observations before and after it, as displayed in Eq. 4.1:

$$\text{Moving Average at time t} = \frac{1}{n} \sum_{i=\frac{1-n}{2}}^{\frac{n-1}{2}} x_{t+i} \tag{4.1}$$

As can be seen looking at Figure 4.2, this process results in a new time series that smoothes out the fluctuations in the original series, making trends easier to spot and allowing to take into consideration seasonal and cyclical behaviors.

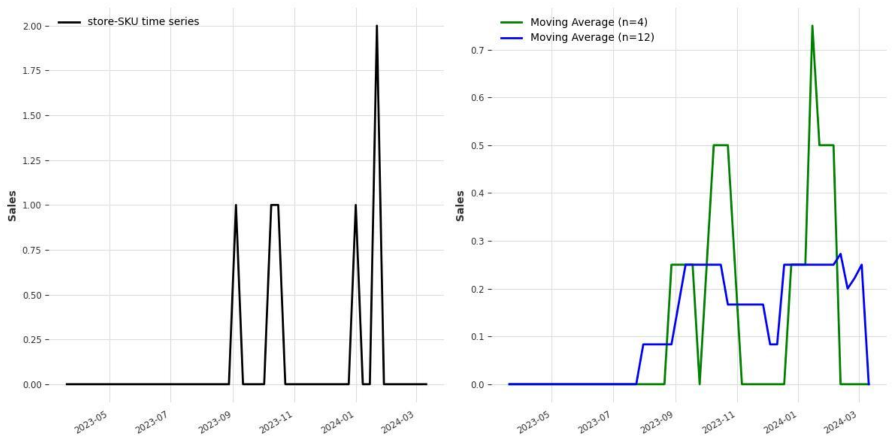

Figure 4.2: Example of moving average filter of window 4 and 12 applied to a store-SKU time series.

4.3 Future variables

The future variables taken into account are the following:

- Four temporally related time series, used to represent the month of the year and the week of the year. In order to be useful, they undergo cyclic encoding to capture their inherent cyclical characteristics, ensuring that the first and last values in their numerical ranges are close to each other. Cyclic encoding involves normalizing the time-related data, followed by the application of *sine* and *cosine* functions, resulting in two novel features. As can be seen in Figure 4.3, these transformations map each value to a point on the unit circle, thus preserving the cyclic nature of the data.

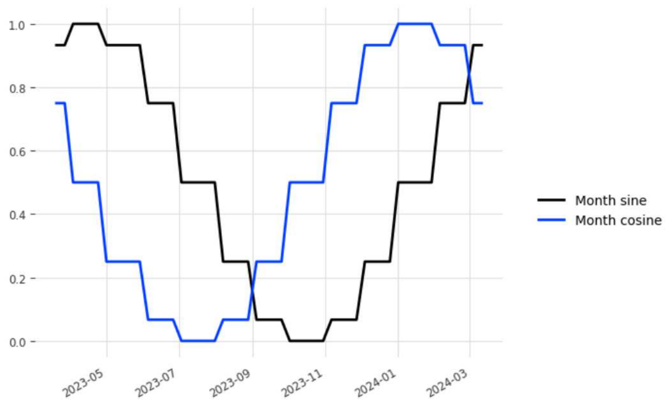

Figure 4.3: Cyclical encoding of the 'Month' temporal attribute.

- The time series obtained by combining the aggregated sales data for the store, grouped by the SKU's *group id*, with its future forecasts. Furthermore, it must be noted that:

    1. The aggregation takes into account only the store-SKU combinations of the 10 stores with at least a registered sale in 2024, as described in Chapter 2. This decision was made based on the results discussed in Chapter 6, which highlight how these forecasts are more accurate than the ones computed over all the available historical data;
    2. The forecasts are computed by an Auto-Regressive Integrated Moving Average (ARIMA) model.

5 Forecasting strategy

Considering the characteristics of the dataset and the work previously conducted [7], it was decided to use a deep learning model, namely the Temporal Fusion Transformer (TFT) [16], to produce the forecasts. This choice was driven by the unique features of this Deep Neural Network (DNN) model, discussed in Section 5.1, and its ability to perform cross-learning. In the context of time series forecasting, cross-learning refers to the simultaneous training of a model across multiple time series. This approach enables the model to identify and learn from global patterns that might exist across different time series, thereby improving the precision of the forecasts. Then, with respect to the series-by-series approach, which requires fitting a model for each time series to be forecasted, cross-learning allows to better exploit large amounts of data. Moreover, it also only requires fitting a single model for the entire dataset.

Additionally, to enhance the forecasts accuracy, it was decided to further integrate the company's business logic into the forecasting process. This was achieved through hierarchical forecasting [17], presented in Section 5.2, which was applied to the business hierarchy of the time series obtained from aggregating the store-SKU combinations. An insight into such organization, mirroring the structure of a tree, is showcased by Figure 5.1:

- On top of the hierarchy there is the ”Total” time series, obtained from aggregating the sales of all the 10 stores;
- The children of the ”Total” node are the 10 nodes representing the sales aggregated for each store;

- Each store node then serves as the root of a tree that reflects the business grouping discussed in Section 4.1. Here, the customer level of aggregation is mapped to the category level, which in turn is mapped to the subcategory level, representing the lowest level of aggregation;
- Lastly, the store-SKU time series represent the leaves of the tree. Each of these, depending on the store they fall under, is then assigned to a specific subcategory of a specific category of a specific customer type.

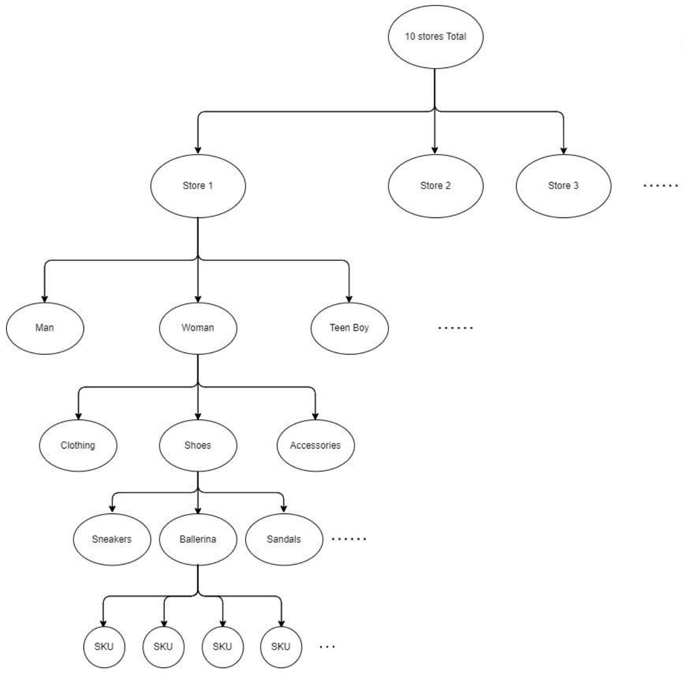

Figure 5.1: Business hierarchy of time series aggregations.

5.1 Temporal Fusion Transformer

The temporal Fusion Transformer is an attention-based architecture for multi-horizon time series forecasting, which represents an evolution of the original Transformer architecture [18]. Particularly, the TFT can be trained in a cross-learning fashion and can provide predictions for homogeneous, chronologically aligned time series sharing the same set of covariates, together with performing variable selection on the regressors and employing an attention-based mechanism to capture long-term dependencies present in the entities. Furthermore, the TFT simultaneously outputs forecasts for τ_{max} time steps, by incorporating all past information within a finite look-back window k, using target and known inputs only up till and including the forecast start time t, and known inputs across the entire range. The model's architecture can be seen in Figure 5.2.

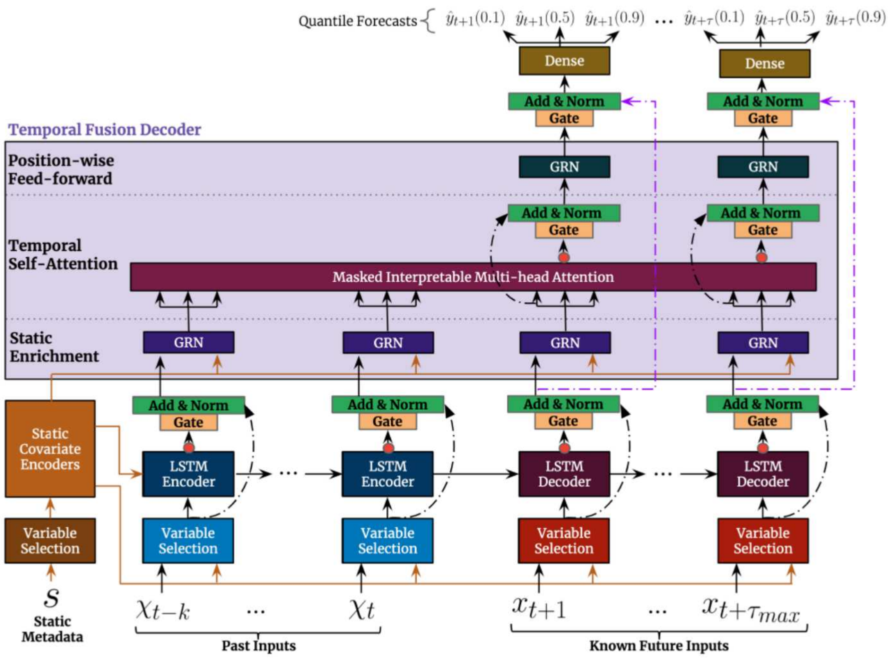

Figure 5.2: TFT architecture.

The building block of the TFT is represented by the Gated Residual Network (GRN), which provides the model with the flexibility to apply non-linear processing only where needed. This property is particularly useful given that the precise relationship between exogenous inputs and targets is often unknown in advance, making it difficult to anticipate which variables are relevant. Moreover, it can be difficult to determine the extent of required non-linear processing, and there may be instances where simpler models can be beneficial. As shown in Figure 5.3, the GRN processes a primary input a and an optional context vector c, which are then fed to a dense layer employing the Exponential Linear Unit (ELU) activation function. Then, when the input is significantly less than 0, the activation generates a constant output, resulting in linear behavior. Conversely, if the input is significantly greater than 0, the ELU activation acts as an identity function. After that, component gating layers based on Gated Linear Units (GLUs) are used to provide the flexibility to suppress any parts of the architecture that are not required for a given dataset. The gating mechanism is then defined as in Eq. 5.1:

$$\mathrm{GLU}(\mathbf{x}) = \mathbf{x} \odot \sigma(\mathbf{W}_g\mathbf{x} + \mathbf{b}_g) \tag{5.1}$$

where $\odot$ denotes element-wise multiplication, σ is the sigmoid activation function and $\mathbf{W}_g$, $\mathbf{b}_g$ are learnable parameters. GLU then allows TFT to control the extent to which the GRN contributes to the original input, potentially skipping over the layer entirely if necessary as the GLU outputs could be all close to 0 in order to suppress the nonlinear contribution. Lastly, during training dropout is applied before the gating layer.

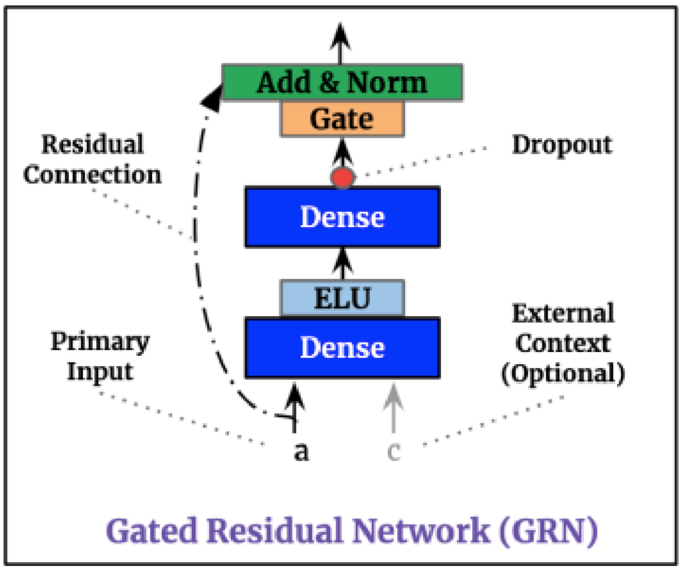

Figure 5.3: Gated Residual Network architecture.

Another prominent feature of the TFT model is the ability to provide instance-wise variable selection through the use of variable selection networks. These networks operate on both static and time-dependent variables, allowing the TFT to eliminate unnecessary noisy inputs that could harm performance. Notably, separate variable selection networks with distinct weights handle static, past, and future inputs. The general architecture of such networks is displayed in Figure 5.4. Before entering the network, each of the numerical variables is transformed through linear transformations into a (d_model)-dimensional vector ξ. This dimension, which is also referred to as hidden size, is one of the most important hyperparameters in the model and represents the size at which inputs will be processed throughout the architecture. On the other hand, categorical variables are dealt with with entity embeddings [19]. Then, at every timestep t input embeddings for each variable j ($j = 1, \ldots, m_X$) are linearly concatenated in a vector

Ξ_t. This vector, together with an external context vector c_s obtained from embedding the static covariates (thus not needed by the variable selection network dedicated to the static variables), is used to generate the variable selection weights through a GRN followed by a Softmax layer. Moreover, an additional layer of non-linear processing is employed by feeding each individual variable input embedding at timestep t to specific GRNs. This means then that each variable has its own GRN, with weights shared across all time steps. Lastly, the processed inputs are weighted and combined by means of the previously generated weights.

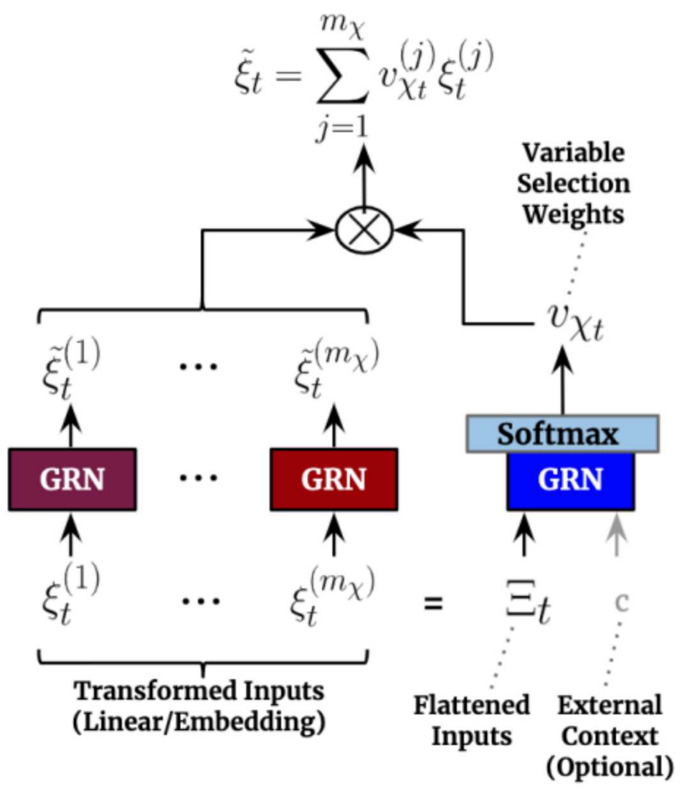
$\tilde{\xi}_t = \sum_{j=1}^{m_\chi} v_{\chi t}^{(j)} \tilde{\xi}_t^{(j)}$
Variable Selection Weights
$v_{\chi t}$
$\tilde{\xi}_t^{(1)}$
$\tilde{\xi}_t^{(m_\chi)}$
Softmax
GRN
GRN
GRN
$\xi_t^{(1)}$
$\xi_t^{(m_\chi)}$
=
Ξ_t
c
Transformed Inputs (Linear/Embedding)
Flattened Inputs
External Context (Optional)

Figure 5.4: Variable selection network architecture.

Regarding the static variables, the output of their variable selection network is fed to static covariates encoders, as can be seen looking at the architecture of the TFT showcased in Figure 5.2. These encoders are separate GRN encoders used to produce four different context vectors c_s, c_e, c_c and c_h, which are then wired into various locations in the temporal fusion decoder where static variables play an important role in processing. Particularly:

- c_s is used in the variable selection networks;
- c_c and c_h are needed for the local processing of temporal features;

- c_e is used to enrich temporal features with static information.

Following the data flow along the architecture, the outputs of the variable selection networks at each timestep t are then subject to temporal processing in a sequence-to-sequence layer, implemented by a Long Short Term Memory (LSTM) encoder-decoder architecture. Moreover, to allow static metadata to influence local processing, the c_h and c_c context vectors from the static covariate encoders are used to initialize the cell state and hidden state respectively for the first LSTM in the layer. This layer then generates a set of uniform temporal features which serves as input for the temporal fusion decoder itself, denoted by $\phi(t,n) \in \phi(t,-k),\ldots,\phi(t,\tau_{max})$, with n being a position index indicating the offset in time steps from the forecast time. Leveraging local context, these features can then provide pattern information on top of point-wise values. Furthermore, a gated residual over this layer is employed, to reduce the complexity of the model if needed.

After this, the data flows into the Static Enrichment layer, the first layer of the transformer's decoder. Here, the temporal features are further enriched with static metadata. Then, for a given position n static enrichment takes the form:

$$\theta(\mathbf{t},\mathbf{n}) = \mathrm{GRN}_\theta(\phi(\mathbf{t},\mathbf{n}), \mathbf{c_e}) \tag{5.2}$$

where the weights of GRN$_\theta$ are shared across the entire layer and c_e is a static context vector coming from a static covariate encoder.

Following static enrichment, all static-enriched temporal features are first grouped into a single matrix (i.e. $\theta(t) = [\theta(t,-k),\ldots,\theta(t,\tau_{max})]^T \in R^{d_{model}\times(k+1+\tau_{max})}$), and interpretable multi-head attention is applied at each forecast time obtaining as output a matrix $B(t) = [B(t,1),\ldots,B(t,\tau_{max})] \in R^{\tau_{max}\times d_{model}}$. Besides preserving causal information flow via masking, the self-attention layer allows TFT to pick up long-range dependencies that may be challenging for Recurrent Neural Network (RNN) architectures to learn. Following the self-attention layer, an additional gating layer is also applied to facilitate training.

Furthermore, additional non-linear processing is applied to the outputs of the self-attention layer through a position-wise feed-forward layer, made of sharing weights GRNs applied to each positional index $n \in \{1,\ldots,\tau_{max}\}$. Lastly, it is applied a gated residual connection that skips over the entire transformer block, providing a direct path to the sequence-to-sequence layer, thus yielding a simpler model if additional complexity is not required. The model, then, can simply output the τ_{max}-step-ahead punctual forecasts or can produce quantile forecasts, obtained by means of a simple linear transformation, for a set of specified percentiles.

5.1.1 Hyperparameters tuning and training procedure

To find the best architecture for the case in use and evaluate the final forecasts, all the time series have been partitioned into three parts: a training set for learning, a validation set for hyperparameters tuning, and a hold-out test set for performance evaluation, both counting 6 observations. Furthermore, it was decided to train the model only on the last 16 training observations, in order to reduce the computational cost and thus the time required to perform training and selection. This choice was also motivated by the characteristics of the data, which only included time series that had at least one week of sales recorded in 2024. Moreover, as described in Section 3.2, most of these time series have less than 27 observations, with 40% spanning less than 14 weeks. Therefore, limiting the consideration to only 22 observations (both for the selection and the performance evaluation) helped mitigate the effects of zero-padding applied to those with fewer than 16 training observations. Regarding the covariates, the past and future ones were normalized in order to favor convergence and thus speed up the training time. On the other hand, the static covariates were encoded with an ordinal encoder, which assigns an integer value to each distinct covariate, and fed in input to the model to generate the needed embeddings.

Parameter	Proposed	Best
Input chunk	$[2, 6, 8, 16]$	4
Output chunk	$[6]$	6
Hidden size	$[8, 16]$	8
LSTM layers	$[1, 4]$	1
Attention Heads	$[1, 4]$	4
Batch size	$[128, 256, 512]$	256
Learning rate	$[1e^{-5}, 1e^{-1}]$	0.001
Dropout	$[0.1, 0.3]$	0.018

Table 5.1: Hyperparameters space and best values obtained after tuning.

Hyperparameters tuning was conducted through *Optuna* [20], a Python library that simplifies the process of finding the best hyperparameter settings for a model by treating it as a

Bayesian optimization problem. The values proposed and the ones selected can be seen in Table 5.1. The training, on the other hand, was performed using *Darts* [21], a Python library that fits cross-learning models by creating from the provided time series a dataset of input/output examples having 'Input chunk' and 'Output chunk' lengths respectively. Each of these pairs, then, represents a sub-sequence of a training time series. Thus, the training dataset is obtained by aggregating all the consecutive pairs of input/output sub-sequences existing in each series.

5.2 Top-Down Reconciliation

Hierarchical forecasting deals with forecasting large collections of time series that aggregate in some way. The challenge, then, is that said forecasts should be coherent across the entire aggregation structure. That is, they should add up in a manner that is consistent with the aggregation structure of the hierarchy that defines the collection of time series [17]. Traditionally, forecasts of hierarchical time series involved selecting one level of aggregation and generating forecasts for that level. These are then either aggregated for higher levels or disaggregated for lower levels, to obtain a set of coherent forecasts for the rest of the structure.

Given the fact that the goal is to improve the forecasts of the store-SKU combinations, which represent the lowest level of the hierarchy as can be seen in Figure 5.1, it was then decided to focus on the top-down approaches, which involve first generating forecasts for the "Total" series and then disaggregating these down the hierarchy. This is achieved by computing a set of disaggregation proportions $p_1, \dots, p_m$ which determine how the forecasts of the "Total" series are to be distributed to obtain forecasts for each series at the bottom level of the structure. The most common approaches specify these proportions based on the historical proportions of the data [22]:

- In the Average Historical Proportions method, the proportions are computed as displayed in Eq. 5.3:
$$p_j = \frac{1}{T} \sum_{t=1}^{T} \frac{y_{j,t}}{y_t} \tag{5.3}$$
for $j = 1, \dots, m$. Each proportion p_j reflects the average of the historical proportions of the bottom-level series $y_{j,t}$ over the period $t = 1, \dots, T$.

- In the Proportions of the Historical Averages method, Each proportion p_j, for $j = 1, \dots, m$, captures the average historical value of the bottom-level series $y_{j,t}$ relative to the average value of the total aggregate y_t, as shown in Eq. 5.4:

$$p_j = \frac{\sum_{t=1}^{T} \frac{y_{j,t}}{T}}{\sum_{t=1}^{T} \frac{y_t}{T}} \tag{5.4}$$

The advantage of these methods is their simplicity, since it is only needed to produce forecasts at the top of the hierarchy to obtain coherent forecasts at the lower levels. On the other hand, using such top-down approaches it is not possible to capture and take advantage of individual series characteristics. Moreover, given that historical proportions methods do not take account of how those proportions may change over time, top-down approaches based on historical proportions tend to produce less accurate forecasts at lower levels of the hierarchy. Given this, and considering that the store-SKU forecasts can be utilized in this scenario, the proportions were calculated based on these forecasts rather than historical data, since this approach indeed requires generating h-step-ahead forecasts of all the time series at all levels. Then, for each level of the hierarchy, starting from the top and moving to the bottom, the proportion of each individual forecast to the aggregate of all the individual forecasts at this level is computed. In the end, then, for a K-level hierarchy each proportion can be computed as displayed by Eq. 5.5:

$$p_j = \prod_{l=0}^{K-1} \frac{\hat{Y}_j^l(h)}{\hat{S}_j^{l+1}(h)} \tag{5.5}$$

where:

- $\hat{Y}_j^l(h)$ is the h-step-ahead initial forecast of the series that corresponds to the node which is l levels above j;
- $\hat{S}_j^{l+1}(h)$ is the sum of the h-step-ahead initial forecasts below the node that is l levels above node j and are directly connected to that node.

These forecast proportions disaggregate the h-step-ahead initial forecast of the ”Total” series to get h-step-ahead coherent forecasts of the bottom-level series.

6 Results

This chapter deals with the analysis of the forecasts obtained by the TFT model, evaluated through the metrics discussed in Section 6.1 and compared to the ones obtained by a Naïve forecasting model, thus used as a baseline. Specifically, Section 6.2 examines the results of the base TFT forecasts with respect to the ones obtained by the Naïve model, while Section 6.3 explores the effect of hierarchical forecasting on the base forecasts. Furthermore, in both Section 6.2 and Section 6.3 performance is evaluated not only at a general level but also within individual forecastability classes, providing useful insights into the model's behavior and forecasts accuracy.

6.1 Evaluation approach

Three evaluation metrics, chosen given their ability with sporadic time series [23], have been used to assess the quality of the generated forecasts:

- The Mean Absolute Error (MAE), computed as in Eq. 6.1. Given a time series of length n, it calculates the average absolute difference between the true values (y_i) and the forecasted ones ($\hat{y}_i$). Then, it is an unsigned error metric that provides an average estimate of the magnitude of forecast errors. Thus, a lower MAE value indicates a more accurate model;

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{6.1}$$

- The Root Mean Squared Error (RMSE), computed as in Eq. 6.2. Given a time series of length n, it calculates the square root of the mean squared error between the true values (y_i) and the forecasted ones ($\hat{y}_i$). Then, the RMSE is a metric that is related to the Mean Squared Error (MSE). However, unlike the MSE, the RMSE is expressed in the same units as the forecasted values, which makes it more intuitive and easier to interpret;

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{6.2}$$

- The Cumulative Forecast Error (CFE), computed as in Eq. 6.3. Given a time series of length n, it calculates the cumulative difference between the forecasted values ($\hat{y}_i$) and the real observations (y_i). This metric is especially effective for evaluating the performance of forecasting models dealing with intermittent time series: positive values of CFE signify overestimation, while negative values denote underestimation. This characteristic of CFE then enhances its interpretability, making it highly valuable also from a business perspective.

$$\text{CFE} = \sum_{i=1}^{n}(\hat{y}_i - y_i) \tag{6.3}$$

Other metrics usually employed to assess forecasts quality, like the Mean Absolute Percentage Error (MAPE), could not be used given that percentage error-based measures have the disadvantage of being infinite or undefined if there are zero values in a series, as in the case of intermittent time series. Similarly, scaled errors like the Mean Scaled Absolute Error were not used given that for the most part the time series are on the same scale. Moreover, in some cases all the historical values of a dataset's time series are equal, thus making the MASE value infinite or undefined [23].

6.1.1 Baseline model

In order to gain a deeper understanding of the TFT model's performance, a Naïve model is employed as a comparative benchmark. This model simply forecasts the last observed value for future data points and can provide valuable insights when dealing with intermittent time series [24]. This is because the last observed value can often be zero in such series, and predicting zero sales for future periods can be a reasonable and safe guess given their sporadic nature. However, while the Naïve model provides a useful benchmark, it lacks the ability to capture more complex patterns and dependencies that may exist in the data. By comparing the performance of the TFT model with that of the Naïve model, it is then possible to better understand the strengths and weaknesses of the TFT model, and assess whether its additional complexity leads to improved forecast accuracy.

6.2 Initial forecasts

Figure 6.1 and Figure 6.2 showcase the distribution of the MAE, RMSE and CFE scores computed over the forecasts of all the $28, 729$ training time series, generated by the Naïve model and the TFT model respectively. Looking at the CFE histograms, it is easy to notice how the Naïve model largely overestimates in a consistent number of cases the number of articles sold during the 6 test weeks, while the TFT model scores more conservative results in terms of both overestimation and underestimation. This behavior is then reflected by a lower mean CFE value for the TFT model with respect to the Naïve's one, as reported in Table 6.1. Furthermore, the TFT model outperforms the Naïve one also in terms of mean MAE. This outcome is quite expected, given the CFE results. Lastly, the Naïve model scores a better result in terms of RMSE, meaning that on average it makes smaller errors in its forecasts compared to the TFT model.

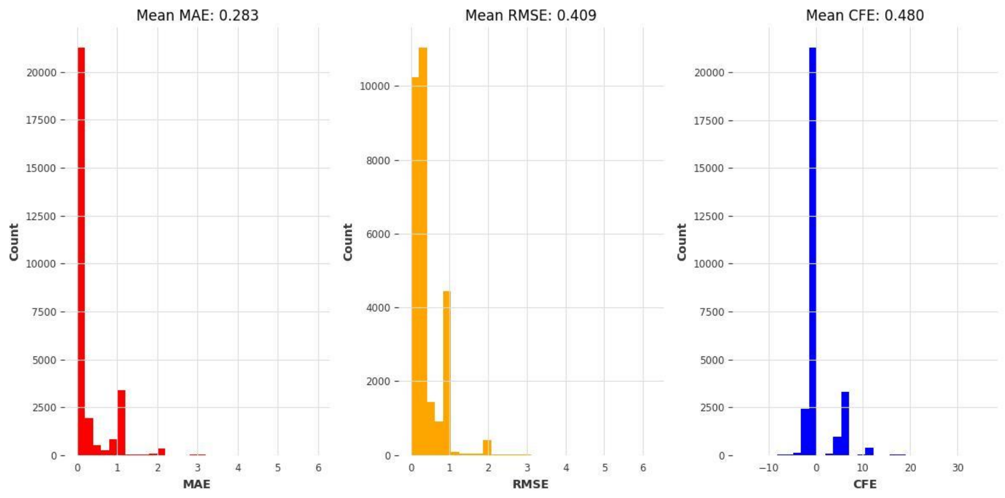

Figure 6.1: MAE, RMSE and CFE scores of the Naïve model.

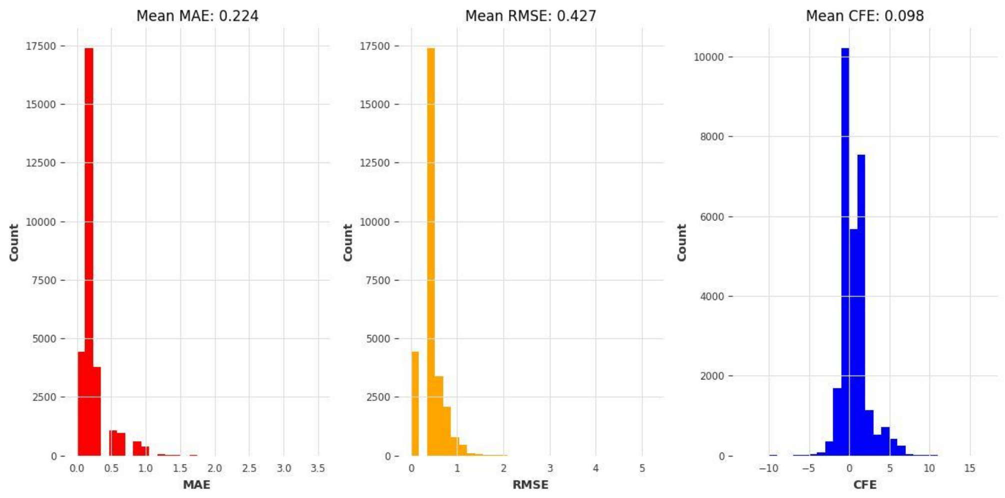

Figure 6.2: MAE, RMSE and CFE scores of the TFT model.

Model	MAE	RMSE	CFE
Naïve	0.283	**0.409**	0.480
TFT	**0.224**	0.427	**0.098**

Table 6.1: MAE, RMSE, CFE scores comparison.

Yet, given the substantial presence of lumpy time series, conducting a comprehensive analysis of the forecasts by examining the average scores across the entire dataset can be challenging. To address this, the MAE, RMSE and CFE scores have been computed separately for all four forecastability classes, discussed in Section 2.2, and for those time series having just one week with non-zero sales. Table 6.2 and Table 6.3 report the results for the Naïve model and the TFT model respectively. It is evident that both models perform the worst on the erratic time series. However, while the MAE and RMSE scores are comparable for the two models, in terms of CFE the TFT model significantly outperforms the Naïve one. This pattern is also reflected in the other three classes, with the TFT model achieving the best results in terms of CFE and MAE, while the Naïve model does slightly better in terms of RMSE. This behavior is likely due to the fact that the Naïve model's forecasts represent a constant line, whereas the TFT model attempts to understand and capture the underlying patterns and reasons behind the behavior of the time series. Regarding the time series with just one week with non-zero sales, it must be noted how the Naïve model scores a CFE of 0.036. This is an expected result, given the very nature of these time series, counting just one non-zero observation.

	MAE	RMSE	CFE
Smooth	0.437	0.693	-2.470
Intermittent	0.494	0.702	-2.815
Lumpy	0.328	0.442	0.771
Erratic	2.452	2.827	6.143
1-sale	0.204	0.348	0.036

Table 6.2: MAE, RMSE, CFE scores of the Naïve model for each forecastability class.

	MAE	RMSE	CFE
Smooth	0.434	0.689	-1.53
Intermittent	0.501	0.707	-2.312
Lumpy	0.256	0.448	0.294
Erratic	1.524	1.982	-3.971
1-sale	0.167	0.389	-0.187

Table 6.3: MAE, RMSE, CFE scores of the TFT model for each forecastability class.

Figure 6.3, Figure 6.4 and Figure 6.5 showcase the forecasts computed by both models for, respectively, a lumpy time series, an erratic time series and a smooth time series:

- In the first case (Figure 6.3), it can be noticed how the Naïve model nails the forecast. As said in Section 6.1.1, this kind of model can be indeed useful when dealing with sporadic time series. On the other hand, the TFT does not produce an unreasonable forecast, predicting just one article sold;

- In the second case (Figure 6.4), the TFT model outperforms the Naïve one, which highlights its limitation. Yet, the TFT fails to capture the surge in demand of the first test week, causing then a consistent underestimation of sold articles.

- In the last case(Figure 6.5), both models produce reasonable forecasts, even though the TFT model is able to correctly predict the right number of sales.

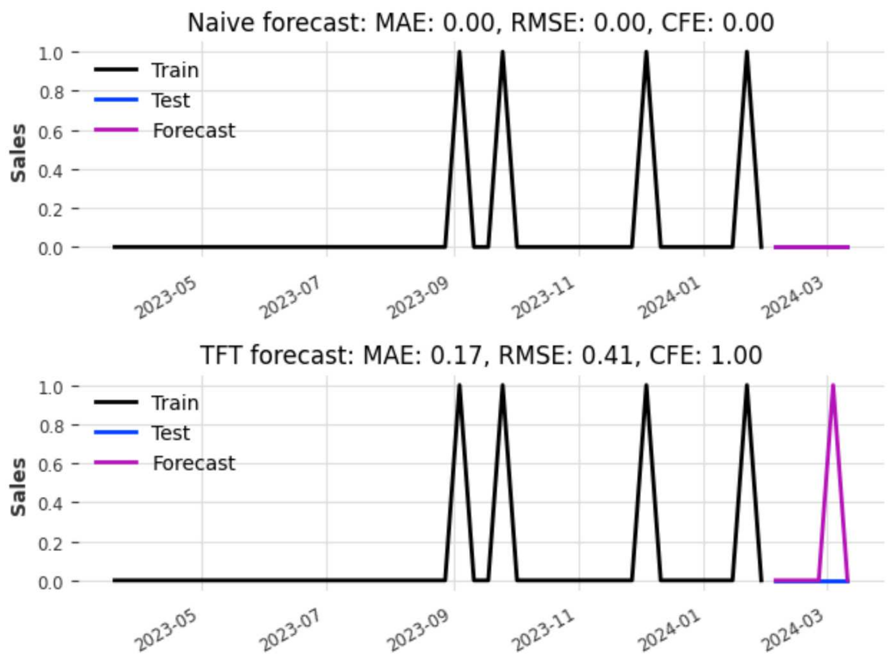

Figure 6.3: Example of lumpy time series forecasts performed by the Naïve and TFT model.

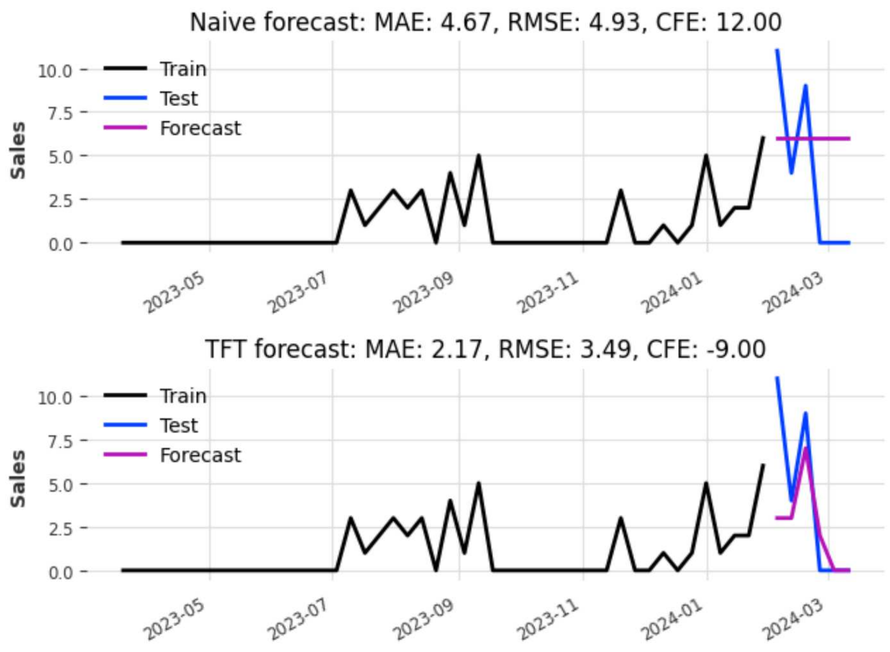

Figure 6.4: Example of erratic time series forecasts performed by the Naïve and TFT model.

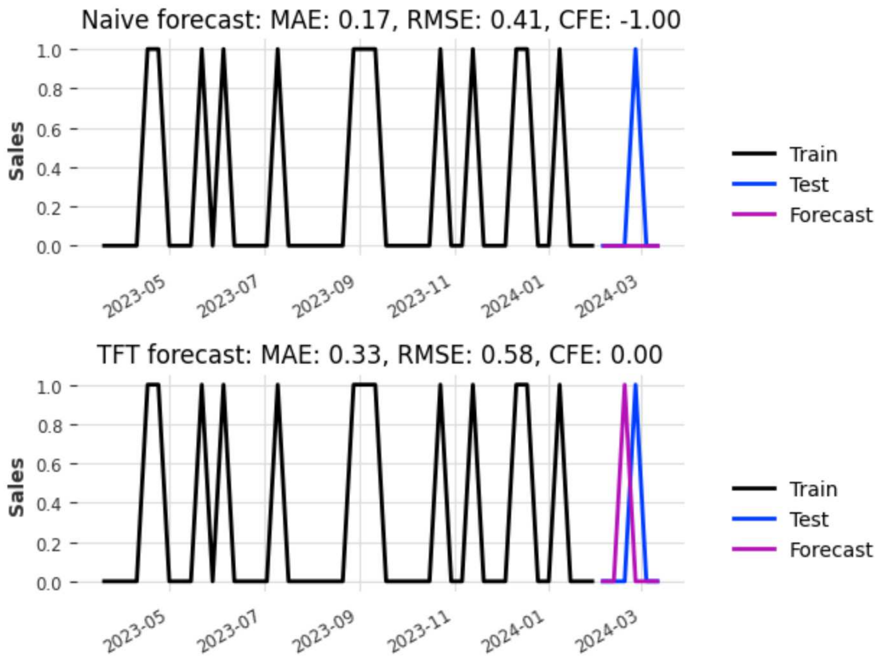

Figure 6.5: Example of smooth time series forecasts performed by the Naïve and TFT model.

6.3 Aggregate forecasts

As explained in Chapter 5, the forecasts at all levels of aggregation are needed to perform hierarchical forecasting. These aggregations were computed twice, summing only the time series of the "1-in-2024" dataset, defined in Chapter 2, and considering all the store-SKU combinations of the original dataset. In both scenarios, each of the 653 aggregate time series had an ARIMA model fitted to it.

Table 6.4 and Table 6.5 display the results obtained for the "1-in-2024" dataset and the original one respectively. At all levels, the forecasts obtained over the "1-in-2024" dataset perform better. Given that top-down hierarchical forecasting works by disaggregating the Total forecasts down the hierarchy, as discussed in Chapter 5, it was then decided to use the "1-in-2024" dataset forecasts.

	MAE	RMSE	CFE
Total	1931	2000	11583
Store level	130	149	−551
Customer level	30	25	−49
Category level	13	15	−26
Subcategory level	4.72	4.81	−3.41

Table 6.4: MAE, RMSE, CFE scores at all levels of aggregations for the "1-in-2024" dataset.

	MAE	RMSE	CFE
Total	2243	2281	13455
Store level	234	254	−1403
Customer level	38	42	−219
Category level	21	23	−116
Subcategory level	4.66	5.28	−22.02

Table 6.5: MAE, RMSE, CFE scores at all levels of aggregations for the original dataset.

6.4 Post-Reconciliation forecasts

Figure 6.6 displays the distribution of the MAE, RMSE and CFE values computed for the reconciled TFT forecasts.

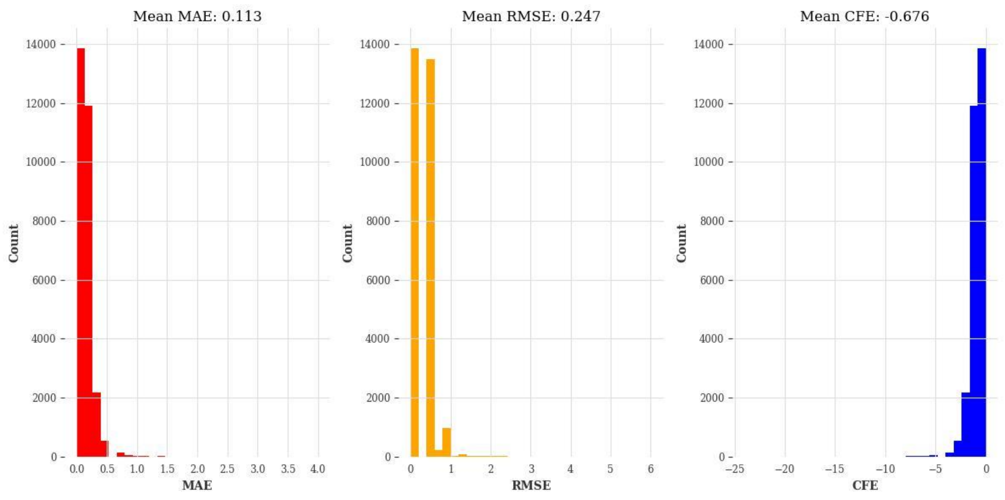

Figure 6.6: MAE, RMSE and CFE of the TFT forecasts after Top-Down Reconciliation.

With respect to the base forecasts, it can be noticed an improvement in terms of average MAE and RMSE, even though it must be registered a noticeable decrease in terms of CFE. Moreover, the CFE histograms highlight how all the forecasts now underestimate the number of sales during the test 6 weeks. This behavior is mirrored by the results filtered according to the type of time series, as reported in Table 6.6. Particularly, it is easy to notice the worsened performance of the erratic time series, which now highly underestimate the articles sold, and, on the other hand, the improvement, particularly in terms of the RMSE, over the lumpy time series, which is the most represented class in the dataset. Lastly, the 1-sale forecasts display a great MAE score, even though their performance in terms of CFE dropped with respect to the base forecasts.

	MAE	RMSE	CFE
Smooth	0.446	0.546	−2.675
Intermittent	0.506	0.711	−3.037
Lumpy	0.117	0.153	−0.704
Erratic	1.976	2.492	−11.857
1-sale	0.099	0.243	−0.595

Table 6.6: MAE, RMSE, CFE scores of the TFT model for each forecastability class post reconciliation.

In Figure 6.7it is possible to observe how the reconciled forecasts behave in comparison to the base forecasts shown in Figure 6.4, Figure 6.5 and Figure 6.6. It is then evident the flattening action performed by the hierarchical forecasting process, which can help mitigate overestimation, as in the "Lumpy" case, or can worsen the final result, as in the "Smooth" scenario.

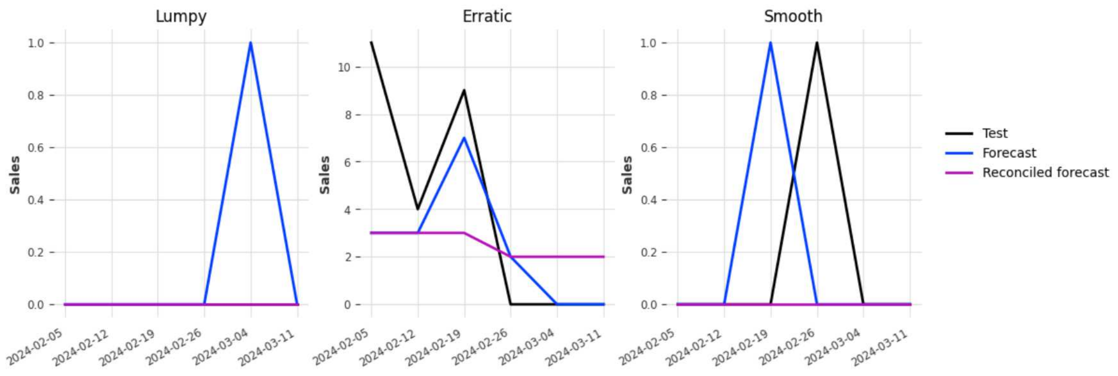

Figure 6.7: Examples of lumpy, erratic and smooth time series reconciled forecasts.

7 Conclusion

In summary, this project offers a practical solution to real-world challenges that characterize the retail fashion industry, particularly related to handling sporadic and highly intermittent time series data. To tackle these issues, the forecastability-based classification of the data was taken into account in order to understand its impact on model performance.

Specifically, the work focuses on improving the accuracy and reliability of forecasts for volatile fashion products, which play a crucial role in decision-making processes within the dynamic fashion retail sector. To achieve this, it was employed an advanced forecasting model known as the Temporal Fusion Transformer, which represents an evolution of the original Transformer architecture proposed with the aim of performing time series forecasting. Particularly, it has been shown how the TFT forecasts outperform the ones produced by a Naïve forecasting model. Additionally, an attempt has been made to enhance forecasts further by incorporating internal business logic into the forecasting process through top-down hierarchical forecasting. However, it is worth noting that this approach resulted in an overall decrease in performance, probably due to the data behavior and the accuracy of the forecasts at higher levels of the hierarchy. Nevertheless, this project tried to bridge the gap between theoretical advancements and practical applications in fashion retail data forecasting, contributing valuable insights to decision-makers.

Further improvements can be carried out. For example, another possible solution to be explored could tackle the problem of high variability between each forecastability class by training a Temporal Fusion Transformer model on each partition of the original training set, where analogous time series belong to the same subset [25]. Furthermore, different types of of hierarchical forecasting could be tried out, like optimal forecast reconciliation through trace minimization [26]. Lastly, future research could be conducted into alternative forecasting models like Temporal Convolutional Networks [27] and other transformer-based models like the Channel-Aligned Robust Dual Transformer [28].

References

[1] F. M. Mahya Seyedan, "Predictive big data analytics for supply chain demand forecasting: methods, applications, and research opportunities," *Journal of Big Data*, vol. 7, no. 53, 2020.

[2] A. Şen, "The us fashion industry: A supply chain review," *International Journal of Production Economics*, vol. 114, no. 2, pp. 571–593, 2008, special Section on Logistics Management in Fashion Retail Supply Chains.

[3] K. Swaminathan and R. Venkitasubramony, "Demand forecasting for fashion products: A systematic review," *International Journal of Forecasting*, vol. 40, no. 1, pp. 247–267, 2024.

[4] W. Y. C. A. Türkmen AC, Januschowski T, "Forecasting intermittent and sparse time series: A unified probabilistic framework via deep renewal processes," *PLoS ONE*, 2021.

[5] O. Besbes and A. Muharremoglu, "On Implications of Demand Censoring in the Newsvendor Problem," *Management Science*, vol. 59, no. 6, pp. 1407–1424, June 2013.

[6] E. Ozhegov and D. Teterina, *Ensemble Method for Censored Demand Prediction*, October 2018.

[7] Massimo Andreetta. Machine and deep learning applications for inventory replenishment optimization. [Online]. Available: https://hdl.handle.net/20.500.12608/61374

[8] A. A. S. J E Boylan and G. C. Karakostas, "Classification for forecasting and stock control: a case study," *Journal of the Operational Research Society*, vol. 59, no. 4, pp. 473–481, 2008.

[9] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[10] L. Ge and S. Chen, "Exact dynamic time warping calculation for weak sparse time series," *Applied Soft Computing*, vol. 96, p. 106631, 2020.

[11] P. Senin, "Dynamic time warping algorithm review," 01 2009.

[12] F. Petitjean, A. Ketterlin, and P. Gançarski, "A global averaging method for dynamic time warping, with applications to clustering," *Pattern Recognition*, vol. 44, no. 3, pp. 678–693, 2011.

[13] J. Brownlee. Why one-hot encode data in machine learning? [Online]. Available: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning

[14] P. Ramos and J. M. Oliveira, "Robust sales forecasting using deep learning with static and dynamic covariates," *Applied System Innovation*, vol. 6, no. 5, 2023.

[15] K. H. Lee, M. Abdollahian, S. Schreider, and S. Taheri, "Supply chain demand forecasting and price optimisation models with substitution effect," *Mathematics*, vol. 11, no. 11, 2023. [Online]. Available: https://www.mdpi.com/2227-7390/11/11/2502

[16] B. Lim, S. Arık, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *International Journal of Forecasting*, vol. 37, no. 4, pp. 1748–1764, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207021000637

[17] R. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*, 3rd ed. OTexts: Melbourne, Australia., 2021.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023. [Online]. Available: https://arxiv.org/abs/1706.03762

[19] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Neural Information Processing Systems*, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:15953218

[20] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," 2019. [Online]. Available: https://arxiv.org/abs/1907.10902

[21] J. Herzen, F. LÃ¤ssig, S. G. Piazzetta, T. Neuer, L. Tafti, G. Raille, T. V. Pottelbergh, M. Pasieka, A. Skrodzki, N. Huguenin, M. Dumonal, J. KoÅ›cisz, D. Bader, F. Gusset, M. Benheddi, C. Williamson, M. Kosinski, M. Petrik, and G. Grosch, "Darts: User-friendly modern machine learning for time series," *Journal of Machine Learning Research*, vol. 23, no. 124, pp. 1–6, 2022. [Online]. Available: http://jmlr.org/papers/v23/21-1177.html

[22] C. W. Gross and J. E. Sohl, "Disaggregation methods to expedite product line forecasting," *Journal of Forecasting*, vol. 9, pp. 233–254, 1990. [Online]. Available: https://api.semanticscholar.org/CorpusID:154568362

[23] R. Hyndman, "Another look at forecast accuracy metrics for intermittent demand," *Foresight: The International Journal of Applied Forecasting*, vol. 4, pp. 43–46, 01 2006.

[24] P. Goodwin, "Using naave forecasts to assess limits to forecast accuracy and the quality of fit of forecasts to time series data," *SSRN Electronic Journal*, 01 2014.

[25] A.-A. Semenoglou, E. Spiliotis, S. Makridakis, and V. Assimakopoulos, "Investigating the accuracy of cross-learning time series forecasting methods," *International Journal of Forecasting*, vol. 37, 12 2020.

[26] G. A. Shanika L. Wickramasuriya and R. J. Hyndman, "Optimal forecast reconciliation for hierarchical and grouped time series through trace minimization," *Journal of the American Statistical Association*, vol. 114, no. 526, pp. 804–819, 2019. [Online]. Available: https://doi.org/10.1080/01621459.2018.1448825

[27] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018. [Online]. Available: https://arxiv.org/abs/1803.01271

[28] W. Xue, T. Zhou, Q. Wen, J. Gao, B. Ding, and R. Jin, "Card: Channel aligned robust blend transformer for time series forecasting," 2024. [Online]. Available: https://arxiv.org/abs/2305.12095