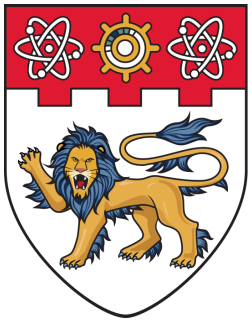

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

COMPUTATIONAL INTELLIGENCE APPROACHES IN FORECASTING DEMAND OF ESSENTIAL MEDICAL PRODUCTS

Suhwan Chung

College of Computing & Data Science (CCDS)

A thesis submitted to the Nanyang Technological University in partial fulfillment of the requirements for the degree of Master of Engineering (M.Eng)

2024

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

04/08/2024

.....

Date

.....

Suhwan Chung

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

04/08/2024

.....

Date

.....

Professor Jagath Rajapakse

Authorship Attribution Statement

This thesis does not contain any materials from papers published in peer-reviewed journals or from papers accepted at conferences in which I am listed as an author.

04/08/2024

.....

Date

Suhwan Chung

.....

Suhwan Chung

Acknowledgements

I would like to deeply thank my supervisor, Jagath C. Rajapakse, for his unwavering guidance and support throughout this research endeavour. His profound expertise in computer science research provided insights that greatly improve quality and depth of my works. At a time when decisions were needed about my research works, his unweavering views and advice on the future career significantly helped making the sound decisions required.

I also extend my thanks to senior management at Becton Dickinson: Paul Holt, Min Yuan, and Raymond Chow, who sponsored the research collaboration program with Nanyang Technological University and trusted me to drive the program internally. Their inspiration and encouragement led me to begin my PhD program while working full-time, for which I am grateful. My colleagues at the company, Soumyakant Dwivedy and Delton Teo, deserve my thanks for their trust in the research team and their hard work to make this program sustainable. Big thanks to all the research assistants and student engineers from the school who supported the research collaboration program throughout this journey.

Finally, I extend my heartfelt thanks to my wife, daughter, and son. Their unwavering support during long, extended days kept me focused on my research work.

Contents

List of Figures

List of Tables

Abstract

It's widely accepted in the forecasting community that complex methodologies may not necessarily result in better out-of-sample predictions compared to simpler methods. The idea is backed by the numerous findings from influential forecasting competitions and previous literature. For instance, in the influential M4 forecasting competition, eight purely neural network-driven models tend to struggle due to insufficient sample size for reliable parameter fitting and suffer from the risk of fitting to the random noise. In fact, having more data in a time series context doesn't necessarily mean to having more reliable insights from each individual time series. Instead, it indicates the availability of numerous "related" series can be considered in the forecasting process that can provide reliable information. In addition, as more related series becomes accessible, the use of more complex models become a more viable approach when trained jointly across all series by "borrowing" information from other series. Forecasting models that can be trained across joint (group) series have a competitive edge over traditional uni-variate forecasting techniques such as Holt-winters, ARIMA, and others. Despite numerous efforts, constructing forecasting models that can train across various disparate series remains challenging since the weak or irrelevant series and features could affect to the prediction of varying forecasting model. To mitigate issues cased by weekly connected series, we recommend developing models that analyze structural features of time series instead of constructing models using past observed values. In our second chapter, we propose a two-stage meta-learning process that resembles the Model Agnostic Meta Learning (MAML) frameworks where the focus is to improve performance on new forecasting tasks with just a few gradient updates. Other challenges related to joint series training observed in our literature review involved with training across heterogeneous time series. The issue becomes more noticeable when a forecasting function is required to be estimated from all relevant time series simultaneously, which is also known as Global Methods. In our third chapter, we propose a unique method to address the issue of identifying similarities among series. This method uses time series characterization algorithms to cluster series and apply forecasting functions to subsets based on their similarity. In our fourth chapter, we bridge the gap between research and industry application of cross-series forecasting by examining the interactions between autoregressive orders and their impact on performance during joint-series training.

Chapter 1

Introduction

1.1 Background

It's widely accepted in the forecasting community that complex methodologies may not necessarily result in better out-of-sample predictions compared to simpler methods. The idea is backed by the numerous findings from influential forecasting competitions including M3 [38], M4 [40], and M5 [39] which have significantly influenced the forecasting research directions throughout the decades, with many novel techniques emerging during submissions. In the M4 competition, eight purely neural network-driven models fell short of outperforming hybrid approaches which leverage statistical and machine learning properties, largely dominated the rankings. When dealing with uni-variate time-series with limited sample sizes, the information that can be extracted becomes limited [71]. In this case, simpler models, which are less sensitive to noise, tend to perform better given their reasonable prior assumptions about the data. In contrast, complex models like the neural networks tend to struggle due to insufficient sample size for reliable parameter fitting and face the risk of fitting to the random noise during the training process.

Even with large sample sizes, the distant past observations may not be as informative features in forecasting tasks, because fluctuations and auto-regressions likely evolve over time. This suggests that unless the sample size is big and consistently show stable patterns, complex neural networks may not provide significantly better results than simpler models. It prevents the success of neural networks from the application of uni-variate forecasting, despite efforts invested by many businesses to collect large volumes of data, making negligible returns. In fact, in the context of time-series forecasting, having more data doesn't always mean to having more reliable insights from each individual time-series. Instead, it indicates the availability of numerous related series can be taken into account in the forecasting process that can provide reliable information. In addition, as more related series becomes accessible, the use of more complex models become a more viable approach when trained jointly across all series by "borrowing" infromation from other series.

1.2 Motivation

Time-series prediction models that can be trained on combined (group) series have a competitive edge over traditional uni-variate forecasting techniques such as Holt-winters [11], ARIMA [8], and others. For

instance, Hartmann et al. (2015) proposed a cross-sectional regression techniques for homogenous time-series to mitigate the issue of missing values in a uni-variate time-series. Similarly, Trapero, Kourentzes, and Fildes (2015) introduced a pooled regression model by pooling similar time-series to generate reliable promotional predictions in situations where historical sales data is lacking. Moreover, Prudêncio and Ludermir (2004) first introduced a meta-learner in time-series forecasting and demonstrated the meta-learning can be an ideal choice when forecasting multiple related series simultaneously.

1.3 Organization of Thesis and Contributions

Despite numerous efforts, constructing forecasting models that can train across various disparate series remains challenging, regardless of the chosen model. For instance, Wang et al. (2023) discussed the potential challenge in achieving consistent performance when using more complex models such as meta-learning algorithms, since weak or irrelevant series and features could affect the prediction of varying forecast model performance. To address this issue derived from weakly connected series, we propose constructing forecasting models that leverages systemic characteristics of time-series rather than directly employing observed values in the past observations. In particular, in our second chapter, we propose a two-stage meta-learning framework that works on feature characterization algorithms. In our experiment, a meta-learner based on OpenAI's Reptile algorithms is implemented with the two stages: (i) training FeatureNet in a fully connected neural network to generate transferable features across uni-variate time-series, and (ii) fine-tuning with CovarNet to refine the final forecasts. The second stage of training is done by incorporating the intermediate outputs from the initial stage and the past observations of time-series. This two-step training process resembles the Model Agnostic Meta-Learning (MAML) frameworks where the focus is on optimizing model parameters to improve performance on new tasks with just a few gradient updates. Our proposed two-step Reptile framework presented in the second chapter contributes to the forecasting domain in several ways:

- Our approach involves using model generated covariate, which is an output from the initial training phase. These are based on a large feature matrix, created using time-series feature extraction algorithms. This covariate is then merged with the original time-series to refine predictions in the second phase. This approach differs from traditional meta-learning methods, which treat any new features as constant covariates in many studies.
- In contrast to traditional meta-learning that mainly focuses on extracting meta-features for model selection tasks, our approach incorporates learning directly into the model via iterative updates. This transition from selecting the "best" external model to building an internal neural networks class allows variations of these networks, such as a two-stage training process.

Other challenges observed in our literature review involved with training across heterogeneous time-series. In particular, the issue becomes more noticeable when a forecasting function is required to be estimated from all relevant time-series simultaneously, which is also known as Global Methods. As mentioned above, when all time-series are combined into one large matrix, the forecasting function tend to fit into the random noise within the combined series. To address this issue of finding similarities among series, in our third chapter, we propose a unique method to identify similarities among series. We first employ time-series characterization algorithms to extract multiple features from these series, and hierarchical and k-means clustering algorithms are applied to these feature subsets, grouping series with unique patterns. Then, we construct forecasting functions that can capture both linearity and non-linearity for each cluster. Here are the benefits of our

work:

- This work contributes to cross-series forecasting methods by providing empirical evidence on how the similarity of time-series impacts prediction results. We propose a method called Feature-based Cross-series Forecasting which is built on the structural homogeneity of each unique series. Due to the interpretable feature subsets, our method provides interpretability, a key factor in forecasting that traditional time-series forecasting methods often lack.
- Our work also contributes to the giant body of time-series research, especially in the context of small-sample size. We address this issue by constructing large matrices of lagged features using data pooling techniques. Each matrix consists of the lagged observed values from similar series defined by clustering techniques. These data pooling techniques, which work on the structural patterns of time-series, can be beneficial for time-series forecasting, and our work could potentially serve as a reference for future research.

Finally, auto-regressive (AR) order is one of the most commonly observed challenges in our review of cross-series forecasting. Also known as the memory of the model, AR order selection is utmost important parameter tuning in time-series forecasting. time-series forecasting is fundamentally dependent on its past observations, and they have historically been key parameter in making predictions [28, 43, 46]. However, there is a noticeable lack in the literature on cross-series forecasting regarding a systematic investigation of the auto-regressive order and its effects on the accuracy of forecasts in the context of cross-series forecasting. This gap is significant because the selection of cross-series forecasting methods typically assumes that the time-series of interest are homogeneous. However, it remains unclear which segments (n-th time step) of the series show similarity. This question can only be answered by auto-regressive selection that restricts the training data for the chosen model. In Chapter 4, we address this gap by providing empirical evidence about the impact of auto-regressive order selection on the effectiveness of cross-series forecasting methods. The contributions of this chapter to demand forecasting research are as follows:

- We investigate whether global cross-series methods (or simply global method), which pool all-time-series of interest outperform in the short memory order. To this end, we first explore the relationships between the auto-regressive order and forecast horizons in terms of errors. Moreover, we examine if these findings can be generalized across different forecasting scenarios.
- The key objective of this research is to understand the effect of various combinations of auto-regressive orders on model results. We aim to establish guidelines for optimal configuration through experimentation using M4 competition data and fill the existing gap of comprehensive empirical analysis in the field and to understand how the "depth" of historical data influences predictive performance.

1.4 Benchmark Forecasting Methods

In time-series forecasting research, benchmark methods are used for evaluating the performance of new techniques and reference points for comparing novel forecasting models. Widely used benchmark forecasting techniques which are known to capture non-linearity includes XGBoost [12], Random Forest [9], Gradient Boosting [7], Decision Tree [51], and Long-Short-Term Memory models [22]. For models known to capture linearity includes Linear Regression [73] and Ridge Regression [6]. In this thesis, the proposed benchmark techniques present a comprehensive analysis of the new forecasting methods. We compare the performance of

seven modeling techniques and evaluate the impact of various forecasting methods. The selected benchmarks include:

***Random Forest*:** It builds multiple decision trees during training and averages their predictions for regression. As demonstrated by Tyralis and Papacharalampous (2017), the model combines predictions from individual trees, which can capture non-linear relationships between time-series variables to improve accuracy and prevent overfitting. During implementation, each tree is trained on a different bootstrap sample with randomly selected features. The prediction for a given input is the mean of predictions from all trees. For each univariate time series $x_i = \{x_{i,t}\}_{t=1}^{T}$, where t represents the forecast at each time point for the univariate time series x_i, and $f_K(x_{i,t})$ is the prediction from each tree K, the prediction $\hat{x}_{i,t}$ using RF is given by:

$$\hat{x}_{i,t} = \frac{1}{K} \sum_{k=1}^{K} f_k(x_{i,t}) \tag{1.1}$$

XGBoost: It is an optimized gradient boosting algorithm that uses decision trees in a parallel fashion while fitting new models to correct the errors of existing models[66]. The objective function is defined by:

$$\mathcal{L} = \sum_{i=1}^{I} \sum_{t=1}^{T} l(x_{it}, \hat{x}_{it}) + \sum_{k} \Omega(f_k) \tag{1.2}$$

$\Omega(f_k)$ is a regularization term to prevent over-fitting with a loss function l that measures the delta between predicted and actual values. Its effectiveness in time-series forecasting has been demonstrated in several studies.

***Linear Regression*:** It assumes a straight-line correlation between the input variables and the target variable in which the prediction $\hat{x}_{i,t}$ is given by:

$$\hat{x}_{i,t} = \beta_0 + \sum_{t=1}^{T} \beta_l x_{i,t-l} \tag{1.3}$$

where the model estimates the coefficients (β_l) that minimize the sum of squared errors. In the above equation, (β_l) is the coefficients for the lagged input variables $x_{i,t-l}$ where l is the maximum lag used as a predictor. In time-series forecasting, Linear Regression is commonly used as a benchmark due to its simplicity.

***Ridge Regression*:** It is a regularized form of linear regression that minimizes the residual sum of squares while penalizing the magnitude of the coefficients using an L2 regularization term. Its objective function follows:

$$\mathcal{L} = \sum_{i=1}^{I} \sum_{t=1}^{T} (x_{i,t} - \hat{x}_{i,t})^2 + \lambda \sum_{t=1}^{T} (\beta_t)^2 \tag{1.4}$$

where λ is the regularization parameter. Ridge regression is also known to handle multicollinearity makes it suitable for time-series data with correlated predictors [68].

***Decision Tree Regression*:** It partitions the data into subsets based on feature values to minimize the mean squared error (MSE). The final prediction is the mean value of the target variable in the leaf node. For a node with i sample series, the MSE is minimized by:

$$\text{MSE} = \frac{1}{I}\sum_{i=1}^{I}\frac{1}{I}\sum_{t=1}^{T}(x_{i,t} - \hat{x}_{i,t})^2 \tag{1.5}$$

***Gradient Boosting*:** Gradient Boosting [58] builds models in stages by fitting new models to correct the residual errors made by prior models. For instance, the prediction at stage t for time-series x_i is given by:

$$\hat{x}_{i,t} = \hat{x}_{i,(T-1)} + \eta f_k(x_{i,1}, x_{i,2}, \ldots, x_{i,(T-1)}) \tag{1.6}$$

where η is the learning rate. It minimizes a differentiable loss function by iteratively updating predictions with a weighted combination of weak learners f_k. Several studies have shown that the gradient boosting can effectively learn from complex patterns in time-series data.

The rest of the thesis is organized as follows. In Chapter 2, we introduce a novel two-stage Reptile forecasting framework that works on both the underlying characteristics of time-series and historical real values like conventional forecasting. Chapter 3 presents a new method for cross-series forecasting using structural similarities. Chapter 4 explores our research on forecasting and its connection to memory. Finally, Chapter 5 concludes the thesis.

Chapter 2

Two-Stage Reptile Meta-Learning for Time-series Forecasting

Abstract

In this chapter, we propose a novel two-stage meta-learning frameworks for time-series forecasting using OpenAI's Reptile algorithm, a gradient-based meta-learning approach that enables task-agnostic initialization and quick adaptation to new tasks. Our approach consists of two stages: FeatureNet, a fully connected neural network that extracts transferable features from univariate time-series data, and CovarNet, a neural network that uses these extracted features as additional covariates to generate final predictions. We benchmark our approach using the Mean Absolute Scaled Error (MASE) against conventional time-series forecasting methods and demonstrate superior performance with greater interpretability which is key for supporting industry decisions. To the best of our knowledge, this is the first attempt to use multi-phase meta-learning implementation of Reptile in time-series forecasting.

Keywords

- Meta Learning
- Reptile algorithms
- Model-Agnostic Meta Learning
- Feature-based forecasting

2.1 Introduction

2.1.1 Background

Accurate demand forecasting is a key success factor for modern supply chain operations. Supply Chain Management (SCM) focuses on meeting customer demand while keeping total supply costs low. The application

of algorithmic demand forecasting models in its operations can help generate accurate demand predictions and synchronize supply chain activities, leading to improve customer satisfaction [55]. However, choosing suitable forecasting techniques for real-world supply chain planning can be challenging for several reasons: (i) no single method is universally effective for all types of forecasting scenarios, (ii) the common, classic uni-variate forecasting model doesn't align with industry demands that require capturing demand sensing signals, (iii) even if the external co-variate are readily available, there isn't a straightforward mechanism to derive algebraic expressions for each co-variate's contribution to the predicted results, and (iv) the variable importance and statistical significance of each feature can fluctuate at different time points, which poses challenge in feature selection, leading to increase complexity of forecasting models. Therefore, automatic feature selection is essential but challenging task in supply chain operation today.

To address the aforementioned problems, feature-based forecasting techniques, which extract underlying time-series characteristics and use them as means of grouping time-series through clustering or direct forecasting, are emerging as a promising alternative to address the aforementioned problem. The key phase involved in the feature-based forecasting technique is feature extraction phase where features are generated from the original time-series. These features, which can range from trend and seasonality to auto-correlation and entropy, serve as inputs to forecasting models [1]. Other common methods for feature extraction include auto-encoders, such as Multi-Layer Perceptron (MLP) and Long Short-Term Memory (LSTM) which process time-series data by using a sliding window method to extract salient features that capture temporal patterns in the original time-series [13]. Another technique in feature-based forecasting is conditional temporal aggregation which uses classification models to choose the most appropriate aggregation level for forecasting and employs correlation between key time-series features and forecasting accuracy to select significant time-series features [31]. Research on demand forecasting often uses direct external signals[56] or inherent characteristics of time-series, applying feature extraction algorithms[72] from the original time-series. These features, which can range from trend and seasonality to auto-correlation and entropy, serve as inputs to forecasting models [1]. In research that considers the inherent characteristics of time-series, meta-learning can offer a promising alternative to this problem. Oreshkin et al. (2021) in research of time-series forecasting provides compelling evidence that meta-learning can significantly improve generalization on new time-series, which the meta-learning could be a strong candidate to mitigate uncertainties involved in forecasting by cross-learning from different datasets.

The concept of employing meta-learning in time-series research has been examined by numerous researchers. More recently, Arango et al. (2021) explored the concept of using meta-learning to adjust parameters of the models for new time-series with limited sample size. He then proposed a method that uses an auxiliary network to condition the model parameters by encoding global information from the time-series to extract meta-features. Further evidence on the superiority of meta-learning is given by Ma and Fildes (2023) that proposed a supervised meta-learning frameworks first learn a feature representation from the serial data. These learned features are then associated with weights, which are used to assign the weights to the base forecasters and produce combined final forecasts. A similar approach is presented by Vaiciukynas et al. (2021) in which the autors first constructs a feature space using feature extraction techniques and then uses these features to train meta-models that can select the best forecasting models. In many cases, a feature matrix is derived from the original time-series data which is then utilized as an input for training a meta-learning algorithm. Despite subtle differences in how the output of the meta-learner is handled in various research, the majority of studies focuses on the objective function of the meta-learner for minimizing the

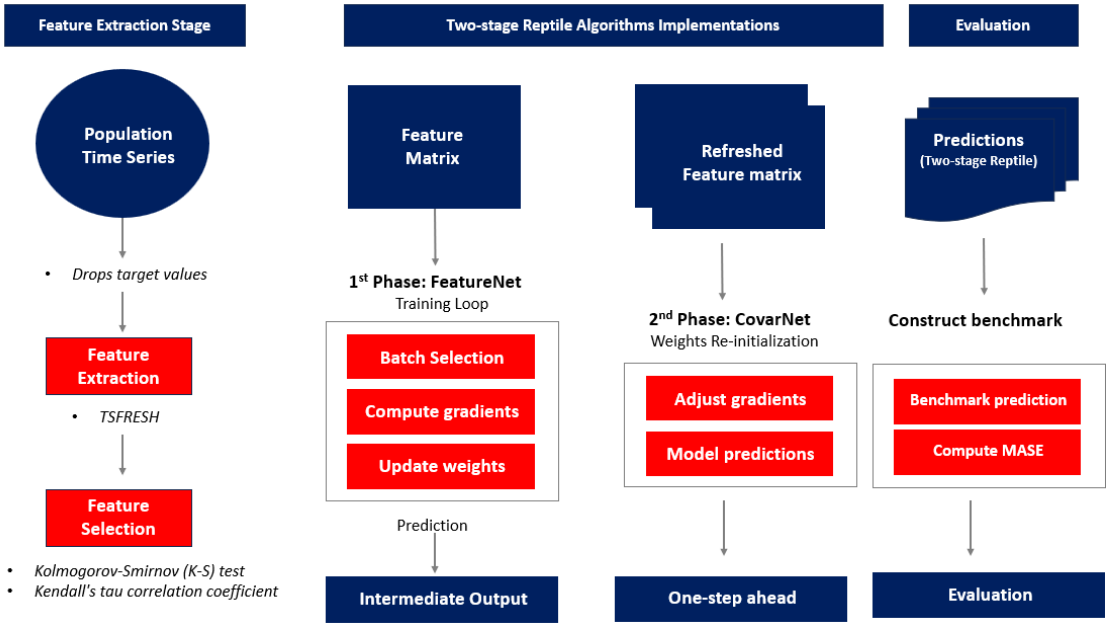

Figure 2.1: Implementation of the two-stage Reptile algorithm: (i) In the first stage, the Reptile algorithm learns from the feature matrix to predict the future value of the time-series. (ii) In the second stage, the intermediate outputs from the first stage are combined with the historical time-series to generate predictions for the next step.

forecast delta [50, 25, 15, 34].

2.1.2 Motivation and Our Approach

While numerous studies have emphasized the effectiveness of a meta-learning approach for time-series forecasting tasks, only a few have concluded their methods as consistently superior to classic forecasting methods. For instance, Meade (2000) demonstrated that summary statistics of time-series can be important factors for choosing a forecasting method, but theses statistics aren't necessarily the deciding factors. In addition, Kang, Hyndman, and Li (2020) presented potential challenge to the consistent performance of the meta-learner: (i) the use of weak or irrelevant features, (ii) incorrect selection of meta-learning algorithms, and (iii) insufficient or unvaried training data, which could affect the prediction of varying forecast model performance. In response, numerous recent studies have addressed these issues, for example, the recent M4 competition emphasize feature-based, meta-learning techniques as a potential approach for predictive time-series forecasting, and Wang et al. (2022) explored feature-based interval forecasting. Considering recent trends in forecasting research, we have concluded that developing feature-based forecasting methods could be a valuable investigation.

In our research, we attempt to address the previously mentioned challenge involved in both feature-based forecasting and meta learning by constructing a two-stage framework, using Reptile algorithms. This algorithm, created by OpenAI, is a meta-learning method which is known as highly adaptive algorithm to process new tasks with minimal data input. It accomplishes this by iterating over a task set and adjusting model weights for all tasks [47]. The Reptile's inherent characteristics to adapt to new tasks with minimal data input makes it also ideal for time-series forecasting setting, where models frequently need to adapt to various

Table 2.1: Extracted time-series features by using TSFRESH algorithms for a total of 448 time-series. This forms the basis of our feature space, which is further refined using feature selection techniques. For easier categorization, we have defined 12 typologies of features in our research and brief descriptions of each typology is provided.

Index	Typology	Typology description	# Features
1	Autocorrelation	Similarity between lagged observations of the same time-series at different time steps	41
2	Basic Statistics	Basic statistical measures	39
3	Binning and Change Quantiles	Quantified shifts in the distribution by binning values	69
4	Entropy and Complexity	Quantified irregularity, unpredictability, and complexity of series	21
5	Fourier Transforms	Decomposed series into frequency components for detecting periodic patterns	121
6	Occurrence (Count base)	Counted occurrence of various pattern	8
7	Peaks	Local maximum and minimum within series	6
8	Positional (Location base)	Parameters representing spatial-temporal dynamics	8
9	Quantiles (Distribution property)	Statistical distribution of series - skewness in various measures	8
10	Reoccurrence	Repeated occurrences of specific patterns	5
11	Trends and Properties Over Time	Measured long-term tendencies and temporal properties	62
12	Wavelet Transforms	Multi-scaled decomposition of series	60

uni-variate time-series with distinct patterns. In our study, we leverage the notion of adaptability provided by Reptile algorithm with a structured two-stage training method to tackle multiple challenges in time-series forecasting. To this end, we construct a two-steps forecasting structure with Reptile where learning from the first stage is transferred through co-variate that represents the feature representaion learning of the time-series. In the initial phase (called FeatureNet), the reptile algorithms is trained based on the feature matrix of the original time-series. It then produces intermediate representations for each time-series in continuous form that captures underlying characteristics from the feature matrix. In the second stage, the intermediate outputs are combined with historical time-series data to take advantage of the previously learned feature representation. To this end, the same neural networks with the learned weights (called CovarNet) are constructed to refine predictions, using this enriched feature set. After reviewing the literature, we found that the implementation of the Reptile algorithms in the context of feature-based time-series forecasting represents an innovative and first attempt to set a new predictive modeling standard in time-series forecasting fields. Our research further expands the concept of meta-learning by treating the features extracted from time-series as multi-variate forecasts which are then fine-tuned with original vector of time-series. As a result, the meta-model is trained to learn the interplays between temporal dependency and the underlying characteristics of time-series represented as features.

In most time-series forecasting research in the application of meta-learning, the focus in on model selection process. This involves choosing the best model with the least errors from a group of potential models by minimizing the difference in forecast accuracy across individual time-series. While the use of the Reptile algorithm within a two-step training framework in our research aligns with the broader notion of meta-learning, the application of a meta-learner using Reptile in this study deviates from traditional model selection approaches. In fact, numerous meta-learning research in the sub-field of time-series forecasting evaluates various models based on a set of meta-features derived from time-series data to determine which model "best" suits specific tasks. For example, Vaiciukynas et al. (2021) train a meta-model on features, such as the basic statistics of the series, characteristics of the residuals from naive forecasting models, and decomposition to predict the most effective forecasting method for a given time-series. Similarly, Barak, Nasiri, and Rostamzadeh (2019) extract features from time-series data and train meta-learners to choose from a subset of high-performing models. In contrast, our proposed methodology with the Reptile algorithm avoids selecting from a pre-existing set of models. Rather, it refines model's parameters of a basic neural networks through a two-stage training procedure: the first step utilizes generalized feature-based learning to construct knowledge base

of the pool of time-series data across multiple contexts and the second stage refines the similar networks by transferring the co-variate from the first phase. This two-step training process is similar to Model Agnostic Meta-Learning (MAML), which focuses on optimizing model parameters in such a way that a few gradient updates will significantly enhance performance on new tasks. Our proposed forecasting framework contributes to the forecasting domain in several ways:

- We employ co-variate of continuous value which is an output of the initial training phase based on binary and continuous features in order to implement the notion of feature-based forecasting. Then this co-variate is merged with the original time-series to fine tune predictions in the second phase. This approach differs from traditional meta-learning methods, which used any new features as constant co-variate in numerous research.
- Unlike traditional meta-learning, which largely focuses on extracting meta-features as a base for model selection, our methodology embeds learning directly within the model through iterative updates. This shift in focus moves away from choosing the "best" external model towards constructing internal neural networks class, which allows variations of the networks such as two-stage training process.
- In this chapter, we demonstrate performance improvements by mapping the outcomes of each time-series in a two-dimensional Principal Component space to create insights into how the quality of feature extraction from the first phase can affect to the evaluation of forecasts in the final stage.

2.2 Methodology

2.2.1 BD Product Sales Dataset

Becton Dickinson (BD), one of the world's leading suppliers of medical devices, has a Medication Delivery Solutions (MDS) division that provides products like needles and syringes. This study uses a dataset that includes the historical demands for consumable medical products manufactured by MDS in the Asia region covering 57 months of historical sales. Each time-series represents the monthly demands for 856 products distributed to pharmaceutical companies or hospitals in Southeast Asia. In our dataset, many series have a zero value which indicates no demand for the corresponding month, potentially due to factors such as sufficient stock levels or inventory from the logistic partners or hospitals. A problem identified during exploratory data analysis is the difficulty of forecasting demand for the subsequent month with no demand, which the pattern of zero demand is quite commonly observed in demand and supply management. In addition, our continued investigation into the data found that less than 10 percent of the total series display seasonality patterns. Since structural patterns such as seasonality or trends are a crucial component of time series forecasting, building forecasting models with samples that contain zero values can introduce significant random noise when detecting patterns in time-series data. To reduce the risk of modeling random noise, in our research, we set the non-zero value thresholds for each series based on Inter-quartile Range (IQR) analysis and excluded any series that do not meet the thresholds as shown in 2.2. As a result, 328 series were included in our research scope.

2.2.2 Two-stage Reptile Frameworks

The two-stage Reptile frameworks as shown in Fig. 2.1 consists of two phases: FeatureNet and CovarNet. In the FeatureNet phase, for each uni-variate time-series, $x_i = (x_{i,t})_{t=1}^{T}$, we apply k feature extraction algorithms

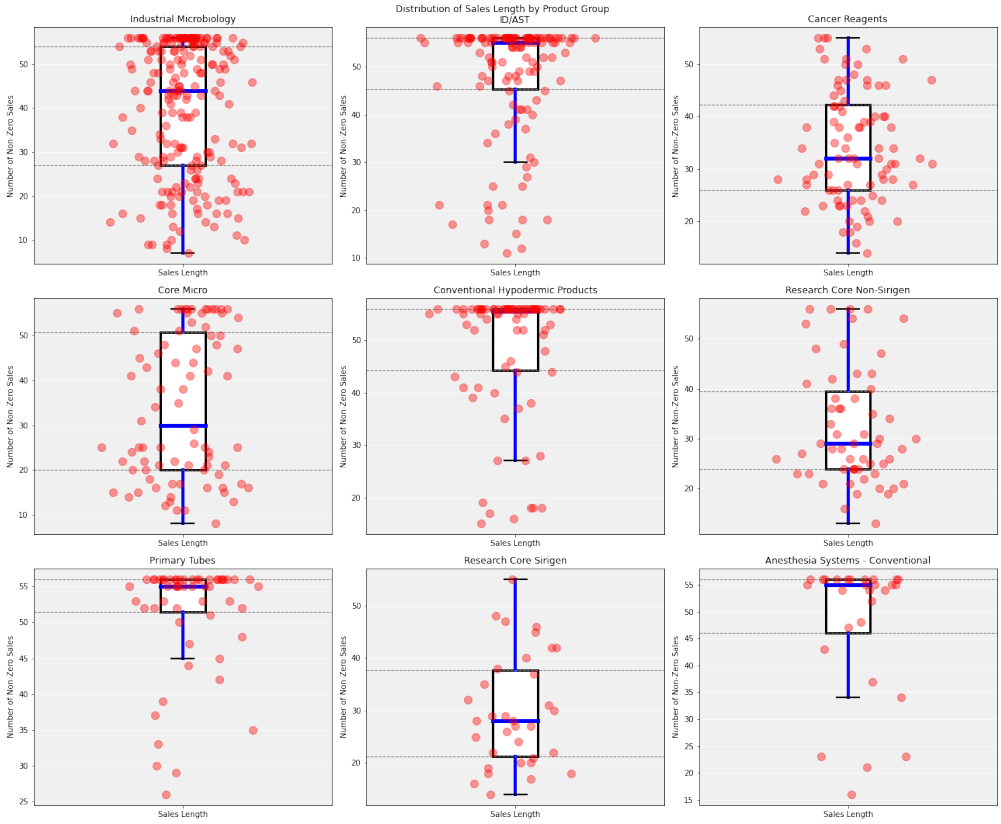

Figure 2.2: Box-and-Whisker plots showing the distribution of non-zero counts from historical demands for 856 medical products. It's important to note that while zero-demand instances can and do occur in the real world, they introduce significant random noise when detecting patterns in time-series data. Based on IQR analyses, we've excluded 528 time-series from the original 856, narrowing down our research scope to 328.

up to $x_{i,t}$, excluding the last two points, to create the feature vector by applying feature extraction algorithms $f_k \{x_{i,t}\}_{t=1}^{T-2}$. The loss function $L(\theta, f_k(x_{i,T-2}), \hat{x}_{i,t})$ is then defined, where θ denotes our model's parameters and $\hat{x}_{i,t}$ represents the forecast at time point t for each uni-variate time series x_i. The goal is to minimize this loss function across all samples where $\hat{x}_{i,t-1}$ denotes the second last forecasted value of the time-series i. To determine the optimal parameters θ^*:

$$\theta^* = \arg\min_{\theta} \sum_{i=1}^{I} L(\theta, f_k(x_{i,T-2}), \hat{x}_{i,t}) \tag{2.1}$$

In the second stage, the parameters θ^* of neural networks obtained from the first stage are used as the initial parameters for fine-tuning. The fine-tuning process focuses on the prediction of the last value of each time-series, which serves as the new target variable x_i. Same as the first stage, the objective in this stage is to further minimize the loss function of the target variable.

Our methodology, employing the Reptile algorithm, aligns closely with Model-Agnostic Meta-Learning (MAML) in that a few gradients updates can optimize model parameters and improve performance on new tasks (each tasks are forecasting each uni-variate time-series). In mathematical terms, Reptile algorithms updates θ over tasks for time series x_i where β is the step size simplifying convergence compared to

traditional MAML updates.

$$\theta \leftarrow \theta + \beta \nabla_\theta \sum_{i=1}^{I} L(\theta, x_i) \tag{2.2}$$

A more detailed description of the methodology is as follows:

- *Frist Phase: FeatureNet*

    1. **Feature Matrix Construction using time-series Characterization**: We extract an extensive set of features from the historical demand of each SKU (Stock Keeping Unit). These features range from statistical summaries and time-based properties to auto-regressive elements that reflect underlying characteristics of series. We use TSFRESH for feature extraction due to its efficiency to extract large volumes of characteristics [14].

    2. **Feature significance testing**: In the subsequent phase, each feature vector is evaluated in terms of its importance for predicting the target value under investigation. For binary feature sets, the importance of each binary feature to a target value is measured through the p-value based on the two-sided uni-variate Fisher test [19]. Similarly, the significance of a real-valued feature to a target value is evaluated using Kolmogorov-Smirnov test [35]. The results of these statistical assessments are expressed as p-values that indicate the importance of each feature in forecasting target. We then select both binary and real-valued features with significant p-value to refine the initial feature matrix.

    3. **FeatureNet training**: We employ the Reptile algorithm for training by constructing a network called FeatureNet. Initially, the model is exposed to a range of sub-tasks, starting with its global weights defined by activating *He initialization (Kaiming normalization)*. The model then enters a training phase for each task using a mini-batch subset (a subset of uni-variate series). In reptile, this rapid training is designed to customize the model to the specific requirements of each task through a finite number of gradient updates. Following the training, the model's weights are adjusted not only based on task-specific learning but also by merging the initial weights with the task-adapted weights. The training is run in a way to minimize the forecast errors for a target value, using a feature matrix as input.

- *Second Phase: TuningNet*

    1. **Merge covariates and make predictions**: In the second stage of training, we construct an additional network, CovarNet. Similar to the first phase, the CovarNet is trained using the Reptile algorithm and uses the weights from the initial phase as a starting point, and adjustments are made to the parameters using the complete range of historical time-series data which is merged with intermediate output from the first stage. Next, a prediction is made on the combined feature sets, and the results from the FeatureNet are used to predict the last point in the series.

    2. **Perform error analysis**: The outputs from the two-stage models are evaluated with classical forecasting models using the Mean Absolute Scaled Error (MASE). For a fair comparison, we used the same prediction setting to generate a one-step ahead forecast using the last point of each series as target variable.

Table 2.2: After filtering statistically significant features with a p-value less than 0.05 using the Kolmogorov-Smirnov test for binary features and Kendall's tau correlation coefficient for continuous feature sets, the total number of time-series features was reduced to 315 from the initial 448. The feature selection process eliminated 133 features through statistical significance testing. Note that most of the selected features are continuous, fitting well with the time-series data.

Index	Typology	# Total Features (p-value < 0.05)	# Binary Features (p-value <0.05)	# Continuous Features (p-value <0.05)
1	Autocorrelation	**19**	0	19
2	Basic Statistics	**26**	6	20
3	Binning and Change Quantiles	**58**	0	58
4	Entropy and Complexity	**11**	0	11
5	Fourier Transforms	**92**	0	92
6	Occurrence (Count base)	**4**	3	1
7	Peaks	**1**	0	1
8	Positional (Location base)	**1**	0	1
9	Quantiles (Distribution property)	**8**	0	8
10	Reoccurrence	**3**	0	3
11	Trends and Properties Over Time	**38**	0	38
12	Wavelet Transforms	**54**	0	54

The subsequent sections provide an in-depth explanation of each component using demands for essential medical product data as an example. In this context, the demand prediction problem involves predicting the next time step forecasts, which equates to the demands for essential medical products in the upcoming month.

2.2.3 Feature Matrix Construction using time-series Characterization

For the last decade, a great deal of study on meaningful time-series characterization (TSC) methods have been published which combines various statistical methods to extract structural information inherent in time-series. Among them, Christ et al. (2018) released a Python package called TSFRESH. The package offers a framework to automate the extraction of hundreds of time-series features from raw time-series data. In the current research, we followed the notion of time-series characterization for $x_{i,t} \in R^n$ which is i-dimensional time-series at t-th time points and n is the feature dimension. In order to extract time-series characteristics from i dimensional time-series vectors consisting of T time steps, the time-series characterization function f_k is constructed using TSFRESH where k is the feature extraction algorithms that follows $(f_1(k_1), f_2(k_2), ..., f_K(x_I)$. The resulting feature matrix would have $I = 856$ rows and $K = 448$ features with each rows and columns corresponds to 448 underlying characteristics extracted from each 856 uni-variate time-series.

Table. 2.1 displays the initial set of values for the extracted features that mimic the time-series in each uni-variate time-series. We extracted 12 typologies of feature sets to produce a total of 448 features that belong to each category respectively. Given that real-world demand patterns are notoriously volatile and fluctuating, features like binning of basic statistical values (e.g., 3-month moving window) and quantile changes can be valuable parameters in understanding high volatile demand patterns.

The thresholds for filtering the time-series based on the non-zero sales values were determined by our observations of box and whisker plots, as shown in Fig. 2.2. The box-and-whisker plot displays the distribution of non-zero time-series lengths of demands for essential medical products across different product categories.

Some outliers can be observed, particularly in the industrial microbiology products, cancer reagents, and ID/AST product categories. In order to setthe minimum thresholds, we use the upper boundaries (Q3+1.5 x IQR) of the box-and-whisker plots. This approach as chosen because we incorporate the length of time-series as features within our meta-learning framework. As a result, the non-zero lengths of the series should encompass all or the majority of time-series we need to forecast.

In addition, to address the challenge associated with the lack of observed characteristics in time-series, such as seasonality, we extracted features of binned change quantiles (e.g., binned variance and binned mean) to reflect the business forecasting context of quarterly trend analysis. We also extracted features like *Fourier transforms* and *wavelet transforms* are also known for beneficial in analyzing time-series data that lack visible systemic patterns or trends [54]. *Fourier transforms* help break down time-series data into their underlying frequency components, which in turn allows the detection of periodicities and hidden cyclic patterns that are not apparent in the time-series analysis. Similarly, *wavelet transforms* allow for the detection of localized, transient events and non-stationary behavior by providing both time and frequency localization. Other features considered in our research include *Positional (location-based) features*, which indicate the spatial-temporal dynamics by identifying where specific events or values occur, quantiles (distribution properties) which revealts the statistical distribution and skewness. Zanin et al. (2012) demonstrated the application of recurrence quantification analysis to characterize non-linear time-series data, hence we also investigated the recurrence feature sets. Using a set of parameters in Table. 2.2, we generated time-series features that characterizes individual 328 uni-variate time-series.

2.2.4 Feature Significance Testing and Feature Selection

In the phase of evaluating feature importance, we individually assess each feature for its significance in making predictions based on the target variable. Suppose we represent the set of feature vectors for uni-variate time series x_i as $f_k(x_i)$ where each feature k is either binary or real-valued. The significance of each feature is determined using suitable statistical tests, which provide a p-value that signifies the relevance of each feature.

For binary features, we measure significance using the Kolmogorov-Smirnov (K-S) test that compares the empirical cumulative distribution function (ECDF) of the binary feature values against that of the continuous target values. Features with significant p-values (e.g., < 0.05) are considered important. Next, after obtaining the p-values for all features, we select those with significant p-values to fine-tune the initial feature matrix. Let $P = \{p_k \mid i = 1, 2, \ldots, K\}$ represent the set of p-values corresponding to each feature. The selection process is $S = \{f_k(x_i) \mid p_k < \alpha\}$ where α is the significance threshold (e.g., 0.05) and S is the final set of selected features. In this study, we used the `tsfresh` library to implement the statistical tests to run the K-S test.

For continuous feature, we use *Kendall's tau* correlation coefficient measures to compute p-value for analyzing the relationship between two sets of continuous variables. This method evaluates the significance of a continuous (real-valued) features set relative to the same continuous target series by calculating a p-value using Kendall's tau correlation coefficient τ. This coefficient measures the ordinal association between the two variables and is defined as follows, where C and D represent the number of concordant and discordant pairs, respectively.

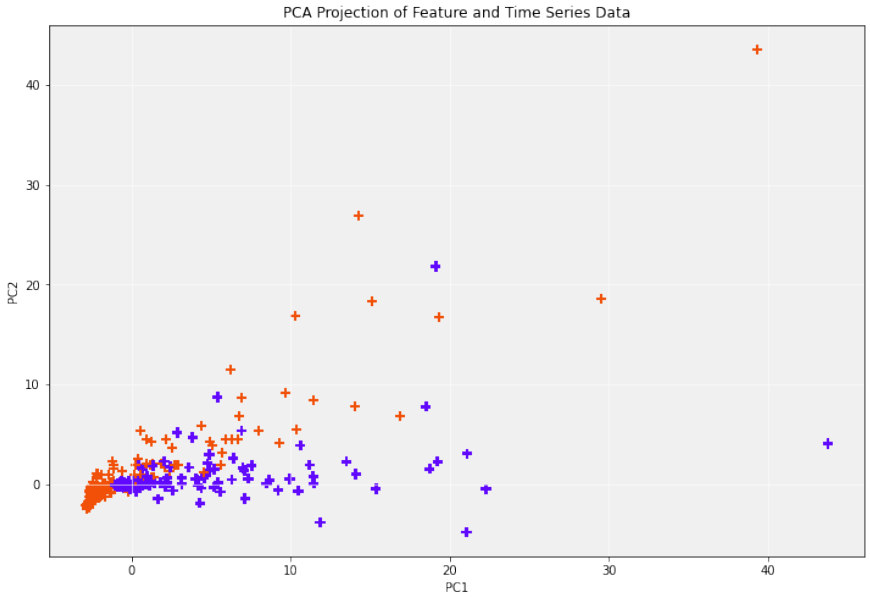

Figure 2.3: The overall distribution of statistically significant, extracted features with a p-value greater than 0.05 in the PCA space is shown. Significant features are represented by dark orange, while blue symbolizes the original historical demands for time-series. The PCA space is calculated from all dimensions of statistically significant features used for training, with the original historical demands time-series then projected into this two-dimensional PCA space. Most product categories for forecasting are within the space of the extracted feature set, with only a few exceptions

$$\tau = \frac{C - D}{\sqrt{(C + D + V) \cdot (C + D + U)}} \tag{2.3}$$

Also the number of ties only in feature set X_k and series $x_{i,t}$ is represented in $V$and U in the equation respectively. Once it computes the correlation coefficient, the p-value for a feature x is calculated using the following hypothesis test:

- Null Hypothesis (H_0): There is no correlation between the feature set $f_k(x_i)$ and the target series $x_{i,j}$.
- Alternative Hypothesis (H_1): There is an correlation between the feature set $f_k(x_i)$ and the target series $x_{i,t}$.

The test statistic is obtained from Kendall's tau as $z = \frac{\tau\sqrt{j(j-1)}}{\sqrt{2(2j+5)}}$ where j denotes the number of observations. The resulting p-value indicates the feature significance, with lower p-values signifying higher significance.

As shown in Fig. 2.2 In our two-stage forecasting framework, we filtered features based on a significance threshold ($\alpha = 0.05$) and retained only those features with a p-value less than the threshold. Figures 2.3 and 2.4 illustrate how well the final feature matrix which is selected through respective significance testing are truly representative of the original time-series data. It's worth noting that using only extracted time-series features to train algorithms can be risky if these features do not accurately represent the population of interest. In cases where forecasters need to deal with high volatile time-series data, whose basic statistics fail to capture any known patterns, extra attention is needed to filter and select features for training in these situations.

2.2.5 First phase: FeatureNet training

The first step of the training process uses the Reptile algorithm to train a neural network (called FeatureNet). The first phase generates a flexible covariate as an intermediate outcome, represented as a continuous value for the 56th time step in each univariate time series. This intermediate outcome captures the underlying characteristics of each series in both binary and continuous feature formats derived from the historical series. In our FeatureNet, we consider a collection of uni-variate forecasting tasks as $\mathcal{J} = \{J_1, J_2, \ldots, J_i\}$, where each task J_i predicts demand of a specific SKUs, the Reptile algorithm starts with a global set of initial weights ($\theta^{(0)}$) and refines them through iterative learning across multiple sub-tasks.

In our experiment, we construct a PyTorch data loader that provides the feature matrix to the model in batches during the training process. The data loader converts the feature and target arrays into tensors and organizes them into mini-batches while ensuringthe model receives data consistently during training. Next, we designed training on each individual tasks as the inner loop which consists of: (i) generating forecasts of the 56-th value in time-series using feature matrix crated up to 55-th time step as input, (ii) loss computation using the Mean Squared Error (MSE) loss between the predictions and the actual values, and (iii) backpropagation. In the backpropagation stage, the loss gradients with respect to the model parameters are computed, and the optimizer updates the model's weights using stochastic gradient descent (SGD). After task-specific training of FeauteNet, the model's weights are interpolated between their pre- and post-training states. In this stage, we clipped the gradients to a fixed value ($clip_value = 1$) to prevent gradient explosion. In this stage of training, the iterative process continues as new tasks are sampled, weights are updated through task-specific learning, and the global weights are adjusted accordingly.

During the initial phase of FeatureNet training with the Reptile meta learning algorithm, there's no traditional split of the input features matrix and a vector of target values into training and test sets, as typically seen in standard supervised learning. Instead, the feature matrix is constructed up to the 55th time step of each uni-variate time-series. In practical forecasting scenarios, forecasters typically use all available historical data to construct predictive models with a goal to optimize the use of the latest data to enhance the model and generate accurate forecasts. We used this strategy in our experiment when training FeatureNet. Furthermore, FeatureNet is primarily designed to learn from the salient feature representations from each uni-variate time-series. Then it generates intermediate output of continuous value which will be transferred to the second stage of training as co-variate (this is equivalent to 56th time step prediction result using feature matrix up to 55th time step).

2.2.6 Second phase: CovarNet

The second phase of training, called CovarNet, refines the meta-learner using the Reptile algorithm. This phase improves prediction accuracy by incorporating intermediate predictions as a new feature, building upon the learned feature representations from FeatureNet (the first phase). The goal is to refine final forecasts by merging the the co-variate with the historical time-series where the result of the first networks conceptualized as the learned representations from the mix of binary and continuous feature matrix. The input to CovarNet is a fresh feature set, new features, composed of the original features used in FeatureNet in the form of continuous co-variate and in-sample time-series. We mathematically depict new features as $(x_{i,t-2}; x_{i,t-1})$ where $X_{i,t}$ symbolizes the vector original features up to $t-2$ th time point, and $x_{i,t-1}$ represents the intermediate predictions made from the first networks. The target data for CovarNet is the

most recent observed value of each uni-variate time-series (57th time step). In contrast to the first phase, which employs the feature matrix to generate intermediate output of a vector of continuous values, the second phase focuses on refining the prediction of the final data point by only considering the continuous values including historical data and the intermediate predictions. The architecture of CovarNet mirrors FeatureNet but incorporates an input layer designed to handle the additional intermediate feature. If we depict the learned weights of CovarNet as θ_{ft}, the model aims to forecast the actual sales data by minimizing the Mean Squared Error (MSE) between the predicted final values and the actual final values, given the new feature matrix. Let x_i symbolize the actual final values, and $\hat{x}_i$ the forecasts by CovarNet; the loss function can be defined as:

$$L(\theta_{ft}) = \frac{1}{I}\sum_{i=1}^{I}(x_i - \hat{x}_i)^2, \tag{2.4}$$

where i is the number of SKUs, and x_i denotes the actual value for SKU i. During the CovarNet training process, the Reptile algorithm modifies its weights across multiple sub-tasks same as the first phase of networks. The inner loop carries out task-specific training by iterating over the mini-batches supplied by the DataLoader. Below is summary of the training mechanism of the CovarNet:

1. **Meta-Training Process:** The model progressively adapts to the task-specific demands across various SKUs.

2. **Task-Specific Training:** For each task (SKU), the model modifies its weights through gradient descent. The model projects the final values (57th) using the updated feature matrix, and the MSE loss between the final predicted and actual values is computed. Then, the gradients of the loss in relation to the model parameters are calculated and used to update the weights via stochastic gradient descent (SGD). Let θ_J symbolize the weights after task-specific training for each SKUs (J), η is the learning rate, and L_J represents the loss function for task j, then the task-specific adaptation can be depicted as:

$$\theta_j^{(k)} = \theta_j^{(0)} - \eta\nabla_{\theta_j} L_j(\theta_j^{(0)}) \tag{2.5}$$

3. **Meta-Update**: Following task-specific training, the global weights which were defined by *He initialization* are updated by interpolating between the initial and task-specific weights where β is the meta-learning rate. The outcome of the second phase of training is a refined model which is then used to predict the final values using both raw historical features and learned intermediate outputs. Once the prediction is made, we then evaluate the performance of our proposed meta-learning framework using out-of-sample data and compare it to benchmarks based on traditional forecasting methods. We employ the Mean Absolute Scaled Error as a measure of forecast accuracy to provides comparable results across various uni-variate time-series.

2.3 Results

2.3.1 Performance Measures

We evaluate the performance of our proposed meta-learning framework using out-of-sample data and compare it to benchmarks based on traditional forecasting methods. We employ the Mean Absolute Scaled Error (MASE) [30] as a measure of forecast accuracy. The Mean Absolute Scaled Error (MASE) is a scale-invariant metric, which provides comparable results across various uni-variate time-series datasets. MASE is calculated

as follows:

$$\text{MASE} = \frac{1}{I}\sum_{i=1}^{I}\left|\frac{x_{i,t} - \hat{x}_{i,t}}{\frac{1}{I-1}\sum_{t=2}^{T}|x_{i,t} - x_{i,t-1}|}\right| \tag{2.6}$$

Here, $x_{i,t}$ represents the actual value of the i^{th} time series at time step t, and $\hat{x}_{i,t}$ represents the forecasted value for the same time series and time step. The variable T denotes the total number of observations within a single time series. The numerator of the MASE formula calculates the Mean Absolute Error (MAE) of the forecast model, while the denominator computes the Mean Absolute Error of a naive forecast, which uses the previous value as the prediction ($\hat{x}_{i,t} = x_{i,t-1}$).

A MASE value below 1 indicates superior performance of the model compared to the naive forecast, while a value above 1 suggests that the naive forecast is more accurate. For time-series with intermittent demand, where many values could be zero, conventional error metrics can be misleading due to high variance and sparse observations. In such cases, MASE remains reliable error measure as it allows to evaluate even for sparse time-series. Moreover, in the presence of high fluctuations (volatility) in time-series, metrics like RMSE can overemphasize large errors. By using MAE in its calculation, MASE provides a balanced view of model performance, despite noise, are not unfairly penalized.

In our experiment, we analyzed MASE across various product groups using two-stage reptile algorithms. As indicated in Table. 2.3, the reptile algorithm demonstrated superior performance in 9 out of 10 product categories included in the experiments. The variability in MASE values within each product group was also lower for products where the two-stage reptile algorithm provided stable forecasts. For example, in the Hypodermic product group, comprised of 12 time-series, the MASE was 0.059 - the lowest of all benchmarked methods. The standard deviation within the 12 SKUs was also observed to be the lowest.

The Principal Component Analysis (PCA) technique is widely used for analyzing the differences between data points that are far apart, rather than the similarities between those that are close. This aspects of the PCA can be useful for our research to evaluate whether the extracted features are different from the original series. For instance, if the features result in isolated or highly dense clusters that are significantly distant from the actual time-series, it could signal an insufficient representation of the actual time-series. It might also suggest that the defined features have not effectively captured the underlying characteristics of time-series. In this chapter, we employ PCA to visualize the dissimilarities between the original time-series and the extracted features in a two-dimensional space in order to investigate whether the variations in feature distribution characteristics reflect what was observed in the original time-series. Since our meta-learner is initially trained on the extracted features, these features must reside within the original time-series data space. This is the reason why we initially compute the principal component projection using the features and subsequently project the time-series to the same low-dimensional feature space. If many features fall outside the space covered by the original time-series, a new meta-learner needs to be trained. The projection also allows us to understand the overall structure of the different collections' locations. To this end, both features and continuous vector of series are standardized using a Z-scale transformation before PCA application. We then calculate the principal components using time-series and project features into a two-dimensional space defined by the first two eigenvectors. In our analyses, the first two principal components account for 92.8 percent and 64.27 percent of the total variations for the time-series and features data, respectively.

2.3.2 Dissimilarity between two datasets

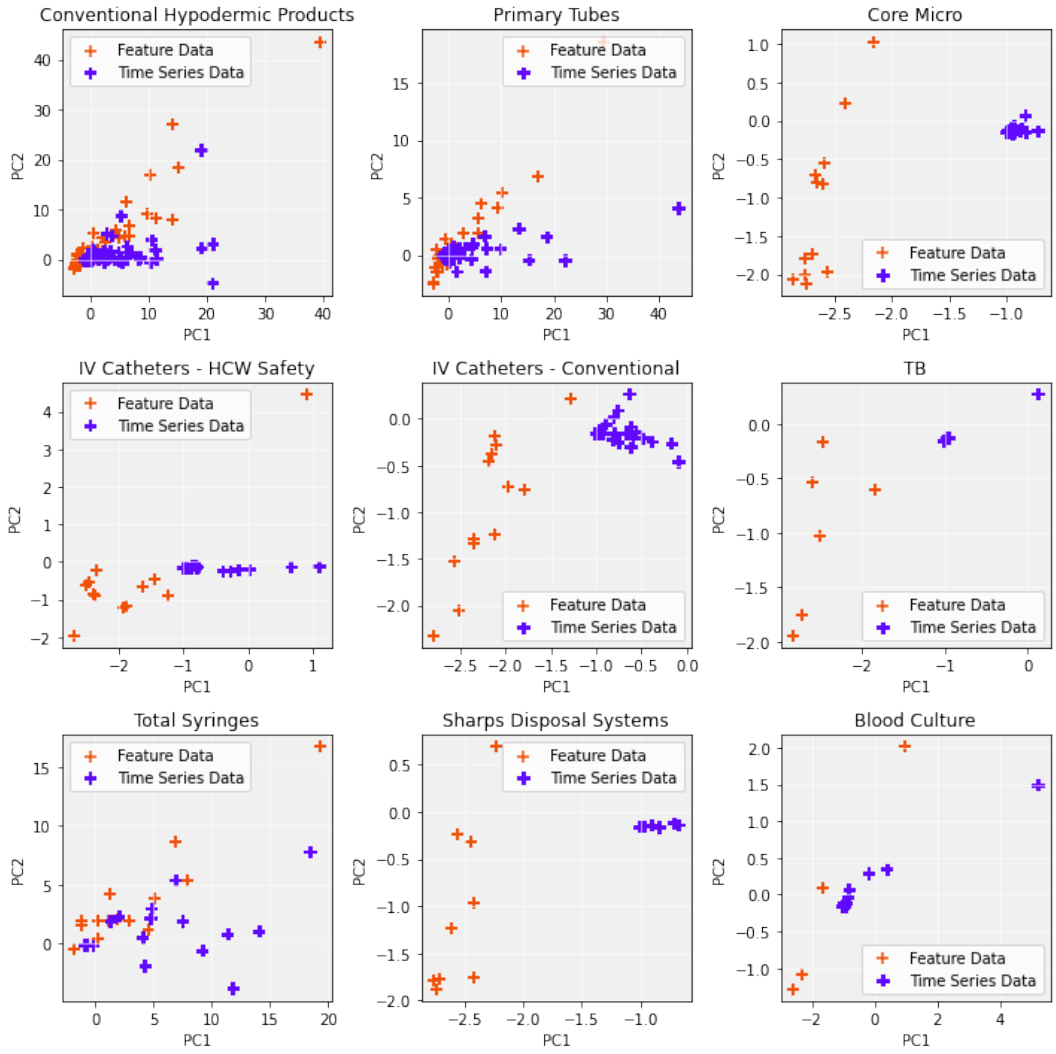

Figure 2.4: The distribution of statistically significant, extracted features with a p-value greater than 0.05 in the PCA space is displayed. Each subfigure uses dark orange to represent significant features and blue to indicate the original historical demands for time-series. The PCA space is calculated from all dimensions of statistically significant features used for training, with the original historical demands time-series then projected into this two-dimensional PCA space. Again, most product categories for forecasting are within the space of the extracted feature set, except for a few product categories. It should be noted that IV Catheters and TB products have a relatively lower number of samples

We also observe that in Fig. 2.3, the features' distribution (represented by dark orange dots) conspicuously nests and fills the continuous time-series space, despite being independent from the original time-series. This observation aligns with our theoretical expectation that the features can capture underlying characteristics. This is crucial because our two-step Reptile framework is trained based on the features and its subsequent output across the two stages. Therefore, the model is valid over the original time-series' space. We also observed a few product categories fall outside the space covered by the original time-series set in the first

Table 2.3: MASE and standard deviation of MASE values across different predictions for each product category. The standard deviation of MASE is computed at the product category level. The table only displays product groups with series that have a count greater than 3.

	Hypodermic	**Core Micro**	**ID/AST**	**Industrial Micro**	**IV Convention**
# Series	12	51	84	85	14
Two-step Reptile-Feature-Based	**0.059 (±0.003)**	**0.080 (±0.008)**	**0.055 (±0.001)**	**4.341 (±16.668)**	**0.137 (±0.005)**
Random Forest Regressor	3.660 (±0.033)	2.986 (±0.594)	3.608 (±0.243)	2.179 (±1.003)	1.100 (±0.033)
XGBoost	3.907 (±0.636)	3.201 (±0.261)	3.825 (±1.075)	2.295 (±0.033)	1.196 (±0.002)
Linear Regression	3.872 (±0.025)	3.144 (±0.658)	3.796 (±0.261)	2.253 (±1.087)	1.117 (±0.037)
Ridge Regression	3.816 (±0.029)	3.087 (±0.650)	3.772 (±0.258)	2.255 (±1.066)	1.126 (±0.038)
Decision Tree Regression	3.847 (±0.024)	3.141 (±0.648)	3.794 (±0.258)	2.299 (±1.038)	1.137 (±0.054)
Gradient Boosting Regressor	3.897 (±0.025)	3.188 (±0.636)	3.815 (±0.261)	2.293 (±1.066)	1.195 (±0.036)
	IV Catheters	**IVA Extension**	**Molecular**	**MolecularMax**	**Wingsets**
# Series	7	5	12	5	30
Two-step Reptile-Feature-Based	**0.115 (±0.012)**	**0.131 (±0.043)**	**0.080 (±0.001)**	**0.648 (±0.023)**	**0.090 (±0.016)**
Random Forest Regressor	1.149 (±0.018)	1.477 (±0.777)	2.141 (±0.035)	0.891 (±0.007)	1.907 (±0.415)
XGBoost	1.250 (±0.025)	1.584 (±0.810)	2.277 (±0.036)	0.865 (±0.007)	2.043 (±0.422)
Linear Regression	1.176 (±0.015)	1.529 (±0.823)	2.189 (±0.035)	0.852 (±0.008)	1.966 (±0.434)
Ridge Regression	1.182 (±0.017)	1.527 (±0.816)	2.125 (±0.039)	0.876 (±0.006)	1.917 (±0.423)
Decision Tree Regression	1.189 (±0.015)	1.576 (±0.799)	2.194 (±0.037)	0.911 (±0.005)	1.972 (±0.425)
Gradient Boosting Regressor	1.250 (±0.024)	1.584 (±0.805)	2.264 (±0.037)	0.882 (±0.007)	2.036 (±0.420)

two principal components due to the deep variability of time-series and fewer samples.

2.3.3 Experiment with different forecasting models

We evaluate the performance of our proposed two-stage reptile algorithm using out-of-sample data against benchmark methods that do not take extracted time-series features (non feature-based approach). In contrast to the two-stage reptile algorithms — where the first stage is pre-trained using extracted features only, and the second stage uses training based on historical real-time series — the benchmark method relies solely on the historical real value of the series. We calculate the Mean Absolute Standard Error (MASE) to compare the performance of our proposed method with the benchmark methods, and compute the standard deviations of the MASE for further investigation. The benchmark techniques used in this research are presented in Section 1.4.

As shown in Table. 2.3, the outcomes from the two-step Reptile frameworks proposed in the current research shows an improvement over the benchmark techniques employed in this research. We evaluate all predicted outcomes for each SKU which is out of sample using a consistent forecast horizon, one step ahead forecast. We chose to evaluate performance with the consistent forecast horizon, as with most studies in time-series forecasting employed a fixed horizon for performance evaluation. As a result, a consistent forecast horizons allows for easier comparison of prediction outcomes across various benchmarked methods.

2.3.4 Evaluating Forecast Outcomes Through Statistical Significance Tests

To evaluate whether predictions made by the two-stage Reptile frameworks are statistically significant, we employ two statistical significance tests: the paired t-test and the Wilcoxon signed-rank test.

Paired t-Test The paired t-test is used in many cases to determine significant differences in the means of two interconnected groups. It works on the assumption that the discrepancies between pairs follow a normal distribution [52]. When considering our predicted results $\hat{x}_{i,t}$ and the actual observations $x_{i,t}$, we employ

the paired t-test to verify if the average error between the predicted and actual figures deviates significantly from zero. For instance, in t-test $= \frac{\bar{d}}{\frac{s_d}{\sqrt{n}}}$, $\bar{d}$ is the mean of the differences between the predictions and actual values which is computed by $\bar{d} = \frac{\sum_{i=1}^{n}(\hat{x}_{i,t} - x_{i,t})}{n}$ where n is the total number of paried observations. Here, s_d is the standard deviation of the differences that follows:

$$s_d = \sqrt{\frac{\sum_{i=1}^{n}(\hat{x}_{i,t} - x_{i,t} - \bar{d})^2}{n-1}} \tag{2.7}$$

For the t-test, the null hypothesis is (H_0) the mean of the differences is zero, whereas the alternative hypothesis (H_1) is the mean difference is not zero which is as follows:

- $H_0 : \mu_d = 0$
- $H_1 : \mu_d \neq 0$

Table 2.4: The paired t-test and Wilcoxon Signed-Rank test indicate that the mean differences between the predictions made by the two-stage Reptile and the actual values are statistically significant (p-value < 0.05). Please note that the p-value is rounded to the fourth decimal point

Test	Test statistics	P-value
T-test	-4.554	0.001
Wilcoxon Signed-Rank Test	20329	0.001

Our experimentation reveals in Table. 2.4 t-test statistic of -4.55396 from the paired t-test by a p-value of less than 0.05. Given that the p-value <0.05 threshold, we dismiss the null hypothesis, thereby concluded with confidence that there's a substantial statistical difference between predictions by two-stage Reptile frameworks and the actual observations.

Wilcoxon Signed-Rank Test

The Wilcoxon signed-rank test serves as a non-parametric alternative to the paired t-test. It disregards the normality of the differences between pairs and is particularly suited for data sets where the differences aren't normally distributed [67]. The Wilcoxon signed-rank test first compute the differences between predictions ($\hat{x}$) and actuals (x). It then assign ranks based on the absolute values of differences while ignoring zero differences. For instance, positive ranks (R^+) is assigned if the sum the ranks of pairs is $\hat{x}_i - x_i > 0$, whereas it takes negative ranks (R^-) if the sum the ranks of pairs where $\hat{x}_i - x_i < 0$. It then calculate the test statistic $W = \min(R^+, R^-)$ and decides whether to reject the null hypothesis or not:

- H_0 : The median difference between prediction and actual is zero
- H_1 : The median difference is not zero

When we apply the Wilcoxon signed-rank test, we obtain a test statistic of W=20329 and a p-value of 0.000109. As the p-value is significantly below the 0.05 benchmark, we reject the null hypothesis and infer a notable statistical discrepancy between the forecasted and real values as shown in Table. 2.4.

2.4 Conclusion

The study introduces a two-step meta-learning framework for feature-based time-series prediction. Initially, this framework predicts intermediate output (continuous value) from the extracted features (a mix of binary

and continuous features), and then uses it co-variate, along with historical values, to make final predictions in the second stage. Our introduced two-stage Reptile frameworks outperforms standard forecasting methods commonly used in time-series prediction research.

Our findings indicate that carefully selected time-series features with statistical significance can provide valuable input for the training phase, as they can capture the underlying characteristics of each uni-variate time-series. This is particularly true when the sample size is small and the data patterns are difficult to discern in visual inspection or basic statistical tests. This method of feature extraction and selection techniques demonstrated in this chapter can provide alternative solutions for forecasters who want to integrate systemic characteristics of time-series in their forecasting models, which could impact forecast results.

A promising future extension of this research is its application to generate probabilistic forecasts. Generally, a model's performance greatly relies on the dataset used for training. We explored the potential of the Reptile approach to improve predictive accuracy by diversifying feature space. This approach is advantageous when there is a small sample size to construct a reliable prediction model or when shared characteristics or patterns among the time-series of interest are not evident with conventional statistical parameters such as seasonality or trends. However, some may argue that the co-variate - the output of our initial training phase based on binary and continuous features - could introduce noise, and learning models of the second phase might be burdened with fitting this unnecessary input component. However, in our experiment, we carried out a feature selection process, filtering features based on statistical significance to ensure only salient features are included before applying the feature matrix to our Reptile algorithm. Moreover, we analyzed the feature distributions in the 2-D Pincipal Component space by overlaying the original time-series to verify whether the extracted features accurately represent the original time-series.

In our experiment on the real world time-series, the two-phase Reptile algorithm demonstrates the highest accuracy as features are projected onto a 2-D PCA space aligning with the original time-series. This is particularly noticeable for both Conventional Hypodermic and Wingset product lines. However, due to the random walk drift often seen in time-series data, this observation does not apply to all instances. Hence, co-variate can serve as highly informative features in forecasting as it could remove noise involved in the massive feature matrix. The future work can also explore techniques to more dynamically express the co-variate from the feature space such as constructing rolled sub-series to implement cross validation.

We observed that the accuracy of the Two-Stage Reptile model decreases for parts of the time-series when the calculated features are not representative of the original time-series because of the inherent, random noise in time-series, such as random walk drift. Given that noise contributes to forecast uncertainty, co-variate can also be selected based on the errors that best exhibit the linear and non-linear aspects. Given the current method employed in this research generates co-variate based on past events, future research could consider using error-derived uncertainty measures as co-variate.

The current implementations are limited to the real-world historical demand from medical product manufacturers who funded this research. As a result, the application of the feature space and the two-stage Reptile algorithm space discussed in this chapter is restricted to time-series with similar features. Expanding these frameworks to different datasets from various application areas is a crucial research direction. Additionally, when considering extensions of our proposed framework for different forecasting applications, the feature space should be revised to include domain-relevant features that capture the characteristics that matter the most.

Chapter 3

Investigating cross-learning time-series forecasting with structural similarities

Abstract

Cross-learning approaches in time-series forecasting allow models to learn from multiple series. Clustering techniques group individual series based on their similarities and simultaneously learns the forecasting functions. Previous clustering approach pairs a specific distance metric for time-series. However, methods based on distance-based clustering often fail to account for the similarities among sub-sequences within individual time-series, which may not provide accurate comparison of the time-series. In this chapter, we suggest feature-based cross-learning forecasting methods that work on the structural similarities of individual time-series which consists of two phase. In the first stage, time-series feature extraction algorithms are applied to characterize the structural patterns of time-series, which is followed by conventional clustering algorithms to group time-series with inherent similarities. The resulting clusters are used to represent time-series with similar features and group them in the same dimensions within the matrix. In the second stage, data pooling technique is employed to construct lag-embedded matrices based on similar series defined by clustering methods. This is then applied to various forecasting techniques we constructed. Our proposed framework is tested using the real world time-series data and is compared against various of other methods that define similarity. Our approach provides comparable performance to other benchmark methods but at greater interpretability of forecasting performance due to understandable features, which is crucial factor for decision-making support. We also provide valuable perspectives on which prediction models are likely to perform better for specific types of time-series and how various underlying factors influence the forecasting performance.

Keywords

- time-series clustering

- Feature-based clustering
- Cross-series training
- Global forecasting

3.1 Introduction

3.1.1 Background

Most real-world forecasting problems involves predicting future time points of numerous time-series simultaneously. In the pharmaceutical supply chain [74], a drug manufacturer is required to predict the future demands for numerous products. For many years, the conventional data-driven forecasting approach has considered this problem as a separate uni-variate forecasting problem [36]. The uni-variate technique has become the standard framework for solving time-series prediction problems, and researchers and practitioners have focused their efforts on modeling an individual time-series [57]. However, the limitations of the local method has been evident in practice especially in cases where data sample sizes are small, leading to lower-than-expected forecast accuracy [4]. When the sample size is small, the local method faces challenge in learning functions from individual time-series. Therefore, many efforts have been made by researchers to incorporate prior knowledge into the uni-variate technique, which requires excessive manual work and expert supervision, limiting scalability.

To overcome the limitations of the local method, a new forecasting method that works on a group of time-series has been introduced [27, 4, 74, 44, 63, 53, 70, 20]. This approach "borrow" information from other time-series making it more effective method when the sample size is small. The time-series grouping mechanism has been referred by various names (cross series, multiple series, joint series), the latest grouping approach is recognized as a global method. The global method involves pooling all time-series of interest and learning a single prediction function, resulting in less sensitivity to over-fitting than its local counterpart. The global method, however, assumes that all time-series must be originated from the same source, and thus homogeneous [44]. Based on this assumption, the global method estimates a single function by pooling all series together. Such assumption of the global method can weaken the justification for selecting the global method over its local counterpart. For example, the idea that time-series are homogeneous because they originate from the same data generation process can be interpreted in different ways. Furthermore, forecasters who are responsible for developing forecasting models may be unaware of the data generation process, leading to uncertainty when using a global method.

3.1.2 Motivation and Our Approach

To address the above-mentioned problems, numerous studies have explored the idea of grouping time-series [3, 49, 18, 23]. However, these studies primarily benchmark against traditional naive methods, and there is a lack of research evidence showing which techniques and systemic characteristics of time-series impact the performance of the pooled data approach based on the similarity. Additionally, it's not clear whether grouping series based on similarity would outperform a naive global forecasting method that pools data from all series, due to a lack of comprehensive evaluation. In our research, we propose a method that involves extracting time-series features using the *TSFRESH* algorithm [14] to characterize each series and groups with those interpretable feature sets. This allows us to better understand the characteristics of subgroups

of series within the context of the joint-series (cross-series) forecasting methods. We then propose feature-based clustering techniques to create clusters from the vector of features. We build the cross-series forecasting functions within each cluster using linear and non-linear model classes, which will then evaluate cross-group average accuracy using a benchmark methods that considers all available time-series. Finally, we characterize the groups of series based on interpretable features to answer key research question: under what conditions does the cross-series forecasting perform well? The benefits of this work can be summarized as below:

- This work contributes to the Global Forecasting Methods (GFM) by providing empirical evidence regarding time-series relatedness. Although the GFM heavily relies on the assumption of homogeneity among all time-series of interest, verification of this relatedness assumption has been largely overlooked in the previous literature. In this chapter, we thoroughly evaluate the effects of time-series similarity within the GFM context. Our findings confirm the validity of the GFM's similarity assumptions based on empirical evidence.
- We propose a methodology called Cross-series Forecasting that works on structural similarities of time-series. This method focuses on the inherent characteristics of series expressed by feature extraction algorithms and introduces an interpretability that traditional time-series forecasting methods often lack. Our method develops understanding of the key factors in series that influence forecast results, leading to more actionable and insightful practices in time-series forecasting.
- Our work contributes to the fields of demand forecasting and time-series forecasting, particularly when dealing with real-world data of small sample sizes. We address this challenge by creating large matrices of lagged features using data pooling techniques. Each matrix is constructed based on the lagged observed values of similar series. These techniques have practical applications in time-series forecasting and our work can serve as a benchmark for further research.

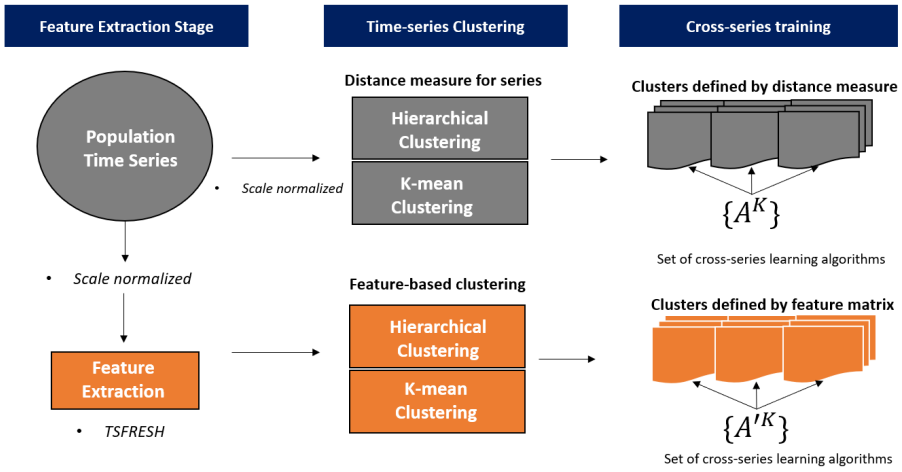

Figure 3.1: The implementation of cross-series forecasting frameworks involves three stages:(i) In the first stage, time-series feature characterization algorithms express underlying patterns from the time-series. (ii) In the second stage, time-series clustering techniques group series based on distance measure (value-based) and feature similarity (feature-based). (iii) In the final stage, a forecasting function fitted to each cluster is constructed and compared to global forecasting functions that fit all series.

3.2 Methodology

3.2.1 BD Logistics Datasets

The dataset used in this chapter details the secondary demands of consumable medical products sold by logistics partners of Becton Dickinson (BD). The raw data was collected directly from BD's logistics database which includes monthly sales quantities at the SKU level across 2,303 Stock Keeping Units (SKUs) and additional attributes such as product categories, geographical regions sold, time indices, and SKU numbers over a span of 57 months. To convert the time series forecasting into a supervised learning framework, each time step in the series was treated as a training sample, with the target variable being the demand value for the subsequent month. This mapping allows the dataset to be used in a supervised learning context. The key difference between the datasets used in the previous chapter and the current one is the type of demands they track. The prior product sales datasets track primary demands, which are products delivered and sold directly by the company and recorded in the company's finances. In contrast, the logistic datasets focus on secondary demands sold by the company's logistics partners and provide more detailed SKU-level product information. Both datasets serve different functions: the total sales data provides insights for the finance department, while the logistics dataset is key for inventory and distributor management from a supply chain standpoint. Although many series have zero values similar to the previous chapter, which could be due to low or no demand from the end customers, we did not exclude series with non-zero values during the data preparation stage in order to cater to the company's requirement for comprehensive forecasts.

To transform time-series forecasting into a supervised learning problem, we convert raw time-series data into labeled pairs of inputs and outputs. To create a tagged dataset for supervised learning, we follow these steps:

1. Feature extraction - We use the TSFRESH library to extract statistical, temporal, and frequency-domain features for each univariate time series. Detailed feature extraction techniques and attributes are explained in Section 3.2.2.

2. Feature-based clustering - We group SKUs with similar characteristics by applying clustering algorithms to the extracted features. Each SKU receives a cluster label, which is then used to create pooled series and define joint series, as detailed in Section 3.2.3.

3. Lag Embedding and Data pooling - Each time series is transformed into a lag-embedded matrix, where the previous n-time steps serve as input features, and the next time step serves as the target variable. For example, for an AR(6) model that looks back 6 time steps, the input features for time step t are $[x_{t-6}, x_{t-5}, \ldots, x_{t-1}]$, and the target variable is x_t. After creating the lag-embedded matrices, they are stacked together based on the clustering algorithms to create a large global training matrix, where each row represents a training sample. The detailed data pooling process is described in Section 3.2.4.

3.2.2 Feature generation for clustering

Our cross-series forecasting framework (Fig. 3.1 begins with time-series feature extraction. In this chapter, we followed the notion of time-series characterization for $x_{i,t} \in R^n$ which is i-dimensional time-series at t-th time points. In order to extract time-series characteristics from i dimensional time-series vectors consisting of t time steps, the time-series characterization function f_k is constructed using TSFRESH where k is the number of feature extraction algorithms that follows $(f_k(x_1), f_k(x_2), ..., f_k(x_I)$. Table. 3.1 displays the key set of features characterizing time-series of interests.

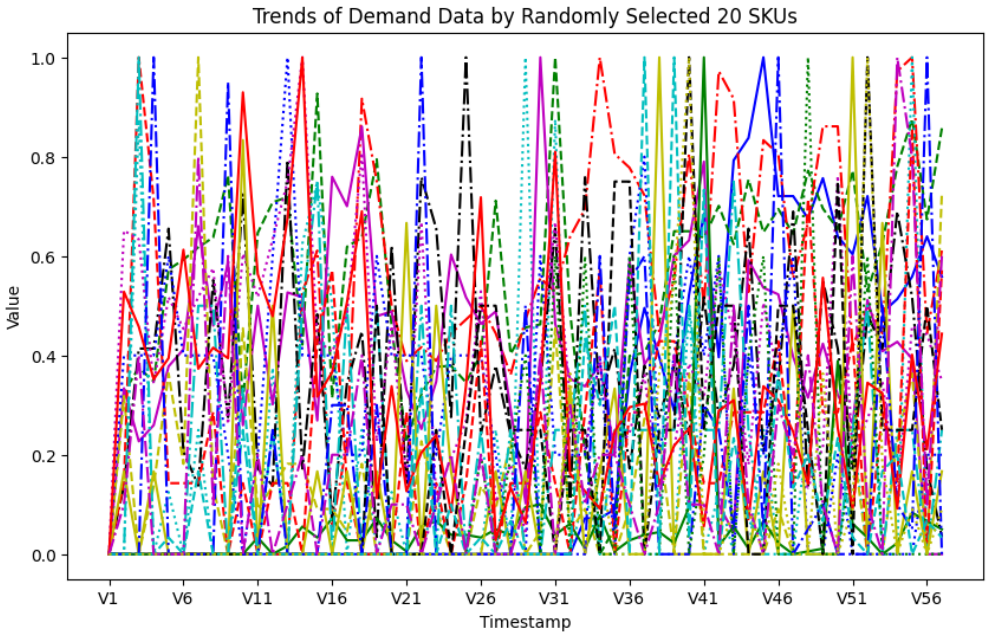

Figure 3.2: The line plots show 20 series sampled from the BD logistics datasets. The sales values are scaled to emphasize volatility and isolate scale effects. Visual inspection of these plots reveals that unlike conventional time-series data used in previous literature review, this dataset is highly fluctuating with no apparent trends.

3.2.3 Feature-based Clustering using K-mean

The second phase of cross-series forecasting method begins with time-series clustering, grouping all series according to their structural characteristics. We take into account both value-based and feature-based similarities during the clustering process, using value-based clustering as our benchmark. In the implementation of value-based clustering, each time-series x_i is directly used for clustering based on its values. The clustering aims to minimize the within-cluster variance, defined as:

$$\min \sum_{k=1}^{K} \sum_{x_i \in C_k} ||x_i - \mu_k||^2 \tag{3.1}$$

where C_k represents the cluster K is the total number of clusters, and μ_k is the centroid of cluster k. For feature-based clustering, time-series x_i are first transformed into a feature space $f_k(x_i)$ using l number of feature extraction algorithms using *TSFRESH* algorithm that follows $X_l = (f_1(x_1), f_2(x_2), ..., f_l(x_I))$. The clustering is then performed in the resulting feature space to group time-series based on the extracted features of series. The mathematical depiction of feature-based clustering is as follows:

$$\min \sum_{k=1}^{K} \sum_{f_l(x_i) \in C_k} ||f_l(x_i) - \mu_k||^2 \tag{3.2}$$

Table 3.1: Summary of key features

Feature Typology	**Identifiable Feature Name**	**Detailed Descriptions**
Basic Statistics	Presence of Variance	Indicates if the variance of the dataset is larger than the standard deviation, which can suggest a non-Gaussian distribution.
Energy Measures	Absolute Energy	The sum of the squares of the values, which corresponds to the energy of the signal.
Change Features	Mean Absolute Change	The average of the absolute differences between consequtive time-series values, which reflects volatility.
Central Tendency	Median Value	The median of the time-series data.
Statistical Features	Standard Deviation and Variance	Measures the amount of variation or dispersion in the time-series.
Shape Features	Skewness and Kurtosis	Reflects the asymmetry of the value distribution or the tailedness of the distribution of values in the time-series.
Signal Features	Root Mean Square	The square root of the mean of the squares of the values, often used as a measure of signal magnitude.
Trend Features	Linear Trend Slope	The gradient of a linear regression line fitted to the series, indicating upward or downward trends.
Autocorrelation Features	Autocorrelation Measure	Correlation of the time-series with a delayed version of itself at different lags.
Frequency Domain Features	FFT Coefficients	The coefficients obtained from the discrete Fourier transform which represents the time-series in the frequency domain.
Complexity Features	Approximate Entropy	Measures complexity of the time-series indicating the predictability of future values.

3.2.4 Data pooling

After clustering, the data is pooled from each cluster to construct training sets for forecasting models. In cross-learning time-series forecasting, models are cross-trained on a group of time-series defined by the previous clustering algorithms. To facilitate the application of learning functions for forecasting models, a data pooling technique is employed. In supervised learning of time-series forecasting, each series is lag-embedded into a matrix, and these matrices are then stacked together to create C_k metrices, defined by the clustering algorithms, achieving data pooling. For instance, for a uni-variate time-series $x_{i,t}$, the series is lag-embedded up to a certain auto-regressive order l, which involves creating lagged versions of the series up to l time steps back:

$$X_{i,t} = [x_{i,t}, x_{i,t-1}, \ldots, x_{i,t-l+1}] \tag{3.3}$$

Here, $X_{i,t}$ is a vector containing l lagged values of $x_{i,t}$. The lag-embedded series $X_{i,t}$ can be represented as a matrix:

$$X_i = \begin{bmatrix} x_{i,l} & x_{i,l-1} & \ldots & x_{i,1} \\ x_{i,l+1} & x_{i,l} & \ldots & x_{i,2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{i,T} & x_{i,T-1} & \ldots & x_{i,T-l+1} \end{bmatrix} \tag{3.4}$$

where X_i is the lag-embedded matrix for series i, and T is the total number of observations in series i. By stacking the lag-embedded matrices, the cross-series learning method can leverage shared patterns and dependencies across grouped series of interests by learning a single forecasting function using the pooled information.

3.2.5 Forecasting models and classes

Our research uses a variety of forecasting models based on the principle of auto-regression. These models aim to minimize loss across a large, aggregated matrix, which is created by using lag-embedding on each series. In essence, we turn forecasting problems into supervised machine learning problems using past time-series observations, also known as lags. This approach allows us to draw data from multiple time-series to build a data matrix defined by clusters. We then apply forecasting models that are suitable for all sub-series within these clusters. This method helps forecasting functions tap into the collective dataset information. In this time-series matrix, the final column represents the target values, while the preceding columns serve as input features. Our focus is primarily to produce one-step-ahead forecasts, but this can also be extended to longer horizons through recursive forecasting. The benchmark techniques used in this research are presented in Section 1.4.

It's important to note that our modeling approach intentionally avoids the use of advanced techniques, including seasonality adjustments and complex neural network architectures. Instead, we use basic features to ensure our findings derive from the models' inherent predictive abilities, without any external enhancements. The key model parameters of the respective forecasting models employed in this work are presented in Table. 3.2.

Table 3.2: Choice of values for parameters for respective model classes.

Model	Parameters
Random Forest	n_estimators=300, max_features='sqrt', max_depth=5, random_state=18
XGBoost	objective="reg:squarederror", random_state=42, subsample=0.5, colsample_bytree=0.5, colsample_bylevel=0.5
Decision Tree	max_depth=5
Gradient Boosting	n_estimators=100, learning_rate=0.1, max_depth=1, random_state=0, loss='ls'
LSTM	epochs=10, batch_size=32

3.3 Experiments

Our cross-series forecasting process starts with clustering of series to recognize structural similarities. This is followed by constructing a lagged time-series matrix, which uses data pooling based on the defined clusters. In the clustering application, we increase the experiment's complexity by including different auto-regressive order parameters. This allows us to assess the impact of the auto-regressive order on each clustering performance, helping us determine the optimal clusters. The final prediction stage uses a one forecast horizon, drawing from a pooled dataset of various time-series and an array of forecasting models, which will be detailed in the next section. The experiment follows these steps for each series and model combination:

3.3.1 Characterizing time-series

As defined in the section 2.4, in order to extract time-series characteristics from i dimensional time-series vectors consisting of j time steps, the time-series characterization function f_k is constructed using TSFRESH

where k is the number of feature extraction algorithms that follows $(f_1(x_1), f_2(x_2), ..., f_k(x_i)$. The feature extraction algorithm [14] offers automatic methods to extract thousands of features. In the previous chapter, all time-series features provided by TSFRESH by default were used, followed by the application of statistical techniques such as feature significance testing to filter out insignificant features. Instead, this study takes subject matter expertise (SME) based approach in the supply chain to determine the significant variables. For this purpose, we conducted interviews with five supply chain experts of Becton Dickinson to identify commonly used variables for predicting the demand for essential medical products during the Sales and Operation Planning process (SOP). Next, the feature extraction algorithms have been applied to generate time-series features relevant to 11 typologies defined based on the SME inputs, which is presented in Table. 3.1. The resulting feature matrix would have $i = 2303$ rows and $k = 457$ features with each rows and columns corresponds to 457 underlying characteristics extracted from each 2303 uni-variate time-series. After constructing the feature matrix, as explained in the previous chapter, we refine the feature matrix by applying feature significance testing and feature selection. In the phase of evaluating feature importance, we individually assess each feature for its significance in making predictions based on the target variable. Suppose we represent the set of feature vectors for uni-variate time-series x_i as $f_k(x_i)$ where each feature k is either binary or real-valued. The significance of each feature is determined using suitable statistical tests, which provide a p-value that signifies the relevance of each feature. The detailed feature significance testing for each binary and continuous feature is outlined in section 2.2.4.

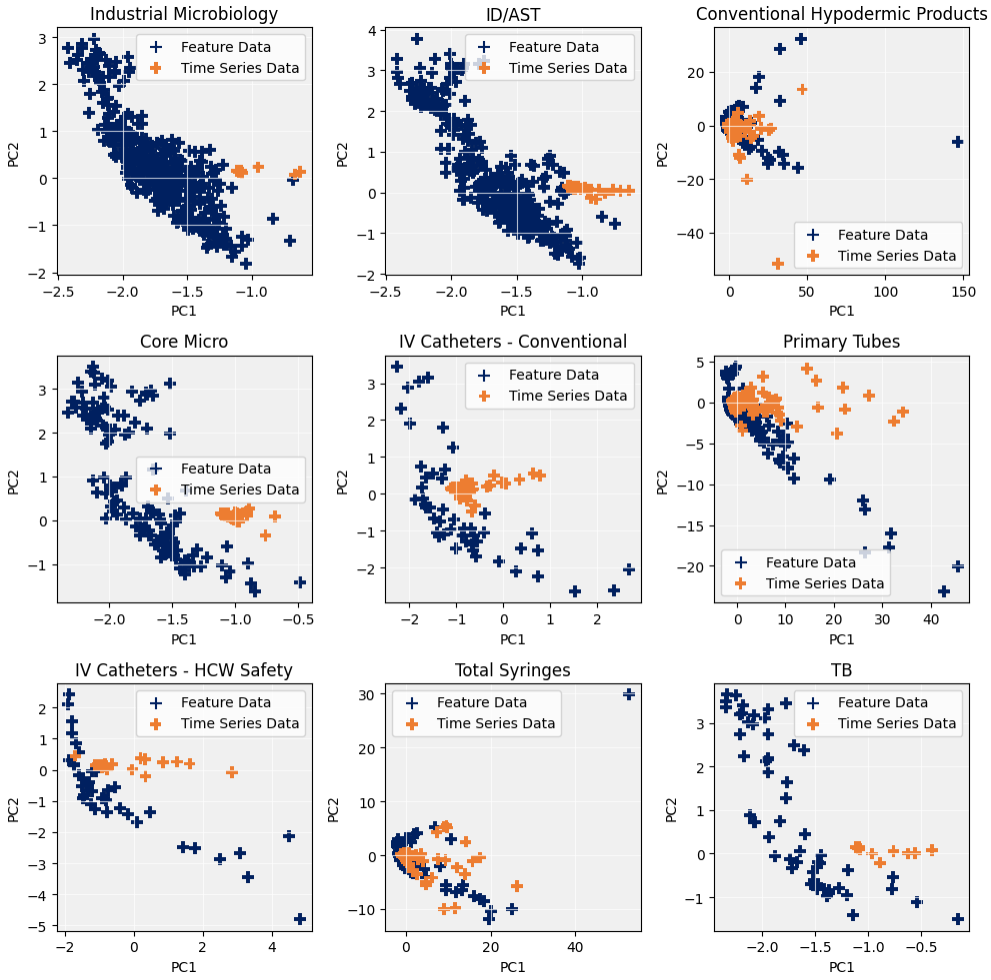

Figure 3.3: The distribution of statistically significant features with a p-value (< 0.05) in the PCA space. Each sub-figure uses dark orange to represent significant features and blue to indicate the original historical demands for time-series. The PCA space is calculated from all dimensions of statistically significant features used for training, with the original historical demands time-series then projected into this two-dimensional PCA space. Again, most product categories for forecasting are within the space of the extracted feature set, except for a few product categories

After implementing the feature significance test and removing features based on pre-determined significance thresholds ($\alpha = 0.05$), the test results eliminated 54 features from the initial set, leaving a final feature space with 403 features [2303, 403]. Fig. 3.3 illustrate how well the final feature matrix which is selected through respective significance testing are truly representative of the original time-series data. In case where forecasters need to deal with high volatile time-series data like the one in the current study, whose basic statistics fail to capture any known patterns, carefully designed time-series characterization approach can be used to simulate the original time-series and compare similarity.

Table 3.3: Clustering metrics calculated from K-means and hierarchical clustering applied to the feature matrix: (i) K-means Silhouette: A high silhouette score suggests that clusters are compact and well-separated. (ii) Sum of Squared Distances (SSD): Measuring the compactness of clusters, lower values means more compact clusters. However, SSD typically decreases as the number of clusters increases because more clusters cover the data space, even if these clusters may not be significant. (iii) The silhouette score was also calculated from hierarchical clustering.

Clusters (K)	**KMeans Silhouette**	**KMeans SSD**	**Hierarchical Silhouette**
2	0.936	700194	0.936
3	0.938	624329	0.931
6	0.199	468322	0.212
12	0.088	349807	0.101
17	0.077	308580	0.068
22	0.056	270723	0.068
27	0.061	248651	0.053
32	0.051	235949	0.031

3.3.2 Clustering

The number of clusters in specific applications may not always be known beforehand. However, for the K-means clustering algorithm, the number of clusters should be predetermined. In our case, we have 65 product groups, where each stock keeping unit (SKU) is an individual uni-variate series. As each SKU (a lower hierarchy of product groups) follows the behaviors and trends of higher group products, we aim to find 65 clusters. We first apply the K-means clustering algorithm to both the original time-series (value-based clustering) and feature matrices (feature-based clustering) with varying numbers of clusters. We then select the best solution using the Sum of Squared Distances (SSD), a commonly used metric to evaluate the number of clusters in K-means clustering. The SSD for K-means clustering is defined as follows:

$$\text{SSD}(k) = \sum_{i=1}^{k} \sum_{x'_i \in X'_i} \|x'_i - \mu_i\|^2 \tag{3.5}$$

where k is the number of clusters ranging from 2 to 33, x'_i represents a feature vector corresponding to i uni-variate time-series, and μ_i is the centroid of cluster i, calculated as the mean of all feature vectors in x'_i. As shown in Table. 3.3, the Sum of Squared Distances (SSD) decreases as the number of clusters increases due to less variance within each cluster. However, this doesn't necessarily mean better clustering. At its extreme, the lowest SSD would be achieved when each data point is its own cluster, which is not useful[24]. Next, we proceed to hierarchical clustering, focusing on the Silhouette Score. This metric measures both the cohesion and separation of clusters. A high silhouette score indicates compact, well-separated clusters. In Table. 3.3, it is noted that the silhouette score for 2-3 clusters is close to 1. High silhouette scores with fewer clusters suggest that the data points fit well into fewer clusters, which are also well-separated from each other. This indicates that the clustering structure is strong and meaningful with fewer clusters.

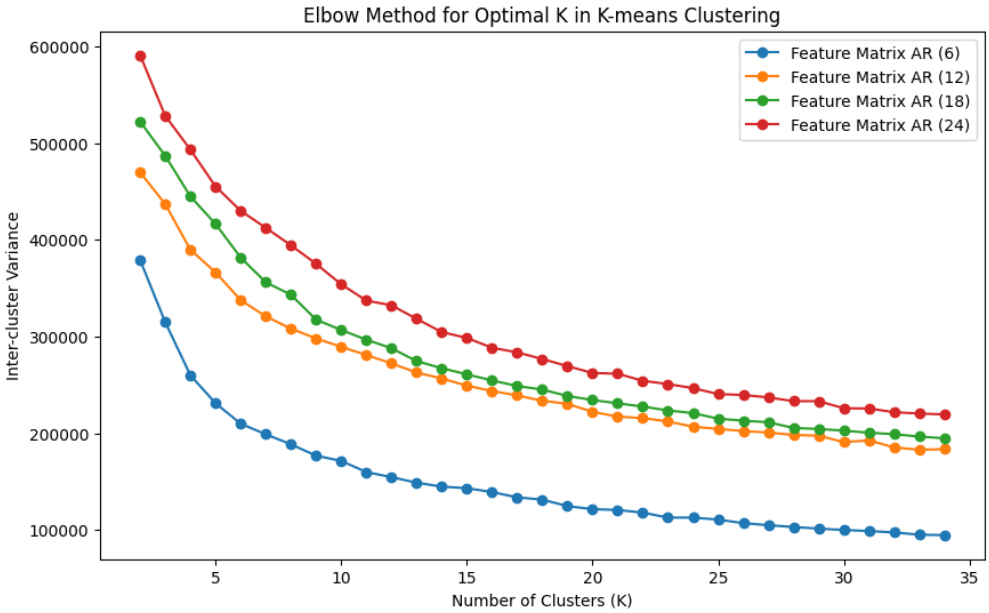

Figure 3.4: An elbow plot in K-means clustering is conducted based on time-series characteristics from the original series. Different auto-regressive orders are experimented with before characterizing the time-series. For instance, the AR(6) method takes the latest 6-month time-series, characterizing recent trends, compared to AR(24), which has longer lookback periods.

Based on the key metrics involved with clustering, Fig. 3.4 is a graphical representation of K-means clustering, known as an elbow plot, which is used to identify the optimal number of clusters. The elbow plot method typically has the number of clusters (K) on the x-axis and the inter-cluster variance (also known as inertia) on the y-axis. Lower values of inter-cluster variance indicate that data points within a cluster are closer to the cluster center, generally signifying better K-means clustering quality.

In our experiments, we drew different elbow plots by adjusting the auto-regressive order (input data range) used for clusters constructed based on the feature matrix derived from the original time-series. It is noted from the elbow plots that for all different auto-regressive orders, the elbow point is observed at 2-3 for shorter auto-regressive orders (AR6 and AR12) and 5-6 for longer auto-regressive orders (AR18 and AR24). These points are where the rate of decrease in inter-cluster variance sharply slows down, aligning with the previous findings using K-mean Silhouette method, where the optimal cluster is identified at cluster 3 when all auto-regressive orders are used for analysis without time-series segmentation.

Although clustering analytics provide insights into the range of clusters, we did not limit our cross-series forecasting experiments to just the optimal number of clusters (2-6) found in different time-series segmentations. Instead, we considered various cluster parameters, ranging from 2 to 33, to observe the impact of different cluster combinations on feature-based cross-series models.

After applying clustering techniques with varying parameters to both the original time-series and feature matrix, we move on to the cross-series training and prediction stage. This stage involves training models on data from each cluster. Each forecasting model, defined in the previous section, estimates a single function from a group of series in each cluster. With joint series training by clusters, the model can learn specific

patterns and trends within the time-series of each cluster group. We also construct a Global Forecasting Model that learns a single function from all the time-series of interest used in this study as a benchmark. The forecast horizon tested is one time-step ahead, and the sMAPE forecast metric is primarily used to compare the accuracy and performance of different approaches.

To assess whether the feature-based cross-series forecasting approach improves the forecast, we also trained a distance-based cross-series forecasting approach using all the model classes defined in section 2.4.2, including linear and non-linear modes. We incorporated all these model classes into the global forecasting approach to facilitate fair comparisons among various forecasting techniques. In these experiments, we added complexity by adjusting the number of auto-regressive orders used for clustering the series. We also experimented with different cluster settings and monitored the performance change in forecasting results by gradually increasing the number of clusters. For each technique, we calculated the median forecasting errors using the chosen error metrics (sMAPE) from 8 model classes.

- **Experiment 1**. We controlled the segments of series used as inputs into the K-means clustering algorithm for both the feature-based and distance-based clustering approaches. We assumed that the range of series used for clustering would impact the definition of series similarity, so we included this as a key factor in our experiment.
- **Experiment 2**. In addition to variations in input clusters, we also simulated different numbers of clustered groups using K-means. We experimented with a range of clustered groups from 3 to 35, increasing by 2 groups at each stage of the experiment.
- **Experiment 3**. We used 8 model classes known to capture either linearity or non-linearity, as introduced in the previous section. Each forecasting approach experimented with the 8 sets of models. For instance, feature-based cross-series training incorporated 8 forecasting models, and the results were computed using the median sMAPE of each forecasting model for all the series.

3.4 Forecasting Results

We tested the performance of our proposed cross-series forecasting framework using out-of-sample data. We compared it to benchmarks based on cross-training that employs distance-based similarity and global forecasting methods. To measure forecast accuracy, we used the symmetric Mean Absolute Percentage Error (sMAPE), which is known to handles outliers better than MAPE. This is because the sMAPE formula's denominator helps stabilize the metric when actual or predicted values are very small or close to zero which is commonly observed in the dataset used in this study. The sMAPE can be mathematically represented as follows:

$$\text{sMAPE} = \frac{100\%}{n} \sum_{i=1}^{I} \sum_{t=1}^{T} \frac{|x_{i,t} - \hat{x}_{i,t}|}{(|x_{i,t}| + |\hat{x}_{i,t}|)/2} \tag{3.6}$$

In this context, $x_{i,t}$ denotes the observed value of the i^{th} series at time t, while $\hat{x}_{i,t}$ stands for the predicted value for the same series. We compare the forecasting performance of cross-series methods and global methods. The main difference between these two frameworks lies in how each model's learning function is configured. In cross-series training, functions are learned from each defined cluster group. In contrast, global

methods learn a single function from all the time-series used in the experiment. According to Table. 3.4, the median sMAPE computed from all series by all forecasting models for the global method is 145.1. However, the cross-series approach yields a lower sMAPE of 135.6. Additionally, most results from the cross-series training do not exceed 140. This evidence suggests that grouping time-series that are functionally similar can outperform approaches that estimate from all series. This indicates that the global methods' assumption of series homogeneity is accurate, as grouping series improves forecasting performance. In other words, the cross-series approach, which "borrows" information from other series, can more effectively manage error volatility, making forecasts more accurate, especially for small sample sizes with random noise patterns.

Table 3.4: The median of the individual Symmetric Mean Absolute Percentage Error (sMAPE) values calculated over 2,303 time-series is shown. The cross-series method is further divided into feature-based and distance-based approaches. For each forecasting method, we examine eight models known for capturing both linear and non-linear trends. Each experiment also involves testing different cluster numbers and assessing the impacts of auto-regressive order on clustering inputs.

	Cluster Number									
Forecasting Framework	**3**	**5**	**7**	**11**	**15**	**19**	**23**	**27**	**31**	**35**
Cross-series-feature-AR6	142.3	143.8	145	140.9	140	**135.6**	138.6	138.2	137.4	141.6
Cross-series-feature-AR12	141.3	140.4	140.2	140.4	140.4	140.5	140.9	139.8	140.5	140.2
Cross-series-feature-AR18	143.6	145.1	142.4	143	147	144.3	140.8	140.7	141.5	**139.9**
Cross-series-feature-AR24	146	141.2	142	**141.8**	142.1	142.7	143.8	143.1	142.5	142.5
Cross-series-value-AR-6	138.4	140	133.7	136	134.4	134.8	132.6	132.9	**131.1**	133
Cross-series-value-AR-12	140.1	138.4	139.3	137.8	137.7	140.1	135.7	134.8	136.7	**133.4**
Cross-series-value-AR-18	142.5	141.2	141.3	140.8	140.6	136.5	137	138.9	141.8	136.2
Cross-series-value-AR-24	143.4	142.7	142.2	141.8	141.9	141.5	140.1	143.8	146	140.7
Global-All-Series					145.1					

Table. 3.4 also shows that the least errors are mostly observed with clusters beyond index 19. The sMAPE values on the right side of the table, which represents larger cluster numbers used in cross-series training, are generally smaller than those on the left side, which represents smaller cluster numbers. This contradicts our initial clustering analysis, where the optimal number of clusters for a feature-based approach was 2-3, as discussed in Table. 3.4. The difference in auto-regressive orders, which set the number of series used for clustering input, also shows variability. For feature-based cross-series forecasting methods in Table. 3.4, errors accumulate as we increase the number of series. This is also observed in value-based clustering methods. This suggests that auto-regressive and partial auto-regressive patterns exist in the dataset, as confirmed by significant features described in Table. 3.4.

3.5 Conclusion

Our research proposes a novel framework for feature-based, cross-series forecasting that leverages the similarity of time-series. As an automated forecasting framework, it uses similarity factors as key influences in generating forecasting performance. The results show the robustness of our approach compared to other recently emerged global methods in the forecasting community. The data used in this experiment is highly volatile, as described in the previous section, and these characteristics are not easily defined. We use time-series characterization algorithms and clustering techniques to find similarities among variations in series, which helps us understand the importance of series similarity in constructing learning functions from the combined time-series.

Our experiment also investigates the effect of lag feature quantity on the inputs for clustering, as highlighted in the previous section. Notably, a decrease in lag features correlates with a decrease in error rates. This suggests that incorporating distant past observations in the clustering process may add noise or reduce their relevance to the current data point, negatively impacting the concept of similarity or relatedness of series. This is a key assumption of the semi-global forecasting method. Our observations suggest that we should refine our clustering methodologies beyond conventional k-mean and hierarchical clustering algorithms. Techniques like Self-Organizing Maps (SOM)[16] and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [17] should be incorporated into our investigation. Our focus is to uncover a complete set of clustering techniques that work well in cross-series forecasting settings. This research is a foundational step towards improving the accuracy of demand forecasting in the medical device and pharmaceutical industries. Ultimately, we aim to propose an end-to-end selection mechanism for clustering strategies based on similar structures and memory components in time-series.

Chapter 4

Impact of Auto-regressive Order on Global Forecasting Performance: Empirical Evidence

Abstract

The accuracy of time-series forecasting largely depends on the 'memory' or auto-regressive (AR) order of models, which influences how historical observations shape future forecasts. Despite its importance, there is a lack of comprehensive studies on the role of auto-regressive order in the pooled data approach which stacks all series of interest together and estimates a single forecasting function, known as global forecasting model. This chapter aims to fill this gap in the global forecasting research by conducting experiments with popular M4 competition datasets. To this end, this research focuses on the variations in predictive performance with different levels of memory coefficients. It also monitors how increased memory complexity impacts the performance of global forecasting methods. In our empirical study, we suggest generalization memory bounds that produce optimal results in global forecasting models using M4 competition datasets. This could guide future research in finding the best auto-regressive orders for implementing global forecasting models.

Keywords

- time-series forecasting
- Global forecasting model
- Muti-time step forecasting
- Pooled regression

4.1 Introduction

4.1.1 Background

Successful multi-time step forecasting is important in most forecasting scenarios since the mid-to-long term predictions determine the practical effectiveness of those forecasting tools. They are also decisive factors for incorporating the method into the actual planning process. As more companies rely on algorithmic forecasting, various techniques have been explored, ranging from traditional uni-variate forecasting method relying on the statistical models to combined, hybrid models mixed of statistical method and advanced Artificial Neural Networks (ANN). A successful forecasting method can reduce the manpower needed for demand planning, increase workflow efficiency, prevent supply chain disruptions, and maintain a high level of demand visibility. In the world of data-driven decision-making, algorithmic forecasting has become significant for strategic planning in many industries. The ability to predict future demands, market trends, and risks can significantly influence a company's operational and financial success. For instance, the semiconductor industry, characterized by rapid technological advancements and cyclical demand patterns, relies heavily on multi-time step forecasting to navigate complex pricing and demand dynamics [42]. Similarly, the medical sector relies on accurate multi-horizon forecasting for inventory management, production scheduling, and uninterrupted essential medical and pharmaceutical device delivery [10]. These forecasts, often extending three to six months into the future, are crucial for companies aiming to optimize their operations and supply chain management.

Over the past decades, numerous time-series forecasting literature has been published across different sectors including semi-conductor, energy, retail, pharmaceutical to identify superior time-series forecasting methodologies. Since the 1970s, forecasting methodologies have typically relied on statistical methods, with Auto-regressive Integrated Moving Average (ARIMA) [21] and Holt-Winter's exponential smoothing [29] being the most common techniques used for forecasting time-series. The techniques developed during the early days of research are characterized as uni-variate forecasting. In this local method, each model parameter is estimated separately from the respective individual series. However, in many cases, the results did not meet expectations, mainly due to high fluctuations of series and the qualitative factors also affecting forecast outcomes [33]. In the 20th century, advanced computational methods introduced a new dimension to forecasting research. Tong [59], through his work on nonlinear time-series analysis, suggested that linear models like ARIMA were insufficient for capturing the complex dynamics of real-world data. He highlighted the importance of non-linear models in advancing time-series research.

Most recently, global forecasting methods, which use pooled time-series data to estimate a single predictive function, have gained attention due to their potential accuracy and computational efficiency. In the early stages, global methods were also referred to as cross-series or joint-series training, which involved learning from multiple time-series simultaneously. The initial investigation of the global method was conducted using the deep neural networks and referred to as DeepAR [53] which achieves significant improvements over traditional local counterparts. Subsequent study has emerged, emphasizing the computational efficiency and comparable performance of the global method [44].

4.1.2 Motivation

Although global forecasting methods theoretically provide considerable advantages over its local counterparts, there is a lack of empirical evidence that allows a comprehensive understanding of the factors contributing to

Table 4.1: Review of the literature on time-series forecasting with global forecasting methods

Authors (Year)	Forecasting algorithm	Forecast Horizon	AR order Evaluation
Bandara et al. (2020)	Recurrent Neural Networks (RNN)	Various, based on available data	Not mentioned
Chen et al. (2021)	DeepDGL (encoder-decoder architecture with convolutional and transformer layers)	Various, including long-term patterns	Not mentioned
Rovzanec et al. (2021)	Anomaly detection-based models (XAI)	One time step ahead (n=1)	Not mentioned
Godahewa et al. (2021)	Various classical methods including ARIMA and exponential smoothing	Varies, including daily NN5's given horizons	Various AR models are used, but not mentioned the AR impacts
Lopez et al. (2022)	Various classical regression models	One time step ahead (n=1)	AR order predetermined for clustering
Hewamalage et al. (2022)	Seasonal AR, Chaotic Logistic Map, Self-Exciting Threshold Auto-Regressive (SETAR)	Multiple horizons based on data	AR(3) and SAR(1)
Kontopoulou et al. (2023)	Classical time-series and machine learning methods such as linear regressions and XGBoost	One time step ahead (n=1)	Not mentioned
Ibanez et al. (2023)	time-series Transformers	Quarterly	Not mentioned
Selmy et al. (2024)	Hybrid CNN-LSTM model	Various (focus on sequence-to-sequence forecasting)	Not mentioned

their superior performance. time-series forecasting is fundamentally dependent on its past observations, and the auto-regressive order, also known as the memory of model, has historically been key parameter in making predictions [28, 43, 46]. However, there is a noticeable lack in the literature on global forecasting methods regarding a systematic investigation of the auto-regressive order and its effects on the accuracy of forecasts. This gap is significant because the selection of global forecasting methods typically assumes that the time-series of interest are homogeneous. However, it remains unclear which segments (nth time step) of the series show similarity. This question can only be answered by auto-regressive selection that restricts the training data for the chosen model. Moreover, the applicability of the global forecasting methods remains uncertain in the real world scenarios. For instance, the M4 forecasting competition, which evaluates various forecasting techniques across diverse horizons, indicates that pooled data approaches do not consistently outperform local methods in all forecast horizon scenarios. The study by Montero-Manso et al. on meta-learning for global time-series forecasting investigated model selection processes that works on the combined/pooled time-series across different forecasting horizons. However, the study does not address the notion of auto-regressive order, leaving a gap in the empirical assessment of the global forecasting method for different auto-regressive orders.

4.1.3 Our Approach

This chapter intends to bridge this gap by exploring the impact of the auto-regressive order selection on the effectiveness of global forecasting methods. This research attempts to answer three key questions: (i) Can global forecasting methods outperform in the short memory order by "borrowing" information from other series with the pooled series? (ii) What are the relationships between the auto-regressive order and forecast horizons in terms of errors, and can these findings be generalized across different forecasting scenarios? This research aims to clarify their effect on model results and establish guidelines for their optimal configuration by experimenting with various combinations of auto-regressive orders in global forecasting methods. This approach not only addresses the existing gap of comprehensive empirical analysis in the field, but also develop understanding of how the "depth" of historical data directly influence predictive performance. Through a thorough analysis, this research aims to provide valuable insights that could guide future applications and enhancements in the global forecasting methods .

Table 4.2: Number of M4 time-series per data frequency and domain

Time interval between successive observations	Micro	Industry	Macro	Finance	Demographic	Other	Total
Yearly	6,538	3,716	3,903	6,519	1,088	1,236	**23,000**
Quarterly	6,020	4,637	5,315	5,305	1,858	865	**24,000**
Monthly	10,975	10,017	10,016	10,987	5,728	277	**48,000**
Weekly	112	6	41	164	24	12	**359**
Daily	1,476	422	127	1,559	10	633	**4,227**
Hourly	0	0	0	0	0	414	**414**
Total	**25,121**	**18,798**	**19,402**	**24,534**	**8,708**	**3,437**	**100,000**

4.2 Methodology

4.2.1 M4 Datasets

We use the M4 competition dataset [40], an enhancement of the M1 and M3 competitions, which offers a diverse range of time-series data across various frequencies. The dataset as shown in Table. 4.2 contains 100,000 time-series split into six categories - Demographic, Finance, Industry, Macro, Micro, and Others. Our focus in this chapter is on the Hourly subset of the M4 dataset as we are interested in high-frequency data to assess the effectiveness of the global forecasting method for multi-time step forecasting. This hourly subset enables us to identify short, medium, and long-term patterns.

4.2.2 Global Forecasting Function

Let us assume that ensemble $x_i = (x_{i,t})_{t=1}^{T}$ of time-series coming from the same source is available for forecasting where i denotes the number of time-series in the set at t-th time point. We assume each uni-variate time-series x_i is scaled such that the value $x_{i,t} \in [0, 1]$ in order to account for individual variability of time-series. The global approach fits the same forecasting function to all time-series in the dataset while the local approach, also known as the uni-variate approach, fits unique functions to each time-series. In contrast, the semi-global approach fits distinct functions to subsets of time-series.

4.2.3 Forecasting models

Our research employs diverse sets of forecasting models built around the principle of auto-regression. These models aim to minimize loss across a large, aggregated matrix, which is formed by lag-embedding each series. Essentially, we treat the forecasting models as a regression task, using past time-series observations (lags) to transform the forecasting problems into supervised machine learning problems. This approach enables us to pool data from different time-series, constructing a global data matrix for global forecasting models, which uses the collective information within the dataset. In the global matrix (pooled data), a final column represents the target values, while the preceding columns are input features. Our forecasting strategy primarily produces one-step-ahead predictions, extending to farther-ahead horizons using recursive forecasting. The benchmark techniques used in this research are presented in Section 1.4. It's important to note that our modeling approach intentionally avoids the use of advanced techniques, including seasonality adjustments and complex neural network architectures. Instead, we use basic features to ensure our findings stem from the models' inherent predictive abilities, without any external enhancements.

Algorithm 1 Model Fitting and Forecasting

```
procedure GlobalModelFitting(time_series, AR_orders, n_horizons, global_methods)
    pooled_series ← DataPooling(Series)
    for all AR_order in AR_orders do
        for all n_horizon in n_horizons do
            model ← TrainGlobalModel(pooled_series, AR_order, method)
            RecursiveForecast(model, pooled_series, n_horizon)
        end for
    end for

end procedure
procedure LocalModelFitting(time_series, AR_orders, n_horizons, local_methods)
    for all AR_order in AR_orders do
        for all n_horizon in n_horizons do
            model ← TrainLocalModel(time_series, AR_order, method)
        end for
    end for

end procedure
procedure RecursiveForecast(model, data, n_horizon)
    X_test ← GetParameters(methods, models)          ▷ methods includes global and local methods
    for i ← 1 to n_horizon do
        forecast ← model.Predict(X_test)
        X_test ← UpdateTestData(X_test, forecast)          ▷ Update test set with new forecast
    end for
end procedure
```

4.2.4 Recursive forecasting methods for multi-step forecasting

In this chapter, we employ a recursive forecasting strategy to generate multi-step ahead predictions. This technique is especially effective for models that inherently generate one-step ahead forecasts using the pooled matrix of lag features. Recursive forecasting uses model predictions as inputs for future forecasts, "recursing" through the forecast horizon. The methodological framework for the recursive forecasting process is defined as follows:

Given a uni-variate time-series from the set $x_{i,t}$ where $t = 1, 2, \ldots, T$ (assuming T is the total length of time-series) and a forecasting model f, the objective is to predict h steps ahead. The predictions are denoted as $\hat{x}_{i,T+1}, \hat{x}_{i,T+2}, \ldots, \hat{x}_{i,T+h}\}$. For a model that uses l lags, the input at any time t (for $t > l$) is $\{x_{i,T-l}, \ldots, x_{i,T-1}\}$, and it predicts the value at $\hat{x}_{i,T+h}$.

The recursive forecasting process starts with the actual observed values up to time T to predict $\hat{x}_{i,T+1}$. This prediction, $\hat{x}_{i,T+1}$ is then appended to the lagged values, replacing the oldest observation, to form a new input set for predicting $\hat{x}_{i,T+2}$, and so forth. Mathematically, this can be represented as follows:

1. Predict $\hat{x}_{i,T+1}$ using $x_{i,T-l+1}, ..., x_{i,T}$
2. For each step subsequent step:
    - Update the input set by appending $\hat{x}_{i,T+l-1}$ and removing the oldest value.
    - Predict $\hat{x}_{i,T+l}$ using the updated input set.

This recursive approach enables our forecasting framework to extend single-step predictions into multi-step forecasts while capturing the inherent characteristics such as dynamics and dependencies within individual time-series.

4.2.5 Memory

In time-series forecasting, the concept of 'memory' or AR order is important as it controls the complexity of forecasting models and help understand predictions based on historical data. Given a time-series $\{x_{i,t}\}$ where t indexes time ($t = 1, 2, \ldots, T$), the memory of the model, denoted by l (number of lags), is the number of past observations used to predict the future value $\hat{x}_{i,T+1}$. This can be expressed in the auto-regressive (AR) model form as:

$$\hat{x}_{i,T+1} = \phi_0 + \sum_{i=1}^{l} \phi_i x_{i,T+1-i} + \epsilon_{i,T+1} \tag{4.1}$$

where ϕ_0 is the intercept term andϕ_i are the coefficients of the lags (i.e., past values) up to l, and $\epsilon_{i,T+1}$ is the error term at time $T + 1$.

4.2.6 Experiments

Our implementation begins with data pooling to create a lag-embedded matrix, which is then split for model fitting using both global and local forecasting methods. The final stage of predictions uses the recursive forecasting method, leveraging a pooled dataset from various time-series and an array of forecasting models. The end-to-end implementation of the experiment is depicted in algorithm 1. Table. 4.3 shows parameters used in the experiment. For each series and model combination, the implementation follows these steps:

1. **Uni-variate model fitting (Local benchmark methods)**: In this research, we employ a combination of statistical and deep learning techniques frequently mentioned in literature reviews for benchmarking local methods, as described in section 2.4.1. The first linear auto-regressive model is used for general-purpose methods, while more complex models like the Long Short-Term Memory Network (LSTM) and Gradient Boosting Regressor are proposed to handle non-linearity of the hourly M4 dataset. To ensure a fair comparison between local and global methods, we applied the same 8 model classes to both.

2. **Pooled data approach (Global methods)**: All global methods are constructed based on auto-regressive (AR) or limited memory models. Each series is pooled and embedded into a matrix based on the AR order. These matrices are then stacked to form a single, large matrix, thus realizing data pooling using uni-variate time-series. Various model classes are designed to minimize a loss on this extended global matrix, similar to a conventional regression problem. The final column in the matrix represents the target values, while the rest of the columns represent the input variables. All global models are experimented to provide the same combinations of forecast horizons. Recursive forecasting technique is used for the extended forecasting horizons.

3. **time-series Partitioning**: Another experiment we explored to increase complexity is partitioning the time-series. Each series is divided into three equal-sized groups (the initial 20/60/80 percent of the observations), and both local and global models are applied to each group for fitting and forecasting.

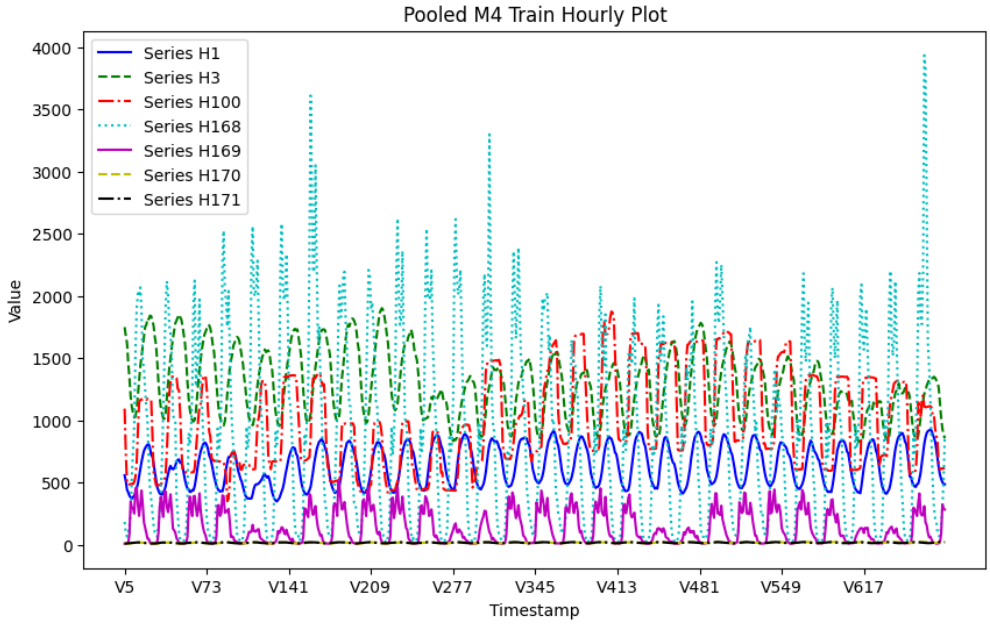

Figure 4.1: The line plots for 7 sampled series from the M4 Hourly set are depicted. Variability beyond the normal seasonal behavior is observed between the V73 and V141 time steps and around the V277 and V617 time steps. This variability of patterns may impact the performance of the series. Hence, the time-series partitioning technique could be beneficial to evaluate results across different segments.

We perform partitioning of the time-series to ensure our evaluation would not be influenced by the variability of specific segments of time-series data, as indicated in Fig.4.1. Besides potentially providing a more accurate comparison of various methods, partitioning also allows for faster training times by limiting the training size.

4. **Performance metrics**: Once the model is fitted, we apply recursive forecasting techniques to both local and global methods. Starting from the last observed value, the forecasting process iteratively predicts the future value. Each prediction is then incorporated back into the input set for subsequent forecasts. The results of recursive forecasting are compiled to provide a detailed view of forecast accuracy across different metrics. Among the various evaluation metrics, Root Mean Squared Error (RMSE) is primarily used to assess the results in this chapter to evaluate with other benchmark studies.

4.3 Results

Memory Effects in Global Forecasting Models

Table. 4.4 demonstrates the impact on the forecast accuracy when the memory of a global model is increased. This design principle is first tested with linear models, the simplest class of models. However, we observed no convergence in accuracy and the error increased significantly, as the global model experienced over-fitting. Although the observed over-fitting can be mitigated in practice by applying cross-validation technique, we exclude the linear outliers from the analysis. As shown in Fig.4.2, we notice the clear trend of reduction in out-of-sample error in global forecasting methods compared to local methods. However, as the global AR model order increases, errors start to accumulate in both global and local methods, roughly around AR(12).

Table 4.3: With each forecasting method (local and global), we examine eight forecasting models known for capturing both linearity and non-linearity across five different windows of the AR order. The required prior knowledge is derived from AR correlation analyses. We also test three different groups of time-series partitioning across six unique forecast horizons. For each uni-variate time-series, a total of 1,440 combinations of experiments are conducted.

Methods	**Forecasting Techniques**	**AR order**	**Partitioning**	**Forecast Horizons**
Global	Linear Auto-regressive regression	3	20%	1
Local	Ridge Regression	6	60%	3
	Random Forest Regressor	12	80%	7
	XGBoost Regressor	24		14
	Random Forest Regressor	36		23
	Decision Tree Regressor			30
	Gradient Boosting Regressor			
	Long Short-Term Memory Network			
2	**8**	**5**	**3**	**6**

For further investigation, we extend our analysis to examine auto-correlations of M4 Hourly series under experimental conditions. As stated previously, the global method's performance is best achieved at AR(6) compared to other AR models. We assume this pattern is likely due to the inherent features of the M4 Hourly series, such as auto-correlation.

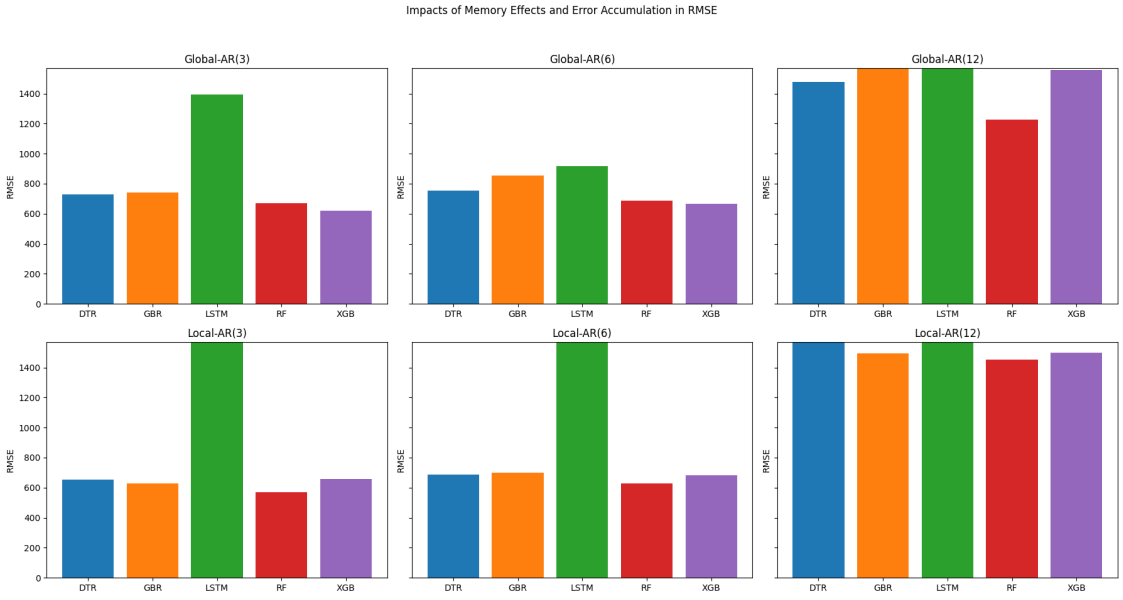

Figure 4.2: Median Root Mean Squared Error (RMSE) of 441 series of global and local methods in different auto-regressive orders, measured as out-of-sample RMSE over their respective forecasting classes.

Table 4.4: Impacts of memory effects and error accumulation in Root Mean Squared Error (RMSE). Each RMSE value for a combination of experiments, such as a Decision Tree Regressor with an auto-regressive order of 3, is computed based on the median RMSE values for 441 series.

Memory Order	DTR	GBR	LSTM	RF	XGB
Global-AR(3)	727	742	1395	670	619
Global-AR(6)	753	856	917	686	665
Global-AR(12)	1476	1568	1854	1228	1557
Local-AR(3)	653	630	3967	569	658
Local-AR(6)	687	701	3837	628	681
Local-AR(12)	1567	1493	5021	1454	1498

To analyze how observations correlate over time lags in M4 series, we compute the Auto-correlation Function (ACF), which represents the correlation between the series and its delayed version at lag k, as depicted in Fig. 4.3. We note no decay of correlations, a phenomenon usually observed when a series has a short memory and falls to zero rapidly. In contrast, the ACF values decline slowly which suggests a long memory in the series. We found that AR(6) model provides the best performance, even though the ACF value at lag 6 is nearly 0 in the extended analysis. During our evaluation with the M4 hourly series, we found that the trend of improved accuracy with increased memory for global methods doesn't consistently apply to most forecasting models.

As shown in by the ACF analysis, the general error bounds for global methods more than double with extended memory, which reveals negative correlations at AR(12). The loss of serial dependency leads to increased error accumulation across all tested global methods. Among the tested models, AR(6) showed the best performance on the M4 hourly series, which could indicate that the dataset's inherent characteristics of short serial correlations as well as the short lead times in data collection. When time series data is collected or aggregated at short intervals (e.g., hourly), it captures immediate trends and patterns that models with limited memory can analyze without needing historical data points from the distant past. These findings align with our observation that error rates increase when memory thresholds exceed AR(6). In particular, we observe that linear models show greater error increase beyond AR(6) compared to non-linear models, which indicates that linear models struggle more with longer memory. In contrast, non-linear models (e.g., LSTM and XGBoost) showed greater resilience to extended memory lengths, though they too suffered performance degradation at AR(12). Based on these observations, our findings suggest that global non-linear models are more appropriate for forecasting and perform best with short memory without the need to learn from distant past data.

Nonlinear models: deep networks

The experiment under discussion evaluate increased complexity by adding the different model classes in both global and local scenarios. We incorporate nonlinear models like deep networks (Long Short-Term Memory Networks) and regression trees as alternatives to linear classes. We evaluate these non-linear models for defined auto-regression orders to isolate the model class effects from the memory expansion effects.

Table. 4.5 shows the impact of model class on the M4 Hourly dataset. The accuracy of the global linear model improves as its memory capacity increases, however, it does not surpass local benchmark methods.

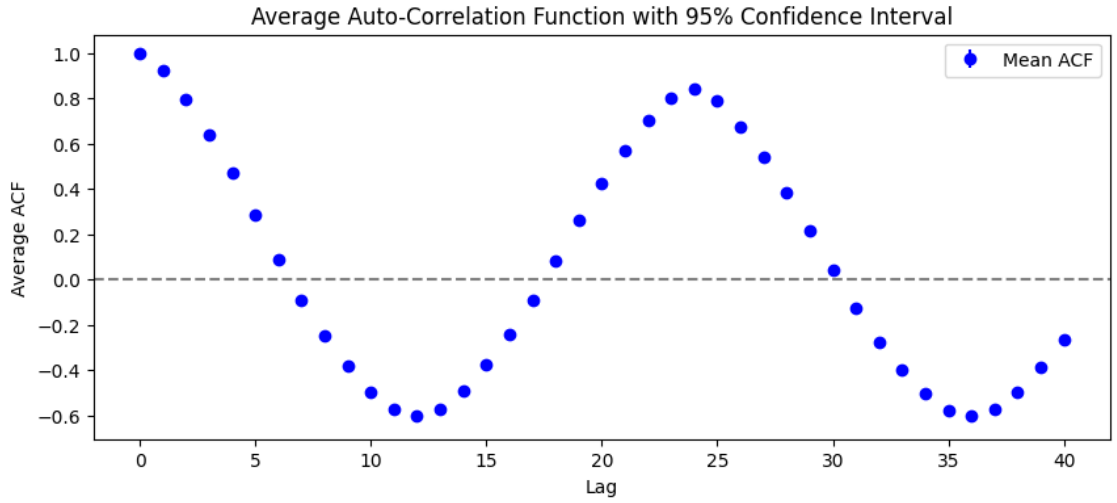

Figure 4.3: Auto-correlation represents the correlation between series and its delayed version at lag k

Table 4.5: Impact of model class across different auto-regressive and forecast horizon parameters. The accuracy is denoted as Symmetric Mean Absolute Percentage Error (sMAPE), which multiplies the metric values by 2 to obtain a range of sMAPE = [0, 200]. A score of 0 indicates a perfect match between the actual and predicted values. The rows of the table correspond to the forecast horizons in relation to the auto-regressive order.

	Global Methods							**Local Methods**						
	DTR	GBR	LR	LSTM	RF	RR	XGB	DTR	GBR	LR	LSTM	RF	RR	XGB
AR(3)	***87.0***	***118.1***	***89.9***	***42.9***	***88.5***	***89.9***	***44.1***	***33.9***	***38.0***	***54.1***	***74.1***	***30.2***	***53.9***	***41.5***
AR(3),H(1)	95.1	125.8	51.8	30.1	95.2	51.8	34.8	28.3	38.5	42.4	53.0	26.3	42.4	38.8
AR(3),H(3)	90.5	119.8	40.9	9.5	90.7	40.9	31.0	15.7	21.0	23.9	45.7	12.9	23.9	28.0
AR(3),H(7)	83.2	112.8	66.4	34.2	83.9	66.4	39.4	35.2	35.8	39.4	73.0	28.7	39.4	39.2
AR(3),H(14)	80.8	111.2	99.7	42.1	81.9	99.7	49.3	42.2	43.0	55.3	85.9	38.4	55.2	47.5
AR(3),H(23)	88.2	119.0	134.2	58.2	89.8	134.2	55.6	42.0	45.8	75.7	92.3	38.9	75.4	48.9
AR(3),H(30)	88.1	120.2	146.6	83.5	89.7	146.6	54.6	40.2	43.6	87.8	94.8	36.1	87.4	46.7
AR(6)	***87***	***127.5***	***72.3***	***49.2***	***89.1***	***72.3***	***37.8***	***38.1***	***43.1***	***54.0***	***60.6***	***35.0***	***53.8***	***39.1***
AR(6),H(1)	98.5	137.6	71.6	54.5	98.3	71.6	50.2	55.3	59.9	64.8	66.1	52.7	64.8	52.3
AR(6),H(3)	90.1	131.4	56.8	47.8	91.0	56.8	38.1	34.7	41.5	45.9	48.8	30.7	45.9	31.1
AR(6),H(7)	82.8	121.7	36.7	28.6	83.1	36.7	28.9	25.1	30.1	33.7	50.6	23.3	33.7	29.2
AR(6),H(14)	80.5	119.6	62.9	39.0	81.5	62.9	30.6	36.4	40.0	46.6	61.2	32.6	46.5	38.3
AR(6),H(23)	88.7	126.8	96.5	51.5	90.4	96.5	40.1	41.0	45.7	63.6	67.0	37.4	63.3	43.4
AR(6),H(30)	88.2	127.7	109.4	74.0	90.2	109.4	39.1	35.9	41.4	69.2	70.0	33.5	68.7	40.3
AR(12)	***92.4***	***125.6***	***50.6***	***53.4***	***94.1***	***50.6***	***43.5***	***40.1***	***43.1***	***48.0***	***70.1***	***39.1***	***47.9***	***39.8***
AR(12),H(1)	104.6	143.6	59.7	76.8	108.4	59.7	55.9	53.7	63.6	64.0	70.2	56.7	64.0	55.0
AR(12),H(3)	97.2	138.2	56.1	49.2	100.7	56.1	49.2	44.9	54.2	58.7	64.3	45.3	58.7	44.9
AR(12),H(7)	85.3	125.7	44.9	52.4	85.9	44.9	37.5	31.7	29.3	35.5	63.0	26.3	35.5	26.4
AR(12),H(14)	82.1	103.1	28.5	31.5	81.3	28.5	32.7	26.5	24.7	25.0	69.5	26.1	25.0	29.3
AR(12),H(23)	90.5	113.0	53.0	44.0	90.9	53.0	39.0	37.1	35.7	44.2	78.3	35.3	44.1	38.4
AR(12),H(30)	91.9	114.3	56.7	57.8	90.6	56.7	40.4	37.6	34.9	47.0	78.4	32.8	46.9	34.7

Even though a decrease in error is noticed as memory expands, an exponential rise in forecast error occurs beyond extended forecast horizons (H=23), indicating that the linear model's complexity is limited by under-fitting in global scenario. Recursive forecasting approach appears to make the simple linear models numerically unstable (beyond 23-step-ahead forecast for this dataset) and is thus not recommended for automatic time-series forecasting. Regression trees, although more complex than linear models, perform poorly on this dataset without noticeable effects on both auto-regressive order and forecast horizons, making them an unsuitable model class for global forecasting scenarios. Lastly, the Long Short-term Memory Networks (LSTM) generally outperforms global methods and delivers the best results among them, close to the XGBoost model in this situation. Although some studies advocate for direct forecasting (separate modeling of each horizon), we demonstrate that the LSTM networks don't suffer as much from recursive forecasting instability as other model classes. Both XGBoost and LSTM show that nonlinear model classes can improve accuracy, even though significant effects of auto-regressive orders in global scenarios haven't been observed in this experiment. The LSTM model, when applied to the pooled series (global), outperforms local methods in all scenarios except AR(12) regarding 1-step ahead forecasts. We view these findings as strong empirical evidence supporting globality using deep neural networks. However, these findings should not be seen as preferring one model class over others. For instance, with some hyperparameter adjustments, regression trees could perform better.

Computational Efficiency

We conducted experiments to compare the computational efficiency of forecasting models on the M4 Hourly dataset which consists of 414 series. The experiments were performed on a workstation with a 24-core CPU, NVIDIA RTX 5000 GPU, and 32GB of memory using Python 3.7. The best-performing models are selected, and horizon 1 predictions are tested across simulations using three different autoregressive orders.

Table 4.6: Training and testing times (in seconds) for global and local methods across autoregressive orders and forecasting models. The numbers are rounded to the nearest tenth.

AR Order	**Global Methods**				**Local Methods**			
	LR	**DTR**	**XGB**	**LSTM**	**LR**	**DTR**	**XGB**	**LSTM**
AR(3)	410	1,120	3,200	3,850	2,400	6,500	9,800	11,200
AR(6)	720	1,700	5,200	6,800	3,800	10,100	14,500	16,500
AR(12)	1,150	2,500	7,300	11,200	6,200	15,300	21,000	24,500

Table 4.6 reveals that global methods are computationally more efficient than local methods across all models and autoregressive (AR) orders. For instance, with AR(3), the LSTM's local methods runtime is almost three times higher than its global counterpart. Global methods benefit from a pooled data approach, where the computational load is shared across the series. Conversely, local methods train separate functions for each series and increase complexity. Additionally, computational time increases substantially with higher AR orders across both global and local methods. The AR order affects the size of the lag-embedded matrix, leading to increases the computational load during training and recursive forecasting. For example, training times for AR(12) are approximately 1.5 times longer than those for AR(6) across all model classes. Notably, the increase in runtime as AR order increases is more pronounced for non-linear models. The LSTM global runtime shows more than a 3x increase from AR(3) to AR(12), while linear models show more modest scaling. This indicates that non-linear models are more sensitive to AR order changes than linear models.

4.4 Conclusion and future work

This study has examined the capabilities of global forecasting methods (GFM) compared to traditional local methods in handling high-frequency time-series data from the M4 hourly dataset. Our empirical investigation focused on understanding the impact of autoregressive (AR) order and forecast horizons on forecasting accuracy. We found that for shorter autoregressive (AR) orders, both global and local forecasting methods deliver similar performance levels. This implies that for immediate forecasting, choosing between global and local methods may not be critical. However, as the AR order increases, introducing more complexities, global forecasting methods show clear sign of outperformance. This indicates that global methods are more appropriate for leveraging long-term historical information to improve forecast accuracy. This implies that for forecasting applications which require long-term predictions, prioritizing the use of global forecasting methods can be more applicable in real-world settings.

Our empirical evidence also indicate that the benefits of the global forecasting method become more noticeable as forecast horizons widen. In fact, global methods reduce error accumulation as the forecast horizon extends in contrast to local methods, where error rates tend to increase with longer forecast horizons. This reduction in the overall performance could be attributed to the underlying assumption of the global method that synthesizes information across a larger, pooled dataset, leading to minimize the cumulative error derived from the recursive forecasting techniques. This finding will lead to explore the relationship between AR order and forecast horizons in the context of the global forecasting methods to better understand their synergistic effects on the forecasting performance and refine global forecasting methods, which is yet underexplored aspects in the sub-field of globality.

Our most recent finding suggests that nonlinear regression models, especially those with extended polynomial orders, outperform local methods when applied to pooled data. This suggests that nonlinear global models are more capable of capturing the complex patterns inherent in aggregated time-series data. However, these findings should not be seen as favoring one model class over others, until the empirical evidence is based on large, diverse sets of time-series data with distinct structural patterns.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In many time series applications, we possess substantial amounts of time series data. Recently, strategies have been devised to construct models that take advantage of the similarities among various time series. Nonetheless, when dealing with diverse time series, the precision of such a model can be reduced, making it necessary to take into account the degree of similarity between the time series. With this requirement in mind, we've proposed various forecasting frameworks that leverage cross-series information or borrow inherent characteristics of time series.

In Chapter 2, we introduce a two-step meta-learning framework for feature-based time-series prediction. This framework first predicts intermediate outputs using a mix of binary and continuous features from time series. It then uses these outputs and historical values to make final predictions. By using the Reptile algorithm, this approach shows superior performance compared to traditional forecasting methods. Additionally, we present a framework for capturing underlying time series characteristics and selecting statistically significant features. We use the Kolmogorov-Smirnov test for binary features and Kendall's tau correlation test for continuous features to demonstrate how salient feature selection can be implemented for the feature-based forecasting stage. Moreover, we demonstrate how many features fall outside the space covered by the original time series data using Principal Component Analysis. All these tests have proven effective in implementing Reptile algorithms. Despite potential concerns about noise introduction, covariates derived from the initial training phase have served as informative predictors for making final forecasts. Experiments conducted on real-world time-series data reveal the framework achieved high accuracy.

Chapter 3, proposes a new framework for feature-based, cross-series forecasting that utilizes the similarity of time series. This automated forecasting framework uses similarity factors among series to minimize forecast errors. Our study also explored how the features extracted from the original time series represent each univariate series. This was done through statistical significance testing and Principal Component Analysis. While experiments with real-world data using feature-based and distance-based similarity measures resulted in mixed outcomes, we concluded that carefully measured series similarity can outperform emerging global models when cross-trained together.

In Chapter 4, we investigated the impact of auto-regressive order selection on the effectiveness of cross-

series forecasting methods. Our study compared global forecasting methods (GFM) with traditional local methods using high-frequency M4 time-series data. The study revealed that global methods, specifically those incorporating long-term historical data and extended polynomial orders, outperform local methods. This is particularly true for longer forecast horizons. The results also indicate the potential of recurrent neural networks to reduce error accumulation over extended periods and to capture complex patterns in aggregated time-series data.

5.2 Future Work

Our study on the two-step meta-learning framework confirmed that feature-based forecasting frameworks can outperform conventional forecasting methods that operate on time series values. In our experiment, the current framework's dependence on a specific dataset from medical product manufacturers restricts its generalizability. Therefore, the two-step Reptile approach can also be evaluated using the widely recognized M4/M5 competition datasets. In addition, our approach can also be further explroed dynamic ways to increase forecasting accuracy across various applications and data sets. This includes expressing covariates from the feature space, such as creating rolled sub-series for cross-validation. Furthermore, using error-derived uncertainty measures as covariates could also introduce a new dimension to forecast modeling which can be particularly useful when dealing with random noise and forecast uncertainty.

The clustering methods discussed in Chapter 3 for cross-series forecasting can be expanded. Beyond traditional techniques like K-means or hierarchical clustering, advanced methods such as Self-Organizing Maps (SOM) can help identify similarities in time series from large feature subsets. Finally in Chapter 4, we uncovered the relationships between the AR order and forecast horizons based purely on forecast accuracy, without considering the auto-regression analysis of the pooled data matrix. Since the global method is based on data pooling techniques by combining all series of interest, deep learning techniques like the self-attention mechanism for automatic modeling of auto-regressive order can help increase forecast accuracy.

Bibliography

[1] Hossein Abbasimehr and Reza Paki. "Prediction of COVID-19 confirmed cases combining deep learning methods and Bayesian optimization". In: *Chaos, Solitons & Fractals* 142 (2021), p. 110511.

[2] Sebastian Pineda Arango et al. "Multimodal meta-learning for time series regression". In: *Advanced Analytics and Learning on Temporal Data: 6th ECML PKDD Workshop, AALTD 2021, Bilbao, Spain, September 13, 2021, Revised Selected Papers 6*. Springer. 2021, pp. 123–138.

[3] Kasun Bandara, Christoph Bergmeir, and Slawek Smyl. "Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach". In: *Expert systems with applications* 140 (2020), p. 112896.

[4] Kasun Bandara et al. "Improving the accuracy of global forecasting models using time series data augmentation". In: *Pattern Recognition* 120 (2021), p. 108148.

[5] Sasan Barak, Mahdi Nasiri, and Mehrdad Rostamzadeh. "Time series model selection with a meta-learning approach; evidence from a pool of forecasting algorithms". In: *arXiv preprint arXiv:1908.08489* (2019).

[6] Eren Bas et al. "Type 1 fuzzy function approach based on ridge regression for forecasting". In: *Granular Computing* 4 (2019), pp. 629–637.

[7] Candice Bentéjac, Anna Csörgő, and Gonzalo Martínez-Muñoz. "A comparative analysis of gradient boosting algorithms". In: *Artificial Intelligence Review* 54 (2021), pp. 1937–1967.

[8] George EP Box et al. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[9] Leo Breiman. "Random forests". In: *Machine learning* 45 (2001), pp. 5–32.

[10] Charles W Chase. *Demand-driven forecasting: a structured approach to forecasting*. John Wiley & Sons, 2013.

[11] Chris Chatfield. "The Holt-winters forecasting procedure". In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 27.3 (1978), pp. 264–279.

[12] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[13] Alexey Chernikov et al. "FRANS: Automatic Feature Extraction for Time Series Forecasting". In: *arXiv preprint arXiv:2209.07018* (2022).

[14] Maximilian Christ et al. "Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package)". In: *Neurocomputing* 307 (2018), pp. 72–77.

[15] Mariel Abigail Cruz-Nájera et al. “Short time series forecasting: Recommended methods and techniques”. In: *Symmetry* 14.6 (2022), p. 1231.

[16] Simon Dablemont et al. “Time series forecasting with SOM and local non-linear models-Application to the DAX30 index prediction”. In: *WSOM 2003, Workshop on Self-Organizing Maps*. 2003.

[17] Fatoumata Dama and Christine Sinoquet. “Time series analysis and modeling to forecast: A survey”. In: *arXiv preprint arXiv:2104.00164* (2021).

[18] Xin Ding et al. “A novel similarity measurement and clustering framework for time series based on convolution neural networks”. In: *Ieee Access* 8 (2020), pp. 173158–173168.

[19] Ronald Aylmer Fisher. “Design of experiments”. In: *British Medical Journal* 1.3923 (1936), p. 554.

[20] Gene Fliedner. “Hierarchical forecasting: issues and use guidelines”. In: *Industrial Management & Data Systems* 101.1 (2001), pp. 5–12.

[21] Michael Geurts. *Book review: time series analysis: forecasting and control*. 1977.

[22] Alex Graves and Alex Graves. “Long short-term memory”. In: *Supervised sequence labelling with recurrent neural networks* (2012), pp. 37–45.

[23] David Guijo-Rubio et al. “Time-series clustering based on the characterization of segment typologies”. In: *IEEE transactions on cybernetics* 51.11 (2020), pp. 5409–5422.

[24] Manoj Kumar Gupta and Pravin Chandra. “An empirical evaluation of K-means clustering algorithm using different distance/similarity metrics”. In: *Proceedings of ICETIT 2019: Emerging Trends in Information Technology*. Springer. 2020, pp. 884–892.

[25] Xiao Han et al. “Multi-Task Time Series Forecasting Based on Graph Neural Networks”. In: *Entropy* 25.8 (2023), p. 1136.

[26] Claudio Hartmann et al. “Exploiting big data in time series forecasting: A cross-sectional approach”. In: *2015 IEEE international conference on data science and advanced analytics (DSAA)*. IEEE. 2015, pp. 1–10.

[27] Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. “Global models for time series forecasting: A simulation study”. In: *Pattern Recognition* 124 (2022), p. 108441.

[28] David C Hill et al. “Application of auto-regressive models to UK wind speed data for power system impact studies”. In: *IEEE Transactions on Sustainable Energy* 3.1 (2011), pp. 134–141.

[29] Charles C Holt. “Forecasting seasonals and trends by exponentially weighted moving averages”. In: *International journal of forecasting* 20.1 (2004), pp. 5–10.

[30] Rob J Hyndman and Anne B Koehler. “Another look at measures of forecast accuracy”. In: *International journal of forecasting* 22.4 (2006), pp. 679–688.

[31] Anastasios Kaltsounis, Evangelos Spiliotis, and Vassilios Assimakopoulos. “Conditional Temporal Aggregation for Time Series Forecasting Using Feature-Based Meta-Learning”. In: *Algorithms* 16.4 (2023), p. 206.

[32] Yanfei Kang, Rob J Hyndman, and Feng Li. “GRATIS: GeneRAting TIme Series with diverse and controllable characteristics”. In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 13.4 (2020), pp. 354–376.

[33] Dimitrios E Koulouriotis and Georgios Mantas. "Health products sales forecasting using computational intelligence and adaptive neuro fuzzy inference systems". In: *Operational Research* 12 (2012), pp. 29–43.

[34] Pedro Lara-Benítez, Manuel Carranza-García, and José C Riquelme. "An experimental review on deep learning architectures for time series forecasting". In: *International journal of neural systems* 31.03 (2021), p. 2130001.

[35] Hubert W Lilliefors. "On the Kolmogorov-Smirnov test for normality with mean and variance unknown". In: *Journal of the American statistical Association* 62.318 (1967), pp. 399–402.

[36] Winston T Lin. "Modeling and forecasting hospital patient movements: Univariate and multiple time series approaches". In: *International Journal of Forecasting* 5.2 (1989), pp. 195–208.

[37] Shaohui Ma and Robert Fildes. "Large-Scale Time Series Forecasting with Meta-Learning". In: *Forecasting with Artificial Intelligence: Theory and Applications*. Springer, 2023, pp. 221–250.

[38] Spyros Makridakis and Michele Hibon. "The M3-Competition: results, conclusions and implications". In: *International journal of forecasting* 16.4 (2000), pp. 451–476.

[39] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. "M5 accuracy competition: Results, findings, and conclusions". In: *International Journal of Forecasting* 38.4 (2022), pp. 1346–1364.

[40] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. "The M4 Competition: Results, findings, conclusion and way forward". In: *International Journal of forecasting* 34.4 (2018), pp. 802–808.

[41] Nigel Meade. "Evidence for the selection of forecasting methods". In: *Journal of forecasting* 19.6 (2000), pp. 515–535.

[42] Nigel Meade and Towhidul Islam. "Technological forecasting—Model selection, model stability, and combining models". In: *Management science* 44.8 (1998), pp. 1115–1130.

[43] Prabhat Mittal. "Impact of Auto-regressive (AR) Process in Bullwhip Analysis in a Multi-location Supply Chain Network". In: *Journal of Business Management and Information Systems* 6.1 (2019), pp. 19–26.

[44] Pablo Montero-Manso and Rob J Hyndman. "Principles and algorithms for forecasting groups of time series: Locality and globality". In: *International Journal of Forecasting* 37.4 (2021), pp. 1632–1653.

[45] Pablo Montero-Manso et al. "FFORMA: Feature-based forecast model averaging". In: *International Journal of Forecasting* 36.1 (2020), pp. 86–92.

[46] Davide Nardi et al. "Detection of low-velocity impact-induced delaminations in composite laminates using Auto-Regressive models". In: *Composite Structures* 151 (2016), pp. 108–113.

[47] Alex Nichol, Joshua Achiam, and John Schulman. "On first-order meta-learning algorithms". In: *arXiv preprint arXiv:1803.02999* (2018).

[48] Boris N Oreshkin et al. "Meta-learning framework with applications to zero-shot time-series forecasting". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 10. 2021, pp. 9242–9250.

[49] Ángel López Oriona, Pablo Montero Manso, and José Antonio Vilar Fernández. "Time series clustering based on prediction accuracy of global forecasting models". In: *arXiv preprint arXiv:2305.00473* (2023).

[50] Ricardo BC Prudêncio and Teresa B Ludermir. "Meta-learning approaches to selecting time series models". In: *Neurocomputing* 61 (2004), pp. 121–137.

[51] J. Ross Quinlan. "Induction of decision trees". In: *Machine learning* 1 (1986), pp. 81–106.

[52] Amanda Ross et al. "Paired samples T-test". In: *Basic and Advanced Statistical Tests: Writing Results Sections and Creating Tables and Figures* (2017), pp. 17–19.

[53] David Salinas et al. "DeepAR: Probabilistic forecasting with autoregressive recurrent networks". In: *International Journal of Forecasting* 36.3 (2020), pp. 1181–1191.

[54] Yan-Fang Sang. "A review on the applications of wavelet transform in hydrology time series analysis". In: *Atmospheric research* 122 (2013), pp. 8–15.

[55] Mahya Seyedan and Fereshteh Mafakheri. "Predictive big data analytics for supply chain demand forecasting: methods, applications, and research opportunities". In: *Journal of Big Data* 7.1 (2020), p. 53.

[56] Mahya Seyedan and Fereshteh Mafakheri. *Predictive big data analytics for supply chain demand forecasting: methods, applications, and research opportunities. J Big Data 7, 53 (2020).*

[57] Slawek Smyl. "A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting". In: *International Journal of Forecasting* 36.1 (2020), pp. 75–85.

[58] Souhaib Ben Taieb and Rob J Hyndman. "A gradient boosting approach to the Kaggle load forecasting competition". In: *International journal of forecasting* 30.2 (2014), pp. 382–394.

[59] Howel Tong. "Nonlinear time series analysis since 1990: some personal reflections". In: *Acta Mathematicae Applicatae Sinica* 18.2 (2002), pp. 177–184.

[60] Juan R Trapero, Nikolaos Kourentzes, and Robert Fildes. "On the identification of sales forecasting models in the presence of promotions". In: *Journal of the operational Research Society* 66 (2015), pp. 299–307.

[61] Hristos Tyralis and Georgia Papacharalampous. "Variable selection in time series forecasting using random forests". In: *Algorithms* 10.4 (2017), p. 114.

[62] Evaldas Vaiciukynas et al. "Two-step meta-learning for time-series forecasting ensemble". In: *IEEE Access* 9 (2021), pp. 62687–62696.

[63] Neal Wagner et al. "Intelligent techniques for forecasting multiple time series in real-world systems". In: *International Journal of Intelligent Computing and Cybernetics* 4.3 (2011), pp. 284–310.

[64] Xiaoqian Wang et al. "Forecast combinations: An over 50-year review". In: *International Journal of Forecasting* 39.4 (2023), pp. 1518–1547.

[65] Xiaoqian Wang et al. "The uncertainty estimation of feature-based forecast combinations". In: *Journal of the Operational Research Society* 73.5 (2022), pp. 979–993.

[66] Yan Wang and Yuankai Guo. "Forecasting method of stock market volatility in time series data based on mixed model of ARIMA and XGBoost". In: *China Communications* 17.3 (2020), pp. 205–221.

[67] Robert F Woolson. "Wilcoxon signed-rank test". In: *Wiley encyclopedia of clinical trials* (2007), pp. 1–3.

[68] Xinying Yu and Shie-Yui Liong. "Forecasting of hydrologic time series with ridge regression in feature space". In: *Journal of Hydrology* 332.3-4 (2007), pp. 290–302.

[69] Massimiliano Zanin et al. “Permutation entropy and its main biomedical and econophysics applications: A review”. In: *Entropy* 14.8 (2012), pp. 1553–1577.

[70] Bing Zhang, Jhen-Long Wu, and Pei-Chann Chang. “A multiple time series-based recurrent neural network for short-term load forecasting”. In: *Soft Computing* 22 (2018), pp. 4099–4112.

[71] Guoqiang Zhang, B Eddy Patuwo, and Michael Y Hu. “Forecasting with artificial neural networks:: The state of the art”. In: *International journal of forecasting* 14.1 (1998), pp. 35–62.

[72] Ruixue Zhang and Yongtao Hao. “Time Series Prediction Based on Multi-Scale Feature Extraction”. In: *Mathematics* 12.7 (2024), p. 973.

[73] Zheng Zhou, Cheng Qiu, and Yufan Zhang. “A comparative analysis of linear regression, neural networks and random forest regression for predicting air ozone employing soft sensor models”. In: *Scientific Reports* 13.1 (2023), p. 22420.

[74] Xiaodan Zhu et al. “Cross-Series Demand Forecasting Using Machine Learning: Evidence in the Pharmaceutical Industry”. In: *Production and Operations Management, Forthcoming* (2021).