University of Padova

Department of Mathematics

Master Thesis in Data Science

Multiple Time Series Forecasting with Temporal Fusion Transformers

Supervisor
Professor Mariangela Guidolin
University of Padova

Master Candidate
Gaia Ziraldo

Academic Year
2022-2023

This thesis is dedicated to my grandparents, which support and words of encouragement have accompanied me for the entirety of this academic journey. I hope this achievement can make them proud.

Abstract

The goal of this thesis is to present the Temporal Fusion Transformer model and to evaluate its forecasting capabilities across multiple time series. Its contribution to the field of multi-horizon, multiple time series forecasting is explored, with great focus on the interpretability feature offered by the model. It is observed how improvements to the model performances can be achieved when paired with a form of clustering on the target entities, either by exploiting the natural categorization of the time series considered or by associating similar entities by means of a clustering algorithm on the target variable.

Contents

Listing of figures

Listing of tables

Listing of acronyms

TFT ..... Temporal Fusion Transformer

GLU ..... Gated Linear Units

GRN ..... Gated Residual Networks

LSTM ..... Long Short Term Memory network

1 Introduction

Multiple time series forecasting, i.e. the task of performing forecasts over several time series at once, is a problem which has been predominantly tackled by statistical forecasting models, but has recently been taken into consideration in the machine learning field as well. In many real-world cases it arises the need to provide multi step-ahead forecasts for an ensemble of time series. A large-scale retail brand for example, may require monthly forecasts for the sales of all its products, or an infrastructure company needs traffic flow predictions on a set of regional driveways in order to plan maintenance interventions. The challenges posed by providing forecasts for a large set of time series data concern not only the accuracy of the predictions but also the computational burden required to perform such a task.

Generally the majority of models, either machine learning based or not, are able to provide forecasts for one time series at the time, thus limiting the scope of the prediction only to the data of a single entity without taking into account the global context or the possible interactions between similar time series. A recent addition to the forecasting scene however is designed to bypass such problem. Cross-learning models represent a family of forecasting methods capable to generate predictions for a set of time series simultaneously, and as such have been receiving increasing interest in the multiple time series forecasting field.

Another challenge that arises when tasked with forecasting is represented by model explainability. Most models function in a ”black-box” fashion, meaning that the computations within their architectures are complex and involve a great number of parameters, thus limiting the capability of the user to understand how the predictions were carried out and which features of

the time series had the most weight in the forecast. Especially in real world applications is required a tradeoff between model accuracy and forecast interpretability, thus prediction models capable of offering insights on the underlying relationships present in the data are extremely valued.

This thesis examines the contribution of Temporal Fusion Transformers, a cross-learning model introduced in 2020 by K. Bandara, C. Bergmeira and S. Smyl [1], to the multiple time series forecasting scenario. In particular, after examining the state of the art of multiple time series forecasting models in the first chapter, it focuses on its architecture and its interpretability mechanism. The last chapter presents the results of a series of experiments carried out with the support of the internship organization ACT Operations Research IT s.r.l., aiming to assess the Temporal Fusion Transformer performance in forecasting a real world retail dataset, comparing the results against two frequently employed statistical benchmarks. Moreover, given the cross-learning nature of the model and its capability of generating predictions over a set of entities requiring a single training pass, its performance was tested after combining it with two forms of partitioning on the training set as well, with the goal of ascertaining whether the employment of such approach could signify in an improvement of forecast accuracy and computational cost. The insights offered by the interpretability feature of the model were taken into account as well, and their value was weighted by the context in which the Temporal Fusion Transformer was trained.

2

State of the Art in Multiple Time Series Forecasting

Time series forecasting is a well-known task in the fields of statistics and machine learning, and has been tackled in many different ways across the years. Several models for multi-horizon forecasting have been developed, compared and thoroughly tested, each catering to different needs and situations. In real-world applications however it's often required to perform forecasts over a large number of time series, either homogeneous and chronologically aligned or widely unrelated to each other. Most importantly, in many occasions the number of time series to be handled is so large to not allow a one-by-one approach. In this fields, not only prediction accuracy is required, but also autonomy in model construction and computational complexity. The latter are relevant factors due to time being an essential resource in realistic settings. This chapter presents a literature overview of the state-of-the-art for multiple time series forecasting, presenting different approaches to the problem and highlighting their advantages and disadvantages.

For clarity, in this thesis the term *entity* will be used to identify a single time series with its relative regressors belonging to a set of N time series. A set of entities can be represented by a chain of stores, a set of products in a supermarket or a group of patients in healthcare for example.

This thesis originates from an internship activity carried out in 2022 at the company ACT Operations Research IT s.r.l., where I was allowed access to a wide library of forecasting models

and real world datasets. During the internship period I collaborated in the implementation and testing of the Temporal Fusion Transformer model, which results are presented in the following chapters.

2.1 Forecasting multiple time series

Multiple time series forecasting identifies the problem of providing predictions for a batch of time series. The definition is very general on the set of entities to be predicted: it can refer to a dataset of contemporary series sharing the same frequency as well as a portfolio of series widely different in nature. The considered entities can be univariate, thus presenting only one response variable, or multivariate, featuring a number of explanatory variables (or covariates) either fixed or time dependent in addition to the target.

There are many real-world scenarios in which multiple time series forecasting is applied. One of the prime examples is represented by the retail sector: in this field demand forecasting is often required for a very large set of products. Such predictions are necessary to manage the supply chain, reduce warehouse costs and to formulate proper purchase plans in line with the needs of the customers. In retail systems items are identified with a SKU, or Stock Keeping Unit, and are often organized in product categories at various levels of granularity. Time series are usually multivariate, with the demand variable being combined to information regarding promotions, holidays or store location for instance. Wagner et al. [2] developed a hybrid forecasting system currently used in a large-scale food distribution company, while [3] compared several statistical and Machine Learning models on the task of SKU demand forecasting. Another example of system which employs multiple time series forecasting is the manufacturing industry. Management systems in this sector feature thousands of articles distributed across one or more warehouses, thus making demand estimation for each finite product a necessity to plan purchases of components, workforce scheduling and inventory management. In this sector, depending on the industry, product time series are available in a wider granularity, usually weekly or monthly and are mostly univariate.

Energy management and planning in various types of buildings (residential, entertainment, business, industrial and more) has become a crucial problem requiring the support of electricity consumption forecasting. Energy load predictions at building level represent a form of multiple time series forecasting as well. Such scenario features data collected at high frequencies, mostly hourly or daily, and dependent from a wide variety of exogenous variables such as holidays, building use and destination, weather, energetic efficiency level of the structure, number and

behaviour of occupants, electricity price fluctuations and many others. The insights provided by energy forecasts at household/factory level can be used to optimize heating and ventilation systems, as well as to improve network stability. Another field in which forecasting multiple time series can support the decision making process is represented by the healthcare system. Several problems such as capacity planning for different wards in hospitals and predictive screening on high-risk patients require forecasting models able to identify extreme events, which can help the system improve its robustness under critical conditions. Additionally, healthcare systems can benefit from price or demand predictions on a set of frequently prescribed medications to better manage pharmaceutical supply chain, as suggested by [4]. Healthcare data can consist in both univariate and multivariate time series, in the second case possible covariates can represent weather conditions, genetic features of a patient, average occupancy and more. Traffic flow data on a set of roads and highways, either hourly aggregated occupancy rates or flow data at 5-minute intervals [5], represent another application for multiple time series forecasting. Traffic data are gathered through road sensors and are dependent on a wide variety of covariates such as geographical location, holidays, atmospheric conditions, accidents or planned road works. Forecasting traffic flow for a set of roads in a region can aid in planning maintenance operations and infrastructure management, while real-time forecasting gives drivers the opportunity to select better routes.

Multiple time series forecasting received high interest in the academic field as well: the Makridakis Competitions, M-Competitions for short, are a series of open forecasting competitions in which the participants are tasked with a multiple time series forecasting problem, having to provide predictions for a large number of time series, all belonging to different application domains and featuring different frequencies and forecasting horizons. The goal of such challenges was to empirically ascertain which method or family of methods was the most suited in forecasting a large and diverse dataset, testing the best practices for time series prediction and comparing the results on standard benchmark approaches. In particular, the M4 Competition [6] held in 2018 featured a dataset of 100.000 entities, the largest to date, and received 61 submissions.

In order to provide a concrete example of the variety of entities that can possibly constitute a multiple time series forecasting problem, in Figure 2.1 are illustrated the target variables of 7 time series present in a dataset offered by the internship organization, reporting daily demands for 15593 products of a retail chain. For reasons of corporate confidentiality it is offered a sample representation of the retail dataset by showing only few series which can however sufficiently convey the complexity and high variability between entities present in many multiple time series forecasting problems. It is possible to observe the pattern dissimilarity in the sam-

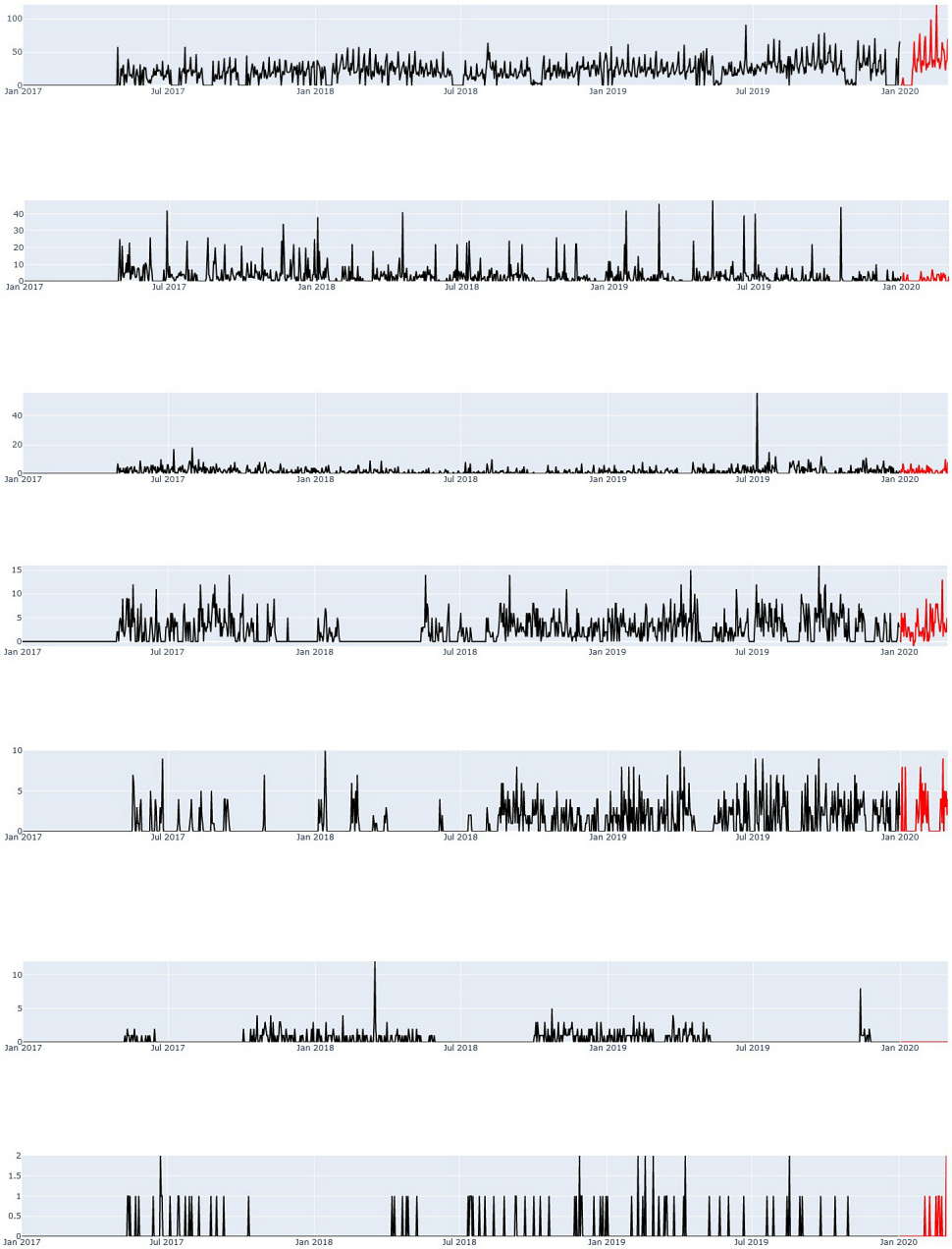

Figure 2.1: Sample representation of the target variable in the retail dataset offered by the internship organization.

ple, which comprises both series presenting large and frequent periods of zero demand and entities showing a smoother behaviour. In some entities the spikes are more regular in magnitude, while in other a strong dissimilarity can be noticed, seasonal behaviours are not consistent as well. In such cases tackling the task of forecasting a dataset as diverse as the one illustrated should involve a flexible approach, not relying on specific time series regimes.

The biggest challenges when forecasting multiple time series arise when the number of entities is very large, making manual model selection and construction infeasible in terms of time. When faced with a prediction task involving a huge quantity of targets, autonomy in model construction is highly valued. For this reason simple statistical models or global cross-learning models, which will be expanded upon in the following sections, are the most suited approaches to such task since they require less computational resources and are able to provide reliable results in terms of forecast accuracy. Another issue regarding the task of providing forecasts for multiple entities lies in possible relationships between considered time series. In the case in which the portfolio presents targets influencing each other, models performing predictions over one series at a time fail to capture such connections, possibly overlooking some temporal patterns shared between the entities. The following sections will present an overview of the forecasting approaches more suited to tackle the problem of multiple time series forecasting.

2.2 Statistical approaches

Statistical forecasting models for time series represent the first developed approaches for the task of providing predictions for historical data before the advent of Machine Learning techniques. This family includes a wide variety of methods and algorithmic procedures, from simple models such as Exponential Smoothing or Moving Averages, to more complex tools, all having in common the fact of relying on a sample of the available data to infer future values. More specifically, statistical models aim to represent the entire data population on the basis of a sample of variable size, performing algorithmic manipulations to reconstruct the underlying data process which is assumed to generate the distribution. Due to their nature, models belonging to this class utilize only a part of all historical data available, either by setting a sample size or lookback window or by assigning decreasing weights to older observations.

Exponential smoothing defines the one step-ahead forecast as the weighted average of all past observations, where the weights decrease exponentially the older the observations get. Different variations of Exponential Smoothing models have been formulated from Simple Exponential Smoothing (SES), where forecast at time $t + 1$ is defined as a convex combination between the

previous forecast f_t and the previous observation of the series y_t:

$$f_{t+1} = \alpha y_t + (1 - \alpha) f_t \tag{2.1}$$

In presence of time series presenting a trend, Holt's Exponential Smoothing was introduced, while when dealing with data having both a trend and a seasonal pattern (either additive or multiplicative) Holt-Winters model was formulated to capture such features. These are some of the many variants of the Exponential Smoothing method, each designed to deal with different types of situations. Despite its simplicity this class of models represents a reliable forecasting benchmark to this day.

ARIMA models are a class of statistical forecasting methods born from the combination of Autoregressive models (AR) with Moving Average models (MA) together with an integration feature. The basic concept behind these methods relies on representing a forecast as a linear combination of past observations of the differentiated time series and past forecast errors. More specifically, an $\mathrm{ARIMA}(p, d, q)$ model is defined as follows:

$$(1 - \phi_1 B - ... - \phi_p B^p)(1 - B)^d y_t = c + (1 - \theta_1 B - ... - \theta_p B^p)\varepsilon_t \tag{2.2}$$

where p indicates the level of the autoregressive part, q indicates the level of the moving average part and d represents the degree of integration. The lag operator B is defined as $B^d y_t = y_{t-d}$. A number of variants has been developed for this family as well, from SARIMA which is designed to model seasonal components of the time series as well, to ARIMAX, a multivariate version of the ARIMA model, capable to include explanatory variables in the forecast. ARIMA models represent a class of extremely versatile forecasting methods, having the feature to adapt to a wide variety of time series. Due to its adaptability and performances this family is still one of the most used in the field of single and multiple time series forecasting, either on its own, combined with other techniques or used as base forecast and further refined.

A large number of other forecasting models belong to the class of statistical approaches, each one catering to a specific case: the Croston method [7] for example is designed to model sparse and intermittent time series. The Vector Autoregression method [8], VAR for short, is a multivariate forecasting model used mostly in econometric scenarios and is able to provide predictions for multiple time series at once. The Theta model [9] on the other hand is a recently developed method based on curvature decomposition and known for its simplicity, flexibility and reliable performances. In particular it achieved remarkable results in the M3 and M4 com-

petitions, outperforming most of the pure Machine Learning models submitted.

Most statistical forecasting approaches such as ARIMA and Exponential Smoothing are univariate, i.e. they require as input only the target variable in order to perform the predictions. This feature leaves such methods at disadvantage with respect to multivariate models, since by considering only the target variable they do not exploit the information provided by other regressors. In general however, univariate statistical models compensate their neglect for exogenous variables with relatively accurate forecasts. Moreover, statistical forecasting methods are known for having considerably faster training times and lighter computational costs with respect to Machine Learning models, which in multiple time series scenarios represents a major advantage, especially when the number of entities is very large. In such cases, training a set of Machine Learning models and performing hyperparameter optimization on them becomes infeasible in terms of time and computational resources. Statistical models represent a valid and reliable alternative, having proven empirically on several occasions [10] to be a valid tradeoff between forecast quality and training complexity.

2.3 Cross-Learning approaches

Another approach to forecasting multiple time series is represented by cross-learning methods [11]. Traditional forecasting methods are trained and perform predictions in a series-by-series fashion, meaning that N independently trained models are required to forecast N different time series. Most statistical methods such as Exponential Smoothing, ARIMA models and Theta methods, as well as Machine Learning based methods like recurrent neural networks belong to this category. Cross-learning methods on the other hand can be used to forecast N time series simultaneously by training one single model on the entire dataset as illustrated in Figure 2.2.

Both approaches have their advantages and disadvantages. Pure machine learning models such as DNNs or sequence-to-sequence networks have proved extremely effective in specific tasks such as image classification (with the Resnet [12] architecture representing a solid benchmark for this task), Natural Language Processing (NLP), object detection and many others. Their performances when applied to time series forecasting however are often poor when compared to pure statistical models, which is surprising given the big difference in computational complexity between the two categories. The core difference between the two families of methods is that, unlike statistical models which assume an underlying data generating process, machine learning based models approximate said distribution by capturing non-linear relationships in the data [11]. For this reason they require a large amount of training samples in order

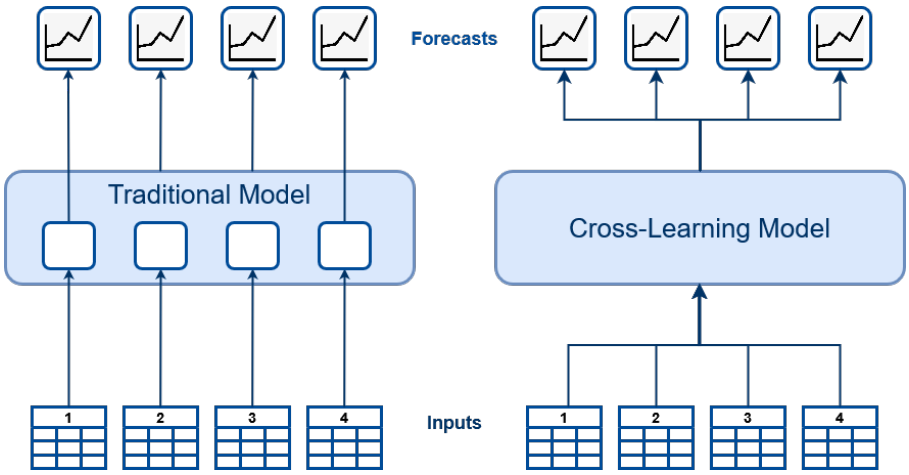

Figure 2.2: Difference between traditional and cross-learning models in handling multiple time series forecasting.

to properly infer their distribution and capture trend and seasonality. Since traditional machine learning models for forecasting are trained on one time series at a time, this translates in the necessity of long or high-frequency time series for training. Depending on the application, this requirement is often not satisfied, thus resulting in an unsatisfactory performance even when compared to forecasts provided by simpler and faster statistical models. Moreover as observed by [13], even if the considered time series are long and present a large number of data, in forecasting frameworks often occurs that the distant past has a considerably lower predictive relevance as temporal patterns and relationships will change after long periods. It follows that machine learning models trained in a series-by-series fashion are usually not able to outperform their statistical counterparts unless provided with very long and stable time series, due to their inability to distinguish between signal and noise when provided with scarce training samples.

One possible solution to improve this behaviour is to train machine learning models in a cross-learning fashion, allowing the methods access to a wider variety of time series and giving them the possibility to better capture global patterns. Another advantage gained from this implementation is a reduction in running times and computational complexity due to the fact that only one model needs to be fitted to the data instead of many. In general, especially when the available time series are short and/or present low frequency (monthly or yearly), which is a common occurrence in many real-world applications, machine learning models can greatly benefit from a cross-learning approach.

At the same time, while pure statistical methods applied in a series-by-series fashion constitute a solid and reliable choice for many forecasting tasks, they too can benefit from a cross-

learning approach. If combined with some machine learning component it can allow them to enrich their high-performance single-entity forecasts with global context from the whole dataset.

On the downside, cross-learning implementations also present the following limitations:

1. They require datasets populated with numerous time series in order to allow the model to capture meaningful relationships. Providing individual forecasts necessitates a large and possibly diverse set of series so that the method can be able to learn different patterns and take advantage of the similarity between previously observed inputs.

2. Machine learning methods trained with a cross-learning approach usually perform better when paired with different pre-processing techniques on the input data as highlighted by [11]. These techniques include but are not limited to: normalization, de-seasonalization, de-trending (which can be obtained by differencing) and feature extraction. Depending on the implementation, different pre-processing operations may be required and forecasting accuracy varies according to which of these techniques have been adopted. Thus the choice of optimal pre-processing methods gains more weight and adds complexity to the task of building a reliable cross learning model.

In conclusion, as remarked by [11], cross-learning is a technique that in determinate situations can greatly benefit the performances of machine learning models with respect to their series-by-series trained counterparts. Additional approaches such as enriching the inputs with feature extraction or the combination in a meta-learning fashion with traditional models can further improve the accuracy of such models. Another improvement strategy suggested by the article consists in balancing the training sample and/or applying some form of clustering on the data based on their particular characteristics.

2.3.1 Hybrid approaches

Another solution is the use of hybrid approaches, namely models that combine cross-learning methods with traditional forecasting methods. Both first and second place in the M4 competition [6] belong to this category. The first classified [14] is a model that combines exponential smoothing with Long Short Term Memory networks by means of a dynamic graph neural network trained globally on all time series. Different data pre-processing techniques including normalization and de-seasonalization are also applied prior to training. The second submission to the competition [15] proposes the FFORMA framework (Feature-based FORecast Model Averaging), which features a meta-learner model trained across the entire time series dataset. The

output of this model is a vector of weights that are used to combine the forecasts of a pool of traditional models (ARIMA, exponential smoothing, random walk, naive etc...). Both cited methods are also hybrid in the sense that they combine machine learning models and statistical models to provide forecasts, thus exploiting the advantages of both approaches while dampening their drawbacks.

According to [11] there are multiple ways to combine cross-learning models with traditional ones to produce a reliable hybrid method:

- Averaging forecasts from a cross-learning model and a traditional model;
- Integrating the training inputs of a cluster learning model with features obtained from traditional models;
- Using information on the entire time series set extracted by means of a cross-learning method to combine the forecasts produced from different traditional models;

The biggest advantage of hybrid models is that they combine cross-learning, allowing them to capture informations at a global level for the forecasting of single entities, with traditional methods, which emphasize the individual characteristics of the time series. The results of the M4 competition highlight the potential of cross-learning approaches for forecasting, and in particular of hybrid methods. Many methods of this nature also happen to be a combination of machine learning and statistical models, which also distinguished themselves in the M4 competition for their improved performances, thus suggesting further research in this field of multiple time series forecasting.

2.3.2 Clustering

The opportunity of training a model globally over multiple entities opens the possibility in accuracy deterioration due to wide differences between single time series, in particular when the entities present a highly nonlinear or volatile behaviour. To overcome this issue a clustering methodology can be applied. This approach suggests to subdivide the original entity set into smaller subsets containing entities sharing similar patterns. Then a forecasting model is trained on each subset of time series. The goal of such approach consists in allowing the model to take advantage of similarities between time series during training phase. The outcomes of this process are strongly dependent on the similarity criterion that is adopted when grouping entities together, which opens up a variety of solutions according to the application. An improvement

in forecasting accuracy from adopting a clustering methodology over training a model globally was reported by [16] and [17].

In some applications, time series data are already provided with a form of *natural grouping* or categorization. Examples of this occurrence can be found in retail data, where items are often organized in a product tree, or in the manufacturing field, in which articles present a hierarchical structure in the database. This form of partition does not provide a measure of similarity between entities but exploits the domain knowledge of the case in exam and relies on natural characteristics intrinsic to the data.

When a natural categorization is not present or it's not sufficiently representative of similarity between entities, algorithmic clustering methods can be adopted to provide an appropriate partition. T. Warren Liao [18] classified clustering approaches for time series data in three categories:

- *Distance-based clustering*: also called *raw-data-based* approaches, this family includes algorithms that group time series based solely on the distance between raw data points. Their results are strongly dependent on the chosen distance metric and the time series often require a form of preprocessing if their lengths are different. The clustering algorithm applied can range from k-means, k-medioids, fuzzy c-means, agglomerative hierarchical clustering and so on.

- *Feature-based clustering*: algorithms belonging to this category perform grouping indirectly, by applying a standard clustering procedure to a static vector of features extracted from each time series. Global features usually describe relevant information of an entity such as mean, variance, autocorrelation at a specific order, strength of trend/seasonality and many others. Such methods offer more flexibility since they are more robust to noisy and missing data, can work on time series of differing lengths and usually perform clustering on lower-dimensional vectors. Furthermore, feature-based clustering approaches are also more interpretable with respect to the other categories since they perform grouping on a set of application dependent features encoding specific information on the time series.

- *Model-based clustering*: this class of grouping algorithms performs clustering on the basis of parameters of a model that has been fitted to the data. In this family of approaches each time series is assumed to be generated by a model or an underlying probability distribution. Entities are then grouped according to a similarity measure between the model parameters or the parameters themselves can be used to identify the clusters. The interpretability of these methods is strongly dependent on the model chosen to represent the data. One example of this approach was the one introduced by Piccolo [19], where clustering was performed on the Euclidean distance between the autoregressive expansion vectors obtained by fitting ARIMA models to the time series.

In general, most clustering approaches necessitate the number of subsets to be specified beforehand. Such hyperparameter requires tuning by trial and error to identify the best value. However some algorithms such as the Snob clustering method proposed by [20] can autonomously identify the most optimal number of clusters, thus reducing the user's workload.

This approach extends cross-learning methods by allowing them to take account of similarity between time series, thus improving prediction accuracy since globally trained cross-learning model can experience a deterioration in accuracy from the high variability of entities.

3

Temporal Fusion Transformers

The Transformer architecture, proposed in 2017 by a team of researchers at Google Brain [21], represented an evolution in sequence-to-sequence deep neural networks and opened the way to more advanced machine learning based models for sequence processing. Transformers employ an encoder-decoder structure paired with an attention mechanism which is able to capture relationships between different positions in the input sequences. Such architecture gained considerable success, and is frequently utilized in many fields such as Natural Language Processing, Machine Translation, Computer Vision and time series forecasting.

In 2020 Lim, Arik, Loeff and Pfister proposed the Temporal Fusion Transformer [1], an attention-based architecture for multi-horizon time series forecasting which represented an evolution of the original Transformer architecture and introduced numerous new features and functionalities. One of the most important additions consisted in its ability to forecast multiple time series simultaneously, thus employing what in Section 2.3 Chapter 2 has been defined as a *cross-learning* approach. Moreover, it introduced an interpretability mechanism, which allows the user access to informations relative to the fitting process, enhancing model explainability and enhancing the understanding of the data.

In this chapter the structure of the Temporal Fusion Transformer model is presented. Temporal Fusion Transformers can be classified as a pure machine learning model for multi-horizon forecasting, trained in a cross-learning fashion. It 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

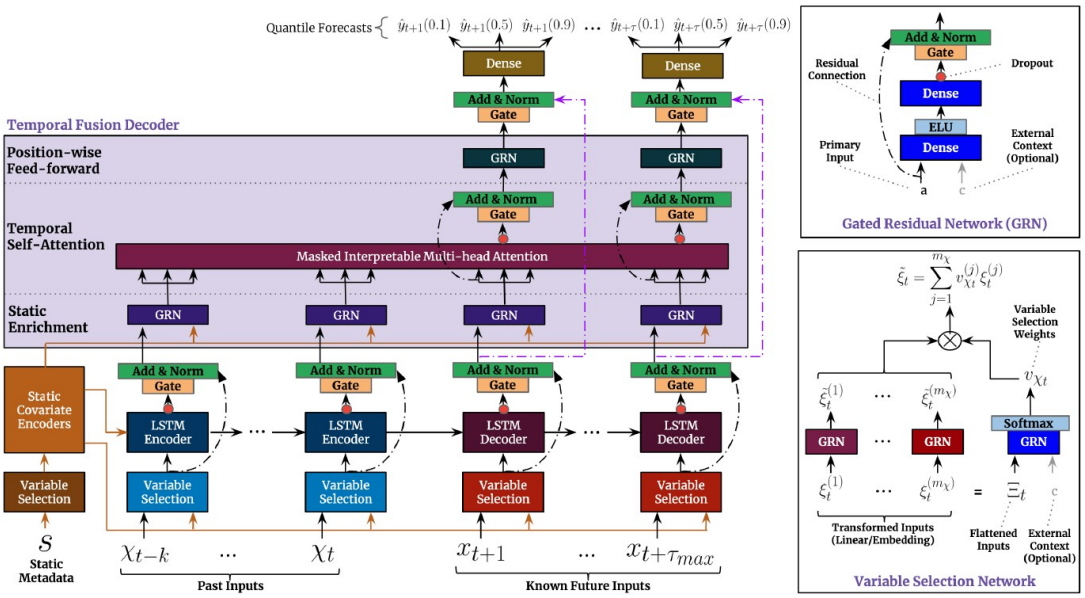

Figure 3.1: Architecture of the Temporal Fusion Transformer model.

long-term dependencies present in the entities.

3.1 Model Structure

The Temporal Fusion Transformer model presents different components, each serving a specific function for the task of multi-horizon forecasting across multiple time series:

1. Variable selection networks
2. Gating mechanisms
3. Static variable encoders
4. Sequence-to-sequence networks
5. Interpretable multi-head attention

In this section the architecture of the TFT model is presented in detail, with particular attention to each component's function.

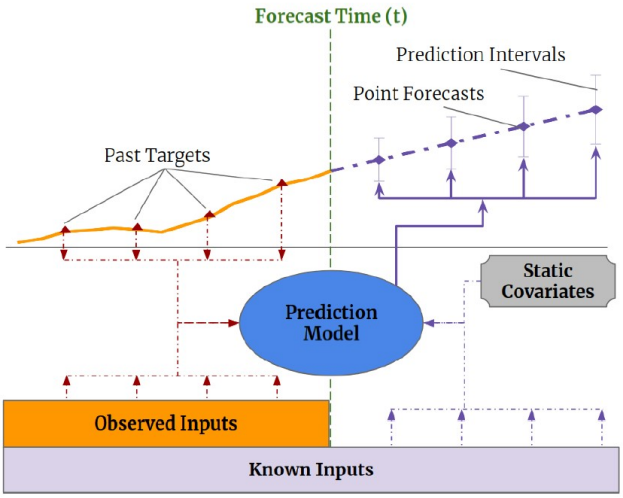

Figure 3.2: Different covariate types handled by the Temporal Fusion Transformer model.

3.1.1 Different covariate types

In many real-world applications the types of available data are often heterogeneous and very different in nature. The TFT model deals with covariates by subdividing them in the following categories:

- *Static covariates*: time independent variables, unchanging throughout the time series, although they can be different between entities. They usually represent global, static characteristics of the entity in question (unique item identifiers, store location, genetic information etc...);
- *Known future covariates*: time dependent variables which values are known in the future (holidays, days of the week etc...);
- *Unknown future covariates*: time dependent variables which future values are unknown, also called observed variables (product demand, price, weather etc...).

Many machine learning models for time series forecasting neglect this input heterogeneity by either ignoring static metadata altogether or by assuming all covariates known into the future [1]. The Temporal Fusion Transformer on the other hand reserves each covariate type a different treatment and at several points in its architecture allows static information to flow in the model state and enrich the forecasts.

Figure 3.2 portrays the different input types handled by a multi-horizon forecasting model. Remaining consistent with the original article's notation [1], the inputs are represented as such:

Let $I = \{1, ..., N\}$ be the set of entities in the dataset, then to each item $i \in I$ and timestep $t \in [0, T_i]$ are associated the following:

- A set $s_i \in \mathbb{R}^{m_s}$ of static covariates;
- a scalar target variable $y_{i,t} \in \mathbb{R}$;
- a set of input features $\chi_{i,t} = \left[z_{i,t}^T, x_{i,t}^T\right]^T$, where $z_{i,t} \in \mathbb{R}^{m_z}$ represents the observed inputs and $x_{i,t} \in \mathbb{R}^{m_x}$ the known inputs.

The Temporal Fusion Transformer model is able to produce both point forecasts and quantile forecasts for the target variable. Being a cross-learning multi-horizon forecasting method, it will simultaneously generate predictions for every entity $i \in I$ and every step-ahead timestep $\tau \in \{1, ..., \tau_{max}\}$, where τ_{max} indicates the maximum prediction length. The point forecast at time t for the i-th entity, $\hat{y}_i(t, \tau)$, generated by the TFT model, can then be expressed as:

$$\begin{aligned} \hat{y}_i(t, \tau) &= f(\tau, y_{i,t-k:t}, z_{i,t-k:t}, x_{i,t-k:t+\tau}, s_i) \\ y_{i,t-k:t} &= \{y_{i,t-k}, ..., y_{i,t-k}\} \\ x_{i,t-k:t+\tau} &= \{x_{i,t-k}, ..., x_{i,t}, ..., x_{i,t+\tau}\} \end{aligned} \tag{3.1}$$

where k represents the length of the lookback window applied by the model. Through this window, only inputs dating back at most k steps are being utilized for the forecast. It is important to observe that target and observed values are considered up until the forecast start time t, while known future inputs range from the lookback to the prediction window.

3.1.2 Variable selection

It can be observed from Figure 3.1, that in the encoder section of the model at each time step both static metadata as well as past and known future covariates are fed through variable selection networks. This preliminary operation benefits the model by allowing it to consider the most salient variables and give smaller weights to the ones with lest predictive content.

Static, observed and future covariates get processed through different variable selection networks, which structure is illustrated in Figure INS. First, for each timestep all inputs relative to all entities are encoded in a d_{model}-dimensional vector ξ_t^j. This dimension, which will also be referred to *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.

Before expanding on the structure of variable selection networks however, is important to define two gating modules that will be employed in different sections throughout the Temporal Fusion Transformer's architecture: Gated Linear Units and Gated Residual Networks.

1. *Gated Linear Units* (GLU): this element allows the model to completely suppress an input if necessary.

$$\mathrm{GLU}_{\omega}(\gamma) = \sigma(W_{1,\omega}\gamma + b_{1,\omega}) \odot (W_{2,\omega}\gamma + b_{2,\omega}) \tag{3.2}$$

   It achieves this by performing an Hadamard element-wise product between the sigmoid of a linear transformation of the input γ and another linearly transformed input vector. Since the sigmoid function can assume values in the range $]0, 1[$, a Gated Linear Unit introduces the possibility of negating the input if necessary, thus enabling the TFT model to control the flow of informations in its architecture.

2. *Gated Residual Networks* (GRN): in order to introduce optional nonlinear processing in the TFT model, GRNs were introduced. This gating module employs a Gated Linear Unit controlling the flow of a nonlinear transformation of the input a. An optional context vector c can be integrated in the component:

$$\mathrm{GRN}_{\omega}(a, c) = \mathrm{LayerNorm}(a + \mathrm{GLU}_{\omega}(\eta_1)) \tag{3.3}$$
$$\eta_1 = W_{3,\omega}\eta_2 + b_{3,\omega} \tag{3.4}$$
$$\eta_2 = \mathrm{ELU}(W_{4,\omega}a + W_{5,\omega}c + b_{4,\omega}) \tag{3.5}$$

   By modifying the parameters ω, it is possible to entirely skip the nonlinear transformation applied by the Exponential Linear Unit funtion (ELU), and to only consider the original input a.

At every timestep input embeddings for each variable j are linearly concatenated in a vector Ξ_t. This flattened input is fed through a GRN and a Softmax layer, together with a static context vector c_s obtained from embedding the static covariates s, to generate variable selection weights. All input embeddings are then further processed through a GRN, which weights are shared across all timesteps. Lastly, the processed inputs are weighted and combined by means of the previously generated weights vector $v_{\chi_t} \in \mathbb{R}^{m_\chi}$. The output of every variable selection network, for each timestep and variable category (static, observed and future), is a vector $\tilde{\xi}_t \in \mathbb{R}^{d_{model}}$.

$$\Xi_t = \left[\xi_t^1, ..., \xi_t^{m_\chi}\right]$$
$$v_{\chi t} = \text{Softmax}\left(GRN(\Xi_t, c_s)\right)$$
$$\tilde{\xi}_t = \sum_{j=1}^{m_\chi} v_{\chi t} * \xi_t^j$$

This variable selection features makes the Temporal Fusion Transformer model particularly effective when applied to datasets with a large number of correlated covariates, as it makes it able to utilize only the most relevant regressors. Furthermore, as it will be detailed in Section 3.2, the variable selection weights can be used to determine the significance and the predictive content of each covariate.

3.1.3 Static variable encoders

Unlike observed and future covariates, which are directed towards temporal processing after being fed through variable selection networks, static variables undergo another level of manipulation. The TFT employs *static covariate encoders* to extract context information from static inputs. The purpose of this operation is to integrate global metadata at various locations in the architecture to further refine the predictions.

This model components outputs four context vectors c_s, c_c, c_h and c_e of dimension d_{model} generated from four separate GRN encoders. Each of this vectors serves as an additional input to different components of the TFT model, their respective purposes are reported below:

1. c_s : context vector for variable selection;
2. c_c : context vector for local processing of temporal features, used to initialize the cell state of first LSTM encoder;
3. c_h : context vector for local processing of temporal features, used to initialize the hidden state of first LSTM encoder;
4. c_e : context vector for the enrichment of temporal features with static information.

Static metadata contain valuable informations, especially in cross-learning contexts where multiple time series are being considered. In this settings time-independent covariates assume further relevance with respect to single-series frameworks, as they can be employed by the

model to distinguish and assign forecasts to different entities. A Deep Learning model for wildfire danger forecasting [22] applied an approach similar to the one employed in the TFT architecture, separating dynamic and static variables and reserving an ad-hoc processing for the latter.

3.1.4 Sequence-to-sequence networks

The time-dependent outputs of variable selection networks $\tilde{\xi}_t$ are then subject to temporal processing in a sequence-to-sequence layer. This model component employs Long Short Term Memory networks (LSTMs) to capture local patterns in the data and serves as positional encoding to provide uniform temporal features to the decoder.

LSTM networks [23] are a sequence-to-sequence architecture considered to be an evolution of Recurrent Neural Networks (RNNs). Unlike the latter, LSTMs are not subject to vanishing gradient issues and are capable to learn dependencies between timesteps several lags apart. They are able to capture both long and short-term temporal dependencies thanks to a gate-controlled cell state. As can be seen in Figure INS, the gate regulates the information flowing inside the cell state, allowing it to stay unchanged for a large number of lags and determines which inputs are relevant to update it.

In the current application in the TFT architecture, an hyperparameter regulates the number of stacked LSTM layers (with 1 layer being the default option). Furthermore, the cell state c_1 and hidden state h_1 at the beginning of the sequence are initialized by means of the context vectors c_c and c_h (defined in Subsection 3.1.3) respectively.

3.1.5 Static Enrichment layer

For every time step t the sequence-to-sequence layer outputs a set of uniform temporal features $\phi(t, n) \in \mathbb{R}^{d_{model}}$, where $n \in \{-k, ..., \tau_{max}\}$ is a positional index. The distinction between the indices t and n lies in the fact that t represents the forecast time, while n indicates the offset in time steps from the forecast time. Afterwards, a gated skip connection is employed to enhance the outputs with the weighted embeddings $\tilde{\xi}_t$:

$$\tilde{\phi}(t, n) = \text{LayerNorm}(\tilde{\xi}_t + \text{GLU}(\phi(t, n))) \tag{3.6}$$

Subsequently, the processed temporal features enter a *static enrichment layer*, where they are

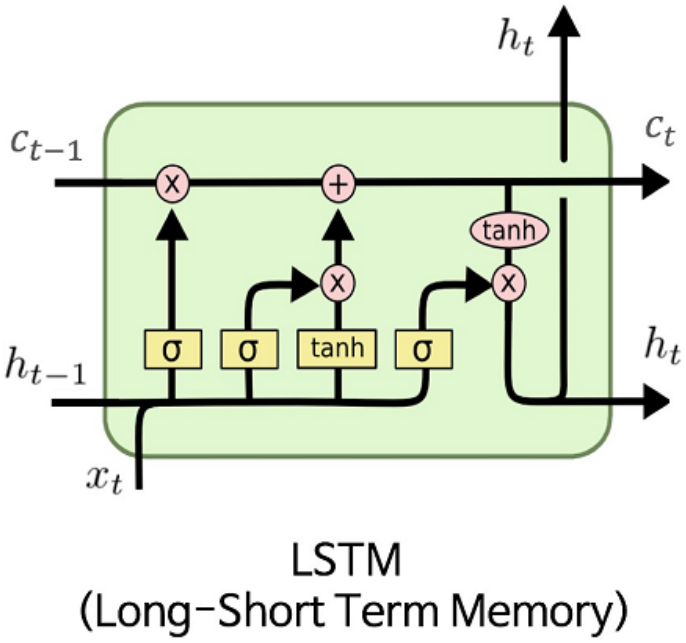

Figure 3.3: Long Short Term Memory network architecture.

further enhanced with the static context vector c_e by means of a Gated Recurrent Network:

$$\theta(t, n) = \text{GRN}_\theta(\tilde{\phi}(t, n), c_e) \tag{3.7}$$

This operation is carried out for every positional index n. The weights of the GRN employed in this model component are shared across the layer.

3.1.6 Temporal Self-Attention layer

Finally, the static-enriched temporal features enter the *Temporal Self-Attention layer*, the core part of the TFT's architecture. The purpose of this model component is to capture long-term dependencies in the data. It achieves this by processing the static-enriched temporal features $\theta(t, n)$ through an Interpretable Multi-Head Attention module. The finer details of this component will be explained in Section 3.2, but its operation is similar to classic multi-head attention modules.

For every time step t, the $k + 1 + \tau_{max}$ temporal features $\theta(t, n)$ are concatenated in a matrix

$\Theta(t)$, which serves as input for the Interpretable Multi-Head Attention.

$$\Theta(t) = [\theta(t,-k), ..., \theta(t,\tau), ..., \theta(t,\tau_{max})]^T \in \mathbb{R}^{d_{model} \times (k+1+\tau_{max})} \tag{3.8}$$

The matrix obtained this way is used simultaneously as query, key and value for the attention block and outputs are generated only for future horizons $\{1, ..., \tau_{max}\}$:

$$B(t) = \text{InterpretableMultiHead}(\Theta(t), \Theta(t), \Theta(t)) \tag{3.9}$$

$$B(t) = [\beta(t,1), ..., \beta(t,\tau_{max})]^T \in \mathbb{R}^{\tau_{max} \times d_{model}} \tag{3.10}$$

In the formula above the τ_{max} output vectors are grouped in a single matrix $B(t)$. The number of attention heads employed in this component is an hyperparameter chosen by the user.

Following this component, a gating layer featuring a skip connection is used to preserve the information from the static-enriched temporal features. For each time step t this operation is applied at every positional index n.

$$\delta(t,n) = \text{LayerNorm}(\theta(t,n) + \text{GLU}_\delta(\beta(t,n))) \tag{3.11}$$

3.1.7 Position-wise feed-forward layer

Before generating the forecasts, the outputs of the temporal self-attention layer are subject to a last instance of non-linear processing in a *position-wise feed-forward layer*. In this component for every positional index $n \in \{1, ..., \tau_{max}\}$ another Gated Recurrent Network is used.

Lastly, to preserve context from the processed temporal features $\tilde{\phi}(t,n)$ another gated skip connection is employed:

$$\tilde{\psi}(t,n) = \text{LayerNorm}(\tilde{\phi}(t,n) + \text{GLU}_\psi(\psi(t,n))) \tag{3.12}$$

The advantage of the numerous skip connections utilized in different sections of the TFT architecture consists in providing further unprocessed inputs to the deepest sections of the network, thus enabling feature reusability. Many established complex machine learning architectures such as ResNet [12] or DenseNet [24] make frequent use of shortcut connections. The advantage of deep architectures lies in their ability to learn extremely complex relationships directly from the data. On the downside, however, they often fail to represent simple interactions and after a certain depth their training and optimization becomes flawed and is subject to

failure. The use of skip connections allows the following benefits:

- *Feature reusability* : Allowing a model to utilize in deeper layers features learned in earlier layers is beneficial to training. In [25] is suggested that reusing features from previous layers can be more advantageous than adding new features in every layer, when tasked with learning a mapping. Thus feature reuse makes possible to lower the amount of model parameters required to represent a family of functions.

- *Increased generalization* : Skip connections, integrated with gating mechanisms, allow a deep architecture to entirely skip or limit the influence of some sections of the network. Machine Learning models with a large number of layers not employing shortcut connections are not guaranteed to represent even an identity function over the data, as proven in [25]. This follows from the fact that in deep neural networks small changes to the training data translate into large changes in the forecast. Thus being able to cut off sections of the network and reducing depth allows DNNs to represent a wider range of relations.

- *Improved optimization* : Deep neural networks training relies on the gradient descent algorithm. If the gradient is ill-conditioned it causes optimization to fail. As highlighted by [25], for deep architectures depth growth translates into asymptotically singular error gradients, which causes an accumulation of precision errors in the backpropagation algorithm. The result is an erratic weights update and optimization failure. The integration of skip connections in the model architecture avoids the singularity of error gradients, thus ensuring a successful optimization with consistent weight updates.

The Temporal Fusion Transformer can either output τ_{max}-step-ahead punctual forecasts $\hat{y}(t, \tau)$ for every prediction time t, or quantile forecasts $\hat{y}(q, t, \tau)$ for a set of user-specified percentiles identified by q. Both results are obtained by means of a simple linear transformation on $\tilde{\psi}(t, \tau)$. Figure 3.4 offers a representation of the 60-step-ahead punctual forecasts (in blue) for the 7 sample time series presented in Chapter 2 provided by a Temporal Fusion Transformer model.

3.2 Interpretability feature

One of the most important advantages offered by the Temporal Fusion Transformer model is its ability to provide insights on how the predictive process was carried out, highlighting the subsets of inputs which brought the most value to the forecast as well as suggesting the underlying temporal relationships present in the data. Additionally the TFT can identify which

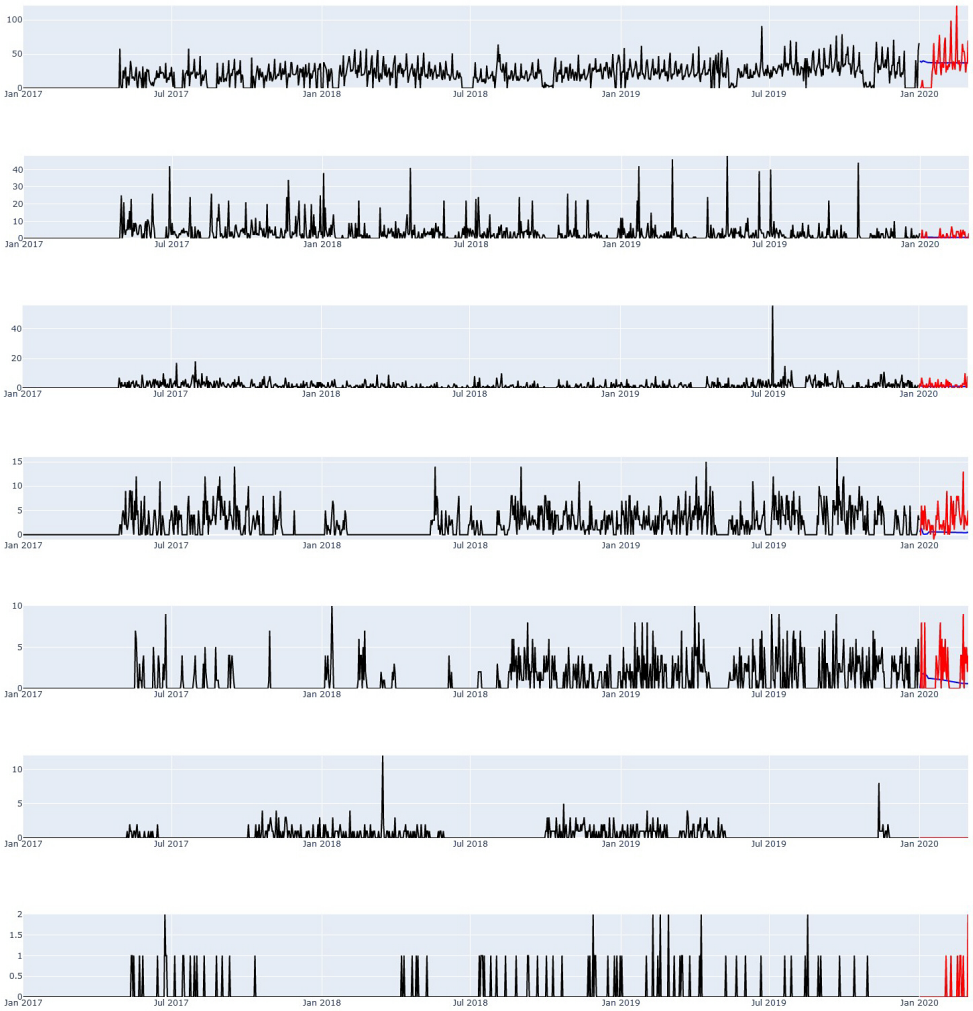

Figure 3.4: In blue the 60 step-ahead forecasts for the sample time series from the retail dataset.

temporal pattern present in the entities was useful to infer future forecasts and recognize to a certain extent variations in the regime of the considered time series.

Aside from prediction accuracy, a model offering such interpretable results is extremely valuable with respect to other models which carry out their forecasts in a "black-box" fashion. The latter category, which includes the majority of machine learning based models, outputs predictions on the data without providing additional context on the process through which this was achieved. Not only that, but in general the computations such methods employ in their architectures are complex and do not carry much informative content to a user. A recurrent neural network, for example, utilizes a large number of parameters tied with nonlinear activation functions in a way that is hard to understand how variables are combined to generate a forecast. In this sense the TFT is no different, however in certain components of its architecture the learned parameters can be traced back to original features of the data, such as input variables or time steps, and thus be exploited to gain additional knowledge.

The value these informations bring to the user is procedural. Knowledge regarding variable importance can be exploited in order to influence future targets, while context concerning temporal patterns can be utilized to improve the forecast of the model by applying additional processing to the input data. For example, a model can be retrained after excluding the variables which achieved the lowest importance, or after applying a form of feature engineering incorporating the learned seasonal information. If the model recognized valuable temporal patterns in early time steps it might prove useful to retrain it with a larger input window.

For practical implementations, especially when the user has the capability to influence some of the quantities related to the model inputs or when forecast results have a role in a decision-making process, model explainability can be considered more valuable than high accuracy [26]. A manufacturing firm can serve itself of information about variable relevance to modify its business strategy, focusing on the aspects which more conditioned the final forecast.

The current chapter focuses on how the Temporal Fusion Transformer model is able to extract valuable information from its architecture, and how the provided knowledge can be interpreted to further understand the underlying aspects present in the data.

3.2.1 Variable importance

The variable selection networks weights can be used to assess the predictive relevance of each covariate in the dataset. For every time-varying regressor j, included the auxiliary regressors built by the model itself, the variable importance v_j is obtained by averaging the variable selection

weights v_{j_t} across all their relative time steps:

$$v_j = \sum_{t=s_j^{(0)}}^{s_j^{(max)}} v_{j_t} \tag{3.13}$$

In the formula above $s_j^{(0)}$ and $s_j^{(max)}$ indicate the ranges of past inputs for encoder variables and future inputs for decoder variables. Static variable weights v_s on the other hand do not require averaging over time steps, and are returned without additional processing.

It is important to observe that for time-dependent known future variables, which are processed both in the encoder variable selection networks and in the decoder variable selection networks, the encoder weights and decoder weights are different. This is due to the fact that the GRN present in these components share their weights separately across the encoder and the decoder.

These parameters represent the learned relevance each input had for the purposes of providing a forecast. The highest the parameter value, the more significant that specific covariate's contribution was for the model. In a finer sense, they represent the influence each regressor bears over the target variable. To mitigate overfitting, a subset of covariates can be determined by setting a cutoff threshold and by selecting only the regressors which variable selection weights resulted higher than the threshold value. Then the TFT or another forecasting model can be trained on the input data considering only the covariates present in the subset. Alternatively, the features can be ranked according to their selection weights to determine where to direct resources for the purpose of conditioning the target variable.

3.2.2 Interpretable multi-head attention

The *Interpretable Multi-Head Attention* mechanism, employed in the decoder of the TFT model and introduced in Section 3.1, represents the central component of its architecture. Its function is to capture long-term dependencies and to learn patterns in the input data. However, the main feature that distinguishes it from other attention-based decoders is the information it provides along the processed inputs.

Regular Multi-Head Attention linearly combines the concatenation of the outputs of each

head by means of a weight matrix $W_H \in \mathbb{R}^{(m_H \cdot d_v) \times d_{model}}$:

$$\text{MultiHeadAttention}(Q, K, V) = [H_1, ..., H_{m_H}] W_H \tag{3.14}$$

$$H_i = \text{Attention}(QW_Q^{(h)}, KW_K^{(h)}, VW_V^{(h)}) \quad \forall i \in \{1, ..., m_H\} \tag{3.15}$$

where m_H indicates the number of heads used in the attention block.

Interpretable Multi-Head Attention on the other hand, averages the outputs of the attention heads and applies a linear transformation W_H on the average (not the concatenation). Another notable difference is that the value matrix weights W_V are shared across all heads:

$$\text{InterpretableMultiHead}(Q, K, V) = \tilde{H} W_H \tag{3.16}$$

$$\tilde{H} = \frac{1}{m_h} \sum_{h=1}^{m_h} \text{Attention}(QW_Q^{(h)}, KW_K^{(h)}, VW_V) \tag{3.17}$$

$$= \left\{ \frac{1}{m_h} \sum_{h=1}^{m_h} \text{Softmax}(QW_Q^{(h)} W_K^{(h)T} K^T / \sqrt{d_{attn}}) \right\} VW_V \tag{3.18}$$

$$= \tilde{A}(Q, K) \cdot VW_V \tag{3.19}$$

In this variant the weight matrix $W_H \in \mathbb{R}^{d_{attn} \times d_{model}}$ is used to ensemble the additive outputs of the heads, where d_{attn} indicates the dimension of the attention weights. Moreover temporal masking is applied to the every attention head, such that only past features can be used to predict future ones. All the time steps in the decoder can attend to all positions in the decoder up to and including themselves.

This way, the shared weights can be exploited to extract the temporal patterns captured by the component. This operation was not feasible with regular Multi-Head Attention, due to the fact that attention weights are unique to each head and their behaviour is related to a single temporal pattern instead of the global temporal relationships in the data. As can be seen in Equation 3.9 in the Temporal Fusion Transformer use case, for every time step t the Interpretable Multi-Head Attention is used on a concatenated matrix of temporal features, so for every forecast horizon τ the outputs of this model component can be represented in the follow-

ing way:

$$\beta(t,\tau) = e_\tau^T \cdot \tilde{A}\left(\Theta(t),\Theta(t)\right) \cdot \Theta(t) \cdot W_V W_H \tag{3.20}$$

$$= e_\tau^T \cdot \tilde{A}\left(\Theta(t),\Theta(t)\right) \cdot \tilde{\Theta}(t) \tag{3.21}$$

$$= \sum_{n=-k}^{\tau_{max}} a(t,\tau,n) \cdot \tilde{\theta}(t,n) \tag{3.22}$$

where $\tilde{\Theta}(t) = \Theta(t) \cdot W_V W_H$ and $a(t,\tau,n)$ represents the (τ,n)-th entry of the matrix $\tilde{A}\left(\Theta(t),\Theta(t)\right)$. This way, attention outputs can be seen as linear combinations of weighted, uniform temporal features. The distribution of attention weights for a fixed forecast horizon τ across all time steps can then be exploited to determine the importance of each previous time step n.

This new approach introduces a tradeoff between model performances and explainability: inhibiting pattern-recognition capabilities of the attention block by reducing the number of head-specific parameters, but at the same time enabling model interpretability by providing a uniform set of weights across all heads, relating to the importance of the input temporal features $\theta(t,n)$. The attention weights obtained this way can be exploited in the following ways:

1. *Temporal pattern recognition*: Global seasonal patterns can be identified by means of studying the attention weights. For time series, trend and seasonal analysis is often carried out entity-wise by observing the autocorrelation function (ACF) or partial autocorrelation function (PACF). This process however is unfeasible when the number of time series to study is large. Average attention weights across all time steps t for fixed forecast horizons τ can highlight, if present, a global common pattern across all considered entities thus providing a powerful instrument to further understand the nature of the data.

$$\frac{1}{T}\sum_{t=1}^{T} a(t,\tau,n) \quad \text{for } n \in \{-k,...,-1\} \tag{3.23}$$

2. *Regime change identification*: The advantage of Multi-Head-Attention architectures consists in their ability to capture different patterns present in the data. Interpretable-Multi-Head-Attention is no different, with the additional feature of being able to visualize the extracted information. For a fixed time step t, the average distance across all forecast horizons τ between attention vectors and the mean attention pattern is a metric that, when achieving significance over a threshold, can highlight global regime shifts of the target series.

$$dist(t) = \frac{1}{\tau_{max}} \sum_{\tau=1}^{\tau_{max}} dist(a(\tau), a(t,\tau)) \tag{3.24}$$

Being able to identify events that alter the regime of an ensemble of entities is extremely valuable, and as such the attention weights offer the user an additional tool to further understand the target series.

In conclusion, the data provided by the Temporal Fusion Transformer along its predictions not only bring value to the user by shedding light on the forecasting process, but also provide valuable informations relative to the dataset itself. Usually a tradeoff is present between model accuracy and explainability, and especially machine learning models are lacking in the latter category. The goal of TFT models consists in offering reliable forecasts paired with a form of interpretability which, especially when dealing with extremely large sets of entities, is a most appreciated feature. Global information relative to time series belonging to the same distribution is extremely valuable in cases when it is not feasible to deal with each time series singularly.

4

Experiments and clustering approach

In order to ascertain the performances of the Temporal Fusion Transformer model, its multi-step forecast capabilities, the insights provided by its interpretability feature and the advantages of performing simultaneous predictions over multiple time series, an experiment framework was set.

After choosing an appropriate dataset for the task, a simple Moving Average baseline was established. Then the Temporal Fusion Transformer model was trained on all the entities of the selected dataset, to test its behaviour in forecasting many time series largely different in nature. Afterwards, an hypothesis was formulated: whether the performance of the model could improve if paired with a form of grouping on the entities. The goal was to ascertain if training several TFT models on a partition of the entity set could yield better forecasts than a single model trained on the whole set. The partition should be based on a similarity criterion, thus allowing the transformer to take advantage of time series with analogous features in order to achieve better performances. The first selected partition method consists in an intuitive grouping based on the product tree categories of the entities, while the second method associates similar time series by means of a clustering algorithm run before training.

4.1 Dataset and Testing Framework

For the experiments the retail dataset mentioned in Chapter 2 was made available by the internship organization. This dataset compiles historical daily data from 01/01/2017 to 28/02/2022 for 15593 products. Each product time series is multivariate and presents the following features:

- `qty`: the target variable, representing the sales of the product;
- `qty_promo`: regressor indicating the quantity of the product subject to promotion;
- `sales_promo`: binary value indicating the sales subject to promotion;
- `id`: code number identifying uniquely each product;
- `date`: datetime index;
- `holiday`: binary value indicating whether the shop is closed or not.

To the previous features a sixth regressor `time_idx` was manually added. This is an incremental integer for each step of the time series and was required by the TFT model.

For testing purposes 1000 time series were randomly selected among the 15593 products. Due to the high computational cost required to train and validate a Temporal Fusion transformer model on a large number of entities, a blocked cross-validation approach [27] was adopted since it involves only one training and evaluation step. Such approach is consistent with the one employed in [1]. The training set was defined from 01/01/2017 to 01/11/2019 for a total of 3225 time steps, while the validation set was selected as the 60 days following the training set, namely between 02/11/2019 and 31/12/2019. Lastly, the test set was chosen from 01/01/2020 to 29/02/2020 for a total of 60 time steps following the validation set.

After running the Augmented-Dickey-Fuller test on every time series it resulted that most entities presented a stationary behaviour and thus did not require detrending or differencing as suggested by [28] and [29]. Input normalization was not applied either due to the fact that the Temporal Fusion Model handles this operation internally. A check of the ACF however revealed that a large number of entities presented a form of seasonality (mostly weekly). No deseasonalization was performed to test the pattern capturing capabilities of the Temporal Fusion Transformer model.

Training of all models was performed on a notebook instance with one GPU, all training times have been recorded for comparison.

4.1.1 Demand pattern classification

It is important to note that most of the time series considered, in particular the response variable `qty`, present long streaks of zero values sporadically interrupted by nonzero demand. As observed by [3], this is a common occurrence when dealing with daily SKU demand forecasting. This type of regime has been classified by [30] as *irregular* or *sparse demand*. This does not apply to all entities, however many products present this characteristic and thus it is necessary to address properly this issue [31]. Irregular demand data are different from common time series and more difficult to predict.

Following Syntetos & Boylan categorization scheme, demand patterns are characterized according to two parameters:

- The squared coefficient of variation (CV^2): measures the variability in demand magnitude for the time series, is computed by squaring the standardized standard deviation on nonzero demands.

$$CV^2 = \left[\frac{\text{nonzero demand standard deviation}}{\text{nonzero demand average}}\right]^2 \tag{4.1}$$

- The average inter-demand interval (ADI): represents the average interval in time periods between two consecutive nonzero demands.

$$ADI = \frac{\sum_{i=1}^{N} t_i}{N} \tag{4.2}$$

  where t_i indicates the length in time periods between two consecutive demand periods and N represents the number of all periods [32].

The corresponding demand categories derived from thresholding the CV^2 and ADI parameters as represented in Figure 4.1 are the following:

1. *Smooth demand*: $CV^2 \leq 0.49$ and $ADI \leq 1.32$
2. *Erratic demand*: $CV^2 > 0.49$ and $ADI \leq 1.32$
3. *Intermittent demand*: $CV^2 \leq 0.49$ and $ADI > 1.32$
4. *Lumpy demand*: $CV^2 > 0.49$ and $ADI > 1.32$

Regimes presenting regular demand occurrence, such as smooth and erratic, are often addressed as *regular demand*, while *irregular* or *sparse demand* indicates infrequent demand regimes such as lumpy and intermittent.

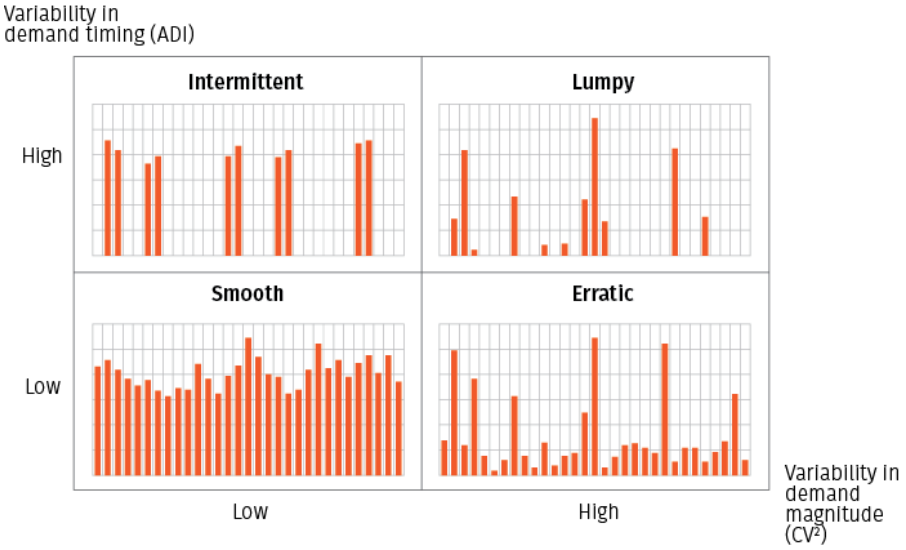

Figure 4.1: Demand pattern classification according to Syntetos & Boylan.

In order to classify the time series in the retail dataset with the aim of selecting an appropriate benchmark and evaluation metrics, the CV^2 and ADI parameters were computed for all 1000 entities. The classification results revealed a predominance of lumpy time series (686), followed by 156 entities with intermittent demand, 121 erratic and only 36 smooth. The majority of the considered entities thus presents an irregular demand regime, portrayed in Figure 4.2.

According to [33], the most suited models to forecast lumpy and intermittent time series are ad-hoc or iterative methods such as Simple Exponential Smoothing or the Croston method [7]. Further research [34] however has shown that Recurrent Neural Networks proved extremely reliable, achieving similar if not superior performances with respect to SES and Croston models. Temporal Fusion Transformers feature an attention-based architecture, different from RNNs, but still machine learning based, thus promising acceptable results in terms of forecasting accuracy for the current application. Thus the following experiments also aim to assess the TFT performance in this particular field.

4.1.2 Metrics

The lumpy/intermittent nature of the target variable has an impact towards metric selection. To evaluate forecasting accuracy Mean Absolute Percentage Error (MAPE) cannot be used due to the occurrences of zero periods of demand [35]. The metrics considered for model selection and evaluation were therefore Mean Absolute Error (MAE), Mean Squared Error (MSE), Root

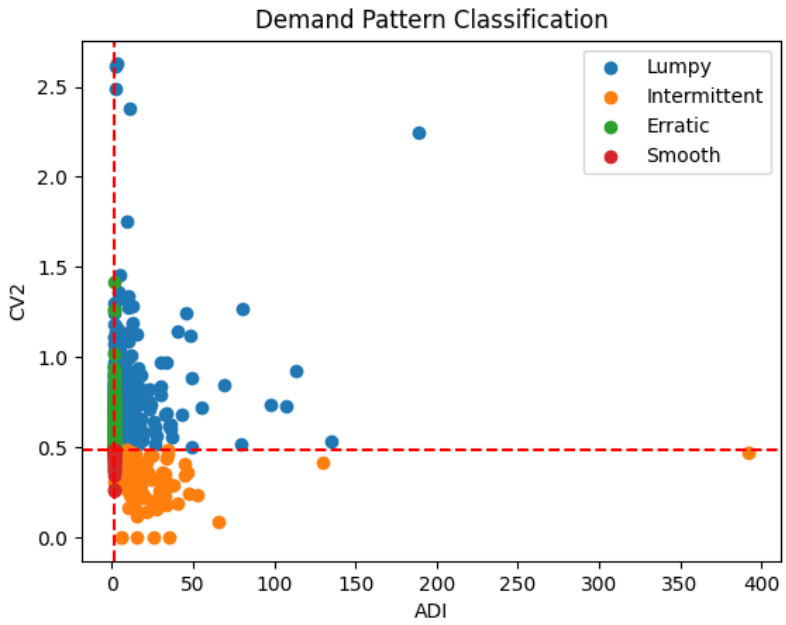

Figure 4.2: Demand pattern distribution of the 1000 entities in the retail dataset. The predominance of irregular demand can be observed.

Mean Squared Error (RMSE) and Mean Absolute Scaled Error (MASE). The selected metrics are defined as follows:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_{t+i} - y_{t+i}| \tag{4.3}$$

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_{t+i} - y_{t+i})^2 \tag{4.4}$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_{t+i} - y_{t+i})^2} \tag{4.5}$$

$$\text{MASE} = \frac{\text{MAE}}{\frac{1}{t-1}\sum_{i=2}^{t}|y_i - y_{i-1}|} \tag{4.6}$$

Where y_i indicates the actual value of the target variable, while $\hat{y}_i$ is the forecast produced by the model, both at the i-th timestep.

The last metric, MASE, has the feature of being scale independent, and thus is better suited

to evaluate forecast accuracy across multiple time series. For this reason it was chosen as the model selection metric. To evaluate the global performances of a model performing forecasts over multiple time series, the selected metric was computed on each entity, then the average of said metric across all entities was considered.

4.2 Holt-Winters benchmark

After settling for a Moving Average model MA(1) as baseline, another statistical model was chosen as benchmark. For this task, due to the presence of some trend in a few time series and seasonality in almost all of them, Holt-Winters exponential smoothing model was selected. The choice of this method was dictated by the necessity of comparing the Temporal Fusion Transformer with a deeply different forecasting model. According to [36] is highly recommended to compare machine-learning models with simpler statistical benchmarks. Holt-Winters exponential smoothing is a traditional forecasting method (opposed to the cross-learning nature of TFT), featuring a much lower complexity than its attention-based counterpart but retaining a high reliability and good performances.

Holt-Winters model, also referred to as triple exponential smoothing [37], is designed to forecast time series that exhibit both a trend and a seasonal pattern. It is available in two versions: one designed to model additive seasonalities and the other to model multiplicative seasonalities. The first version is more suited when the seasonal patterns are roughly constant in scale throughout the time series. Due to the features of the considered dataset, the additive Holt-Winters model was selected, which formula in reported below. The forecast at time $(t + i)$ is expressed as:

$$F_{(t+k)} = (L_i + k * B_i) + S_{(i+k-m)} \tag{4.7}$$

The formula features a level component L_i, a trend component B_i and a seasonality component S_i, where m represents the seasonal period length and y_i the value of the time series at the i-th timestep. Each component is defined as follows:

- $L_i = \alpha * (y_i - S_{(i-m)}) + (1 - \alpha) * \left[L_{(i-1)} + B_{(i-1)}\right]$
- $B_i = \beta * \left[L_i - L_{(i-1)}\right] + (1 - \beta) * B_{(i-1)}$
- $S_i = \gamma * [y_i - L_{(i-1)} - B_{(i-1)}] + (1 - \gamma) * S_{(i-m)}$

In the current application the Holt-Winters model was applied with a 7-lag seasonality, as inferred by the autocorrelation plots.

Number of entities	1000	Dropout rate	0.1
Number of samples	3000 k	Hidden size	32
Lookback window	180	Attention heads	2
Prediction length	60	Minibatch size	32
Trainable model parameters	105 k	Learning rate	0.05
Model parameters size	0.421 MB	Number of epochs	50

Table 4.1: Optimal TFT model summary and dataset informations.

4.3 TFT on entire dataset

First the Temporal Fusion Transformer model was trained on the entire set of 1000 entities. The dataset presented different types of covariates, their respective input types are reported below:

- Static categoricals: `id`
- Time varying unknown reals: `qty, qty_promo`
- Time varying unknown categoricals: `holiday, sales_promo`
- Time varying known reals: `time_idx`
- Datetime: `date`

Hyperparameter optimization was conducted via grid search over 50 epochs, listed below are the search ranges:

- Learning rate: 0.01, 0.05, 0.1
- Hidden size: 32, 64, 128
- Attention head size: 2, 4

The hidden continuous size was arbitrarily fixed as half of the hidden size, while all other hyperparameters were fixed as the recommended values. Table 4.1 presents an overview of the final model.

As can be seen in Table 4.2 the TFT model outperformed both the Moving Average baseline (30.5% of MASE) and the Holt-Winters benchmark (21.7% of MASE), requiring however a

Model	MASE	MAE	MSE	RMSE	Running Time
TFT Global	**0.754**	**1.482**	**26.653**	2.093	2319 min
Holt-Winters	0.963	1.522	29.909	**1.979**	235 sec
MA baseline	1.085	1.787	30.995	2.272	**24** sec

Table 4.2: Global TFT model evaluation vs. Moving Average baseline and Holt-Winters benchmark.

much longer training time as expected from its deep-learning nature. The average performance may be due to the intermittent nature of the time series in the dataset, and the wide difference between each entity. Many products present wide periods of zero sales while others present frequent fluctuations of the response variable in the same period.

4.3.1 Interpretation

By means of the interpretability feature of the Temporal Fusion Transformer model, informations such as variable selection weights for encoder covariates, decoder covariates and static covariates were extracted. Moreover, also the interpretable multi-head attention weights were recovered and are shown in Figure 4.3.

The plot shows the attention weights for one-step-ahead forecasts across each time step in the lookback window (180 time steps in total). As explained in chapter 3 section 3.2, attention weights can be used to determine global patterns in the data and to identify which past time steps were the most relevant to the model in order to generate forecasts. The plot in Figure 4.3 presents an erratic behaviour for the first 20-25 lags, which can be partially attributed to the weights still stabilizing during training phase. A way to solve this problem is to increase the training epochs. Another possible cause for the spike at lag 180 can be attributed to the necessity of a wider lookback window Apart from the first lags, the graph clearly presents a logaritmic increase in attention weights towards more recent time steps. A seasonal pattern can also be observed, which becomes more evident at lower lags. In particular the global model learned a weekly pattern in the attention weights, which translated in an increased forecasting relevance for the 7 previous time lags in the data.

By extracting the information from the variable selection modules in the Temporal Fusion Transformer architecture, it was also possible to visualize the encoder, decoder and static variable weights and their importance for the prediction. Figure 4.4 shows each variable importance, divided by input type. It can be noticed that the TFT model generates and adds au-

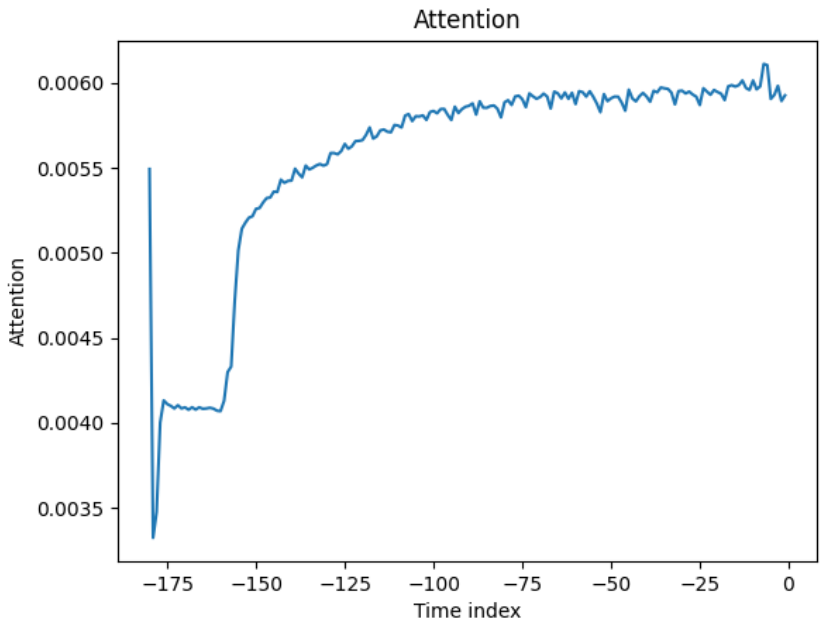

Figure 4.3: Attention weights for one-step-ahead forecasts relative to the global TFT model.

tonomously the following variables:

- `qty_center`: static covariate, average of the target variable;
- `encoder_length`: static covariate, length of the lookback window;
- `qty_scale`: static covariate, scale of the target variable;
- `relative_time_idx`: the distance in time steps from the considered lag and the lag to be predicted.

This operation can be considered a form of input enrichment with feature extraction, although very limited to simple quantities, which in many use cases proved to greatly improve forecasting accuracy, especially in cross-learning models [11].

By observing the graphs it can be gathered that among static covariates the target variable average `qty_center` was the most relevant by achieving a weight of 996.767, more than a 1000 times larger than the time series identifier variable `id`. This result appears surprising, since empirical experience suggests that variables uniquely identifying the entities usually achieve the highest importance. The reason might be attributed to the wide diversity present in the time

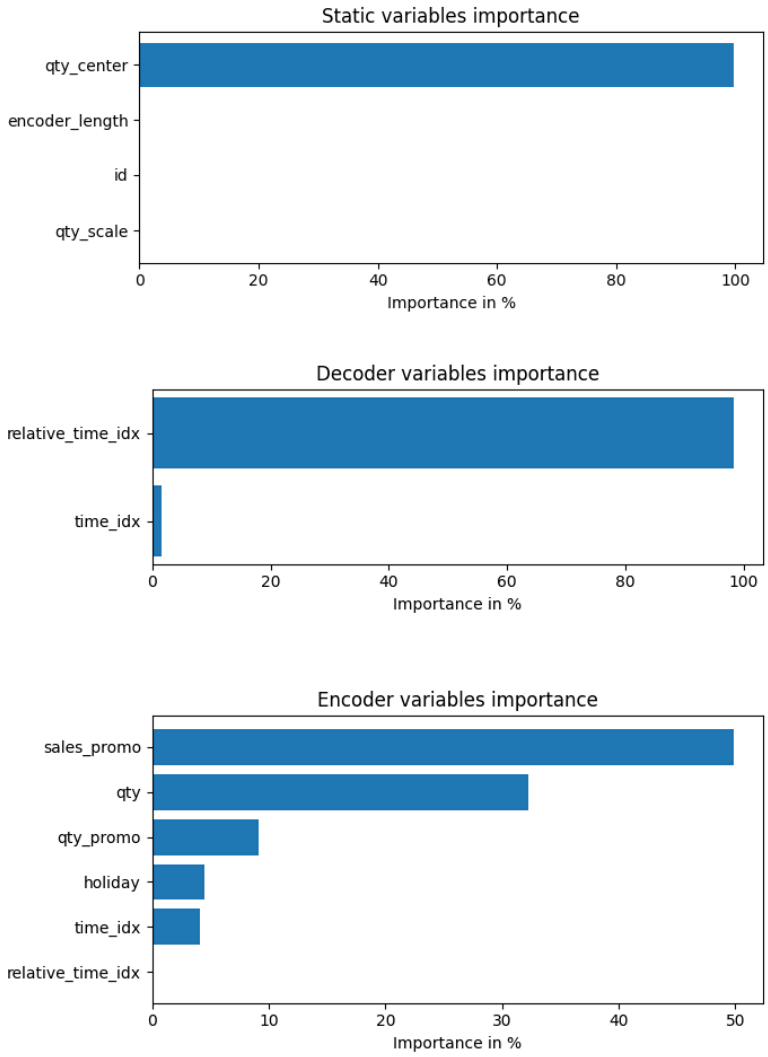

Figure 4.4: Variable selection weights relative to static, encoder and decoder variables for the global TFT model.

series dataset, forcing the model to focus on each series' average as a base to constructing more accurate forecasts.

Among the decoder variables, namely the known future inputs, due to the nature of the dataset only the manually added covariate `time_idx` and its counterpart `relative_time_idx` were featured. Nonetheless it appeared that the lag gap covariate resulted more relevant than the absolute time coordinate, further confirming the ability of the model in identifying a form of seasonal behaviour in the time series.

Lastly, for past regressors the target variable `qty` was critical as expected, with an importance weight of 322.732, surpassed only by `sales_promo` with 498.997 importance. This evidence suggests that, for the global TFT model, information about the presence of promotions resulted more valuable for forecasting than information about past sales.

4.4 TFT on product tree clustering

One possible solution to the problem of high variability between each entity is to train a higher number of Temporal Fusion Transformer models on a partition of the original training set, where analogous time series belong to the same subset. The expected outcome is an improvement in forecast accuracy due to the models taking advantage of entities similar in nature. Such approach consisting in clustering the series based on their characteristics was suggested by [11] as a technique to improve the performances of cross-learning models. First however it is necessary to establish a similarity criterion to subdivide the original entity set. The more intuitive solution is to exploit the natural categorization of entities, based on the hypothesis that entities belonging to the same product category present (to a certain degree) similar values in the response variable.

In the use case considered the dataset is provided with a product tree subdividing the entity set in product categories for 7 levels of refinement. Reported below is the number of categories for each subdivision level:

- Level 1: 11 categories;
- Level 2: 18 categories;
- Level 3: 53 categories;
- Level 4: 341 categories;
- Level 5: 1037 categories;

Grocery	Chemicals	Fresh	Very Fresh
353	128	173	307
Personal Hygiene	Home	Seasonal & hardware	TOTAL
6	10	23	**1000**

Table 4.3: Subset cardinality and entity distribution at 1st subdivision level.

Model	MASE	MAE	MSE	RMSE	Running Time
TFT ProdTree	**0.706**	**1.356**	**21.126**	**1.943**	1910 min
TFT Global	0.754	1.482	26.653	2.093	2319 min
Holt-Winters	0.963	1.522	29.909	1.979	235 sec
MA baseline	1.085	1.787	30.995	2.272	**24** sec

Table 4.4: Global TFT model evaluation vs. Moving Average baseline and Holt-Winters benchmark.

- Level 6: 1957 categories;
- Level 7: 2610 categories;

The level of subdivision is an important hyperparameter to be determined. Given that the selected training set features only 1000 entities, in order to avoid empty and single-entity subsets the subdivision levels 5, 6 and 7 were discarded a priori. Due to computational limitations the final selected subdivision level was the 1st (11 categories), although it may be interesting to research the model performances at finer levels of partition.

Of the selected categories in the partition, 4 of them resulted empty as no entity belonged to those subsets, thus the 1000 time series were subdivided between categories as shown in Table 4.3. All the 7 Temporal Fusion Transformers models created on the partition were trained using the same number of epochs and hyperparameters than the global model.

The evaluation metrics were computed both at a global level on the entire set of 1000 entities, combining the predictions of each model, and separately on the forecasts of the single models on the partition. The global performances of the product tree approach, with respect to the baselines and the model trained on the whole set, are reported in Table 4.4.

The main result that can be observed is that the TFT model trained on the product tree partition not only outperformed both statistical benchmarks, but proved to be more accurate than its globally trained counterpart. Another important observation can be made on the running

Category	MASE	MAE	MSE	RMSE	N. of entities
Grocery	0.690	0.954	5.666	1.503	353
Chemicals	0.644	0.755	5.559	1.142	128
Fresh	0.704	1.344	20.473	1.899	173
Very Fresh	0.784	2.229	48.408	3.011	307
Pers. Hygiene	0.456	0.180	0.392	0.432	6
Home	**0.358**	**0.050**	**0.073**	**0.155**	10
Seas. & hardware	0.510	0.180	0.363	0.406	23

Table 4.5: Single TFT models evaluation on product tree partition

time of the partition models, which resulted lower than the one of the TFT on the whole entity set. Computational efficiency is an important topic in Machine Learning, and is especially sought after in practical applications on par with forecasting accuracy. The test results proved that subdividing the original entity set in partitions, prior to training a cross-learning model such as the Temporal Fusion Transformer, is particularly advantageous not only in terms of predictive performances, but also in regard of learning times. The partitions should be based on a similarity criterion, providing the model with the intuition that similar time series belong to the same subset. In the current case even an intuitive partition based on natural entity categorization proved effective in improving forecasting accuracy.

Performances of the single models on each category are reported in Table 4.5. It can be observed that, on average, the TFT models trained on fewer entities achieved better results than the ones trained on a larger number of time series. This evidence suggests that in terms of forecast accuracy is more advantageous to train on smaller subsets, so it might prove useful to employ a finer subdivision on the entity set.

4.4.1 Interpretation

As was done for the global model, the attention weight plots and variable importance weights were extracted for the TFT models on product tree partition as well. In this case it is worth noting that weights attend only to a smaller set of entities and represent the relevance of covariates or past time steps to the specific model trained on that subset.

In Figure 4.5 are displayed attention weights for each category. It appears a great variability

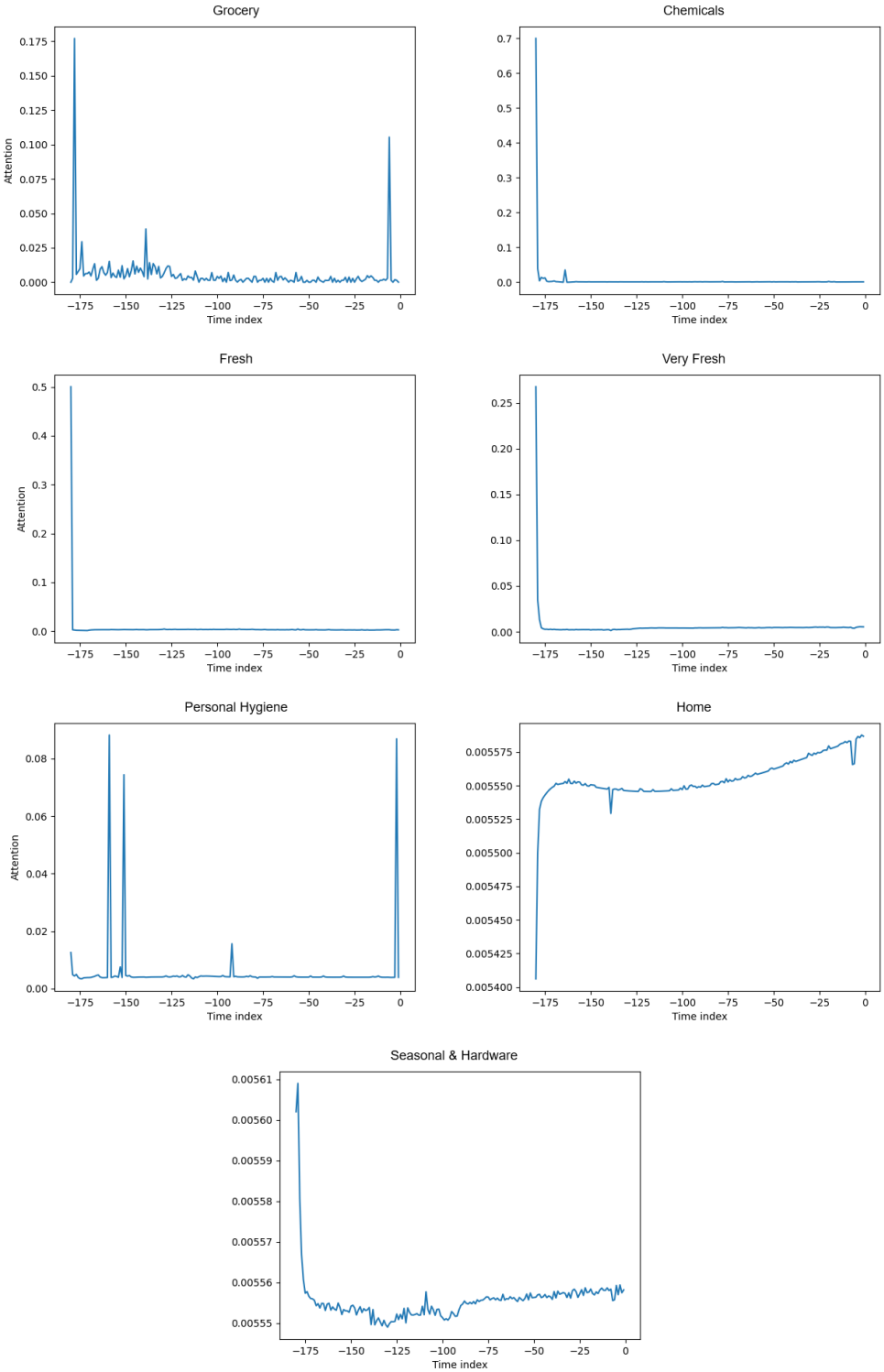

Figure 4.5: Attention weights for all 7 TFT models trained on product tree partition.

in attention patterns between products, with plots being fundamentally different from each other. For the model trained on Grocery subset (353 entities), the graph shows no particular trend but 3 significant attention spikes can be observed at lags 7, 140 and 178. Seasonal behaviour is still present but less prominent than the one observed in the global TFT model. The attention graphs for the categories Chemicals (128 entities), Fresh (173 entities) and Very Fresh (307 entities) are similar and show no strong persistent patterns, exception made for the spike at lag 180 suggesting an increase in the lookback horizon. Personal Hygiene (6 entities) plot showed significant spikes at lags 1, 93, 151 and 158 with no relevant seasonal patterns. One of the most different behaviours is observed in the model trained on the Home product subset (10 entities): the plot shows an increasing trend on smaller lags with a sudden decrease in attention weights for lags larger than 165. A very faint weekly seasonal pattern can also be noticed. Lastly, the plot relative to the Seasonal & hardware category (23 entities) portrays a slight increasing trend paired with a weekly pattern and a significant spike at lags 179-180.

In a similar fashion, Figure portrays variable importance weights for all models on each product category. Consistently with attention patterns, these plots also turned out widely different from subset to subset, thus suggesting that the partition resulted in fundamentally diverse models, which relied on entirely different parameters to forecast the assigned entities.

- *Decoder variables importance*: for the categories Grocery, Fresh and Very Fresh the variable `relative_time_idx` appeared the most significant for the prediction, meaning that the TFT models in order to forecast products belonging to those subsets relied more on the notion of relative time steps from current forecast time, instead of global time indicators. This result is consistent with intuition, since the categories in question include items that, due to their nature and limited shelf life, are purchased more frequently with respect to the other subsets.

- *Encoder variables importance*: regarding past regressors, TFT models trained on most subsets assigned the largest weight to the target variable `qty`, consistently with the model trained on the entire entity set. In particular, for subsets Chemicals, Very Fresh and Home, the models assigned weights close to 0 to all covariates different from `qty`, thus structuring the prediction around the target variable alone.

  Two exceptions were represented by the Temporal Fusion Transformers trained on Personal Hygiene and Seasonal & Hardware categories. In the first, the most relevant variable resulted `qty_promo`, followed by `holiday` and `qty` with similar weights. This deviation in behaviour from the global TFT model can be explained by the nature of the entities belonging to the set. Intuitively, products with longer shelf life are more likely to be purchased when subject to promotions. Regarding importance weights for

Figure 4.6: Decoder, encoder and static variable importance for the 7 TFT models trained on product tree partition.

the Seasonal & Hardware category, it appeared that `holiday` represented the most influential variable to determine the prediction. The subset includes time series relative to products that are often sold around festivities, thus justifying this particular behaviour.

- *Static variables importance*: finally, concerning static variable importance, no model acted consistently with the global TFT, which identified `qty_center` as the most relevant regressor. Four models relative to the categories Chemicals, Very Fresh, Personal Hygiene and Home relied the most on the unique item identifier `id`, suggesting high variability between entities and thus requiring additional context to correctly assign the forecast to the proper time series. The subset Grocery appeared more correlated to the `encoder_length`, which was fixed as 180 time steps for all models. Lastly, the models relative to the categories Fresh and Seasonal & Hardware presented increased importance in the variable `qty_scale`.

Both previous sections highlighted substantial differences between the Temporal Fusion Transformers used for forecasting over a product tree partition. In many cases their behaviour also diverged from the global model relative to the entity set in its entirety. Overall it appeared that training several models on subset of similar items allowed the architectures to capture more specific informations. Each transformer adapted to the features of the relative entity group, learning relationships on the data that the global method missed. As previously observed in Chapter 2 Section 2.3, the advantage of a cross-learning approach consists in enabling a machine learning model access to a wider variety of time series, thus allowing it to better infer the underlying distribution in the data. The increase in forecasting accuracy for the models trained with a product tree partition approach suggests that subdividing the original time series set in groups consistent with sub-distributions present in the data allows such mechanisms to better understand the dynamics present in each partition. This results in a more specific representation of target variables, combined with a lighter computational load.

Additionally, partitioning on an intuitive categorization based on natural features of the considered entities, results beneficial especially in light of the interpretability feature offered by the Temporal Fusion Transformer. The insights obtained on variable importance and the extracted temporal patterns are model-specific, thus by definition category-specific. In many real-world applications, informations relative to user-defined subsets in the data can result even more valuable than global explainability due to the fact that they allow a specific understanding of the dynamics inside each category. Moreover, subset-level interpretations present an important resource also in a procedural sense, since they can enable aimed and ad-hoc interventions on sets of items the user is familiar with.

4.5 TFT on k-means clustering

As observed in Section 4.4, the clustering approach proved successful in improving model performances. Natural item categorization however, may not be representative enough of the underlying sub-distributions present in the data. Entities belonging to the same category can present very different behaviours and at the same time it can occur a high similarity between apparently unrelated time series. To provide an example consistent with the considered dataset, products such as milk and biscuits are often purchased together despite being part of different branches at various levels of the product tree. Moreover, as it was observed in Chapter 2 Section 2.3, a natural partitioning of the entities may not always be available, thus enabling the use of a clustering approach as an option to avoid accuracy deterioration in the forecasts. The aim of the current section is to ascertain whether performing clustering based on a correlation criterion between time series, instead of exploiting their natural categorization, can bring an improvement in forecasting accuracy. Can this form of partitioning allow the TFT model to represent more accurately the dataset distribution or will it hinder the process by not providing enough input variety?

For the grouping criterion, k-means clustering was selected. k-means is a centroid-based partitioning algorithm often used for vector quantization, feature learning and cluster analysis, but has been employed in time series categorization as well [38]. The metric to be minimized is the squared Euclidean distance between vectors, which can serve as a similarity measure between entities. Other more sophisticated distance-based time series clustering methods have been formulated, such as Dynamic Time Warping (DTW) [39], Longest Common Subsequence models (LCSS) [40] and Spatial Assembling Distance method (SpADe) [41] which can match temporal patterns also when subject to deformations, shifting and scaling or can deal with series of varying length. Given the nature of the retail dataset, which provides uniform, same-length entities, simple k-means clustering was chosen instead of more complex partitioning methods. Another reason for settling on this simpler model can be justified by the necessity of a fast preprocessing algorithm, which purpose is to prepare and subdivide the data prior to training several Temporal Fusion Transformers. It is important to observe that the k-means clustering algorithm tends to produce local optimal solutions dependent on the centroids initialization. This is not a big issue in the current application, since it has been already proven in Section 4.4 that the TFT model can provide good forecasts even when trained on subsets of highly variable entities. The aim is to obtain a good enough time series partition based on a closeness criterion, hence k-means is a good method for the task at hand.

Cluster 0	Cluster 1	Cluster 2	Cluster 3
857	1	6	28
Cluster 4	Cluster 5	Cluster 6	TOTAL
105	2	1	**1000**

Table 4.6: Cluster cardinality on 7-means partition.

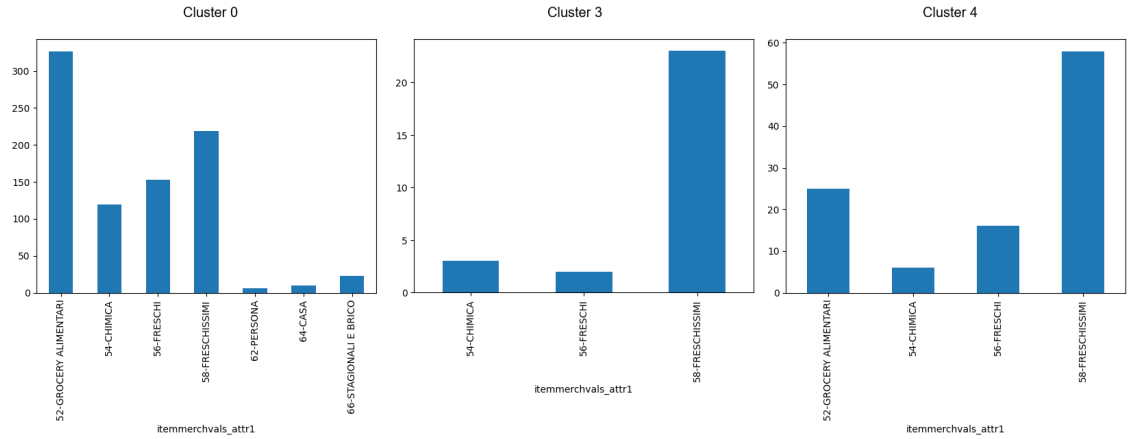

Figure 4.7: Product tree categories distribution in clusters 0, 3 and 4.

To better compare training approaches the number of clusters k was set to 7, as the subsets derived from partitioning the entity set by means of the first product tree level. The set of entities provided as input to the k-means algorithm featured only the target variable `qty`. The reason for this choice was to limit the size of the input vectors, since k-means is a model that greatly suffers from the curse of dimensionality. Table 4.6 presents the cardinality of each cluster generated from the algorithm, while Figure displays the distribution of first-level product tree categories between the most numerous clusters.

It can be observed that the clusters identified by the algorithm are highly different in size. The vast majority of the entities was grouped in cluster 0, while the remaining time series were subdivided between smaller subsets. Of particular interest are the clusters 1, 2, 5 and 6 containing respectively 1, 6, 2 and 1 entities, all belonging to the Very Fresh product tree category. This result suggests that entities belonging to such clusters presented a widely different behaviour from the majority of the other time series. Since the similarity metric for the algorithm is represented by the squared euclidean distance, it is possible that many time series were grouped together despite presenting different temporal patterns due to the global distance resulting small

Model	MASE	MAE	MSE	RMSE	Running Time
TFT k-means	0.713	1.395	22.708	1.961	1912 min
TFT ProdTree	**0.706**	**1.356**	**21.126**	**1.943**	1910 min
TFT Global	0.754	1.482	26.653	2.093	2319 min
Holt-Winters	0.963	1.522	29.909	1.979	235 sec
MA baseline	1.085	1.787	30.995	2.272	**24** sec

Table 4.7: K-means TFT model evaluation vs. Moving Average baseline and Holt-Winters benchmark.

enough. Figure 4.8 represents the cluster centroids, namely the points in each cluster minimizing the total squared euclidean distance between them and the vectors belonging to that subset. Exception made for the singleton clusters 1 and 6, in which the centroids coincide with the contained entity, in all other cases the centroids do not represent a time series present in the dataset but an average of the points in the cluster if the problem was solved by means of Lloyd's algorithm CITA. Clear weekly seasonal patterns are noticeable in the centroids relative to clusters 0, 2, 3 and 4, while the 6th centroid presented a similar pattern as well but paired with an additional yearly oscillation as well. A yearly seasonal behaviour featuring spikes interrupted by a period of zero demand is noticeable in the centroids of clusters 1 and 5.

Following the approach adopted when testing the Temporal Fusion Transformer on the product tree partition, all models were trained with the same epochs and hyperparameters as the global model in Section 4.3. Error metrics were computed both globally and at cluster level as well. Table 4.7 reports the global performances with respect to the statistical baselines and the two previous training approaches for the TFT. It can be observed that applying the k-means clustering approach to the training process of the model improved its performances with respect to the global framework, both in terms of error metrics and running time. In particular regarding the computational costs of the two grouping approaches they appeared almost identical, both methods training 17.59% faster with respect to the TFT on the entire entity set. Product tree partitioning however, provided better results in terms of error metrics than k-means partitioning: the first performed 6.37% better than the global model against the 5.44% improvement brought by the second approach. One reason may be the wider disparity in cardinality between the subsets.

Table 4.8 illustrates the model performances at cluster level, displaying the error metrics of each TFT model on its assigned partition. A large disparity in forecasting accuracy is evident

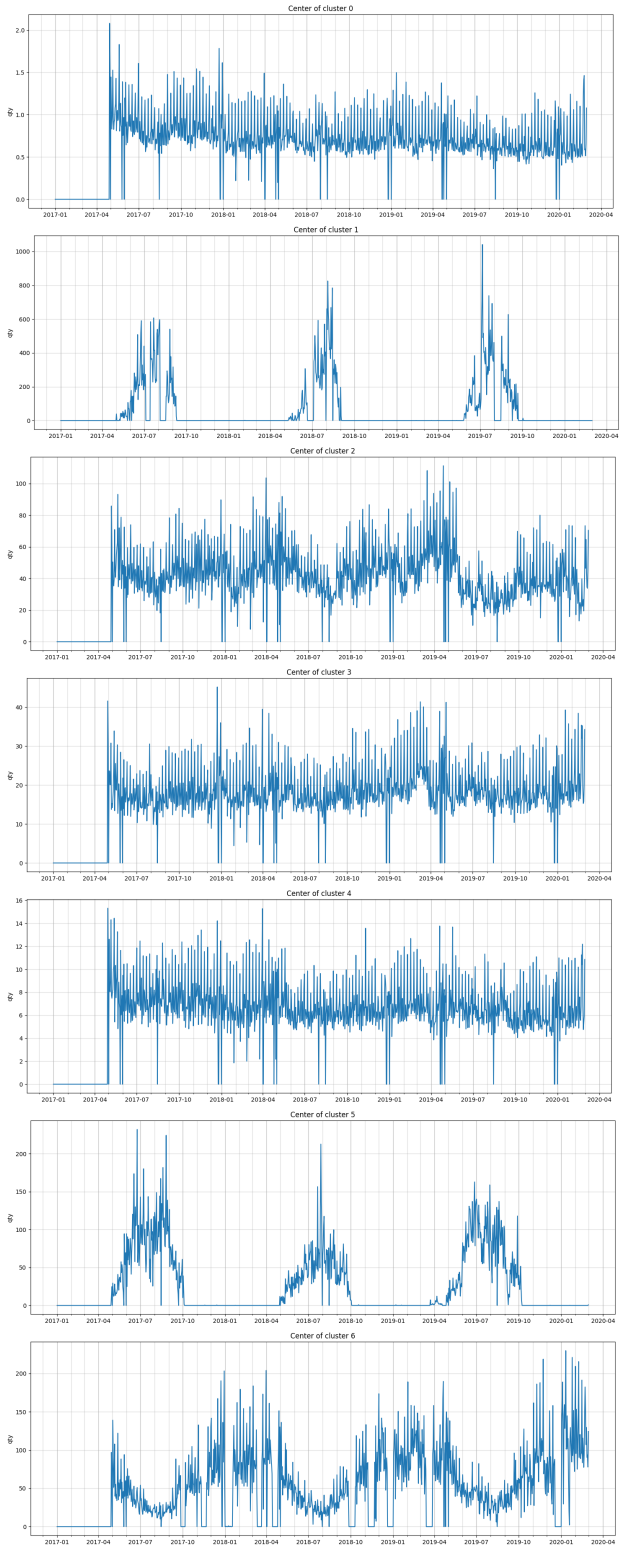

Figure 4.8: Centroid plots for each cluster.

Cluster	MASE	MAE	MSE	RMSE	N. of entities
Cluster 0	0.654	0.594	2.463	0.936	857
Cluster 1	**0.0002**	**0.007**	**0.003**	**0.053**	1
Cluster 2	1.247	20.786	878.777	26.483	6
Cluster 3	1.187	9.662	206.154	12.620	28
Cluster 4	1.045	4.175	45.801	5.494	105
Cluster 5	0.001	0.015	0.017	0.092	2
Cluster 6	2.380	52.279	4743.187	68.870	1

Table 4.8: Single TFT models evaluation on k-means generated clusters.

between subsets, however this occurrence does not appear to be related to cardinality. Cluster 0, the one including the largest number of time series, achieved one of the best results in terms of error metrics, outperformed only by clusters 1 and 5 containing respectively 1 and 2 entities. At the same time, the worst performing TFT model was the one trained on cluster 6, a singleton as well. This result proves that Temporal Fusion Transformers do not necessarily benefit from an excessive fragmentation of the entity set. As highlighted by [11], the biggest advantage of cross-learning models consists in their ability to exploit multiple time series to predict single ones, learning the underlying data-generating process. It follows that global performances are dependent on the chosen partition, which should be representative enough of the sub-distributions present in the data while at the same time provide enough variability in each subset, so that the model can effectively learn to represent effectively each process.

4.5.1 Interpretation

Similarly to the previous two sections, the interpretability feature of the TFT model was exploited to extract information regarding attention patterns and variable importance for each cluster. As remarked by [13] and in Section 4.4 distance-based clustering approaches may yield unintuitive results and limited explainability, but can provide knowledge on the composition of entities subsets nonetheless.

In Figure 4.9 are represented the attention patterns for every TFT model trained on the partition generated by the k-means algorithm. It clearly appears that the 7 model exhibited a widely different behaviour between each other, even more so than in the product tree approach. Clus-

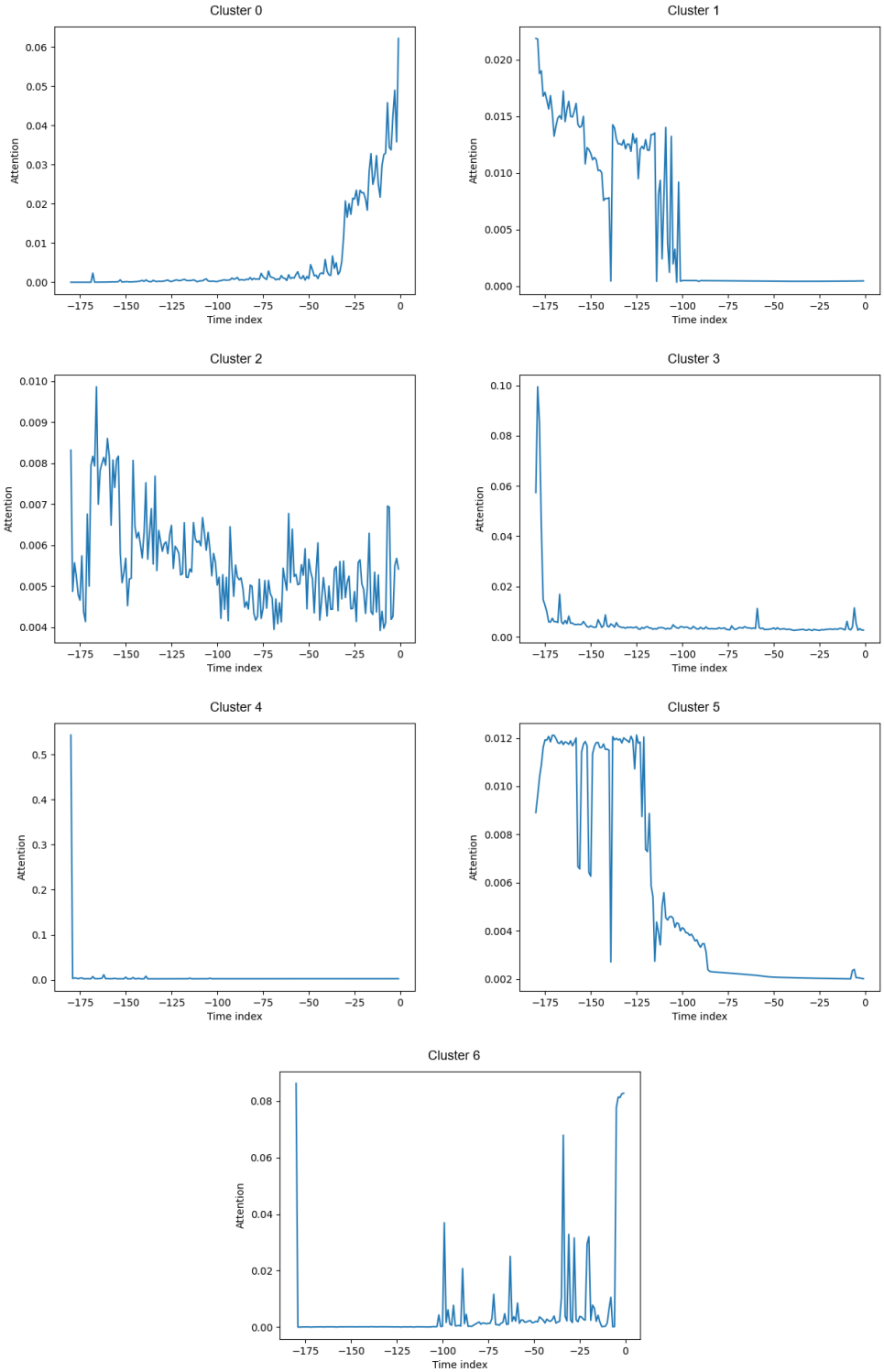

Figure 4.9: Attention weights for all 7 TFT models trained on k-means generated clusters.

ter 0, the largest subset in cardinality, presented an increasing attention towards the 30 most recent time steps, thus relying its forecast on data from the past month. The two best performing models, trained on clusters 1 and 5, appear to share similar attention patterns as it can be observed in the plots. Both models present an increased relevance in time steps greater than 85, almost ignoring earlier data. This behaviour can be explained by observing their relative centroids plots in Figure 4.8, where a strong yearly seasonality is present, thus justifying the decreased relevance in recent time steps. The second and third largest subsets, respectively cluster 3 and 4, exhibited a more extreme behaviour showing no persistent temporal patterns except for a spike on the oldest lag. On the other hand, one of the models presenting the most erratic attention behaviour appeared to be the one trained on cluster 2, the second-to-last in terms of forecasting accuracy. Temporal importance shows no trend or seasonal pattern and increases in variance towards older lags, overall revealing a very noisy behaviour?. Lastly, the TFT which achieved the worst results in terms of error metrics, namely the one relative to the 6th subset, reported high attention peaks at earlier time steps with no apparent regularity. A significant spike can also be noticed at the 180th lag, which can be interpreted as signal of a lookback window too short. This result may suggest the inability of the TFT model in capturing the yearly oscillations visible in the cluster centroid, as lags greater than 120 received little to none attention weight.

Similarly, Figure 4.10 displays the variable selection weights for the TFT models on each cluster. As expected from the previous results, in the current approach covariate relevance between models is not consistent as well. It is possible to observe that, for all types of covariates, the clusters with the highest cardinality (number 0 and 4) presented the widest disparity in variable selection weights with one variable achieving almost 100% importance while all remaining covariates featured weights close to 0. This behaviour was not observed in the TFT adopting the product tree approach and can be attributed to the fact that in the k-means clustering approach each model is allowed to perform predictions from a pool of more similar time series, thus relying on a smaller number of possibly noisy covariates.

- *Decoder variables importance*: models trained on clusters 0, 2, 3, 5 and 6 appeared to be the ones which relied more strongly on the global time indicator `time_idx`. Subsets number 1 and 4 on the other hand featured models that prioritized the offset in lags from the forecast time, namely `relative_time_idx`. This result shows a deviation from the product tree approach, in which the vast majority of the entities belonged to subsets favouring the known future covariate `relative_time_idx`, consistently with the behaviour of the globally trained TFT. Due to the heterogeneity of the entities in each cluster it is hard to formulate a qualitative interpretation of this behaviour.

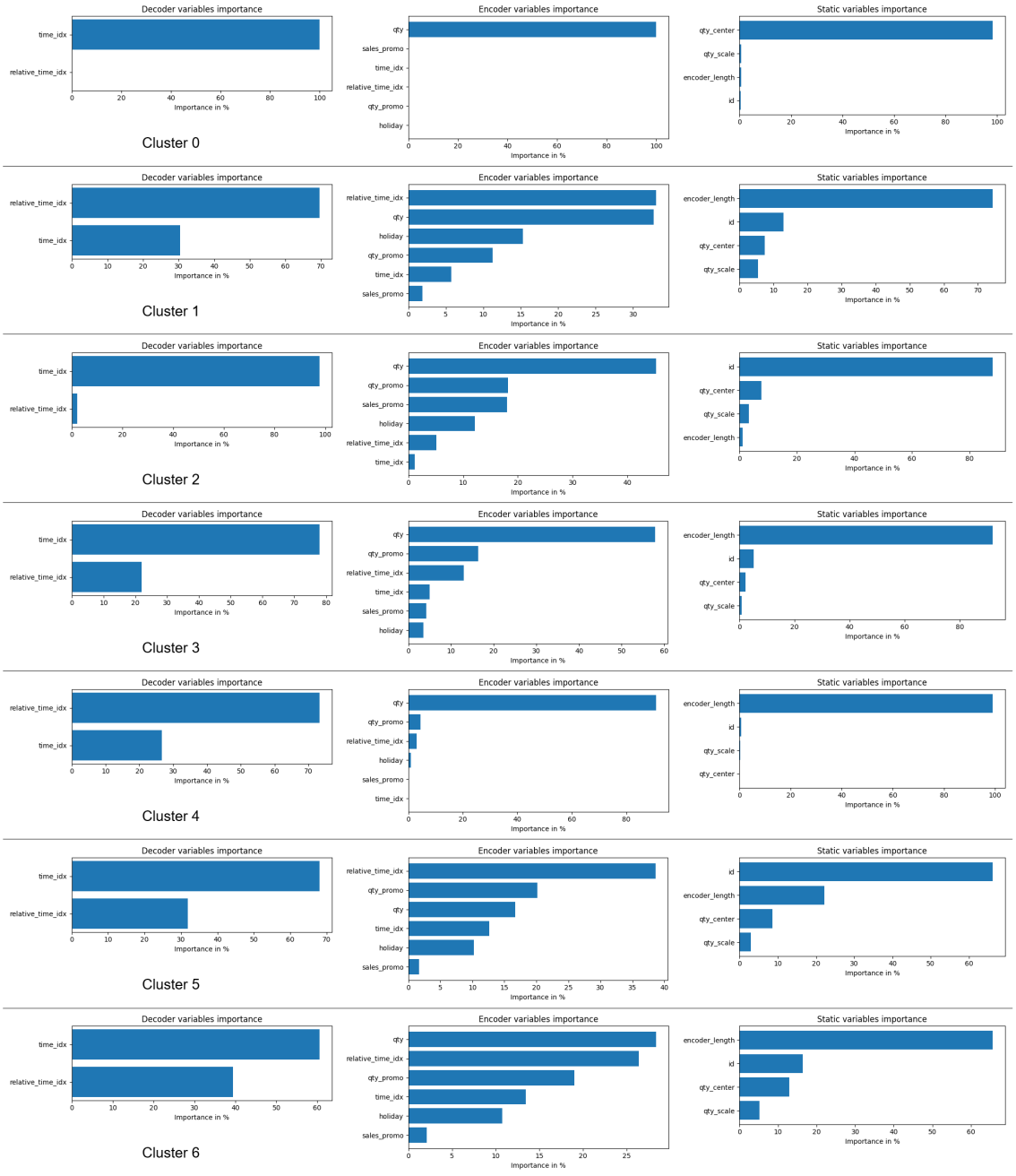

Figure 4.10: Decoder, encoder and static variable importance for the 7 TFT models trained on k-means generated clusters.

- *Encoder variables importance*: As previously observed, clusters 0 and 4 focused almost entirely on the target variable `qty`, assigning negligible importance to the other past covariates. Another possible reason of this behaviour is that in the retail field promotions patterns are often dependent on the item category, and such information can be lost when training a model on a widely heterogeneous subset of entities. Consistently with the global and product tree training approaches, clusters 2, 3 and 6 featured `qty` as the most relevant covariate as well, albeit not as predominantly as in the previous cases. The model trained on subset 5 on the other hand relied more strongly on the `relative_time_idx` covariate than on the target `qty`, while the TFT relative to subset 1 presented comparable weights for both variables. It is important to observe that in both cases the time series presented evident yearly spikes inbetweened by periods of zero demand, which explains the increased importance of the relative time indicator with respect to previous training approaches.
- *Static variables importance*: regarding static covariates, only cluster 0 behaved consistently with the global TFT model, assigning the highest weight to `qty_center`. This notion, combined with the fact that cluster 0 contains the 85.7% of the total entities, can indicate that due to the wide variety of time series belonging to the subset, considering their average value proved more effective in providing accurate forecasts. For all remaining models the variable `encoder_length` achieved the highest importance, with the exception of clusters 2 and 5 which relied the most on the entity identifier `id`. These last two subsets, having respectively cardinality 6 and 2, are expected to contain highly similar time series, thus causing the models trained on them to require more entity-specific information to better distinguish the forecasts.

In conclusion, in this specific application employing a k-means clustering approach to the Temporal Fusion Transformer training did not yield particular improvements in performance over partitioning the entity set on its product tree. Such choice, although providing a considerable increase in forecasting accuracy over the global model, sacrifices part of the information offered by the combination of the TFT's interpretability feature with the contextual knowledge provided by the natural partition of the dataset. Time series clustering represents however a valid option, especially in cases in which the data does not present any form of categorization, moreover grouping time series on the basis of their pattern similarity can offer additional insights towards the nature of the considered entities. Such knowledge, combined with the domain information relative to the specific application field, is a valuable addition to the forecast results. Finally, as remarked by [16] and [17], employing a clustering technique to the training of a cross-learning method over training it globally on the entire set of entities can signify not only an improvement in forecast quality, but also a lowering of the computational cost and time required, a much valued feature in real-world applications.

5 Conclusion

As highlighted by the results of the M4 competition [6] [11], cross-learning models represent a novel and promising approach to multiple time series forecasting. Temporal Fusion Transformers are a recent pure machine learning model belonging to this category and as such have both its merits and limitations. In many occurrences it has been observed that pure Machine Learning methods provided poor performance when compared to hybrid or pure statistical methods, even naive ones, as highlighted in [6] and [36]. The experiments however revealed that Temporal Fusion Transformers were able to provide good predictions in terms of accuracy on a family of entities noticeably harder to forecast, namely lumpy and intermittent time series. In this task the TFT model outperformed a simple moving average baseline and Holt-Winters exponential smoothing, a reliable statistical method for forecasting entities presenting large periods of zero demand. Another disadvantage attributed to machine learning models for time series forecasting is represented by the high computational resources required for training, disadvantage that becomes critical when faced with the task of forecasting hundreds of thousands of entities and is not mitigated by the evolution in speed and cost of high-level computers. In practical cases computational cost bears consistent weight in the choice of a forecasting model, as speed is often required together with prediction accuracy. Temporal Fusion Transformers are no different, presenting high computational complexity that can however be mitigated by applying a clustering approach on the original set. The experiments showed that applying a form partitioning during training phase not only helped in reducing the computational burden, but also resulted in an improvement in forecasting accuracy. Both tested partitioning approaches,

namely exploiting the natural categorization present in the dataset and k-means clustering for time series, yielded promising results and featured similar running times.

Another thing that separates the Temporal Fusion Transformer from other Machine Learning based models, and from most statistical methods is the interpretability feature it provides. Depending on the application it might be useful to access data on the relationships learned by the model, as it can offer important insights on how to intervene to influence a trend in the target variable. In many real-world forecasting scenarios it is often required a tradeoff between prediction accuracy and interpretability and Temporal Fusion Transformers can, to a certain degree, satisfy these requirements thanks to its architecture. Both information on variable importance and global temporal patterns can be extracted by such feature, which gains even more value when combined with a form of natural partitioning based on domain knowledge on the dataset.

In conclusion, Temporal Fusion Transformers represent a novel and flexible approach for tackling multiple time series forecasting problems. Its cross-learning nature combined with the possibility of adopting a partitioning technique during training allow the model to achieve satisfactory results in terms of forecast quality, by capturing global relations present in the data and reducing the computational load with respect to other Machine Learning models trained in a series-by-series fashion. Moreover, the additional information they offer together with the forecast can allow them to be preferred to other higher performing methods lacking such feature.

References

[1] 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

[2] N. Wagner, Z. Michalewicz, S. Schellenberg, C. Chiriac, and A. S. Mohais, "Intelligent techniques for forecasting multiple time series in real-world systems," *Int. J. Intell. Comput. Cybern.*, vol. 4, pp. 284–310, 2011.

[3] E. Spiliotis, S. Makridakis, A.-A. Semenoglou, and V. Assimakopoulos, "Comparison of statistical and machine learning methods for daily sku demand forecasting," *Operational Research*, pp. 1–25, 2020.

[4] S. Kaushik, A. Choudhury, P. K. Sheron, N. Dasgupta, S. Natarajan, L. A. Pickett, and V. Dutt, "Ai in healthcare: Time-series forecasting using statistical, neural, and ensemble architectures," *Frontiers in Big Data*, vol. 3, 2020. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fdata.2020.00004

[5] N. G. Polson and V. O. Sokolov, "Deep learning for short-term traffic flow prediction," *Transportation Research Part C: Emerging Technologies*, vol. 79, pp. 1–17, jun 2017. [Online]. Available: https://doi.org/10.1016%2Fj.trc.2017.02.024

[6] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The m4 competition: 100,000 time series and 61 forecasting methods," *International Journal of Forecasting*, vol. 36, no. 1, pp. 54–74, 2020, m4 Competition. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207019301128

[7] J. D. Croston, "Forecasting and stock control for intermittent demands," *Operational Research Quarterly (1970-1977)*, vol. 23, no. 3, pp. 289–303, 1972. [Online]. Available: http://www.jstor.org/stable/3007885

[8] C. A. Sims, "Macroeconomics and reality," *Econometrica*, vol. 48, no. 1, pp. 1–48, 1980. [Online]. Available: http://www.jstor.org/stable/1912017

[9] V. Assimakopoulos and K. Nikolopoulos, "The theta model: a decomposition approach to forecasting," *International Journal of Forecasting*, vol. 16, no. 4, pp. 521–530, 2000, the M3- Competition. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207000000662

[10] F. Petropoulos, D. Apiletti, V. Assimakopoulos, M. Z. Babai, D. K. Barrow, S. Ben Taieb, C. Bergmeir, R. J. Bessa, J. Bijak, J. E. Boylan, J. Browell, C. Carnevale, J. L. Castle, P. Cirillo, M. P. Clements, C. Cordeiro, F. L. Cyrino Oliveira, S. De Baets, A. Dokumentov, J. Ellison, P. Fiszeder, P. H. Franses, D. T. Frazier, M. Gilliland, M. S. Gönül, P. Goodwin, L. Grossi, Y. Grushka-Cockayne, M. Guidolin, M. Guidolin, U. Gunter, X. Guo, R. Guseo, N. Harvey, D. F. Hendry, R. Hollyman, T. Januschowski, J. Jeon, V. R. R. Jose, Y. Kang, A. B. Koehler, S. Kolassa, N. Kourentzes, S. Leva, F. Li, K. Litsiou, S. Makridakis, G. M. Martin, A. B. Martinez, S. Meeran, T. Modis, K. Nikolopoulos, D. Önkal, A. Paccagnini, A. Panagiotelis, I. Panapakidis, J. M. Pavía, M. Pedio, D. J. Pedregal, P. Pinson, P. Ramos, D. E. Rapach, J. J. Reade, B. Rostami-Tabar, M. Rubaszek, G. Sermpinis, H. L. Shang, E. Spiliotis, A. A. Syntetos, P. D. Talagala, T. S. Talagala, L. Tashman, D. Thomakos, T. Thorarinsdottir, E. Todini, J. R. Trapero Arenas, X. Wang, R. L. Winkler, A. Yusupova, and F. Ziel, "Forecasting: theory and practice," *International Journal of Forecasting*, vol. 38, no. 3, pp. 705–871, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207021001758

[11] 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, no. 3, pp. 1072–1084, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207020301850

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[13] K. Bandara, C. Bergmeir, and S. Smyl, "Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach," 2017. [Online]. Available: https://arxiv.org/abs/1710.03222

[14] S. Smyl, "A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting," *International Journal of Forecasting*, vol. 36, no. 1, pp. 75–85, 2020, m4 Competition. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207019301153

[15] P. Montero-Manso, G. Athanasopoulos, R. J. Hyndman, and T. S. Talagala, "Fforma: Feature-based forecast model averaging," *International Journal of Forecasting*, vol. 36, no. 1, pp. 86–92, 2020, m4 Competition. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207019300895

[16] S. Fan, C. Non-member, and L. Chen, "Electricity peak load forecasting with self-organizing map and support vector regression," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 1, pp. 330 – 336, 09 2006.

[17] H. Mori and A. Yuihara, "Deterministic annealing clustering for ann-based short-term load forecasting," *IEEE Transactions on Power Systems*, vol. 16, no. 3, pp. 545–551, 2001.

[18] T. Warren Liao, "Clustering of time series data—a survey," *Pattern Recognition*, vol. 38, no. 11, pp. 1857–1874, 2005.

[19] D. Piccolo, "A distance measure for classifying arima models," *Journal of Time Series Analysis*, vol. 11, no. 2, pp. 153–164, 1990. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9892.1990.tb00048.x

[20] K. Bandara, C. Bergmeir, and S. Smyl, "Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach," 2018.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[22] M. H. S. Eddin, R. Roscher, and J. Gall, "Location-aware adaptive denormalization: A deep learning approach for wildfire danger forecasting," 2022.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[24] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," 2018.

[25] O. K. Oyedotun, K. A. Ismaeil, and D. Aouada, "Why is everyone training very deep neural network with skip connections?" *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2022.

[26] C. Rudin and J. Radin, "Why Are We Using Black Box Models in AI When We Don't Need To? A Lesson From an Explainable AI Competition," *Harvard Data Science Review*, vol. 1, no. 2, nov 22 2019, https://hdsr.mitpress.mit.edu/pub/f9kuryi8.

[27] C. Bergmeir and J. M. Benítez, "On the use of cross-validation for time series predictor evaluation," *Information Sciences*, vol. 191, pp. 192–213, 2012, data Mining for Software Trustworthiness. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025511006773

[28] M. Nelson, T. Hill, W. Remus, and M. O'Connor, "Time series forecasting using neural networks: Should the data be deseasonalized first?" *Journal of forecasting*, vol. 18, no. 5, pp. 359–367, 1999.

[29] G. Zhang and M. Qi, "Neural network forecasting for seasonal and trend time series," *European Journal of Operational Research*, vol. 160, no. 2, pp. 501–514, 2005, decision Support Systems in the Internet Age. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221703005484

[30] J. M. Rožanec and D. Mladenić, "Reframing demand forecasting: a two-fold approach for lumpy and intermittent demand," 2021. [Online]. Available: https://arxiv.org/abs/2103.13812

[31] A. C. Türkmen, T. Januschowski, Y. Wang, and A. T. Cemgil, "Forecasting intermittent and sparse time series: A unified probabilistic framework via deep renewal processes," *PLOS ONE*, vol. 16, pp. 1–26, 11 2021. [Online]. Available: https://doi.org/10.1371/journal.pone.0259764

[32] G. O. Kaya, M. Sahin, and O. F. Demirel, "Intermittent demand forecasting: A guideline for method selection," *Sādhanā*, vol. 45, pp. 1–7, 2020.

[33] K. R, K. Kayathwal, G. Dhama, and A. Arora, "A survey on classical and deep learning based intermittent time series forecasting methods," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–7.

[34] A. Muhaimin, D. D. Prastyo, and H. Horng-Shing Lu, "Forecasting with recurrent neural network in intermittent demand data," in *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 2021, pp. 802–809.

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

[36] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "Statistical and machine learning forecasting methods: Concerns and ways forward," *PLOS ONE*, vol. 13, no. 3, pp. 1–26, 03 2018. [Online]. Available: https://doi.org/10.1371/journal.pone.0194889

[37] P. S. Kalekar *et al.*, "Time series forecasting using holt-winters exponential smoothing," *Kanwal Rekhi school of information Technology*, vol. 4329008, no. 13, pp. 1–13, 2004.

[38] X. Huang, Y. Ye, L. Xiong, R. Y. Lau, N. Jiang, and S. Wang, "Time series k-means: A new k-means type smooth subspace clustering for time series data," *Information Sciences*, vol. 367-368, pp. 1–13, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025516303796

[39] V. Niennattrakul and C. A. Ratanamahatana, "On clustering multimedia time series data using k-means and dynamic time warping," in *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*, 2007, pp. 733–738.

[40] G. Soleimani and M. Abessi, "Dlcss: A new similarity measure for time series data mining," *Engineering Applications of Artificial Intelligence*, vol. 92, p. 103664, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095219762030110X

[41] Y. Chen, M. A. Nascimento, B. C. Ooi, and A. K. H. Tung, "Spade: On shape-based pattern detection in streaming time series," in *2007 IEEE 23rd International Conference on Data Engineering*, 2007, pp. 786–795.

Acknowledgments

I would like to express my sincere gratitude to my thesis advisor, professor Mariangela Guidolin, for her precious feedback and her valuable insights. My appreciation also goes to my internship tutors at ACT Operation Research IT, Ruben Manganiello and Giuliano Squarcina, who shared with me their knowledge and expertise, and with whom I had the pleasure of working.

Many thanks should also go to my family for their continuous understanding and for their precious words of encouragement. Their support is what motivated me the most during the compilation of this thesis.

Lastly, I would like to mention the staff of San Daniele del Friuli Municipal Library for pointing me towards some useful references and for hosting my writing sessions.