big data and cognitive computing

MDPI

Article

Investigating the Accuracy of Autoregressive Recurrent Networks Using Hierarchical Aggregation Structure-Based Data Partitioning

José Manuel Oliveira [1,2,†] and Patrícia Ramos [2,3,*,†]

[1] Faculty of Economics, University of Porto, rua Dr. Roberto Frias, 4200-464 Porto, Portugal; jmo@fep.up.pt
[2] Institute for Systems and Computer Engineering, Technology and Science, rua Dr. Roberto Frias, 4200-465 Porto, Portugal
[3] CEOS.PP, ISCAP, Polytechnic of Porto, rua Jaime Lopes Amorim s/n, 4465-004 São Mamede de Infesta, Portugal
* Correspondence: patricia@iscap.ipp.pt
† These authors contributed equally to this work.

Abstract: Global models have been developed to tackle the challenge of forecasting sets of series that are related or share similarities, but they have not been developed for heterogeneous datasets. Various methods of partitioning by relatedness have been introduced to enhance the similarities of sets, resulting in improved forecasting accuracy but often at the cost of a reduced sample size, which could be harmful. To shed light on how the relatedness between series impacts the effectiveness of global models in real-world demand-forecasting problems, we perform an extensive empirical study using the M5 competition dataset. We examine cross-learning scenarios driven by the product hierarchy commonly employed in retail planning to allow global models to capture interdependencies across products and regions more effectively. Our findings show that global models outperform state-of-the-art local benchmarks by a considerable margin, indicating that they are not inherently more limited than local models and can handle unrelated time-series data effectively. The accuracy of data-partitioning approaches increases as the sizes of the data pools and the models' complexity decrease. However, there is a trade-off between data availability and data relatedness. Smaller data pools lead to increased similarity among time series, making it easier to capture cross-product and cross-region dependencies, but this comes at the cost of a reduced sample, which may not be beneficial. Finally, it is worth noting that the successful implementation of global models for heterogeneous datasets can significantly impact forecasting practice.

Keywords: global models; deep learning; data partitioning; time-series features; model complexity; intermittent demand; retail

Citation: Oliveira, J.M.; Ramos, P. Investigating the Accuracy of Autoregressive Recurrent Networks Using Hierarchical Aggregation Structure-Based Data Partitioning. *Big Data Cogn. Comput.* **2023**, *7*, 100. https://doi.org/10.3390/bdcc7020100

Academic Editor: Alberto Abelló

Received: 10 April 2023
Revised: 8 May 2023
Accepted: 16 May 2023
Published: 18 May 2023

1. Introduction

Sales forecasts at the SKU (stock-keeping unit) level are essential for effective inventory management, production planning, pricing and promotional strategies, and sales performance tracking [1]. SKUs represent individual products or product variants within a larger product line. By forecasting sales at the SKU level, businesses can optimize their inventory levels to ensure they have enough stock on hand to meet demand without overstocking and tying up capital. This helps to reduce inventory holding costs and avoid stockouts, which can result in lost sales and dissatisfied customers [2]. SKU-level sales forecasts can also help businesses plan their production schedules and ensure they have enough raw materials and resources to meet demand. This can help reduce production downtime and minimize waste and inefficiencies. SKU-level sales forecasts can help businesses determine the optimal pricing and promotional strategies for each SKU. For example, if a particular SKU is expected to have high demand, a business may choose to increase the price to maximize profit margins. Alternatively, if a SKU is not selling as well as expected, a business

may choose to offer discounts or promotions to stimulate sales. SKU-level sales forecasts also allow businesses to track the performance of individual products and identify trends and patterns in consumer behaviour. This can help businesses make data-driven decisions and adjust their strategies accordingly [3].

Retailers typically offer a vast range of products, from perishable items such as fresh produce to non-perishable goods such as electronics and clothing. Each product has distinct demand patterns that may differ based on location, time, day of the week, season, and promotional events. Forecasting sales for each of these items can be a daunting and complicated task, particularly since retailers often sell products through multiple channels, including physical stores, online platforms, mobile apps, and marketplaces, each with its own set of difficulties and opportunities that must be considered when forecasting sales. Additionally, in the retail sector, demand forecasting is a regular occurrence, often performed weekly or daily, to ensure optimal inventory levels. As a result, advanced models and techniques are necessary to tackle the forecasting problem, which must be automated to reduce manual intervention, robust to handle various data types and scenarios, and scalable to accommodate large data volumes and changing business requirements [4].

1.1. Local versus Global Forecasting Models

For decades, the prevailing approach in time-series forecasting has been to view each time series as a standalone dataset [5,6]. This has led to the use of localized forecasting techniques that treat each series individually and make predictions based solely on the statistical patterns observed in that series. The Exponential Smoothing State Space Model (ETS) [7] and Auto-Regressive Integrated Moving Average Model (ARIMA) [8] are notable examples of such methods. While these approaches have been widely used and have produced useful results in many cases, they have their limitations [9]. Currently, businesses often collect vast amounts of time-series data from similar sources on a regular basis. For example, retailers may collect data on the sales of thousands of different products, manufacturers may collect data on machine measurements for predictive maintenance, and utility companies may gather data on smart-meter readings across many households. While traditional local forecasting techniques can still be used to make predictions in these situations, they may not be able to fully exploit the potential for learning patterns across multiple time series. This has led to a paradigm shift in forecasting, where instead of treating each individual time series separately, a set of series is seen as a dataset [10].

A global forecasting model (GFM) has the same set of parameters, such as weights in the case of a neural network [11], for all the time series (all time series in the dataset are forecast using the same function), in contrast to a local model, which has a unique set of parameters for each individual series. This means that the global model takes into account the interdependencies between the variables across the entire dataset, whereas local models focus only on the statistical properties of each individual series. In the retail industry, it is possible to capture cross-product and cross-region dependencies, which can result in more-accurate forecasts across the entire range of products. When we talk about cross-product dependencies, we are referring to the connection between different products. Alterations in one product can have an impact on the demand or performance of another product. For instance, if two products are complementary or substitutable, changes in the sales of one product can affect the sales of the other. Conversely, the demand for a particular product may exhibit a similar pattern for all varieties, brands, or packaging options in various stores. Cross-region dependencies refer to the link between different regions or locations. Changes in one region, such as fluctuations in economic conditions or weather patterns, may have an effect on the demand or performance in another region. Global forecasting models, typically built using advanced machine learning techniques such as deep learning and artificial neural networks, are gaining popularity, as seen in the works of [12–15], and have outperformed local models in various prestigious forecasting competitions such as the M4 [16,17] and the recent M5 [18–20], as well as those held on the Kaggle platform with a forecasting purpose [21]. In summary, the recent paradigm shift in forecasting

recognizes that analysing multiple time series together as a dataset can yield significant improvements in accuracy and provide valuable insights into underlying patterns and trends. This shift has opened up new opportunities for businesses to leverage machine learning and other advanced techniques to gain a competitive advantage in forecasting and decision making. However, there are still many challenges to overcome, such as the need for skilled data scientists, significant amounts of data and time for training the models, and sufficient computational and data infrastructures. Additionally, to promote the adoption and sustained usage of GFMs in practice, it is essential to have expertise within the organization, along with model transparency and intelligibility, which are crucial attributes for establishing user trust.

1.2. Relatedness between Time Series

The successful aforementioned studies are based on the assumption that GFMs are effective because there exists a relationship between the series (all come hypothetically from similar data-generating processes), enabling the model to recognize complex patterns shared across them. Nevertheless, none of these studies endeavours to elucidate or establish the characteristics of this relationship. Some research has connected high levels of relatedness between series with greater similarity in their shapes or patterns and stronger cross-correlation [22,23], while other studies have suggested that higher relatedness corresponds to greater similarity in the extracted features of the series being examined [24].

Montero-Manso and Hyndman's recent work [9] is the first to provide insights into this area. Their research demonstrates that it is always possible to find a GFM capable of performing just as well or even better than a set of local statistical benchmarks for any dataset, regardless of its heterogeneity. This implies that GFMs are not inherently more restricted than local models and can perform well even if the series are unrelated. Due to the utilization of more data, global models can be more complex than local ones (without suffering from overfitting) while still achieving better generalization. Montero-Manso and Hyndman suggest that the complexity of global models can be achieved by increasing the memory/order of autoregression, using non-linear/non-parametric methods, and employing data partitioning. The authors provide empirical evidence of their findings through the use of real-world datasets.

Hewamalage et al. [25] aimed to investigate the factors that influence GFM performance by simulating various datasets with controlled characteristics, including the homogeneity/heterogeneity of series, pattern complexity, forecasting model complexity, and series number/length. Their results reveal that relatedness has a strong connection with other factors, including data availability, data complexity, and the complexity of the forecasting approach adopted, when it comes to GFM performance. Furthermore, in challenging forecasting situations, such as those involving short or heterogeneous series and limited prior knowledge of data patterns, GFMs' complex non-linear modelling capabilities make them a competitive option.

Rajapaksha et al. [26] recently introduced a novel local model-agnostic interpretability approach to address the lack of interpretability in GFMs. The approach employs statistical forecasting techniques to explain the global model forecast of a specific time series using interpretable components such as trend, seasonality, coefficients, and other model attributes. This is achieved by defining a locally defined neighbourhood, which can be done through either bootstrapping or model fitting. The authors conducted experiments on various benchmark datasets to evaluate the effectiveness of this framework. They evaluated the results both quantitatively and qualitatively and found that the two approaches proposed in the framework provided comprehensible explanations that accurately approximated the global model forecast.

Nevertheless, the major real-world datasets are, by nature, heterogeneous, including series that are clearly unrelated, such as the M4 forecasting competition, whose dataset is a broad mix of unaligned time series across many different domains [25].

1.3. Model Complexity

Kolmogorov's theory [27] explains the concept of complexity, which can be technically described as follows. We begin by establishing a syntax for expressing all computable functions, which could be an enumeration of all Turing machines or a list of syntactically correct programs in a universal programming language such as Java, Lisp, or C. From there, we defined the Kolmogorov complexity of a finite binary string (every object can be coded as a string over a finite alphabet—say, the binary alphabet) as the length of the shortest Turing machine, Java program, etc., in the chosen syntax. Thus, to each finite string is assigned a positive integer as its Kolmogorov complexity through this syntax. Ultimately, the Kolmogorov complexity of a finite string represents the length of its most-compressed version and the amount of information (in the form of bits) contained within it. Although Kolmogorov complexity is generally believed to be theoretically incomputable [28], recent research by Cilibrasi and Vitanyi [29] has demonstrated that it can be approximated using the decompressor of modern real-world compression techniques. This approximation involves determining the length of a minimum and efficient description of an object that can be produced by a lossless compressor. As a result, to estimate the complexity of our models in this experiment, we rely on the size of their gzip compressions, which are considered very efficient and are widely used. If the output file of a model can be compressed to a very small size, it suggests that the information contained within it is relatively simple and structured and can be easily described using a small amount of information. This would indicate that the model is relatively simple. Conversely, if the output file of a model is difficult to compress, and requires a large amount of storage space, this suggests that the information contained within it is more complex and is structured in a way that cannot be easily reduced. This indicates that the model is more complex. It is worth noting that this approach to measuring algorithmic complexity of models may depend on the data used, but since all models in our experiment are based on the same data, we do not factor the data into the compression.

The number of parameters in a model can also be a useful heuristic for measuring the model's complexity [30]. Each parameter represents a degree of freedom that the model has in order to capture patterns in the data. The more parameters a model has, the more complex its function can be, and the more flexible it is to fit a wide range of training data patterns. Deep learning models differ structurally from traditional machine learning models and have significantly more parameters. These models are consistently over-parametrised, implying that they contain more parameters than the optimal solutions and training samples. Nonetheless, research has demonstrated that extensively over-parametrised neural networks often show strong generalization capabilities. In fact, several studies suggest that larger and more-complex networks generally achieve superior generalization performance [31].

1.4. Key Contributions

Despite all of the aforementioned efforts, there has been a lack of research on how the relatedness between series impacts the effectiveness of GFMs in real-world demand-forecasting problems, especially when dealing with challenging conditions such as the highly lumpy or intermittent data very common in retail. The research conducted in this study was driven precisely by this motivation: to investigate the cross-learning scenarios driven by the product hierarchy commonly employed in retail planning that enable global models to better capture interdependencies across products and regions. We provide the following contributions that help understand the potential and applicability of global models in real-world scenarios:

- Our study investigates possible dataset partitioning scenarios, inspired by the hierarchical aggregation structure of the data, that have the potential to more effectively capture inter-dependencies across regions and products. To achieve this, we utilize a prominent deep learning forecasting model that has demonstrated success in

numerous time-series applications due to its ability to extract features from high-dimensional inputs.

- We evaluate the heterogeneity of the dataset by examining the similarity of the time-series features that we deem crucial for accurate forecasting. Some features, which are deliberately crafted, prove especially valuable for intermittent data.
- In order to gauge the complexity of our models during the experiment, we offer two quantitative indicators: the count of parameters contained within the models and the compressibility of their output files as determined by Kolmogorov complexity.
- A comprehensive evaluation of the forecast accuracy achieved by the global models of the various partitioning approaches and local benchmarks using two error measures is presented. These measures are also used to perform tests on the statistical significance of any reported differences.
- The empirical results we obtained provide modelling guidelines that are easy for both retailers and software suppliers to implement regarding the trade-off between data availability and data relatedness.

The layout of the remainder of this paper is as follows. Section 2 describes our forecasting framework developed for the evaluation of the cross-learning approaches, and Section 3 provides the details about its implementation. Section 4 presents and discusses the results obtained, and Section 5 provides some concluding remarks and promising areas for further research.

2. Forecasting Models

Due to the impressive accomplishments of deep learning in computer vision, its implementation has extended to several areas, including natural language processing and robot control, making it a popular choice in the machine learning domain. Despite being a significant application of machine learning, the progress of using deep learning in time-series forecasting has been relatively slower compared to other areas. Moreover, the lack of a well-defined experimental protocol makes its comparison with other forecasting methods difficult. Given that deep learning has demonstrated superior performance compared to other approaches in multiple domains when trained on large datasets, we were confident that it could be effective in the current context. However, few studies have focused on deep learning approaches for intermittent demand [32]. Forecasting intermittent data involves dealing with sequences that have sporadic values [33]. This is a complex task, as it entails making predictions based on irregular observations over time and a significant number of zero values. We selected DeepAR, which is an autoregressive recurrent neural network (RNN) model that was introduced by Amazon in 2018 [23]. DeepAR is a prominent deep learning forecasting model that has demonstrated success in several time-series applications.

2.1. DeepAR Model

Formally, denoting the value of item i at time t by $z_{i,t}$, the goal of DeepAR is to predict the conditional probability P of future sales $\mathbf{z}_{i,t_0:T}$ based on past sales $\mathbf{z}_{i,1:t_0-1}$ and covariates $\mathbf{x}_{i,1:T}$, where t_0 and T are, respectively, the first and last time points of the future

$$P(\mathbf{z}_{i,t_0:T}|\mathbf{z}_{i,1:t_0-1}, \mathbf{x}_{i,1:T}). \tag{1}$$

Note that the time index t is relative, i.e., $t = 1$ may not correspond to the first time point of the time series. During training, $z_{i,t}$ is available in both time ranges $[1, t_0 - 1]$ and $[t_0, T]$, known respectively as the conditioning range and the prediction range (corresponding to the encoder and decoder in a sequence-to-sequence model), but during inference, $z_{i,t}$ is not available in the prediction range. The network output at time t can be expressed as

$$\mathbf{h}_{i,t} = h(\mathbf{h}_{i,t-1}, z_{i,t-1}, \mathbf{x}_{i,t}; \Theta), \tag{2}$$

where h is a function that is implemented by a multi-layer RNN with long short-term memory (LSTM) cells [34] parameterised by Θ. The model is autoregressive in the sense that it uses the sales value at the previous time step $z_{i,t-1}$ as an input, and recurrent in the sense that the previous network output $\mathbf{h}_{i,t-1}$ is fed back as an input at the next time step. During training, given a batch of N items $\{\mathbf{z}_{i,1:T}\}_{i=1,\ldots,N}$ and corresponding covariates $\{\mathbf{x}_{i,1:T}\}_{i=1,\ldots,N}$, the model parameters are learned by maximizing the log-likelihood of a fixed probability distribution as follows

$$L = \sum_{i=1}^{N} \sum_{t=t_0}^{T} \log l(z_{i,t}|\theta(\mathbf{h}_{i,t})), \tag{3}$$

where θ denotes a linear mapping from the function $\mathbf{h}_{i,t}$ to the distribution's parameters, while l represents the likelihood of the distribution. Since the encoder model is the same as the decoder, DeepAR uses all of time range $[0, T]$ to calculate this loss (i.e., $t_0 = 0$ in Equation (3)). DeepAR is designed to predict a 1-step forwarded value. To forecast multiple future steps in the inference, the model repeatedly generates forecasts for the next period until the end of the forecast horizon. Initially, the model is fed with past sequences ($t < t_0$), and the forecast of the first period is generated by drawing samples from the trained probability distribution. The forecast of the first period is then used as an input to the model for generating the forecast of the second period, and so on for each subsequent period. As the forecast is based on past samples from the predicted distribution, the model's output is probabilistic and not deterministic, and it represents a distribution of sampled sequences. This sampling process is advantageous as it generates a probability distribution of forecasts, which can evaluate the accuracy of the forecasts.

To address the issue of zero-inflated distribution in sales demands, we employed the negative log-likelihood of the Tweedie distribution for the loss function. The Tweedie distribution is a family of probability distributions that is characterized by two parameters: the power parameter, denoted as p, and the dispersion parameter, denoted as ϕ. The probability density function of the Tweedie distribution is defined as:

$$f(y; \mu, \phi, p) = \frac{y^{p-1} \exp\left(\frac{y\mu^{1-p}}{\phi(1-p)}\right)}{\phi(1-p)y^{p}\Gamma\left(\frac{1}{1-p}\right)}, \quad y > 0, \tag{4}$$

where μ is the mean parameter of the distribution, Γ is the gamma function, and p and ϕ are positive parameters. When $1 < p < 2$, the Tweedie distribution is a compound Poisson–gamma distribution, which is commonly used to model data with a large number of zeros and positive skewness. The dispersion parameter ϕ controls the degree of variability or heterogeneity in the data. When ϕ is small, the data are said to be highly variable or dispersed, while a large value of ϕ indicates low variability or homogeneity in the data.

Our implementation of the DeepAR models used the PyTorch AI framework [35] with the `DeepAREstimator` method from the `GluonTS` Python library [36].

2.2. Benchmarks

Benchmarks are used to evaluate the performance of forecasting models by providing a standard against which the models can be compared [37]. By using benchmarks, researchers and practitioners can objectively assess the forecasting accuracy of different models and identify which model performs best for a given forecasting task. Comparing the accuracy of a forecasting model against a benchmark provides a baseline measure of its performance and helps to identify the added value of the model. The two most-commonly utilized models for time-series forecasting are Exponential Smoothing and ARIMA (AutoRegressive Integrated Moving Average). These benchmark models are good references for evaluating the forecasting performance of more-complex models. They provide a baseline for comparison and help to identify whether a more-complex model is justified based on its added accuracy on the benchmark. The seasonal naïve method can be very

effective at capturing the seasonal pattern of a time series and is also frequently adopted as a benchmark to compare against more complex models.

2.2.1. ARIMA Models

The seasonal ARIMA model, denoted as ARIMA$(p,d,q) \times (P,D,Q)_m$, can be written as:

$$\phi_p(B)\Phi_P(B^m)(1-B)^d(1-B^m)^D\eta_t = c + \theta_q(B)\Theta_Q(B^m)\varepsilon_t, \tag{5}$$

$$\phi_p(B) = 1 - \phi_1 B - \cdots - \phi_p B^p, \qquad \Phi_P(B^m) = 1 - \Phi_1 B^m - \cdots - \Phi_P B^{Pm},$$
$$\theta_q(B) = 1 + \theta_1 B + \cdots + \theta_q B^q, \qquad \Theta_Q(B^m) = 1 + \Theta_1 B^m + \cdots + \Theta_Q B^{Qm},$$

where η_t is the target time series, m is the seasonal period, D and d are the degrees of seasonal and ordinary differencing, respectively, B is the backward shift operator, $\phi_p(B)$ and $\theta_q(B)$ are the regular autoregressive and moving average polynomials of orders p and q, respectively, $\Phi_P(B^m)$ and $\Theta_Q(B^m)$ are the seasonal autoregressive and moving-average polynomials of orders P and Q, respectively, $c = \mu(1 - \phi_1 - \cdots - \phi_p)(1 - \Phi_1 - \cdots - \Phi_P)$, where μ is the mean of $(1-B)^d(1-B^m)^D\eta_t$ and ε_t is a white-noise series (i.e., serially uncorrelated with zero mean and constant variance). Stationarity and invertibility conditions imply that the zeros of the polynomials $\phi_p(B)$, $\Phi_P(B^m)$, $\theta_q(B)$, and $\Theta_Q(B^m)$ must all lie outside of the unit circle. Non-stationary time series can be made stationary by applying transformations such as logarithms to stabilise the variance and by taking proper degrees of differencing to stabilise the mean. After specifying values for p, q, P, and Q, the parameters of the model $c, \phi_1 \cdots, \phi_p, \theta_1, \cdots, \theta_q, \Phi_1 . \cdots, \Phi_P, \Theta_1, \cdots, \Theta_Q$ can be estimated by maximising the log likelihood. The Akaike's Information Criteria (AIC), which is based on the log likelihood and on a regularization term (that includes the number of parameters in the model) to compensate for potential overfitting, can be used for determining the values of p, q, P, and Q. To implement the ARIMA models, we used the `AutoARIMA` function from the `StatsForecast` Python library [38], which is a mirror of Hyndman's [39] `auto.arima` function in the `forecast` package of the R programming language.

2.2.2. Exponential Smoothing Models

Exponential smoothing models comprise a measurement (or observation) equation and one or several state equations. The measurement equation describes the relationship between the time series and its states or components, i.e., the level, the trend, and the seasonality. The state equations express how the components evolve over time [7,40]. The components can interact with themselves in an additive (A) or multiplicative (M) manner; an additive damped trend (A_d) or multiplicative damped trend (M_d) is also possible. For each model, an additive or multiplicative error term can be considered. Each component is updated by the error process, which is the amount of change controlled by the smoothing parameter. For more details, the reader is referred to [7] and [41]. The existence of a consistent multiplicative effect on sales led us to use a logarithm transformation and, consequently, to adopt only linear exponential smoothing models. Table 1 presents the equations for these models in the state-space modelling framework: y_t is the time-series observation in period t, l_t is the local level in period t, b_t is the local trend in period t, s_t is the local seasonality in period t, and m is the seasonal frequency; α, β, γ, and ϕ are the smoothing parameters, and ε_t is the error term usually assumed to be normally and independently distributed with mean 0 and variance σ^2, i.e., $\varepsilon_t \sim \text{NID}(0, \sigma^2)$. To implement the exponential smoothing models, we used the `AutoETS` function from the `StatsForecast` Python library [38], which is a mirror of Hyndman's [7] `ets` function in the `forecast` package of the R programming language.

Table 1. Linear exponential smoothing models.

Trend component		Seasonal Component: N	Seasonal Component: A
	N	$y_t = l_{t-1} + \varepsilon_t$ $l_t = l_{t-1} + \alpha\varepsilon_t$	$y_t = l_{t-1} + s_{t-m} + \varepsilon_t$ $l_t = l_{t-1} + \alpha\varepsilon_t$ $s_t = s_{t-m} + \gamma\varepsilon_t$
	A	$y_t = l_{t-1} + b_{t-1} + \varepsilon_t$ $l_t = l_{t-1} + b_{t-1} + \alpha\varepsilon_t$ $b_t = b_{t-1} + \beta\varepsilon_t$	$y_t = l_{t-1} + b_{t-1} + s_{t-m} + \varepsilon_t$ $l_t = l_{t-1} + b_{t-1} + \alpha\varepsilon_t$ $b_t = b_{t-1} + \beta\varepsilon_t$ $s_t = s_{t-m} + \gamma\varepsilon_t$
	$\mathbf{A_d}$	$y_t = l_{t-1} + \phi b_{t-1} + \varepsilon_t$ $l_t = l_{t-1} + \phi b_{t-1} + \alpha\varepsilon_t$ $b_t = \phi b_{t-1} + \beta\varepsilon_t$	$y_t = l_{t-1} + \phi b_{t-1} + s_{t-m} + \varepsilon_t$ $l_t = l_{t-1} + \phi b_{t-1} + \alpha\varepsilon_t$ $b_t = \phi b_{t-1} + \beta\varepsilon_t$ $s_t = s_{t-m} + \gamma\varepsilon_t$

2.2.3. Seasonal Naïve

The Seasonal Naïve model is a simple time-series forecasting model that assumes the future value of a series will be equal to the last observed value from the same season. It can be formulated as follows:

$$\hat{y}_t = y_{t-m}, \tag{6}$$

where $\hat{y}_t$ is the forecasted value of the series at time t, y_{t-m} is the last observed value from the same season (m periods ago), and m is the number of periods in a season (e.g., seven for daily data with weekly seasonality).

3. Empirical Setup

In this section, we present experimental scenarios that use hierarchical aggregation structure-based data partitioning to investigate quantitatively how the relatedness between series impacts the effectiveness of GFMs and their complexity.

3.1. Dataset

To ensure the significance of a study's findings, it is crucial that it can be reproduced and compared with other relevant studies. Therefore, in this study, we used the M5 competition's well-established and openly accessible dataset, which is widely recognized as a benchmark for the development and evaluation of time-series forecasting models. The M5 dataset is a large time-series set consisting of sales data for Walmart stores in the United States. The dataset was released in 2020 as part of the M5 forecasting competition, which was organized by University of Nicosia and sponsored by Kaggle [19]. The M5 dataset includes daily sales data for 3049 products and spans a period of 5 years, from 29 January 2011 to 19 June 2016 (1969 days). The dataset is organized hierarchically, with products being grouped into states, stores, categories, and departments. The 3049 products were sold across ten different stores, which were located in three states of the USA: California (CA), Texas (TX), and Wisconsin (WI). California covers four stores (CA1, CA2, CA3, and CA4), while Texas and Wisconsin represent three stores each (TX1, TX2, TX3 and WI1, WI2, WI3). For every store, the products are classified into three main categories: Household, Hobbies, and Foods. These categories are further divided into specific departments. Specifically, the Household and Hobbies categories are each subdivided into two departments (Household1, Household2 and Hobbies1, Hobbies2), while the Foods category is subdivided into three departments (Foods1, Foods2, and Foods3). The main goal of the M5 competition was to develop accurate sales forecasts for the last 28 days from 23 May 2016 to 19 June 2016. The M5 dataset has become a standard reference due to its challenging properties, including high dimensionality, hierarchical structure, and intermittent demand patterns (i.e., many products have zero sales on some days).

A dataset is commonly regarded as heterogeneous when it comprises time series that exhibit different patterns, such as seasonality, trend, and cycles, and, conceivably, distinct types of information [9]. Therefore, heterogeneity is often associated with unrelatedness [25]. Our examination of the heterogeneity in the M5 dataset and assessment of the relatedness among its time series followed the methodology proposed by [25], which involved comparing the similarity of the time-series features. Similar to Kang et al.'s methodology [42], we applied Principal Component Analysis (PCA) [43] to decrease the feature dimensionality and depicted the similarity of the time-series features using a 2-D plot. Furthermore, we also identified a set of critical features that significantly impact the forecastability of a series, namely:

- Spectral entropy (Entropy) to measure forecastability;
- Strength of trend (Trend) to measure the strength of the trend;
- Strength of seasonality (Seasonality) to measure the strength of the seasonality;
- First-order autocorrelation (ACF1) to measure the first-order autocorrelation;
- Optimal Box–Cox transformation parameter (Box–Cox) to measure the variance stability;
- Ratio between the number of non-zero observations and the total number of observations (Non-zero demand) to measure the proportion of non-zero demand;
- Ratio between the number of changes between zero and non-zero observations and the total number of observations (Changes) to measure the proportion of status changes from zero to non-zero demand.

The R programming language's `feasts` package [44] was used to calculate time-series features using the `features` function. Additionally, we utilized the `PCA` function from the `FactoMineR` package [45] in the R programming language to conduct principal component analyses. Figure 1 shows the 2-D plot of the M5 dataset's time-series features selected after applying principal component analysis. As expected, the time-series features of the M5 dataset show a scattered distribution in the 2-D space, indicating dissimilarity among them. This dissimilarity is an indicator of the dataset's heterogeneity regarding those features, suggesting that we are examining a broad range of series within a single dataset.

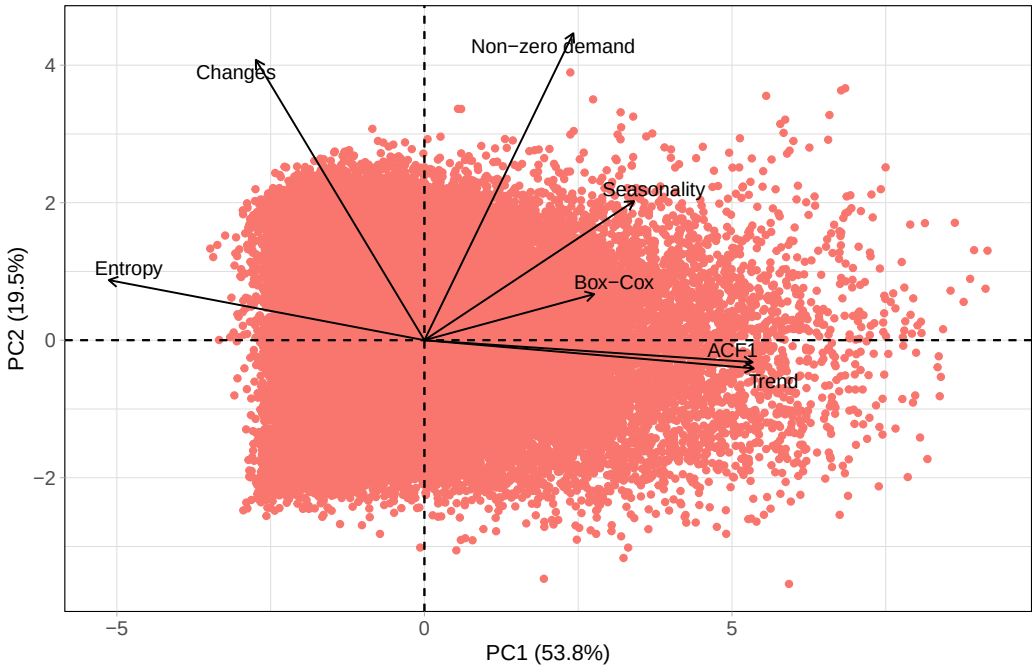

Figure 1. Time-series features of M5 dataset after applying principal component analysis.

3.2. Data Pools

The approach used in the presented framework employs partial pooling and is inspired by the hierarchical structure of Walmart. The multi-level data provided are used to prepare five distinct levels of data, including total, state, store, category, and department, as well as four cross-levels of data, including state–category, state–department, store–category, and store–department. Data pools are then obtained for each level and cross-level. The total pool comprises the entire M5 dataset, which consists of 30,490 time series. At the state level, there are three data pools corresponding to the three states (CA, TX, and WI). CA has 12,196 time series, while TX and WI have 9147 time series each. The store level has ten data pools, including four stores in California (CA1, CA2, CA3, and CA4) and three stores in both Texas and Wisconsin (TX1, TX2, TX3, and WI1, WI2, WI3), each with 3049 time series. The category level has three data pools corresponding to the three distinct categories: Household, Hobbies, and Foods, each with a different number of time series (10,470 for Household, 5650 for Hobbies, and 14,370 for Foods). The department level has seven data pools, consisting of three departments for the Foods category (Foods1, Foods2, and Foods3) and two departments each for the Household and Hobbies categories. The number of time series in each department ranges from 1490 to 8230. The state–category cross-level consists of nine data pools, which result from crossing the three states with the three categories. For instance, CA–Foods contains the products from the Foods category that are available in CA stores. The number of time series in the state–category pools ranges from 1695 to 5748. Similarly, the state–department cross-level comprises 21 data pools that arise from the combination of the three states with the seven departments. For example, CA–Foods3 includes the products from the Foods3 department that are sold in CA stores. The number of time series in the state–department pools varies from 447 to 3292. The store–category cross-level has 30 data pools generated by crossing the ten stores with the three categories. For example, CA3–Foods includes the products from the Foods category that are sold in CA3 store. The number of time series in the store–category pools ranges from 565 to 1437. Lastly, the store–department cross-level has 70 data pools that arise from the combination of the ten stores with the seven departments. For instance, CA3–Foods3 comprises the products from the Foods3 department that are sold in CA3 store. The number of time series in the store–department pools ranges from 149 to 823. All this information is provided in Appendix A.

It is noteworthy that we examined all feasible combinations of partial pools from the multi-level data available. We expect that as the sizes of the data pools decrease and the relatedness of the time series within them increases, the global models' performance will improve, while their complexity will decrease. It is expected that the cross-learning scenarios developed, driven by the product hierarchy employed by the retailer, will result in improved global models that can capture interdependencies among products and regions more effectively. By utilizing data pools at the state and store levels, it may be possible to better understand cross-region dependencies and the impact of demographic, cultural, economic, and weather conditions on demand. Additionally, category and department data pools have the potential to uncover cross-product dependencies and improve the relationships between similar and complementary products. This partitioning method is simpler to implement than the current literature-based clustering methods that rely on feature extraction to identify similarities among the examined series.

3.3. Model Selection

A deepAR model was trained using all the time series available in each data pool, regardless of any potential heterogeneity. For instance, a deepAR model was trained for each state, namely CA, TX, and WI, making a total of three different models for the state level. Similarly, one deepAR model was trained for each store, resulting in ten distinct deepAR models for the store level, and so forth. Moreover, in the case of the total pool, only one deepAR model was trained, using the entire M5 dataset, which consists of 30,490 time series. As a result, a total of 154 separate deepAR models were trained, with each data pool

having one model. Although complete pooling, which involves using a single forecasting model for the entire dataset, can capture interdependencies among products and regions, partial pooling, which uses a separate forecasting model for each pool, is often better suited for capturing the unique characteristics of each group.

We followed the structure of the M5 competition, which kept the last 28 days of each time series as the testing set for out-of-sample evaluation (23 May 2016 to 19 June 2016), while using the remaining data (29 January 2011 to 22 May 2016, 1941 days) for training the models. It is essential to find the appropriate model that can perform well during testing in order to achieve the highest possible level of accuracy. Typically, a validation set is employed to choose the most-suitable model. The effectiveness of a deep learning model largely depends on various factors such as hyperparameters and initial weights. To select the best model, the last 28 days of in-sample training from 25 April 2016 to 22 May 2016 were used for validation. The hyperparameters and their respective ranges that were utilized in model selection are presented in Table 2. The Optuna optimization framework [46] was used to carry out the hyperparameter optimization process by utilizing the Root Mean Squared Error (RMSE) [4] as the accuracy metric for model selection. For both ARIMA and ETS local benchmarks, a model was chosen for each time series using the AICc value, resulting in a total of 30,490 models.

Table 2. DeepAR hyperparameter ranges of values considered in the optimization process.

Hyperparameter	Values Considered
Context length	28
Prediction length	28
Number of hidden layers	$\{1, 2, 3, 4\}$
Hidden size	$\{20, 40, 60, 80, 100, 120, 140\}$
Learning rate	$[1 \times 10^{-5}, 1 \times 10^{-1}]$
Dropout rate	$[0, 0.2]$
Batch size	$\{16, 32, 64, 128\}$
Scaling	True
Number of epochs	100
Number of parallel samples	100
Number of trials	50

3.4. Model Complexity

Data partitioning based on relatedness enhances a dataset's similarities, making it easier for a model to identify complex patterns that are shared across time series, thereby reducing the model's complexity. Therefore, it is essential to have heuristics that can estimate the model's complexity. As discussed in Section 1.3, one way to do this is by counting the number of parameters (NP) in the model of the data pool and measuring the size of the gzip compression (CMS-compressed model size) of its output file, expressed in bytes. Each parameter represents a degree of freedom that the model has to capture patterns in the data. The more parameters a model has, the more complex and flexible it is to fit a wide range of training data patterns. A model's output file can be compressed to a small size if the information contained within it is relatively simple, indicating that the model is simple. Conversely, if the output file is difficult to compress and requires significant storage space, this suggests that the information contained within it is more complex, indicating that the model is more complex. To obtain the total number of parameters (TNP) for each partitioning approach, we added up the number of parameters (NP) in the model for each of its data pools. Similarly, we calculated the total compressed model size (TCMS) in bytes by summing the sizes of the gzip output file of the model for each of its data pools.

Additionally, it should be noted that the complexity of a learned model is affected not only by its architecture but also by factors such as the distribution and complexity of the data, as well as the amount of information available. With this in mind, we also computed

the weighted average number of parameters (WNP) and the weighted average compressed model size (WCMS) per model for each partitioning approach, as shown below.

$$\text{WNP} = \frac{1}{ds}\sum_{i=1}^{n} ps_i \times \text{NP}_i, \tag{7}$$

$$\text{WCMS} = \frac{1}{ds}\sum_{i=1}^{n} ps_i \times \text{CMS}_i, \tag{8}$$

where ds is the dataset size (number of time series), n is the number of data pools of the partitioning approach, and ps_i is the size of the data pool i.

A conservative estimate for the number of parameters in both ARIMA and ETS local benchmark models was considered. For ARIMA, we assumed a maximum of 16 parameters, based on the highest possible orders for the autoregression and moving average polynomials ($p = 5$, $q = 5$, $P = 2$, $Q = 2$) as well as the variance of the residuals. In the case of ETS, we estimated a maximum of 14 parameters per model by taking into account the number of smoothing parameters (α, β, γ, and ϕ), initial states (l_0, b_0, $s_0, \ldots, s_6$), and the variance of the residuals. The TNP for both ARIMA and ETS models was calculated by multiplying the number of separate models (30,490 in total in this case study) by 16 and 14, respectively. As a result, the WNP per model for ARIMA and ETS are 16 and 14, respectively. To obtain the TCMS in bytes for these benchmark models, the sizes of the gzip output file for each individual model were added together. The WCMS per model can be calculated by dividing the TCMS by the number of models.

3.5. Evaluation Metrics

The performance of global and local models was evaluated with respect to two performance measures commonly found in the literature related to forecasting [47], namely the average of the Mean Absolute Scaled Error (MASE) and the average of the Root Mean Squared Scaled Error (RMSSE):

$$\text{MASE}_i = \frac{\frac{1}{h}\sum_{t=n+1}^{n+h} |\text{z}_{i,t} - \hat{\text{z}}_{i,t}|}{\frac{1}{n-1}\sum_{t=2}^{n} |\text{z}_{i,t} - \text{z}_{i,t-1}|}, \tag{9}$$

$$\text{RMSSE}_i = \sqrt{\frac{\frac{1}{h}\sum_{t=n+1}^{n+h} (\text{z}_{i,t} - \hat{\text{z}}_{i,t})^2}{\frac{1}{n-1}\sum_{t=2}^{n} (\text{z}_{i,t} - \text{z}_{i,t-1})^2}}, \tag{10}$$

where $\text{z}_{i,t}$ is the value of item i at time t, $\hat{\text{z}}_{i,t}$ is the corresponding forecast, n is the length of the in-sample training, and h is the forecast horizon (28 days in this case study). RMSSE was employed to measure the accuracy of point forecasts in the M5 competition [18]. MASE and RMSSE are both scale-independent measures that can be used to compare forecasts across multiple products with different scales and units. This is achieved by scaling the forecast errors using the Mean Absolute Error (MAE) or Mean Squared Error (MSE) of the 1-step-ahead in-sample naive forecast errors in order to match the absolute or quadratic loss of the numerator. The use of squared errors favours forecasts that closely follow the mean of the target series, while the use of absolute errors favours forecasts that closely follow the median of the target series, thereby focusing on the structure of the data.

3.6. Statistical Significance of Models' Differences

The MASE and RMSSE errors can be used to conclude if there are any statistically significant differences in the models' performance. First, a Friedman test is performed to determine if at least one model performs significantly differently. Then, the post-hoc Nemenyi test [48] is used to group models based on similar performance. Both of these tests are nonparametric, meaning that the distribution of the performance metric is not a concern. The post-hoc Nemenyi test ranks the performance of models for each time series and calculates the mean of those ranks to produce confidence bounds. If the confidence bounds of different models overlap, then it can be concluded that the models' performances are not statistically different. On the other hand, if the confidence bounds do not intersect, then it can only be determined which method has a higher or lower rank. The `nemenyi()` function in the R package `tsutils` [49] was used to implement these tests, and a significance level of $\alpha = 0.5$ was employed for all tests.

4. Results and Discussion

In this section, a comprehensive examination of the results achieved by the DeepAR global models of the various partitioning approaches and local benchmarks is presented. In addition to evaluating the forecast accuracy using MASE and RMSSE, a comparison of the complexities of the models is also provided. The results of the empirical study are presented in Table 3 and Appendix A. Table 3 includes the percentage difference of each partitioning approach and local benchmark from DeepAR-Total in terms of MASE and RMSSE. This comparison aims to evaluate the enhancement achieved by partial pooling using the hierarchical structure of the data. Furthermore, Appendix A exhibits tables that show the percentage difference of every data-pool model from the most-outstanding one within its aggregation level based on MASE and RMSSE. It is important to note that the results presented in these tables are ranked by MASE in each aggregation level. Table 3 highlights the most-effective data-partitioning approach in boldface within the MASE and RMSE columns. In the field of forecasting, it is common to use forecast averaging as a complementary approach to using multiple models. Numerous studies have demonstrated the effectiveness of averaging the forecasts generated by individual models to enhance the accuracy of forecasts. Based on this idea, we computed the arithmetic mean of forecasts generated by the various partitioning approaches that were developed from the available data pools and denoted this as DeepAR-Comb.

The results presented in Table 3 show that the data-partitioning approaches exhibit significantly better performance than the state-of-the-art local benchmarks. This finding suggests that global models are not inherently more limited than local models and can perform well even on unrelated time-series data. In other words, global models can increase model complexity compared to local models due to their generality. They can afford to be more complex than local models because they generalize better.

Overall, the partitioning approaches outperform DeepAR-Total across all levels of aggregation. DeepAR-State–Department achieves the highest performance according to MASE, while DeepAR-Comb performs best based on RMSSE (which can be explained by the use of RMSE as an accuracy metric for model selection).

Generally, the accuracy of data-partitioning approaches improves as the sizes of the data pools decrease. This can be attributed to the increased similarity among the time series in smaller pools, making it easier to capture cross-product and cross-region dependencies. As a result, models with lower complexity are needed when the data become less heterogeneous. We have observed that both the weighted average number of parameters (WNP) and the weighted average compressed model size (WCMS) decrease accordingly per model. As anticipated, the application of ARIMA and ETS to each time series individually leads to substantially lower WNP and WCMS values than those obtained with global models used in the data-partitioning approaches. The global models tend to be over-parameterised, with a higher number of parameters than training samples. In the case

of WNP, the difference is four orders of magnitude higher, while in the case of WCMS, it is three orders higher.

Table 3. Performance of global and local models evaluated with respect to MASE and RMSSE. Model complexity estimated by TNP (total number of parameters), TCMS (total compressed model size), WNP (weighted average number of parameters), and WCMS (weighted average compressed model size), per model.

Forecasting Methods	No. of Pools	MASE		RMSSE		TNP	WNP	TCMS (Bytes)	WCMS (Bytes)
Partitioning approaches									
DeepAR-Total	1	0.572	—	0.78245	—	204,603	204,603	776,553	776,553
DeepAR-State	3	0.564	−1.38%	0.78060	−0.24%	580,729	203,571	2,178,371	763,894
DeepAR-Store	10	0.560	−1.98%	0.78241	−0.01%	2,121,070	212,107	7,921,985	792,199
DeepAR-Category	3	0.566	−1.08%	0.78094	−0.19%	678,089	296,591	2,595,707	1,136,317
DeepAR-Department	7	0.559	−2.15%	0.78138	−0.14%	1,294,621	226,679	4,821,767	843,542
DeepAR-State-Category	9	0.553	−3.23%	0.78080	−0.21%	1,340,627	168,267	5,015,850	629,156
DeepAR-State-Department	21	**0.551**	**−3.58%**	0.78064	−0.23%	2,419,623	131,080	9,042,778	490,142
DeepAR-Store-Category	30	0.556	−2.74%	0.78340	0.12%	3,708,210	134,902	13,867,558	504,447
DeepAR-Store-Department	70	0.554	−3.11%	0.78421	0.23%	10,310,650	155,903	38,339,800	579,556
DeepAR-Comb	154	0.558	−2.37%	**0.77620**	**−0.80%**	22,658,222	192,634	83,740,297	723,979
Local benchmarks									
ARIMA	1	0.798	39.51%	0.93436	19.42%	487,840 *	16 *	24,431,329	801
ETS	1	0.808	41.24%	0.92853	18.67%	426,860 *	14 *	24,411,454	801
Seasonal Naïve	1	0.905	58.35%	1.23763	58.17%	—	—	—	—

* Conservative estimate of the number of parameters in the models. In ARIMA, they include the orders $0 \leq p \leq 5$, $0 \leq q \leq 5$, $0 \leq P \leq 2$, and $0 \leq Q \leq 2$; c, if it exists, and the residual's variance; thus, a maximum of 16 parameters. In ETS models, they include the smoothing parameters α, β, γ, and ϕ, the initial states l_0, b_0, $s_0, \ldots, s_6$, and the residual's variance; thus, a maximum of 14 parameters. The most-effective data-partitioning approaches within the MASE and RMSE columns are highlighted in boldface.

We have observed that the performance gain of the partitioning approaches over DeepAR-Total is not significant, with an improvement of less than 1% based on RMSSE and up to 3.6% based on MASE. It is noteworthy that the DeepAR-State-Department approach, which uses only 21 data pools, outperforms the other approaches with a higher number of data pools (namely 30 and 70). This suggests that there is a trade-off between data availability and relatedness, where data partitioning can improve the relatedness and similarities between time series by increasing homogeneity. This allows for more-effective capture of the distinct characteristics of the set but at the cost of a reduced sample size, which has been proven to be harmful. Therefore, the primary goal should be to optimize this trade-off. Notably, in addition to achieving the highest forecasting accuracy, the DeepAR-State-Department approach exhibits the lowest weighted average number of parameters (WNP) and weighted average compressed model size (WCMS) per model.

By referring to Appendix A, it can be observed that the DeepAR models associated with the Foods category or Foods1, Foods2, and Foods3 departments generally outperform the models of other categories/departments (see Figures A3–A8). This could be attributed to the higher proportion of non-zero demand (ratio between the number of non-zero observations and the total number of observations) in these data pools. However, it is not possible to establish a direct relationship between data homogeneity and model accuracy due to the different sizes of the data pools (with the exception of the Store level) and the different time series included in each data pool at each aggregation level.

Figure 2 presents the mean rank of the global and local models and the post-hoc Nemenyi test results at a 5% significance level for MASE and RMSSE errors, enabling a more-effective comparison of their performance. The forecasts are arranged by their mean rank, with their respective ranks provided alongside their names. The top-performing

forecasts are located at the bottom of the plot. The variation in the ranks between Table 3 and Figure 2 can be explained by the distribution of the forecast errors. The mean rank is non-parametric, making it robust to outlying errors.

Once again, we have observed that global models outperform local benchmarks. Based on the MASE errors, there is no significant difference between ARIMA and ETS. In addition, DeepAR-State is grouped together with DeepAR-Store-Category and DeepAR-Store, while DeepAR-Comb does not differ from DeepAR-Store-Department and DeepAR-State-Category. The DeepAR-State-Department approach is ranked first and exhibits significant statistical differences from all other approaches. In a similar manner, there is evidence of significant differences among the other four models (DeepAR-Department, DeepAR-Category, DeepAR-Total, and Seasonal Naïve). With regard to the RMSSE, there is no evidence of statistically significant differences between DeepAR-Department, DeepAR-Total, DeepAR-Store, DeepAR-State-Category, DeepAR-Store-Category, and DeepAR-Store-Department. Likewise, DeepAR-State-Department is grouped together with DeepAR-Category and DeepAR-State, ranking on top. The remaining four models exhibit significant differences.

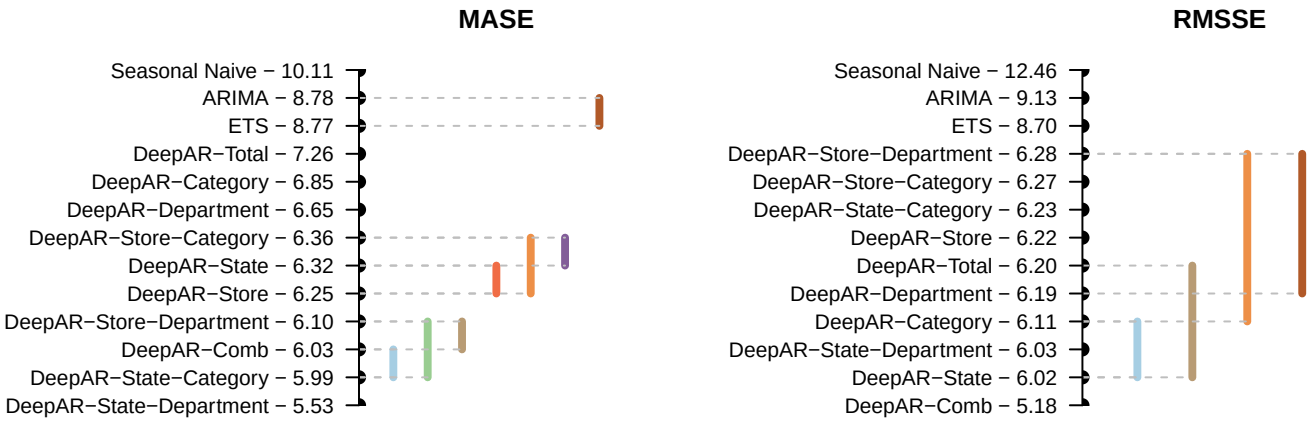

Figure 2. Post-hoc Nemenyi test results at a 5% significance level based on MASE and RMSSE.

5. Conclusions

Retailers typically provide a wide range of merchandise, spanning from perishable products such as fresh produce to non-perishable items such as electronics and clothing. Each of these products exhibits unique demand patterns that can differ based on several factors, including location, time, day of the week, season, and promotional events. Forecasting sales for each product can be a daunting and complex undertaking, particularly given that retailers often sell through multiple channels, including physical stores, online platforms, mobile apps, and marketplaces. Furthermore, in the retail industry, demand forecasting is a routine task that is frequently conducted on a weekly or daily basis to maintain optimal inventory levels. Consequently, advanced models and techniques are required to address the forecasting challenge. These models must be automated to minimize manual intervention, robust enough to handle various data types and scenarios, and scalable to handle vast amounts of data and changing business conditions.

GFMs have shown superior performance to local state-of-the-art benchmarks in prestigious forecasting competitions such as the M4 and M5, as well as those on Kaggle with a forecasting purpose. The success of GFMs is based on the assumption that they are effective if there is a relationship between the time series in the dataset, but there are no established guidelines in the literature to define the characteristics of this relationship. Some studies suggest that higher relatedness between series corresponds to greater similarity in the extracted features, while others connect high relatedness with stronger cross-correlation and similarity in shapes or patterns.

To understand how relatedness impacts GFMs' effectiveness in real-world demand forecasting, especially in challenging conditions such as highly lumpy or intermittent data,

we conducted an extensive empirical study using the M5 competition dataset. We explored cross-learning scenarios driven by the product hierarchy, common in retail planning, to allow global models to capture interdependencies across products and regions more effectively.

Our findings demonstrate that global models outperform state-of-the-art local benchmarks by a significant margin, indicating their effectiveness even with unrelated time-series data. We also conclude that data-partitioning-approach accuracy improves as the sizes of data pools and model complexity decrease. However, there is a trade-off between data availability and data relatedness. Smaller data pools increase the similarity among time series, making it easier to capture cross-product and cross-region dependencies but at the cost of reduced information, which is not always beneficial.

Lastly, it is worth noting that the successful implementation of GFMs for heterogeneous datasets will significantly impact forecasting practice in the near future. It would be intriguing for future research to investigate additional deep learning models and assess their forecasting performance in comparison to the deepAR model.

Author Contributions: Conceptualization, J.M.O. and P.R.; methodology, J.M.O. and P.R.; software, J.M.O. and P.R.; validation, J.M.O. and P.R.; formal analysis, J.M.O. and P.R.; investigation, J.M.O. and P.R.; resources, J.M.O. and P.R.; data curation, J.M.O. and P.R.; writing—original draft preparation, J.M.O. and P.R.; writing—review and editing, J.M.O. and P.R.; visualization, J.M.O. and P.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Publicly available datasets were analysed in this study. These data can be found here: https://www.kaggle.com/competitions/m5-forecasting-accuracy/data (accessed on 12 December 2022).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. Performance of data-pool models evaluated with respect to MASE and RMSSE. Model complexity estimated by NP (number of parameters) and CMS (compressed model size).

Aggregation Level	Data Pool	No. of Time Series	MASE		RMSSE		Model Complexity NP	Model Complexity CMS (Bytes)
Total (1)		30,490	0.572	—	0.7824	—	204,603	776,553
State (3)	CA	12,196	0.545	—	0.7945	5.88%	293,523	1,103,829
	TX	9147	0.566	3.78%	0.7504	—	82,603	310,970
	WI	9147	0.588	7.81%	0.7923	5.59%	204,603	763,572
Store (10)	CA3	3049	0.442	—	0.7715	7.14%	398,443	1,485,258
	TX2	3049	0.484	9.50%	0.7201	—	45,483	172,918
	CA1	3049	0.487	10.21%	0.7585	5.33%	409,683	1,526,328
	CA2	3049	0.542	22.75%	0.8560	18.87%	204,603	764,226
	WI1	3049	0.567	28.34%	0.7949	10.38%	131,683	493,603
	WI3	3049	0.594	34.39%	0.7698	6.89%	398,443	1,484,911
	TX1	3049	0.595	34.76%	0.7537	4.66%	28,003	107,798
	TX3	3049	0.608	37.59%	0.7805	8.39%	33,843	129,442
	WI2	3049	0.613	38.73%	0.8221	14.16%	177,363	660,158
	CA4	3049	0.673	52.42%	0.7969	10.66%	293,523	1,097,343
Category (3)	Foods	14,370	0.431	—	0.7910	4.80%	556,363	2,135,537
	Household	10,470	0.670	55.62%	0.7812	3.50%	74,763	281,886
	Hobbies	5650	0.715	65.97%	0.7548	—	46,963	178,284
Department (7)	Foods3	8230	0.404	—	0.8006	7.15%	285,403	1,063,266
	Foods1	2160	0.435	7.46%	0.7924	6.05%	8923	36,872
	Foods2	3980	0.477	17.86%	0.7815	4.59%	177,363	662,457
	Household1	5320	0.496	22.55%	0.7742	3.61%	285,403	1,064,539
	Hobbies1	4160	0.655	61.97%	0.7472	—	177,363	663,062
	Household2	5150	0.832	105.76%	0.7858	5.16%	285,403	1,051,087
	Hobbies2	1490	0.837	106.99%	0.7645	2.32%	74,763	280,484

Table A1. *Cont.*

Aggregation Level	Data Pool	No. of Time Series	MASE		RMSSE		Model Complexity NP	Model Complexity CMS (Bytes)
State-Category (9)	CA-Foods	5748	0.397	—	0.8067	8.97%	45,483	172,929
	TX-Foods	4311	0.429	7.98%	0.7508	1.42%	183,523	685,556
	WI-Foods	4311	0.433	8.95%	0.8118	9.66%	556,363	2,071,918
	TX-Household	3141	0.630	58.72%	0.7487	1.13%	8923	36,834
	CA-Household	4188	0.639	61.02%	0.7951	7.40%	46,963	178,210
	CA-Hobbies	2260	0.677	70.45%	0.7597	2.62%	28,003	107,975
	TX-Hobbies	1695	0.681	71.57%	0.7498	1.28%	16,203	63,598
	WI-Hobbies	1695	0.708	78.38%	0.7403	—	45,483	171,773
	WI-Household	3141	0.742	86.80%	0.7988	7.90%	409,683	1,527,057
State-Department (21)	CA-Foods3	3292	0.371	—	0.8159	13.73%	131,683	493,415
	WI-Foods3	2469	0.413	11.21%	0.8163	13.78%	240,523	895,494
	TX-Foods3	2469	0.416	12.07%	0.7505	4.62%	293,523	1,095,219
	TX-Household1	1596	0.430	16.02%	0.7174	—	61,203	232,020
	TX-Foods1	648	0.431	16.04%	0.7669	6.90%	61,203	228,161
	CA-Foods1	864	0.438	18.17%	0.8305	15.77%	33,843	128,967
	WI-Foods2	1194	0.442	19.13%	0.8012	11.69%	131,683	491,591
	CA-Foods2	1592	0.477	28.53%	0.7814	8.92%	123,803	464,170
	WI-Foods1	648	0.483	30.05%	0.7993	11.42%	46,963	177,586
	TX-Foods2	1194	0.487	31.22%	0.7438	3.69%	82,603	308,889
	CA-Household1	2128	0.514	38.58%	0.7971	11.11%	79,843	300,371
	WI-Household1	1596	0.522	40.80%	0.8058	12.32%	293,523	1,093,001
	TX-Hobbies1	1248	0.624	68.23%	0.7454	3.90%	240,523	897,953
	CA-Hobbies1	1664	0.625	68.42%	0.7532	4.99%	74,763	281,545
	WI-Hobbies1	1248	0.688	85.39%	0.7481	4.28%	123,803	457,077
	CA-Household2	2060	0.721	94.25%	0.7895	10.05%	5563	24,078
	WI-Hobbies2	447	0.780	110.19%	0.7558	5.36%	28,003	105,157
	TX-Hobbies2	447	0.787	112.00%	0.7507	4.65%	183,523	674,968
	TX-Household2	1545	0.823	121.85%	0.7749	8.02%	46,963	178,373
	CA-Hobbies2	596	0.827	122.87%	0.7820	9.01%	12,283	49,220
	WI-Household2	1545	0.946	154.98%	0.7888	9.95%	123,803	465,523
Store-Category (30)	CA3-Foods	1437	0.306	—	0.7503	5.25%	82,603	311,464
	CA1-Foods	1437	0.356	16.09%	0.7648	7.28%	398,443	1,485,242
	TX2-Foods	1437	0.394	28.48%	0.7244	1.61%	82,603	310,872
	WI2-Foods	1437	0.410	33.94%	0.8221	15.32%	240,523	894,731
	WI1-Foods	1437	0.432	41.13%	0.8241	15.60%	183,523	686,270
	WI3-Foods	1437	0.441	43.86%	0.7875	10.47%	177,363	663,389
	TX1-Foods	1437	0.465	51.94%	0.7393	3.70%	123,803	463,983
	CA2-Foods	1437	0.468	52.91%	0.9308	30.56%	177,363	662,127
	TX3-Foods	1437	0.472	54.08%	0.7949	11.51%	28,003	107,712
	CA4-Foods	1437	0.518	69.28%	0.8060	13.06%	79,843	301,432
	CA3-Household	1047	0.527	72.19%	0.8099	13.61%	8923	36,906
	TX2-Household	1047	0.536	75.11%	0.7129	—	7603	31,610
	CA1-Household	1047	0.570	86.04%	0.7545	5.84%	204,603	763,936
	CA2-Household	1047	0.580	89.37%	0.8127	13.99%	74,763	281,706
	TX2-Hobbies	565	0.582	89.89%	0.7273	2.01%	2203	11,317
	CA3-Hobbies	565	0.591	92.99%	0.7595	6.54%	5563	23,999
	CA1-Hobbies	565	0.607	98.32%	0.7421	4.09%	240,523	896,050
	WI1-Hobbies	565	0.626	104.50%	0.7201	1.02%	7603	31,629
	TX1-Household	1047	0.650	112.07%	0.7657	7.40%	79,843	300,570
	CA2-Hobbies	565	0.656	114.04%	0.7572	6.22%	45,483	172,569
	TX3-Household	1047	0.687	124.35%	0.7710	8.16%	177,363	660,928
	WI3-Hobbies	565	0.697	127.57%	0.7363	3.29%	177,363	658,671
	WI1-Household	1047	0.708	131.05%	0.7951	11.54%	398,443	1,483,148
	TX3-Hobbies	565	0.727	137.51%	0.7692	7.90%	74,763	279,746
	WI2-Hobbies	565	0.765	149.71%	0.7637	7.12%	123,803	459,177
	TX1-Hobbies	565	0.784	155.86%	0.7625	6.96%	43,003	163,129
	WI3-Household	1047	0.786	156.67%	0.7902	10.84%	5563	23,963
	WI2-Household	1047	0.795	159.65%	0.8440	18.40%	204,603	757,647
	CA4-Household	1047	0.796	159.78%	0.7940	11.38%	177,363	663,235
	CA4-Hobbies	565	0.840	174.22%	0.7863	10.30%	74,763	280,400

Table A1. *Cont.*

Aggregation Level	Data Pool	No. of Time Series	MASE		RMSSE		Model Complexity NP	Model Complexity CMS (Bytes)
Store-Department (70)	CA3-Foods3	823	0.279	—	0.7704	12.10%	45,483	172,555
	CA1-Foods3	823	0.313	12.15%	0.7770	13.07%	79,843	300,564
	CA2-Foods1	216	0.349	25.33%	0.8540	24.28%	5,563	23,940
	TX2-Foods3	823	0.353	26.53%	0.7329	6.65%	28,003	107,549
	CA3-Foods2	398	0.364	30.53%	0.7100	3.32%	398,443	1,484,856
	TX2-Foods1	216	0.373	33.83%	0.6884	0.17%	16,203	62,830
	TX2-Household1	532	0.385	38.08%	0.6872	—	285,403	1,056,979
	CA1-Foods1	216	0.389	39.41%	0.7857	14.33%	177,363	654,331
	WI2-Foods2	398	0.389	39.49%	0.8219	19.60%	293,523	1,088,478
	WI2-Foods3	823	0.397	42.40%	0.8250	20.06%	20,723	80,547
	WI1-Foods3	823	0.398	42.65%	0.8334	21.28%	74,763	281,499
	CA3-Foods1	216	0.411	47.28%	0.8176	18.98%	12,283	49,156
	WI3-Foods3	823	0.419	50.46%	0.7887	14.77%	74,763	281,689
	TX1-Foods3	823	0.423	51.73%	0.7380	7.39%	556,363	2,049,755
	CA1-Foods2	398	0.429	53.99%	0.7502	9.17%	293,523	1,094,123
	CA2-Foods3	823	0.440	57.69%	0.9460	37.66%	123,803	463,412
	CA4-Foods3	823	0.440	57.96%	0.7793	13.41%	28,003	107,727
	CA1-Household1	532	0.444	59.14%	0.7654	11.38%	74,763	281,499
	TX1-Foods2	398	0.448	60.77%	0.6969	1.42%	183,523	686,853
	TX2-Foods2	398	0.450	61.38%	0.7342	6.84%	123,803	460,022
	WI3-Foods2	398	0.455	63.39%	0.7636	11.11%	398,443	1,473,384
	TX1-Household1	532	0.457	64.12%	0.7341	6.83%	204,603	753,716
	WI1-Foods1	216	0.458	64.35%	0.8476	23.34%	5,563	23,949
	TX3-Foods3	823	0.461	65.54%	0.8044	17.06%	82,603	310,350
	CA3-Household1	532	0.462	65.76%	0.8322	21.11%	79,843	300,324
	WI2-Foods1	216	0.467	67.63%	0.7934	15.46%	131,683	484,866
	TX3-Household1	532	0.468	67.83%	0.7464	8.61%	46,963	178,155
	CA2-Household1	532	0.472	69.24%	0.8001	16.43%	131,683	493,217
Store-Department (70)	TX1-Foods1	216	0.473	69.64%	0.8169	18.87%	2203	11,276
	WI1-Foods2	398	0.484	73.55%	0.8310	20.92%	79,843	295,408
	WI3-Household1	532	0.497	78.40%	0.7590	10.44%	123,803	463,353
	CA3-Hobbies1	416	0.517	85.63%	0.7440	8.27%	177,363	657,200
	WI2-Household1	532	0.525	88.37%	0.8555	24.49%	123,803	463,933
	WI1-Household1	532	0.531	90.34%	0.8069	17.42%	409,683	1,526,577
	CA2-Foods2	398	0.541	94.17%	0.8720	26.89%	177,363	661,534
	CA1-Hobbies1	416	0.561	101.24%	0.7425	8.05%	285,403	1,060,472
	TX2-Hobbies1	416	0.563	102.04%	0.7317	6.48%	123,803	463,582
	CA4-Foods2	398	0.578	107.44%	0.8096	17.81%	204,603	764,210
	WI3-Foods1	216	0.580	108.05%	0.8108	17.98%	131,683	489,188
	WI1-Hobbies1	416	0.582	108.81%	0.7121	3.62%	177,363	654,673
	CA3-Household2	515	0.603	116.32%	0.7873	14.57%	8923	36,765
	TX3-Foods1	216	0.608	117.95%	0.8813	28.25%	7603	31,574
	TX3-Foods2	398	0.631	126.41%	0.8540	24.27%	204,603	760,157
	CA4-Household1	532	0.641	130.07%	0.7840	14.09%	556,363	2,051,567
	CA4-Foods1	216	0.642	130.47%	0.8920	29.80%	74,763	281,538
	WI3-Hobbies1	416	0.654	134.46%	0.7133	3.79%	177,363	656,325
	TX2-Hobbies2	149	0.657	135.81%	0.7151	4.07%	82,603	307,310
	CA2-Hobbies1	416	0.661	137.24%	0.7674	11.67%	293,523	1,089,744
	CA2-Household2	515	0.688	146.99%	0.8254	20.11%	28,003	106,699
	TX1-Hobbies1	416	0.689	147.29%	0.7367	7.21%	45,483	172,532
	TX2-Household2	515	0.699	150.63%	0.7345	6.88%	293,523	1,074,703
	CA1-Household2	515	0.705	152.98%	0.7429	8.11%	398,443	1,469,196
	CA2-Hobbies2	149	0.705	153.09%	0.7310	6.38%	20,723	80,526
	CA3-Hobbies2	149	0.738	164.93%	0.7762	12.96%	43,003	163,390
	WI2-Hobbies1	416	0.740	165.62%	0.7770	13.07%	123,803	463,481
	CA4-Hobbies1	416	0.754	170.61%	0.7634	11.09%	556,363	2,053,276
	TX3-Hobbies2	149	0.757	171.68%	0.7266	5.73%	293,523	1,088,079
	WI2-Hobbies2	149	0.767	175.25%	0.7197	4.73%	104,043	389,680
	WI1-Hobbies2	149	0.769	175.77%	0.7534	9.64%	46,963	177,610
	TX3-Hobbies1	416	0.790	183.56%	0.8099	17.85%	61,203	229,601
	CA1-Hobbies2	149	0.806	189.04%	0.7656	11.41%	79,843	296,575
	WI3-Hobbies2	149	0.838	200.56%	0.8004	16.48%	12,283	49,227
	WI1-Household2	515	0.852	205.65%	0.7866	14.46%	61,203	231,415
	TX1-Household2	515	0.870	212.20%	0.8052	17.18%	177,363	659,714
	TX3-Household2	515	0.889	218.87%	0.7906	15.05%	183,523	676,504

Table A1. *Cont.*

Aggregation Level	Data Pool	No. of Time Series	MASE		RMSSE		Model Complexity NP	Model Complexity CMS (Bytes)
	WI3-Household2	515	0.900	222.93%	0.7729	12.47%	104,043	386,689
	TX1-Hobbies2	149	0.950	240.72%	0.8169	18.88%	28,003	106,599
	CA4-Household2	515	0.983	252.58%	0.8069	17.42%	45,483	172,371
	CA4-Hobbies2	149	1.006	260.81%	0.8395	22.16%	79,843	299,234
	WI2-Household2	515	1.047	275.49%	0.8084	17.64%	123,803	459,988

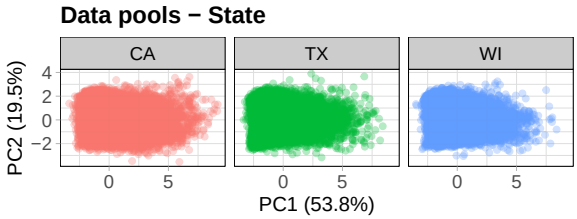

Figure A1. Time-series features of state data pools after applying principal component analysis, ordered by MASE.

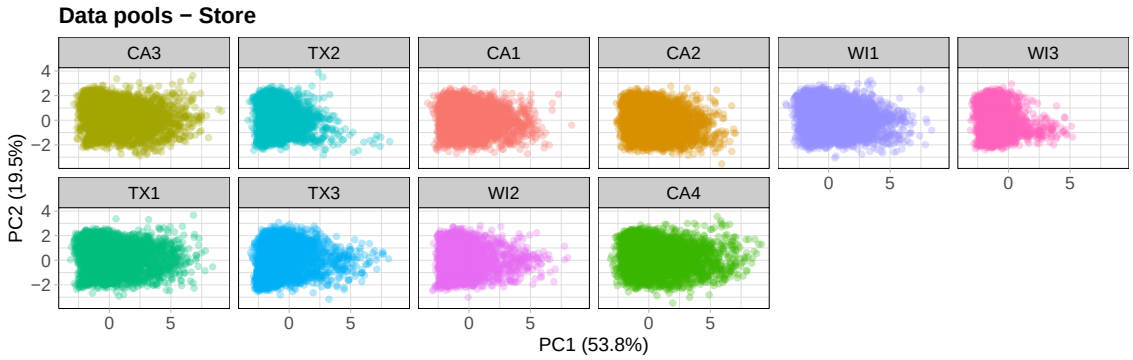

Figure A2. Time-series features of store data pools after applying principal component analysis, ordered by MASE.

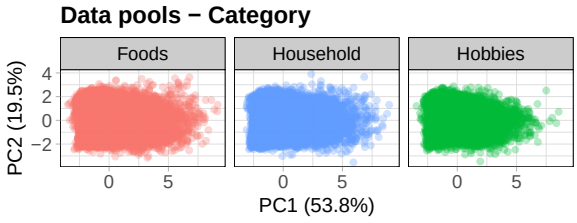

Figure A3. Time-series features of category data pools after applying principal component analysis, ordered by MASE.

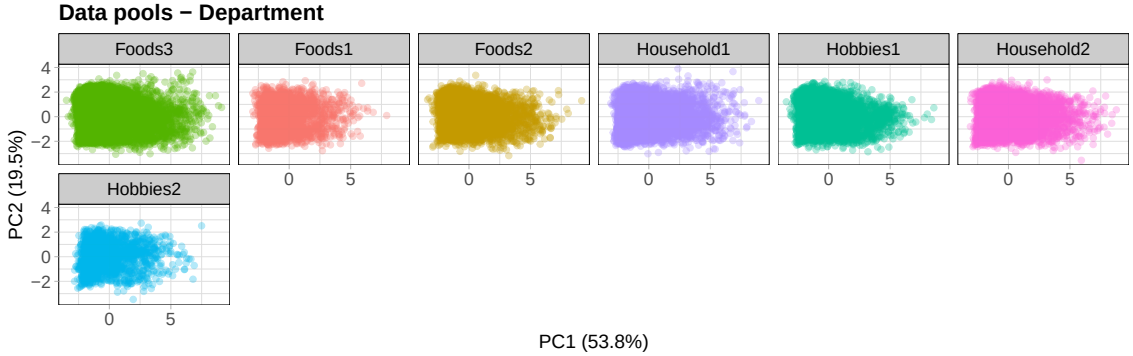

Figure A4. Time-series features of department data pools after applying principal component analysis, ordered by MASE.

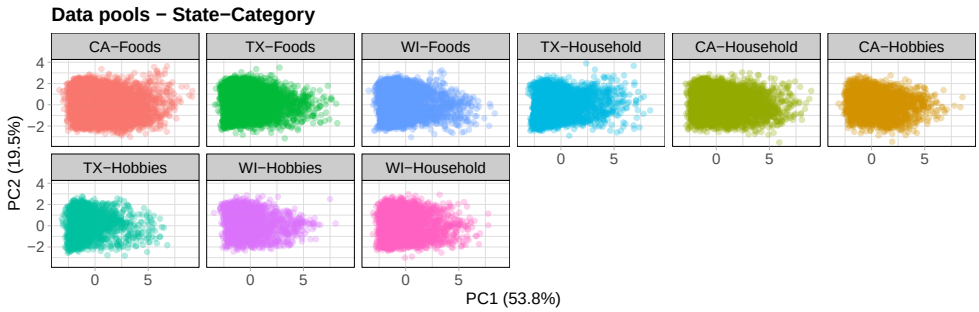

Figure A5. Time-series features of state–category data pools after applying principal component analysis, ordered by MASE.

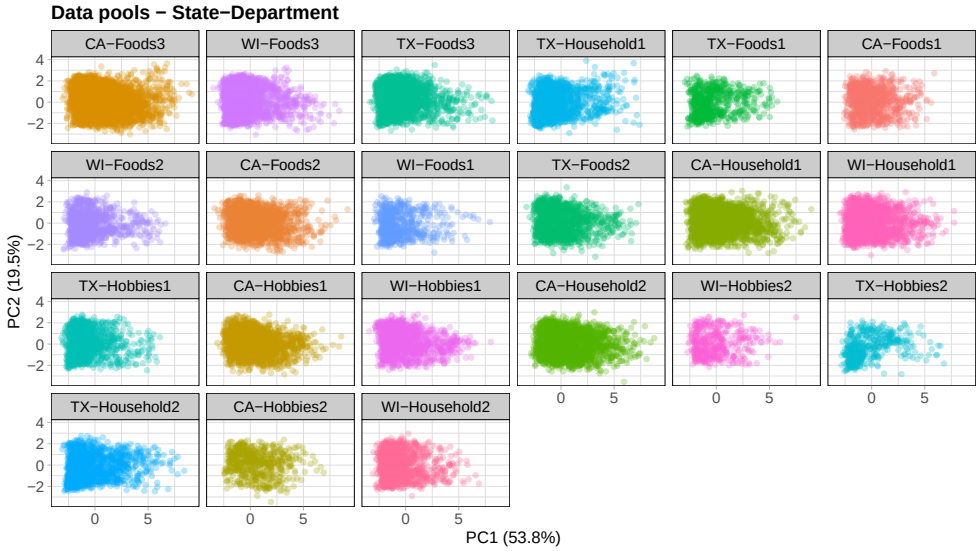

Figure A6. Time-series features of state–department data pools after applying principal component analysis, ordered by MASE.

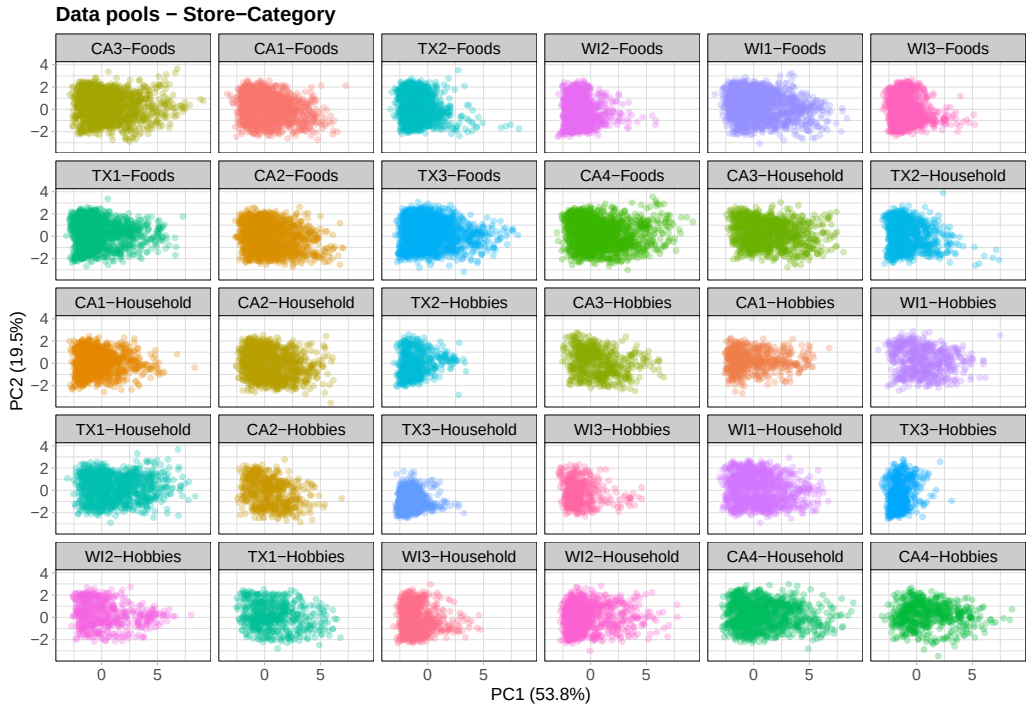

Figure A7. Time-series features of store–category data pools after applying principal component analysis, ordered by MASE.

Figure A8. Time-series features of store–department data pools after applying principal component analysis, ordered by MASE.

References

1. Fildes, R.; Ma, S.; Kolassa, S. Retail forecasting: Research and practice. *Int. J. Forecast.* **2022**, *38*, 1283–1318. [CrossRef]
2. Oliveira, J.M.; Ramos, P. Assessing the Performance of Hierarchical Forecasting Methods on the Retail Sector. *Entropy* **2019**, *21*, 436. [CrossRef] [PubMed]
3. Seaman, B. Considerations of a retail forecasting practitioner. *Int. J. Forecast.* **2018**, *34*, 822–829. [CrossRef]
4. Ramos, P.; Oliveira, J.M.; Kourentzes, N.; Fildes, R. Forecasting Seasonal Sales with Many Drivers: Shrinkage or Dimensionality Reduction? *Appl. Syst. Innov.* **2023**, *6*, 3. [CrossRef]
5. Ramos, P.; Santos, N.; Rebelo, R. Performance of state space and ARIMA models for consumer retail sales forecasting. *Robot. Comput. Integr. Manuf.* **2015**, *34*, 151–163. [CrossRef]
6. Ramos, P.; Oliveira, J.M. A procedure for identification of appropriate state space and ARIMA models based on time-series cross-validation. *Algorithms* **2016**, *9*, 76. [CrossRef]
7. Hyndman, R.J.; Koehler, A.B.; Ord, J.K.; Snyder, R.D. *Forecasting with Exponential Smoothing: The State Space Approach*; Springer Series in Statistics; Springer: Berlin, Germany, 2008. [CrossRef]
8. Box, G.E.P.; Jenkins, G.M.; Reinsel, G.C. *Time Series Analysis*, 4th ed.; Wiley: Hoboken, NJ, USA, 2008.
9. Montero-Manso, P.; Hyndman, R.J. Principles and algorithms for forecasting groups of time series: Locality and globality. *Int. J. Forecast.* **2021**, *37*, 1632–1653. [CrossRef]
10. Januschowski, T.; Gasthaus, J.; Wang, Y.; Salinas, D.; Flunkert, V.; Bohlke-Schneider, M.; Callot, L. Criteria for classifying forecasting methods. *Int. J. Forecast.* **2020**, *36*, 167–177. [CrossRef]
11. Rabanser, S.; Januschowski, T.; Flunkert, V.; Salinas, D.; Gasthaus, J. The Effectiveness of Discretization in Forecasting: An Empirical Study on Neural Time Series Models. *arXiv* **2020**, arXiv:2005.10111.
12. Laptev, N.; Yosinski, J.; Li, L.E.; Smyl, S. Time-series extreme event forecasting with neural networks at Uber. In Proceedings of the International Conference on Machine Learning, Workshop, Sydney, Australia, 6–11 August 2017; Volume 34, pp. 1–5.
13. Gasthaus, J.; Benidis, K.; Wang, Y.; Rangapuram, S.S.; Salinas, D.; Flunkert, V.; Januschowski, T. Probabilistic Forecasting with Spline Quantile Function RNNs. In Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics, Naha, Japan, 16–18 April 2019; Chaudhuri, K.; Sugiyama, M., Eds.; Volume 89, pp. 1901–1910.
14. Oreshkin, B.N.; Carpov, D.; Chapados, N.; Bengio, Y. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *arXiv* **2020**, arXiv:1905.10437.
15. Bandara, K.; Hewamalage, H.; Liu, Y.H.; Kang, Y.; Bergmeir, C. Improving the accuracy of global forecasting models using time series data augmentation. *Pattern Recognit.* **2021**, *120*, 108148. [CrossRef]
16. Makridakis, S.; Spiliotis, E.; Assimakopoulos, V. The M4 Competition: 100,000 time series and 61 forecasting methods. *Int. J. Forecast.* **2020**, *36*, 54–74. [CrossRef]
17. Smyl, S. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *Int. J. Forecast.* **2020**, *36*, 75–85. [CrossRef]
18. Makridakis, S.; Spiliotis, E.; Assimakopoulos, V. The M5 competition: Background, organization, and implementation. *Int. J. Forecast.* **2022**, *38*, 1325–1336. [CrossRef]
19. Makridakis, S.; Spiliotis, E.; Assimakopoulos, V. M5 accuracy competition: Results, findings, and conclusions. *Int. J. Forecast.* **2022**, *38*, 1346–1364. [CrossRef]
20. Makridakis, S.; Spiliotis, E.; Assimakopoulos, V.; Chen, Z.; Gaba, A.; Tsetlin, I.; Winkler, R.L. The M5 uncertainty competition: Results, findings and conclusions. *Int. J. Forecast.* **2022**, *38*, 1365–1385. [CrossRef]
21. Bojer, C.S.; Meldgaard, J.P. Kaggle forecasting competitions: An overlooked learning opportunity. *Int. J. Forecast.* **2021**, *37*, 587–603. . [CrossRef]
22. Duncan, G.T.; Gorr, W.L.; Szczypula, J. Forecasting Analogous Time Series. In *Principles of Forecasting: A Handbook for Researchers and Practitioners*; Armstrong, J.S., Ed.; Springer: Boston, MA, USA 2001; pp. 195–213. [CrossRef]
23. Salinas, D.; Flunkert, V.; Gasthaus, J.; Januschowski, T. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *Int. J. Forecast.* **2020**, *36*, 1181–1191. [CrossRef]
24. Bandara, K.; Bergmeir, C.; Smyl, S. Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. *Expert Syst. Appl.* **2020**, *140*, 112896. [CrossRef]
25. Hewamalage, H.; Bergmeir, C.; Bandara, K. Global models for time series forecasting: A Simulation study. *Pattern Recognit.* **2022**, *124*, 108441. [CrossRef]
26. Rajapaksha, D.; Bergmeir, C.; Hyndman, R.J. LoMEF: A framework to produce local explanations for global model time series forecasts. *Int. J. Forecast.* **2022**. [CrossRef]
27. Kolmogorov, A.N. Three approaches to the quantitative definition of information. *Int. J. Comput. Math.* **1968**, *2*, 157–168.
28. Li, M.; Vitányi, P. *An Introduction to Kolmogorov Complexity and Its Applications*; Springer: New York, NY, USA, 2013. [CrossRef]
29. Cilibrasi, R.; Vitanyi, P. Clustering by compression. *IEEE Trans. Inf. Theory* **2005**, *51*, 1523–1545. [CrossRef]
30. Semenoglou, A.A.; Spiliotis, E.; Makridakis, S.; Assimakopoulos, V. Investigating the accuracy of cross-learning time series forecasting methods. *Int. J. Forecast.* **2021**, *37*, 1072–1084. . [CrossRef]
31. Novak, R.; Bahri, Y.; Abolafia, D.A.; Pennington, J.; Sohl-Dickstein, J. Sensitivity and Generalization in Neural Networks: An Empirical Study. *arXiv* **2018**, arXiv:1802.08760.
32. Kourentzes, N. Intermittent demand forecasts with neural networks. *Int. J. Prod. Econ.* **2013**, *143*, 198–206. [CrossRef]

33. Croston, J.D. Forecasting and Stock Control for Intermittent Demands. *J. Oper. Res. Soc.* **1972**, *23*, 289–303.
34. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780.
35. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proceedings of the Advances in Neural Information Processing Systems*; Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2019; Volume 32.
36. Alexandrov, A.; Benidis, K.; Bohlke-Schneider, M.; Flunkert, V.; Gasthaus, J.; Januschowski, T.; Maddix, D.C.; Rangapuram, S.; Salinas, D.; Schulz, J.; et al. GluonTS: Probabilistic and Neural Time Series Modeling in Python. *J. Mach. Learn. Res.* **2020**, *21*, 4629–4634.
37. Petropoulos, F.; Apiletti, D.; Assimakopoulos, V.; Babai, M.Z.; Barrow, D.K.; Ben Taieb, S.; Bergmeir, C.; Bessa, R.J.; Bijak, J.; Boylan, J.E.; et al. Forecasting: Theory and practice. *Int. J. Forecast.* **2022**, *38*, 705–871. . [CrossRef]
38. Garza, F.; Canseco, M.M.; Challú, C.; Olivares, K.G. *StatsForecast: Lightning Fast Forecasting with Statistical and Econometric Models*; PyCon: Salt Lake City, UT, USA, 2022.
39. Hyndman, R.J.; Khandakar, Y. Automatic time series forecasting: The forecast package for R. *J. Stat. Softw.* **2008**, *27*, 1–22. [CrossRef]
40. Hyndman, R.J.; Koehler, A.B.; Snyder, R.D.; Grose, S. A state space framework for automatic forecasting using exponential smoothing methods. *Int. J. Forecast.* **2002**, *18*, 439–454. [CrossRef]
41. Ord, J.K.; Fildes, R.; Kourentzes, N. *Principles of Business Forecasting*, 2nd ed.; Wessex Press Publishing Co.: London, UK, 2017.
42. Kang, Y.; Hyndman, R.J.; Smith-Miles, K. Visualising forecasting algorithm performance using time series instance spaces. *Int. J. Forecast.* **2017**, *33*, 345–358. . [CrossRef]
43. Jolliffe, I. *Principal Component Analysis*, 2nd ed.; Springer Series in Statistics; Springer: New York, NY, USA, 2002. [CrossRef]
44. O'Hara-Wild, M.; Hyndman, R.; Wang, E. feasts: Feature Extraction and Statistics for Time Series. 2022. Available online: https://github.com/tidyverts/feasts/ (accessed on 12 December 2022).
45. Lê, S.; Josse, J.; Husson, F. FactoMineR: A Package for Multivariate Analysis. *J. Stat. Softw.* **2008**, *25*, 1–18. [CrossRef]
46. Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; Koyama, M. Optuna: A Next-generation Hyperparameter Optimization Framework. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Anchorage, AK, USA, 4–8 August 2019.
47. Hyndman, R.J.; Koehler, A.B. Another look at measures of forecast accuracy. *Int. J. Forecast.* **2006**, *22*, 679–688. [CrossRef]
48. Hollander, M.; Wolfe, D.A.; Chicken, E. *Nonparametric Statistical Methods*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2015. [CrossRef]
49. Kourentzes, N. tsutils: Time Series Exploration, Modelling and Forecasting, R Package Version 0.9.3; 2022. Available online: https://github.com/trnnick/tsutils/ (accessed on 12 December 2022).