UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Algorithm selection and performance understanding for time series forecasting

Moisés Rocha dos Santos

Tese de Doutorado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP
SÃO CARLOS

Moisés Rocha dos Santos

Algorithm selection and performance understanding for time series forecasting

Thesis submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP – in accordance with the requirements of the Computer and Mathematical Sciences Graduate Program, for the degree of Doctor in Science. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho

USP – São Carlos
June 2023

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

R672a

Rocha dos Santos, Moisés
Algorithm selection and performance understanding for time series forecasting / Moisés Rocha dos Santos; orientador André Carlos Ponce de León Ferreira de Carvalho. -- São Carlos, 2023.
101 p.

Tese (Doutorado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2023.

1. . I. Ponce de León Ferreira de Carvalho, André Carlos , orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Moisés Rocha dos Santos

Seleção e compreensão de desempenho de algoritmos para previsão de séries temporais

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho

USP – São Carlos
Junho de 2023

I dedicate this thesis to
my beloved parents and wife

ACKNOWLEDGEMENTS

Agradeço aos meus pais, Ivaldo e Mercês, por sempre me incentivarem na busca por conhecimento. Agradeço a minha esposa Fiama pelas conversas e pela companhia constante. Agradeço aos amigos e familiares pela rede de apoio.

Agradeço ao professor André pela orientação e acompanhamento frequente. Agradeço ao professor Carlos Soares pela supervisão de estágio no exterior e contribuição crucial para a qualidade desta tese. Agradeço também aos meus coautores, uma parte essencial do processo científico são as colaborações.

Agradeço aos meus colegas de laboratório pelas conversas e trocas de ideias constantes.

Agradeço ao projeto de P&D ANEEL desenvolvido em parceria entre a VOLT ROBOTICS, CESP e USP.

Agradeço ao ICMC e à Universidade do Porto pela infraestrutura laboratorial durante a execução desta tese.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Agradeço à FAPESP pelos auxílios referentes aos processos nº 2019/10012-2 e 2021/13281-4, Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP).

“At some point, you gotta decide for yourself who you gonna be.
Can’t let nobody make that decision for you.”
(Juan, Moonlight)

RESUMO

SANTOS, M. R. **Seleção e compreensão de desempenho de algoritmos para previsão de séries temporais**. 2023. 101 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2023.

Previsão de séries temporais é uma tarefa estratégica no suporte à tomada de decisão. A grande disponibilidade, e variabilidade, de algoritmos capazes de induzir modelos preditivos tem gerado uma demanda por formas de seleção de algoritmos. Adicionalmente, para validação do modelo induzido, é importante entender o seu desempenho preditivo quando aplicado a uma série temporal. Esta tese investiga novas abordagens de seleção de algoritmos para combinação de previsões e entendimento de desempenho preditivo de modelos induzidos por algoritmos para previsão de séries temporais. Para isso, foi inicialmente pesquisado o estado da arte em previsão de séries temporais, com foco na seleção de algoritmos e entendimento de desempenho preditivo. Este estudo observou as limitações de abordagens existentes e identificou as lacunas na literatura que puderam ser solucionadas por esta pesquisa. As principais contribuições desta tese são quatro: o desenvolvimento de uma abordagem baseada em meta-aprendizado para seleção de combinações de previsão com decomposição de séries temporais; um método de geração de séries temporais sintéticas baseado em *"dataset morphing"*; a análise empírica de diferentes medidas de desempenho para escolha de meta-alvo na seleção de algoritmos por meta-aprendizado; a análise da aplicação de decomposição de sazonalidade e tendência com Loess como uma etapa de pré-processamento. A abordagem MetaFore combina algoritmos de aprendizado de máquina para as componentes de tendência e resíduo na tarefa de previsão de séries temporais. As componentes são separadas com a decomposição de sazonalidade e tendência com Loess e a sazonalidade é prevista com o método "*naive*" sazonal. MetaFore foi avaliado nas séries temporais mensais da competição M4 e atingiu melhor desempenho preditivo e computacional que um método que é estado da arte, as redes neurais "*Long Short-Term Memory*" (LSTM), em mais de 70% dos conjuntos de dados. Na pesquisa, o método tsMorph gera séries temporais sintéticas de forma gradual transformando uma série temporal de origem em uma série temporal alvo. O método tsMorph foi aplicado para o entendimento da variação de desempenho preditivo de algoritmos de previsão Regressão com Suporte Vetorial e a rede neural LSTM. Os resultados experimentais mostraram que o método tsMorph gerou séries temporais com variação gradual do desempenho preditivo e meta-características. Esta pesquisa contribuiu para o desenvolvimento de novas abordagens de previsão e entendimento de desempenho de algoritmos eficientemente. Os estudos em seleção de combinações de previsões e entendimento de desempenho podem ser facilmente incluídos nos processos de previsão de séries temporais e abrem perspectivas para pesquisa e desenvolvimento na área de meta-aprendizado e aprendizado de máquina automático.

Palavras-chave: Seleção de algoritmos, Desempenho preditivo, Meta-aprendizado, Previsão de séries temporais.

ABSTRACT

SANTOS, M. R. **Algorithm selection and performance understanding for time series forecasting**. 2023. 101 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2023.

Time series forecasting is a strategic task in supporting decision-making. The wide availability of forecasting algorithms has generated a demand for algorithm selection methods and approaches that enable the understanding of predictive performance given a time series. This thesis focuses on developing new approaches for forecast combination selection and understanding the predictive performance of time series forecasting algorithms. The research started with a review of the state of the art in time series forecasting, focusing on algorithm selection and understanding of predictive performance. This study highlighted the limitations of existing approaches and identified gaps in the literature that this research could address. The main contributions of this thesis are fourfold: firstly, the development of a metalearning-based approach for selecting forecasting combinations with time series decomposition. A synthetic time series generation method based on dataset morphing. The empirical analysis of different performance measures for the choice of meta-label in the selection algorithms by metalearning. Finally, an analysis of applying Seasonal and Trend decomposition using Loess as a pre-processing step for machine learning algorithms in the time series forecasting task. The MetaFore approach selects combinations of machine learning algorithms for the trend and residual components in the time series forecasting task. The components are separated with the Seasonal and Trend decomposition using Loess, and the seasonality is forecasted with the seasonal naive method. MetaFore was evaluated in the monthly time series of the M4 competition and achieved better predictive and computational performance than LSTM neural networks in more than 70% of the datasets. The tsMorph method generates synthetic time series by gradually transforming a source time series into a target time series. TsMorph was applied to understand the predictive performance variation of Support Vector Regression and Long Short-Term Memory neural network prediction algorithms. The results showed that the tsMorph method generated time series with gradual predictive performance and meta-feature variation. Empirical analysis of different performance measures as meta-label in the selection of time series forecasting algorithms showed no statistical difference in the predictive performances of the meta and base level. The analysis of the application of Seasonal Trend with Loess decomposition as a pre-processing step in machine learning showed that when the residual component follows a normal distribution, the decomposition improves the predictive performance of the algorithms. In summary, this research contributed to developing new approaches for efficiently predicting and understanding algorithms' performance. Studies on the selection of forecasting combinations and performance understanding can be easily included in time series forecasting processes and open perspectives for research and development in metalearning and

autoML.

Keywords: Algorithm selection, Predictive performance, Metalearning, Time series forecasting.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

CHAPTER

1

INTRODUTION

The practice of forecasting future events has fascinated humanity for thousands of years (HYNDMAN; ATHANASOPOULOS, 2018a). For a long time, predictions were attributed to divine inspiration, among other explanations without scientific rigor. A milestone for formalizing the temporal data forecasting task was the work of Box *et al.* (1970). This started the treatment of forecasting from the perspective of time series analysis.

Time series are sequential data collected equally spaced in time (CRYER; CHAN, 2008). One of the main tasks in time series analysis is forecasting, which consists of predicting the occurrence of future events (MONTGOMERY; JOHNSON; GARDINER, 1990). Forecasting time series is an essential task, as predicting future events is a desirable knowledge in various types of planning and decision-making processes (MONTGOMERY; JENNINGS; KULAHCI, 2015a; HYNDMAN; ATHANASOPOULOS, 2018a), with applications in areas such as finance, industry, government, demography, epidemiology, among others.

The first methodologies to be used for forecasting time series were statistical methods. These methods predict future events in a distribution, whose coefficients are adjusted from historical data (MONTGOMERY; JENNINGS; KULAHCI, 2015a). Examples of statistical methods are autoregressive models, moving averages, and exponential smoothing, still widely used in the literature(HYNDMAN; ATHANASOPOULOS, 2018a). More recently, machine learning algorithms have gained space in the literature due to the versatility of not assuming a prior distribution for historical data (MULLER *et al.*, 1999; DEB *et al.*, 2017).

We found several time series forecasting frameworks in the literature. The task of forecasting time series to real-world applications goes beyond the choice of methodology. Several steps start from defining the problem to using the forecasting model. Figure 1 illustrates a generic forecasting task process based on those proposed by Montgomery, Jennings and Kulahci (2015a), Shmueli and Jr (2016), Hyndman and Athanasopoulos (2018a).

Problem definition is the definition of the data to be used, the understanding the process

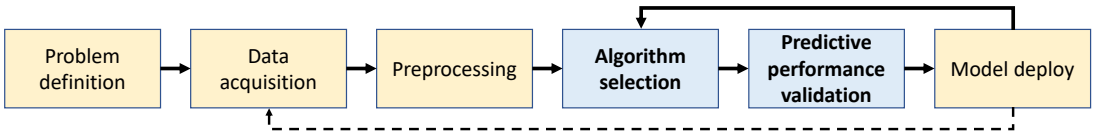


Figure 1 – Steps in a forecasting task.

that produced the data and how the data are distributed. The forecast can be short, medium, or long-term. Which forecast horizon will be considered? How much accuracy is needed for the forecast to be helpful? After this phase, the data of the variable of interest of the event to be predicted are acquired.

The next step is data preprocessing, which consists of preparing the time series for the forecasting algorithms. This step is linked to algorithm selection, as each method requires a specific preprocessing approach. For example, some methods need data normalization to a specific scale. The algorithm selection step consists of choosing and fitting the time series forecasting algorithm for the time series. It is a crucial step, as algorithms are widely available, and testing them all in a trial-and-error approach would be costly.

Predictive performance validation uses performance measures and evaluation techniques to estimate the algorithm's error and behavior in a real-world application. Deploying the forecasting model involves using the forecasts resulting from the process in real problems. Success in deploying a forecasting model is strongly linked to previous steps in the process. The forecasting process in the streaming time series is a repeating cycle from data acquisition to deployment. In case of inadequate performance during model deployment, returning to the algorithm selection step is necessary.

In this thesis, we focus on two steps of the time series forecasting process: algorithm selection, and predictive performance validation. In Figure 1, these steps are highlighted in blue. Algorithm selection and predictive performance validation require great expert knowledge (ARMSTRONG; ADYA; COLLOPY, 2001; HYNDMAN *et al.*, 2006). In the remainder of this section, we delve deeper into these steps and the problems that motivated this work. After the motivation, we introduce the objective and contributions of this thesis.

1.0.1 Motivation

The selection of the algorithm that can induce the best forecasting model for a given time series is a challenging task in this area. To select the most suitable algorithm, it is necessary to explore a wide search space of forecasting algorithms, especially when there is little knowledge of the nature of the time series, which is why the forecasting task is computationally expensive (MEADE, 2000; PRUDÊNCIO; LUDERMIR, 2004). Figure 2 illustrates the algorithm selection problem in time series forecasting.

Metalearning is a set of methods that use the knowledge extracted from past tasks to

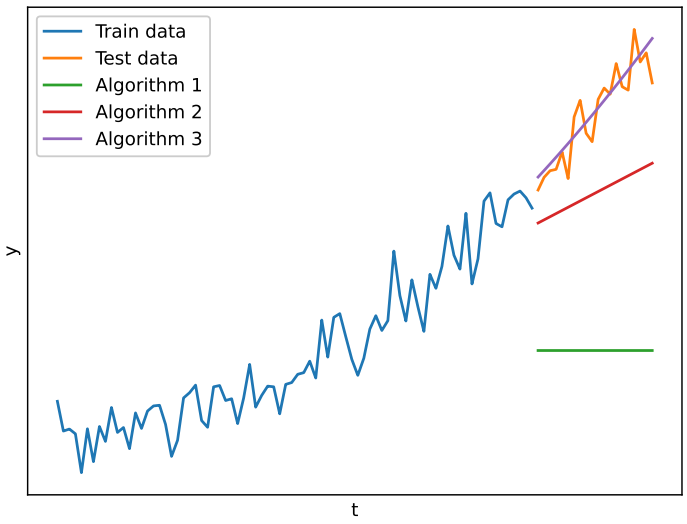


Figure 2 – An example of algorithm selection problem.

perform better for new datasets (VILALTA; DRISSI, 2002; BRAZDIL *et al.*, 2008; FINN; ABBEEL; LEVINE, 2017). In this context, metalearning emerges as an alternative to reduce the computational cost and the high need for specialized knowledge (BRAZDIL *et al.*, 2008). Several works have successfully applied metalearning to select time series forecasting methods (MEADE, 2000; PRUDÊNCIO; LUDERMIR, 2004; LEMKE; GABRYS, 2010; WIDODO; BUDI, 2013; Kück; Crone; Freitag, 2016; ALI; GABRYS; BUDKA, 2018; TALAGALA *et al.*, 2018; BARAK; NASIRI; ROSTAMZADEH, 2019; TALAGALA; LI; KANG, 2019; MONTERO-MANSO *et al.*, 2020; MA; FILDES, 2020). However, some improvements can still be made in selecting metalearning-based time series forecasting techniques, especially regarding exploring new ways of combining forecasts and improvements in the metalearning process for the algorithm selection process.

The forecasting combination is an up-and-coming alternative to be investigated. Clemen (1989), Hibon and Evgeniou (2005) states that combining time series forecasts decreases the risk of choosing a single method with a poor fit to the data. Forecasting combinations gained prominence in the M4 competition (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2020). The first place was received by a hybrid method that combined the predictions of the exponential smoothing method with those of long short-term memory (LSTM) neural networks (SMYL, 2020). This result, among other factors, can be attributed to the great generalization capacity that the prediction combinations have (ZHOU, 2012). Wang *et al.* (2022a) points out as a perspective for future developments in forecasting combination the selection of which forecasts will be combined.

Another point of improvement in the time series forecasting process is the empirical and systematic performance validation experiments. In addition to a simple averaging analysis of

error measures, tools and methods capable of extracting knowledge about the variation in the performance of algorithms are lacking in the literature. In Figure 3, we illustrate an example of applying the same algorithm to two different time series.

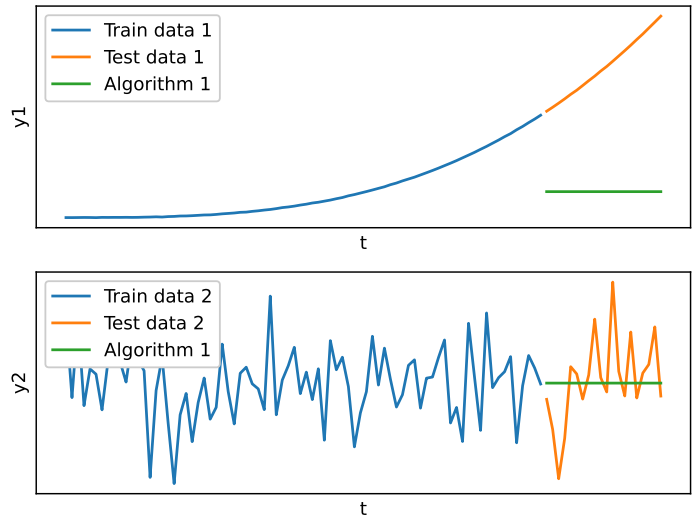


Figure 3 – An example of algorithm performance in different datasets.

An important issue in validating predictive performance is determining under what conditions an algorithm will or will not perform properly. In Figure 3, Algorithm 1 is applied to two data sets. For Data 1, Algorithm 1 cannot follow the general trend of the data, while for Data 2, the performance is much better. These results are usually analyzed by averaging performance measures. However, according to Smith-Miles and Muñoz (2022), we miss the opportunity to get more information about the learning process by looking only at summaries of performance measures.

In time series forecasting, only few works seek to understand the performance of algorithms. Some studies analyze meta-knowledge in metalearning applications to explain the selection of Armstrong, Adya and Collopy (2001), Lemke and Gabrys (2010), Talagala *et al.* (2018) algorithms. However, the need for more metadata in quantity and diversity prevents the extraction of more general knowledge from these approaches.

In other tasks, methodologies were developed to understand the performance of algorithms empirically and systematically. In work by Smith-Miles and Muñoz (2022), the Instance Space Analysis (ISA) methodology is applied to optimization problems. ISA is a methodology that explores the relationships between the complexity characteristics of the test instances and the impact on the performance of the algorithms. With this technique, a 2d plane represents the space of all possible test instances. Instance space analysis allows machine learning algorithms to be used to predict regions where good performance can be expected. The area of this region is called algorithm footprint and is a measure of the applicability and robustness of algorithms.

The dataset morphing (CORREIA; SOARES; JORGE, 2019) method was proposed for recommender systems. Dataset morphing is the generation of synthetic data from the gradual transformation of a source dataset into a target dataset. The synthetic datasets created between the source and the target are used to understand the variation in the predictive performance of a pair of algorithms in datasets. Allied with the performance variation, variations in characteristics of the synthetic datasets are also analyzed. The goal is to understand the predictive performance of algorithms systematically.

1.0.2 Objective and Hypotheses

The main objective of this thesis is to develop and analyze methods for algorithm selection and understanding the predictive performance of time series forecasting algorithms. From this objective, we delimited the scope of the research with two research questions:

- **Q1**: What is the effect of the use of metalearning to select hybrid forecasting combinations on the predictive and computational performance?
- **Q2**: Is it possible to empirically show under which conditions a forecasting algorithm will show a predictive performance?

Based on the research questions mentioned above, we formulated hypotheses that guided the development of methods and experiments throughout the work.

- **Hypothesis 1**: There is a performance measure that induces better predictive performance than the baselines, at base level and meta level. This hypothesis is related to Q1.
- **Hypothesis 2**: Hybrid forecasting combinations with good predictive performance are possible when using metalearning and decomposition methods. This hypothesis is related to Q1.
- **Hypothesis 3**: Seasonal and Trend decomposition using Loess as a pre-processing step can benefit machine learning algorithms when its residual follows a normal distribution. This hypothesis is related to Q2.
- **Hypothesis 4**: It is possible to understand the variation in the predictive performance of forecasting algorithms by using morphing. This hypothesis is related to Q2.

1.0.3 Thesis Organization

This thesis is a collection of papers written by the candidate during the doctoral period. As they are articles, the chapters are self-contained and organized in the order of the explored hypotheses, not necessarily representing the chronological order. Chapter 6 presents the final

remarks, limitations, and possible future work directions. Figure 4 illustrates the relationship between the chapters of the thesis and the problems addressed in the thesis.

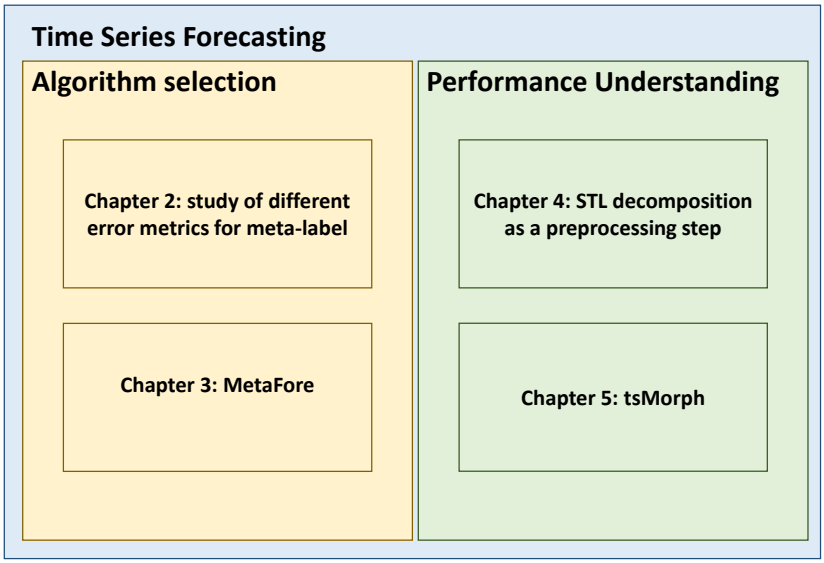


Figure 4 – Thesis organization diagram.

In Figure 4, Chapters 2 and 3 deal with solutions to the algorithm selection problem. Chapter 2 presents an experimental analysis of the predictive performance obtained using different error metrics to define the meta-label value. Chapter 3 proposes and empirically evaluates *MetaFore*, a new time series forecasting approach that uses seasonal trend decomposition with Loess and metalearning to recommend suitable algorithms for time series forecasting combinations.

Chapters 4 and 5 propose and investigate alternatives to understand predictive performance. In Chapter 4, we designed a procedure that uses the Seasonal and Trend decomposition using Loess as a preprocessing step to model the time series components separately using a machine learning algorithm and a seasonal naive forecaster. Chapter 5 proposes *tsMorph*, a simple method for generating synthetic time series based on dataset morphing. Given two datasets, tsMorph creates a sequence of datasets that gradually transform the two.

CHAPTER

2

EVALUATION OF ERROR METRICS FOR METALEARNING LABEL DEFINITION IN THE FORECASTING TASK

Reference: SANTOS, M.R.; MUNDIM, L.R.; CARVALHO, A.C.P.L.F. Evaluation of Error Metrics for Meta-learning Label Definition in the Forecasting Task. In: International Conference on Hybrid Artificial Intelligence Systems, XV, 2020, Gijon-Spain. Hybrid Artificial Intelligent Systems. Springer, Cham, 04 November 2020. p.397–409.

2.1 Abstract

Meta-learning has been successfully applied to time series forecasting. For such, it uses meta-datasets created by previous machine learning applications. Each row in a meta-dataset represents a time series dataset. Each row, apart from the last, is meta-feature describing aspects of the related dataset. The last column is a target value, a meta-label. Here, the meta-label is the forecasting model with the best predictive performance for a specific error metric. In the previous studies applying meta-learning to time series forecasting, error metrics have been arbitrarily chosen. We believe that the error metric used can affect the results obtained by meta-learning. This study presents an experimental analysis of the predictive performance obtained by using different error metrics for the definition of the meta-label value. The experiments performed used 100 time series collected from the ICMC time series prediction open access repository, which has time series from a large variety of application domains. A traditional meta-learning framework for time series forecasting was used in this work. According to the experimental results, the mean absolute error can be the best metric for meta-label definition.

2.2 Introduction

Time series are usually represented by a sequence of data points collected along with the time (CRYER; CHAN, 2008). One of the main tasks in time series analysis is forecasting, the prediction of the occurrence of future events (MONTGOMERY; JOHNSON; GARDINER, 1990). Time series forecasting is a crucial task of data analysis in many domains, such as finance, industry, government, health, and environment.

One of the main challenges in time series forecasting is the selection of the technique able to induce the best forecasting model for a given time series. Several techniques have been proposed, each with its bias, favoring particular data distributions. Thus, to select the most suitable technique from a set of options, it is necessary to explore an ample search space of available forecasting techniques and depends on domain expert knowledge, which has a high cost (MEADE, 2000), (ARMSTRONG, 2001), (PRUDÊNCIO; LUDERMIR, 2004). An alternative to reduce this cost is the use of meta-learning (MtL), which uses the knowledge obtained from past applications of machine learning (ML) algorithms in related tasks to create a predictive model able to recommend the most suitable technique(s) for a new dataset.

MtL has been successfully applied to algorithm recommendation and hyper-parameter selection (VILALTA; DRISSI, 2002),(SOUZA; SOARES; CARVALHO, 2009) (BRAZDIL *et al.*, 2008),(PIMENTEL; CARVALHO, 2019),(MANTOVANI *et al.*, 2019). Like in a conventional ML application, MtL applies a ML algorithm to a dataset, named meta-dataset. This learning process is called meta-level learning, to differ from the conventional learning process in ML, named base-level learning in the MtL literature. In the meta-dataset, each row represents a dataset and is labeled by either the ML algorithms performance applied to this dataset or by the forecasting model with the best predictive performance for a specific error metric. The predictive attributes in the meta-dataset are named meta-features. The values of the meta-features describe a dataset, highlighting important aspects that characterize the dataset. When an ML algorithm is applied to the meta-dataset, it induces a meta-model. Usually, the meta-model is a predictive model, which can be used, as part of a recommendation system, to predict the most suitable technique(s) for a new dataset. The ML algorithm used to induce the meta-model is named meta-learner.

Previous works have successfully applied in time series forecasting (PRUDÊNCIO; LUDERMIR, 2004), (LEMKE; GABRYS, 2010), (WIDODO; BUDI, 2013), (Kück; Crone; Freitag, 2016), (ALI; GABRYS; BUDKA, 2018), (BARAK; NASIRI; ROSTAMZADEH, 2019). However, in these works, the error metric as meta-label was arbitrarily defined, not considering other error metrics.

We believe that the error metric used can affect how well MtL works. This study evaluates the error metrics performance when used for the meta-label definition for several types of meta-learners. For such, we carried out experiments using a set of time series datasets collected from

the ICMC time series datasets repository (PARMEZAN; BATISTA, 2014)[1]. In the experimental analysis, we also look at the computational cost of each error metric used.

As base-learners, we used state of the art time series forecasting techniques (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2018a), based on ML and statistics: Autoregressive Integrated Moving Average (ARIMA) (BOX *et al.*, 2015), state space exponential smoothing (ETS) (HYNDMAN *et al.*, 2008), and neural networks autoregressive (NNAR) (MEDEIROS; TERÄSVIRTA; RECH, 2006). The meta-dataset was created extracting 23 meta-features based on basic statistics, statistical tests for time series, partial and autocorrelations function coefficients and frequency domain measures. For the meta-labels, the following error metrics were assessed: Mean Absolute Error (MAE), Mean Squared Error (MSE), Median Absolute Error (MedAE), Symmetric Mean Absolute Percentual Error (sMAPE), and Mean Absolute Scaled Error (MASE) (HYNDMAN *et al.*, 2006). The ML algorithms used to induce the meta-models were: K-Nearest Neighbors, Decision Tree, Random Forest, and Multilayer Perceptron. More information about these techniques can be found in (MARSLAND, 2014).

2.3 Time series forecasting

Given a time series $Y = y_1, y_2, ..., y_t$ and a forecasting horizon h, a time series forecasting model estimates the Y subsequent values y_{t+h}. Regarding h, the time series forecasting can be classified as one-step ahead or multi-step ahead. One-step ahead forecasting is the prediction of the time series subsequent value, where $h = 1$. Multi-step forecasting predicts many steps into the future, where $h > 1$ (MONTGOMERY; JOHNSON; GARDINER, 1990). This work focuses on the one-step forecasting. Time series forecasting can follow a univariate or multivariate approach. The first uses historical values of a single time series to predict future values. The second use, simultaneously, historical values of multiple time series (TSAY, 2013). Next, we present the main techniques used to induce models for time series forecasting.

2.3.1 Autoregressive integrated moving average

Autoregressive Integrated Moving Average (ARIMA) models (BOX *et al.*, 2015) are widely used for time series forecasting. The autoregressive component builds the interest variable prediction as a linear combination of p previous values. The moving average component is modeled as an error ε average of q previous values to predict the value of the variable of interest. The integrated component of ARIMA models is the inverse of differentiation (HYNDMAN; ATHANASOPOULOS, 2018a). ARIMA models can be generically described to Equation 2.1.

$$y'_t = c + \phi_1 y'_{t-1} + ... + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + ... + \theta_q \varepsilon_{t-q} + \varepsilon_t \quad (2.1)$$

[1] Datasets are available here

where y is the differenced time series, ϕ, and θ are the coefficients of autoregressive and moving average models, respectively, p and q are the model degrees. For simplicity, we can denote an ARIMA model as ARIMA(p,d,q), where p is the autoregressive part degree, d is the differentiation degree, and q is the moving average part degree.

2.3.2 Exponential smoothing state-space model

Exponential smoothing models are based on time series decomposition in a seasonal s and a trend t components. The trend component t can be expressed as a linear combination of a level term l and a growth term b (HYNDMAN *et al.*, 2008). A general state-space model is described by Equations 2.2 and 2.3.

$$y_t = w(x_{t-1}) + r(x_{t-1})\varepsilon_t \tag{2.2}$$

$$x_t = f(x_{t-1}) + g(x_{t-1})\varepsilon_t \tag{2.3}$$

where ε_t is a Gaussian white noise process and $x_t = (l_t, b_t, s_t, s_{t-1}, ..., s_{t-m+1})'$.

2.3.3 Neural network autoregression

Neural networks are ML algorithms inspired in the human brain, specifically in the processing and interaction between neurons (HAYKIN, 1994). Although there are several neural network architectures, the neural networks autoregressive (NNAR) (MEDEIROS; TERÄSVIRTA; RECH, 2006) is the most straightforward for time series forecasting. An NNAR is usually represented as NNAR(p,k), where p is previous inputs in a time series, and k is the number of neurons in the hidden layer. The NNAR weights are originally adjusted by the backpropagation algorithm. Given a NNAR(p,k) and a time series y. The values of output are given by Equation 2.4.

$$y_t = f(\sum_{j=1}^{k} w_j f(\sum_{i=1}^{p} w_{ij} x_i + b_{0j}) + b_0) \tag{2.4}$$

where w are the weights associated to neurons, $x = y_{t-p}, ..., y_{t-1}$ is a set of lagged input and b is the bias, which work like a intercept component of linear equations for make neural networks most robust.

2.4 Meta-learning for forecasting model recommendation

Among other implications, according to the "*no free lunch*" theorem (WOLPERT, 1996), there is no technique that provides the best model for every dataset. This motivates the use of MtL for algorithm recommendation. Formally, given a set of problem instances P represented

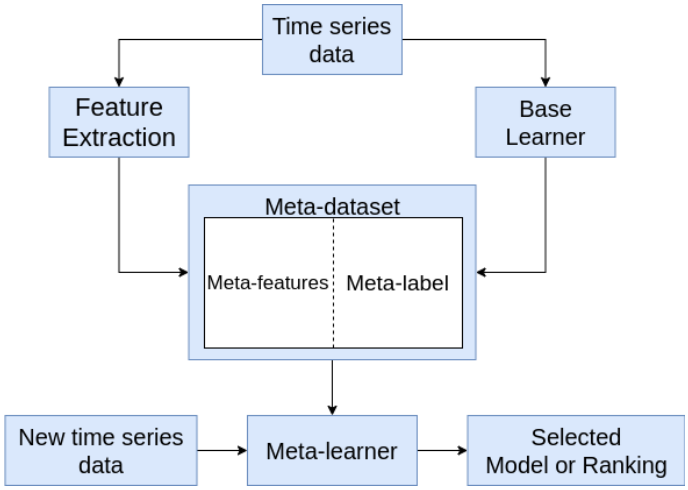


Figure 5 – MtL based recommendation system architecture

by a distribution of datasets D, a set of algorithms A and a performance metric m: $P \times A \rightarrow \mathbb{R}$, the algorithm recommendation problem consists of finding a mapping f: $P \rightarrow A$ that optimizes the expected performance measure regarding m for instances P with a distribution D (RICE, 1976). Figure 5 illustrates a architecture of a MtL based recommendation system for time series forecasting inspired on (PRUDÊNCIO; LUDERMIR, 2004) and (Kück; Crone; Freitag, 2016).

According to this figure, the MtL based model selection process can be described by its components. For such, it uses a meta-dataset from time series data where each meta-example represents a dataset by the values of meta-features (FE) extracted from the dataset and the predictive performance of different techniques, or the technique(s) with the best performance when applied to this dataset (BL). An ML algorithm can be applied to the meta-dataset, as a conventional ML experiment, to extract a predictive meta-model able to recommend, e.g., the most suitable technique for a new time series dataset.

2.5 Experimental setup

2.5.1 Time series data

The experiments carried out used 100 time series datasets from the ICMC time series prediction repository (PARMEZAN; BATISTA, 2014). These time series include synthetic, chaotic, real-world series, from several domains. These time series were selected to present explicitly open policy and domain diversity necessary for MtL.

2.5.2 Meta-features

To create the meta-dataset, we extracted from the 100 time series datasets 23 meta-features, divided in four groups:

- **Basic statistics**: length(Kück; Crone; Freitag, 2016), minimum(Kück; Crone; Freitag, 2016), maximum(Kück; Crone; Freitag, 2016), mean(Kück; Crone; Freitag, 2016), median(Kück; Crone; Freitag, 2016), standard deviation(Kück; Crone; Freitag, 2016), range(Kück; Crone; Freitag, 2016), skewness(Kück; Crone; Freitag, 2016), kurtosis(Kück; Crone; Freitag, 2016), variation coefficient(PRUDÊNCIO; LUDERMIR, 2004), upper quartile(Kück; Crone; Freitag, 2016), lower quartile(Kück; Crone; Freitag, 2016), turning points(PRUDÊNCIO; LUDERMIR, 2004), trend(PRUDÊNCIO; LUDERMIR, 2004);
- **Statistical tests**: Spearman correlation coefficient(Kück; Crone; Freitag, 2016), Mann-Kendall trend test(Kück; Crone; Freitag, 2016), Kruskal–Wallis test(Kück; Crone; Freitag, 2016), Durbin-Watson test(Kück; Crone; Freitag, 2016);
- **Autocorrelation**: lags one and two of autocorrelation function(PRUDÊNCIO; LUDERMIR, 2004), lags one and two of partial autocorrelation function(HYNDMAN; ATHANASOPOULOS, 2018a);
- **Frequency domain**: real part of the Fast Fourier Transform (FFT) maximum value(PRUDÊNCIO; LUDERMIR, 2004).

2.5.3 Base-learners

The base learners used are implemented in the R package "*forecast*" (HYNDMAN; KHANDAKAR *et al.*, 2007). This package includes several techniques to induce forecasting models, such as ARIMA, ETS, and NNAR, which are briefly described next.

- **ARIMA**: produces automatically tuned ARIMA models for univariate time series data. The implementation used in this work was *auto.arima*;
- **ETS**: produces an automatic exponential smoothing state-space model. The implementation *ets* was used;
- **NNAR**: trains feed-forward neural networks NNAR with one hidden layer for univariate time series forecasting. After several tests, we decided to use 25 neurons in the hidden layer. For the normalization of the time series input, we applied the Box-Cox transformation from *nnetar* to the datasets.

2.5.4 Error metrics meta-label

The meta-model recommends, for new time series datasets, the most suitable techniques according to specific error metric, the meta-label. This study investigates how the error metric used affects the recommendations from the meta-model. As five error metrics were investigated, five meta-datasets were created, each one using a different error metric to define the meta-label values.

Given y, the test time series values, $\hat{y}$ are predicted values and n the test set length. The error metrics investigated in this work were:

- **Mean Absolute Error (MAE) Equation 2.5**: magnitude of prediction errors;
- **Mean Squared Error (MSE) Equation 2.6**: average squared of prediction errors;
- **Median Absolute Error (MedAE) Equation 2.7**: median magnitude of prediction errors;
- **Symmetric Mean Absolute Percentage Error (sMAPE) Equation 2.8**: percentage absolute difference of predictions errors. This is symmetric due to original MAPE not work for test set values close to zero (WIDODO; BUDI, 2013).
- **Mean Absolute Scaled Error (MASE) Equation 2.9**: scales the errors based on MAE from the naive forecast method. This naive forecast method defines prediction value as the one-step past value.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n} \tag{2.5}$$

$$MSE = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n} \tag{2.6}$$

$$MedAE = median(|y_i - \hat{y}_i|) \tag{2.7}$$

$$sMAPE = \frac{100}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{|y_i| + |\hat{y}_i|} \tag{2.8}$$

$$MASE = mean\left(\left| \frac{|y_i - \hat{y}_i|}{\frac{1}{n-1} \sum_{i=2}^{n} |y_i - y_{i-1}|} \right| \right) \tag{2.9}$$

2.5.5 Meta-learner

The following meta-learners were used for the experiments carried out in this study. They were selected to cover different learning biases.

- **K-Nearest Neighbors (KNN)**: simple technique based on lazy learning. We used the *caret* (KUHN, 2012) R package;
- **Decision Tree induction algorithm (DT)**: Induce decision trees using divide and conquer. We used the Classification And Regression Trees (CART) algorithm (BREIMAN *et al.*, 1984a) implementation in the *rpart* R package.

- **Random Forest (RF)**: ensemble technique that induces a set decision tree classifiers on various dataset sub-samples. The implementation used was from the *randomForest* (LIAW; WIENER, 2002) R package.

- **Multilayer Perceptron (MLP)**: fully connected feedforward artificial neural networks. The implementation came from the *RSNNS* (BERGMEIR; SÁNCHEZ *et al.*, 2012) R package.

2.5.6 Meta-level evaluation

The experiments with the 100 time series datasets generated meta-datasets. For the meta-learning experiments, each meta-dataset was divided into training (80%) and test subsets (20%), keeping the class proportion. The MtL experiments partitioned the meta-dataset using Leave-One-Out (LOO). For KNN and MLP, the predictive attributes were centralized and scaled to unit variance.

Hyperparameters specific to each technique were tuned using the *caret* (KUHN, 2012) basic hyperparameter tuning technique, with at most 10 tuning interactions: value of k for the KNN technique, complexity for DT, number of variables sampled as a candidate at each split for RF, and number of hidden layer neurons for MLP with a one-hidden layer. The meta-models induced by each technique had their predictive performance assessed using accuracy, as in the related works found in the literature. Each experiment was repeated 30 times, and the Friedman rank sum statistical hypothesis and Nemenyi, multiple comparison post hoc tests, were applied to all results with a p-value < 0.05. The Intel Math Kernel Library for R code was used for performance optimization.

2.5.7 Forecasting level evaluation

The labels predicted for the test data in the meta-level experiment were used for the experimental analyses at the base level. The meta-level predictive performance was measured by the mean and standard deviation of the results obtained by all the meta-learners. Thus, the focus of the first analysis is on the meta-level, i.e., the predictive performance of the recommendation framework.

On the forecasting level, the evaluation focus on whether the MtL use improves the predictive performance of forecasting techniques when compared with the use of a baseline. In this study, the baseline is the average of the predictive performance obtained by the forecasting techniques. Due to intuitive interpretation of the number of times a technique was better than others, Percentage Better (PB) (SHAH, 1997), described by Equations 2.10 and 2.11, is used.

$$PB = \frac{1}{m} \sum_{j=1}^{n} \delta_j \tag{2.10}$$

$$\begin{cases} \delta_j = 1, & \text{if } |e_j^R| \leq |e_j| \\ \delta_j = 0, & \text{otherwise} \end{cases} \tag{2.11}$$

In these equations, δ is incremented when the magnitude of reference error is less than or equal to the magnitude of the MtL predicted error. The variable m is the number of times that these errors are different. Thus, when PB is close to 0, the MtL prediction is better than the reference (baseline), when close to 50, it is similar, and when more than 50, it is worse.

2.5.8 Time complexity analysis of the error metrics

For time complexity analysis, the "microbenchmark" R package (MERSMANN *et al.*, 2015) was used. Error metrics are tested with two synthetic arrays with a length of 10^2 and run 10 times.

2.6 Experimental results

The previous section described the experimental setup adopted to compare the error metrics performance for meta-label prediction in univariate one-step ahead time series forecasting. This section presents the main experimental results. These results aim to answer the two fundamental questions of this study:

1. Does the MtL predictive performance varies for different meta-learners and distinct error metrics?
2. Among the error metrics with the best results for MtL, for each meta-learner, is there any difference in time complexity?

This section presents the results for the MAE, sMAPE, and MASE error metrics for the meta-target selection. Figures 6, 7, and 8 shown the results obtained.

These results show that the results for MedAE and MSE error metrics are redundant since there is a strong Spearman correlation with the MAE measurements for three meta-learners.

2.6.1 Meta-learner performance for meta-label error metrics

2.6.1.1 KNN performance for meta-label error metrics:

Figures 9 and 10 show the KNN meta-learner performance for each error metric used, in the meta and the forecasting level, respectively. On average, the MAE metric presented the best performance in the meta and forecasting levels. However, for both levels, there is no statistically significant difference between MAE and MASE.

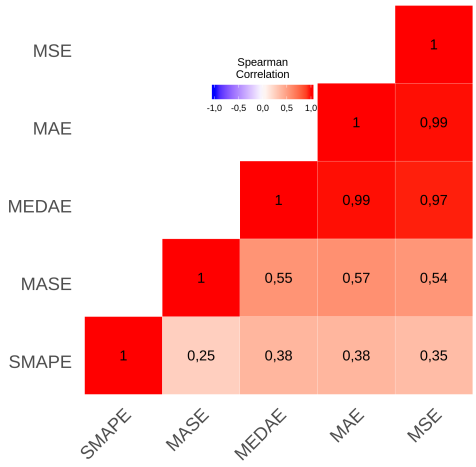


Figure 6 – ARIMA Spearman corr.

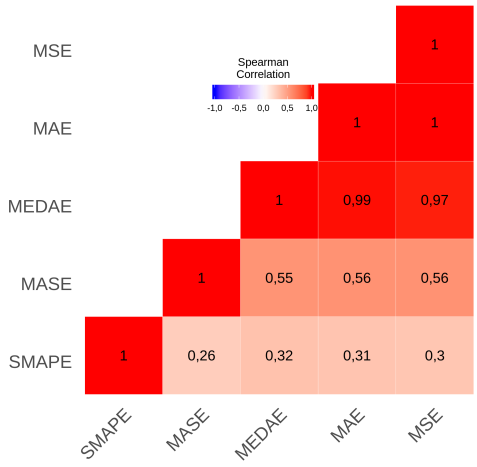


Figure 7 – ETS Spearman corr.

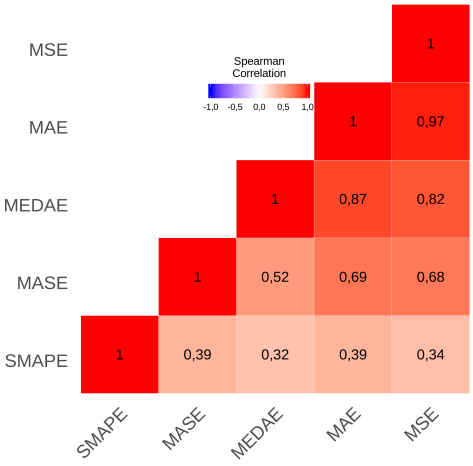


Figure 8 – NNAR Spearman corr.

2.6.1.2 DT performance for meta-label error metrics:

Figures 11 and 12 show the DT meta-learner performance for each error metric used in the meta-label definition in the meta-level and in the forecasting level, respectively. The sMAPE metric presented a better performance in the meta-level, but without significant difference regarding MAE and MASE. For the base level, the best performance was obtained by the MASE metric.

2.6.1.3 RF performance for meta-label error metrics:

Figures 13 and 14 show the RF meta-learner performance for each error metric used in the meta-label definition in the meta and in the forecasting level, respectively. The sMAPE metric presents a superior performance on average in the meta and forecasting levels. The meta-level has no statistically relevant difference for all metrics. In the forecasting level, we did not observe a statistical difference between the results obtained using sMAPE, MAE, and MASE.

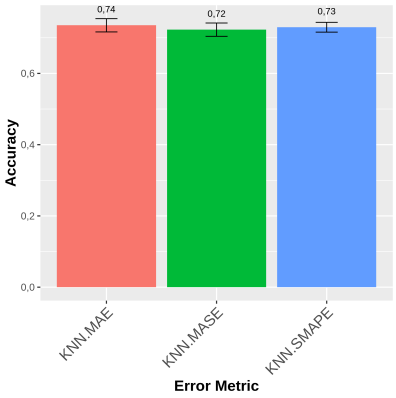


Figure 9 – Accuracy KNN

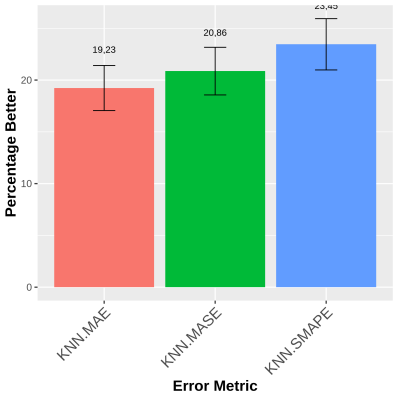


Figure 10 – PB KNN

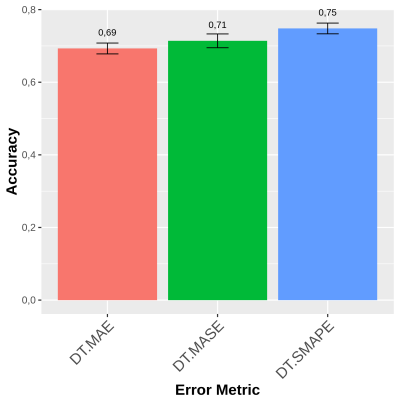


Figure 11 – Accuracy DT

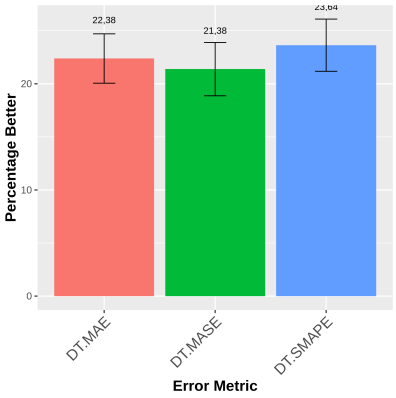


Figure 12 – PB DT

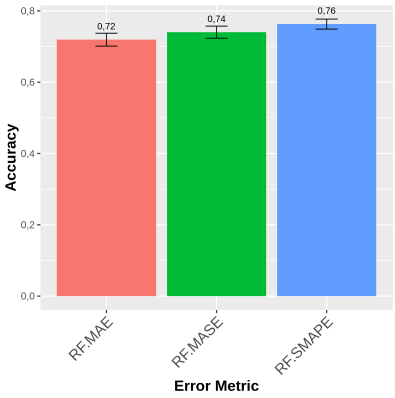


Figure 13 – Accuracy RF

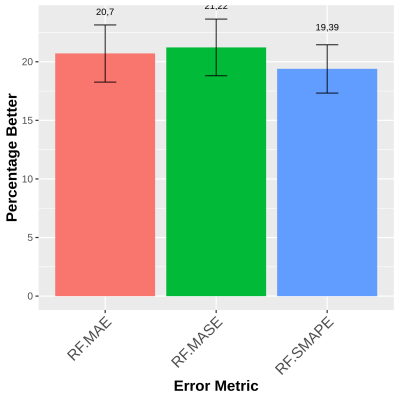


Figure 14 – PB RF

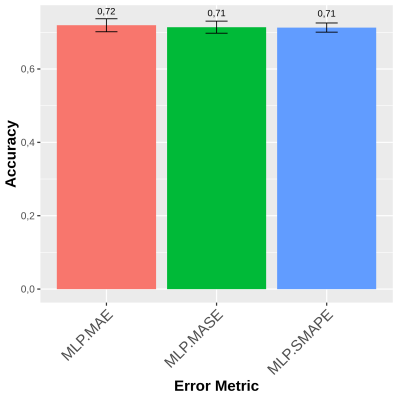


Figure 15 – Accuracy MLP

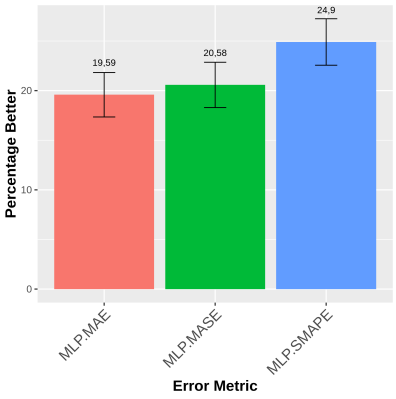


Figure 16 – PB MLP

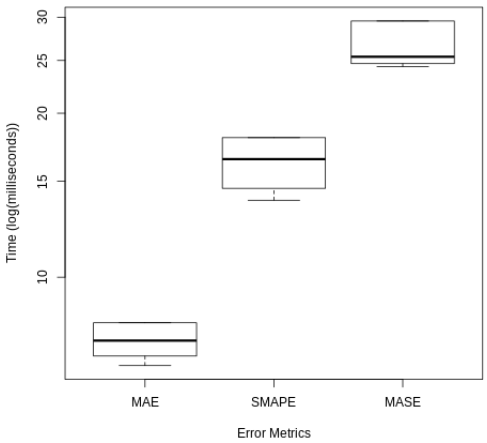


Figure 17 – Error metrics time complexity

2.6.1.4 MLP performance for meta-label error metrics:

Figures 15 and 16 show the MLP meta-learner performance for each error metric used in the meta-label definition, for the meta-level and for the base level, respectively. On average, the MAE metric showed the best performance at both these levels. However, in these meta-level has no relevant difference between all the error metrics. In the forecasting level, has no significant difference between the MAE and MASE metrics.

2.6.2 Time complexity evaluation

Figure 17 shows the boxplot of the time complexity of the three error metrics previously analyzed. In this figure, the abscissa coordinate represents the execution time (milliseconds) in the logarithmic scale. The ordinate coordinate is the error metric.

According to Figure 17, MAE presented the lowest running time among the error metrics evaluated. In terms of analysis of the average relative time, the cost of sMAPE is twice the cost of MAE, and the cost of MASE is three times higher than the cost of MAE. This occurred because sMAPE, and MASE are MAE-based metrics. Thus, they have more operations than MAE, making their computational cost higher.

2.7 Conclusion

This study evaluates the effect of predictive performance metrics on the recommendation of time series forecasting techniques using MtL. According to the experimental results, Regarding the error metrics MAE, sMAPE, and MASE, Mtl produced the best predictive performance, when compared with a baseline, for all tested scenarios.

For most of the meta-learners, the MAE metric presented, on average, the best predictive

performance, despite being the metric with the lowest cost. In the time complexity analysis, the MAE obtained the best performance when compared with the error metrics. Thus, MAE should be used in scenarios requiring low computational costs without performance loss.

A limitation of this work is the low number of meta-examples in the meta-dataset. Although this fact did not impair the MtL process in our experiments, a larger meta-dataset would provide more significant experimental results. Thus, for the future, we want to increase the size of the meta-dataset. Another future work directions are to include meta-features that extract other aspects of time series and to investigate the effect of meta-feature selection on the MtL performance.

Acknowledgments

This study was partially funded by the Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) - Process 2019/10012-2 and Intel Inc.

CHAPTER

3

METAFORE: ALGORITHM SELECTION FOR DECOMPOSITION FORECASTING COMBINATIONS

Reference: This work was submitted to the "Information Sciences" journal. The current status is "Under review.".

3.1 Abstract

Time series forecasting is an important tool for planning and decision-making. Considering this, several forecasting algorithms can be used, with results depending on the characteristics of the time series. The recommendation of the most suitable algorithm is a frequent concern. Metalearning has been successfully used to recommend the best algorithm for a time series analysis task. Additionally, it has been shown that decomposition methods can lead to better results. Based on previously published studies, in the experiments carried out, time series components were used. This work proposes and empirically evaluates *MetaFore*, a new time series forecasting approach that uses seasonal trend decomposition with Loess and metalearning to recommend suitable algorithms for time series forecasting combinations. Experimental results show that MetaFore can obtain a better predictive performance than single models with statistical significance. In the experiments, MetaFore also outperformed models widely used in the state-of-the-art, such as the Long Short-Term Memory neural network architectures, in more than 70% of the time series tested. Finally, the results show that the joint use of metalearning and time series decomposition provides a competitive approach for time series forecasting.

3.2 Introduction

The predictive performance of time series forecasting methods is affected by several aspects, such as trend, seasonality, and number of attributes. The fast increase in the data volume, sources and complexity make their analysis more challenging. A recent approach to deal with these challenges is to decompose the series into components, modeling each component separately and then combining their output into a final forecast. One challenge of this approach is which method to use for each of the decomposed time series.

Forecasting is one of the main tasks of time series analysis (HYNDMAN; ATHANASOPOULOS, 2018a). It is a set of essential tools for planning and strategic decision-making in various areas such as the financial sector, industry, and public health. Time series forecasting was a field dominated by essentially statistical methods that assumed distribution for the data. However, in the last decade, machine learning (ML) algorithms have achieved very competitive results (DEB *et al.*, 2017; PARMEZAN; SOUZA; BATISTA, 2019c).

When considering the full range of candidate methods for the forecasting task, the expert is faced with choosing the best method. However, testing all possibilities involves a high computational cost and high processing time. In this context, metalearning emerges as an alternative by using the learning acquired in past tasks and ML algorithms to recommend promising algorithms for performing new tasks (BRAZDIL *et al.*, 2008).

Recent research studies have investigated hybrid approaches, when two or more methodologies are used in addition to using single models. These approaches have presented state-of-the-art results in time series forecasting (SMYL, 2020). A topic widely explored is hybrid forecast combinations from time series decomposition. A decomposition method example is Seasonal Trend decomposition using Loess (STL), which presents good predictive performance and stability (GURNANI *et al.*, 2017; SILVESTRE; SANTOS; CARVALHO, 2021).

However, hybrid forecasting approaches usually choose algorithms for each component arbitrarily, based on expert knowledge of the time series and the inductive bias of candidate models. This study aims to empirically analyze the use of metalearning for the automatic recommendation of algorithms for decomposed univariate times series.

The main contributions of this study are: to propose MetaFore: a metalearning approach for forecasting decomposed time series; empirically show the potential of MetaFore in time series of real applications with different domains; analysis of the meta-knowledge obtained through partial dependence plots.

This paper is structured as follows. Section 3.3 defines some essential concepts for designing the approach to the problem; Section 3.4 discusses related work; Section 3.5 introduces MetaFore; In Section 3.6, the experimental setup is described; The results are presented in Section 3.7 and in Section 3.8 the conclusions of the study and future perspectives are shown.

3.3 Background

This section describes the main concepts that are important to follow this work. First, we introduce the main aspects of time series analysis, the STL decomposition, and the forecasting task. Then, we will introduce the idea of recommending models by metalearning.

3.3.1 Time series analysis

Time series can be classified according to the number of variables observed simultaneously. When only one variable of interest is observed in a time series, it is univariate. A univariate time series $Y_t = \{y_t\}_{t=1}^{T}$ is a sequence of observations that are ordered and evenly spaced in time (MONTGOMERY; JENNINGS; KULAHCI, 2015a). When two or more variables of interest, related to the same phenomenon, are observed, it is a multivariate time series.

A time series can also be characterized regarding its frequency, usually daily, weekly, monthly, or annually, when each observation covers one day, week, month or year, respectively. We can also have high-frequency time series, when observations are collected in periods of hours, seconds or even shorter intervals (ANDERSEN, 2000).

A univariate time series Y_t can be divided into three main components:

- **Trend** (T_t): a low-frequency component that describes a long-term direction of the series. It can assume linear or non-linear patterns.
- **Seasonality** (S_t): a cyclical pattern that repeats itself over a period. Multiple seasonal patterns can be present in a time series.
- **Residue** (R_t): a time series component, usually of high frequency, whose expected property is to be random.

Different representations of a time series Y_t can be obtained using T_t, S_t, or R_t (HYNDMAN *et al.*, 2008). Their combination can be additive, defined by Equation 3.1, in the sense that aspects from different components are added to form a new time series.

$$Y_t = T_t + S_t + R_t \tag{3.1}$$

We can also combine the components using a multiplicative combination, as expressed by Equation 3.2.

$$Y_t = T_t \times S_t \times R_t \tag{3.2}$$

where the three components are combined by a product operator.

The inverse can also occur, when a time series is decomposed in its basic components, using a combination of components. Decomposition techniques assume that it is possible to find a function able to decompose a time series into the T, S and R components, as expressed in Equation 3.3.

$$f(Y_t) = (T_t, S_t, R_t) \tag{3.3}$$

Different techniques have been proposed to decompose a time series into these three components, such as Seasonal Extraction in ARIMA Time Series (SEATS) and the X-11 method (DAGUM; BIANCONCINI, 2016). For reasons to be explained later, we will adopt the Seasonal Trend decomposition using Loess (STL) in this work.

The STL method additivelly decomposes a time series Y_t by sequentially applying local regression smoothers (Loess) to the time series.

Loess applies local linear regression to a time series to reduce the effect of extreme observations or outliers. To do this, at each iteration of the local regression, its weights are adjusted to penalize observations with large errors during curve estimation (CLEVELAND *et al.*, 1990).

The STL decomposition uses two iterative procedures, one nested within the other. In each inner loop iteration, the seasonality and trend components are updated once. The outer loop has one or two inner loop iterations, followed by the identification of extreme values. In the next inner loop, the weights of extreme observations are decreased. The complete process takes between 10 and 20 iterations (MAKRIDAKIS; WHEELWRIGHT; HYNDMAN, 1998).

The advantages of using the STL, when compared with the other decompositions previously mentioned are: it supports seasonal components varying over time; it is robust to outliers; and it can control the smoothness of the trend. Its main disadvantage is not being able to handle calendar variation or specific events (HYNDMAN; ATHANASOPOULOS, 2018a).

Time series forecasting tasks use known observations to extrapolate observations to the future. Formally, given a time series $Y_t = \{y_t\}_{t=1}^{T}$, they look for a function f such that $f(t) = \hat{y}_{t+h} \approx y_{t+h}$, $\forall t + h > T$ (HYNDMAN; ATHANASOPOULOS, 2018a). The mapping $\mathscr{A} : \{y_t\}_{t=1}^{T} \rightarrow f$ is a forecasting model. Forecasting models assume the existence of a probability measure $\mu(y_{T+h}|\{y_t\}_{t=1}^{T})$, where h is the forecast horizon. When $h = 1$, the forecasting is one-step ahead forecasting, and when $h > 1$, it is multi-step ahead. Thus, these models assume that there is a direct relationship between past observations and a future observation. To do this, different approaches can be used by forecasting models.

3.3.2 Metalearning for model selection

In this section, we will formally define how metalearning can be used for model selection. Concerning this, let $\mathscr{P}$ be the space or a collection of datasets d of a task such that $d \in \mathscr{P}$. Let $\mathscr{A}$ be the space or portfolio of algorithms L applicable to tasks of P such that $L \in \mathscr{A}$. The mapping $\mathscr{S} : \mathscr{P} \rightarrow \mathscr{A}$ applies all L algorithms to all datasets d. From the $\mathscr{S}$ mapping, the space of performance measures $p : \mathscr{A} \times \mathscr{P} \rightarrow \mathbb{R}^n$ can be accessed. The Algorithm Selection Problem (ASP) searches in $\mathscr{S}$ for an algorithm L that optimizes a performance measure of p given a dataset d (RICE, 1976).

Depending on the complexity and the number of algorithms in the portfolio $\mathscr{A}$, the search for an algorithm L that satisfies the desired predictive performance can have a high cost. An alternative to reduce this cost of exploring all possibilities in $\mathscr{S}$ is to use the metalearning.

Metalearning, or learning to learn, is the set of methods that uses knowledge extracted from tasks, algorithms, or performance evaluation of algorithms to improve predictive performance and make it faster and more efficient in the future (BRAZDIL *et al.*, 2008; FINN; ABBEEL; LEVINE, 2017). As shown in (HUTTER; KOTTHOFF; VANSCHOREN, 2019a), there are several approaches to metalearning. In this study, we will use metalearning for algorithm recommendation based on meta-features.

Meta-feature-based metalearning uses an ML approach to relate the characteristics present in datasets to the relative performance of algorithms applied to them (BRAZDIL *et al.*, 2008). For the sake of simplicity, in this work, we will simplify meta-feature-based metalearning to just metalearning. Figure 18 presents a general metalearning diagram with its main steps.

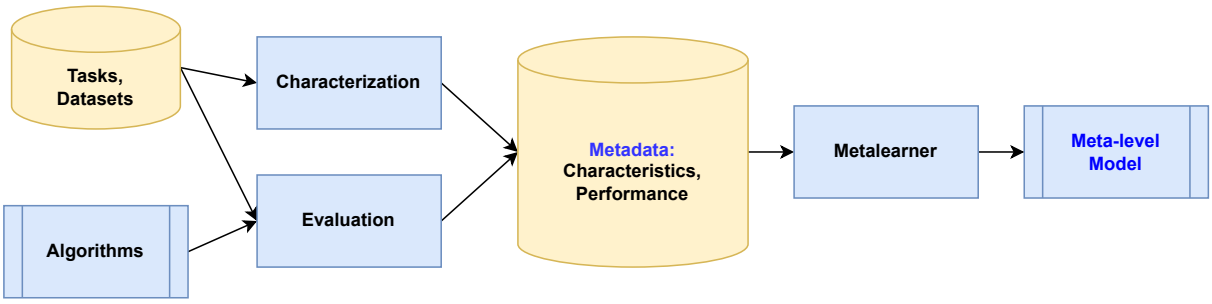


Figure 18 – Metalearning General Diagram based on (BRAZDIL *et al.*, 2022a).

On the left side of the diagram, we have the two main inputs for a metalearning-based solution, at the top a set of n tasks, represented by n datasets $D = \{d_0, d_1, \ldots, d_n\}$, and at the bottom, a portfolio of m candidate algorithms or base learners $L = \{l_0, l_1, \ldots, l_m\}$. Next, we have the characterization of the datasets and the evaluation of the algorithms when applied to the datasets.

The characterization is the extraction of characteristics, called meta-features, from the datasets. A large number of meta-features, based on different approaches, have been proposed.

In (RIVOLLI *et al.*, 2018), the authors formally define a meta-feature f as a function $f: D \rightarrow \mathbb{R}^k$. Function f returns a set of k values that characterize the dataset d_i. The evaluation step applies the base learners L to the datasets D to obtain a performance measure $p(L,D)$. The recommendation to be learned for each dataset, represented by meta-features, is called meta-target. The value of the meta-target is defined according to p and the expected output of the recommendation system, which can be: an algorithm, a set of algorithms, a ranking of algorithms, or a performance prediction (BRAZDIL *et al.*, 2008). The combination of the meta-features with the meta-target forms the metadata.

The next step in the diagram is the application of a ML algorithm to the metadata, inducing a meta-model (BRAZDIL *et al.*, 2008). When a new dataset, in our case, a new time series, needs to be predicted, the meta-model is applied to the meta-feature values representing this dataset to recommend one of the ML algorithms in the portfolio for this task.

3.4 Related work

In this section, we briefly describe previous studies investigating the use of metalearning to recommend ML algorithms for time series forecasting and the combination of different approaches for hybrid forecasting.

3.4.1 Metalearning for time series forecasting

The approaches proposed by (MEADE, 2000; PRUDÊNCIO; LUDERMIR, 2004) were the first to use metalearning to recommend time series forecasting models. (MEADE, 2000) investigated the selection of single statistical models and committees. To evaluate the performance of their metalearning approach, the authors used data from the M-Competition (MAKRIDAKIS *et al.*, 1982) and telecommunication-based time series. (PRUDÊNCIO; LUDERMIR, 2004) analyzed the use of meta-features for the recommendation of statistical methods rankings for time series forecasting. The experiments used data from the M3 competition (MAKRIDAKIS; HIBON, 2000). Both studies obtained promising results towards the effective application of metalearning for selecting time series forecasting models.

Many different approaches were proposed after these works. They can be characterized according to several dimensions. In (MEADE, 2000; PRUDÊNCIO; LUDERMIR, 2004; LEMKE; GABRYS, 2010; WIDODO; BUDI, 2013; Kück; Crone; Freitag, 2016; ALI; GABRYS; BUDKA, 2018), the authors used their own sets of meta-features, which were based on descriptive statistics, statistical tests, frequency domain measures (based on Discrete Fourier Transform), autocorrelation coefficients and relative performance of models. The work of (MA; FILDES, 2020) developed a deep learning approach to learn a meta-level representation of a time series using Convolutional Neural Networks.

Regarding the recommendation strategy based on metalearning, three trends were found in the literature: recommendation of the best model or ranking, recommendation of a combination of models, and prediction of model performance. The approaches by (MEADE, 2000; PRUDÊNCIO; LUDERMIR, 2004; WIDODO; BUDI, 2013; Kück; Crone; Freitag, 2016; ALI; GABRYS; BUDKA, 2018; BARAK; NASIRI; ROSTAMZADEH, 2019; TALAGALA *et al.*, 2018) used the selection strategy of the best model or ranking. The papers (MEADE, 2000; LEMKE; GABRYS, 2010; MONTERO-MANSO *et al.*, 2020; MA; FILDES, 2020; VAICIUKYNAS *et al.*, 2021) built model combinations from metalearning. In most of the studies, the predictive performance of model combinations was compared with the single models, and the combinations obtained the best results. Another approach is the prediction of model performance for a given time series observed in the study by (TALAGALA; LI; KANG, 2019; MONTERO-MANSO *et al.*, 2020; MA; FILDES, 2020). Moreover, considering combination strategies, the study (CERQUEIRA *et al.*, 2017) uses metalearning to dynamically combine models to predict the error of the combination of models within a given time series.

Neural networks are used in many approaches but are the only ML model on the base level in many frameworks. Despite this, ML models presented competitive results when compared to statistical models (PARMEZAN; SOUZA; BATISTA, 2019c). Most of the work focuses on using statistical methods as base learners. In the study by (MA; FILDES, 2020), other ML algorithms are added as base learners: extreme learning machines, Random Forest (RF), Support Vector Machine (SVM), and gradient boosting.

3.4.2 Hybrid forecasting

Recent works in hybrid forecasting propose new alternatives of combining forecasts and applying them to real-world time series analysis problems. The approaches proposed by (GURNANI *et al.*, 2017; LI *et al.*, 2020c; DUDEK; PEŁKA; SMYL, 2021) developed methods for specific domains, drugstore sales forecasting, dam displacement, and midterm load forecasting, respectively. Moreover, the methods proposed by (SMYL, 2020; SILVESTRE; SANTOS; CARVALHO, 2021) were applied to several application domains.

In these studies, the methods used to decompose the time series are exponential smoothing statistical and STL decomposition. In (SMYL, 2020; DUDEK; PEŁKA; SMYL, 2021), the authors use the exponential smoothing method to separate and forecast the seasonality component, while using another method for the other components. The studies published in (GURNANI *et al.*, 2017; LI *et al.*, 2020c; SILVESTRE; SANTOS; CARVALHO, 2021) use STL decomposition to separate a time series into the three components and use different methods for each component forecasting.

The studies that used STL decomposition (GURNANI *et al.*, 2017; LI *et al.*, 2020c; SILVESTRE; SANTOS; CARVALHO, 2021) described that, in addition to the present state-of-the-art predictive results, the hybrid forecasts showed stability and robustness. The method that

uses exponential smoothing (SMYL, 2020) was the best in the M4 competition, when it was compared with several other methods for different forecasting horizons.

A review article (WANG *et al.*, 2022b) with a 50-year overview of the time-series forecasting combination discussed the evolution of research in this area. The complexity of the combination methods does not necessarily increase the predictive performance. Another point raised in the article is that certain types of combination are suitable only in certain situations, and therefore future work should focus on determining which situations the proposed methods improve performance. The article also pointed out several directions for future research in this area.

3.5 The MetaFore approach

Given a time series data $\mathscr{Y} = \{Y_0, Y_1, \ldots, Y_n\}$ and a new time series Y_{new}. The MetaFore framework forecasts the new time series Y_{new} as illustrated in Figure 19. Figure 19 shows that the proposed approach is based on two main steps: the metadata collection step and the model recommendation step.

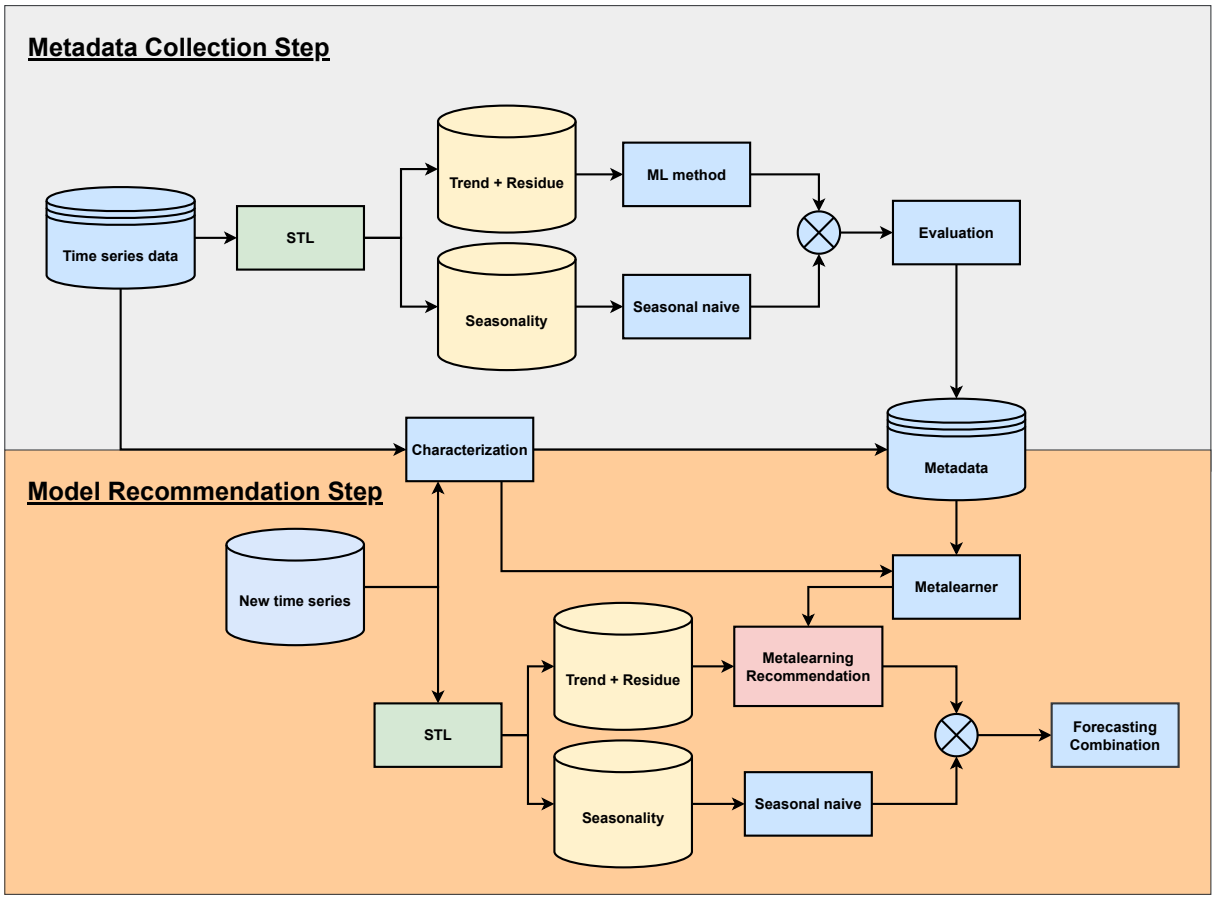


Figure 19 – MetaFore approach Diagram.

In the metadata collection phase, we carry out the STL decomposition for each time series

of $\mathscr{Y}$. The trend, seasonality, and residual components are separated in the STL decomposition. In MetaFore, trend and residual are combined and treated separately from seasonality. The combined trend and residual components are modeled using ML algorithms. The seasonality component is modeled with the seasonal naive method. The choice of the seasonal naive method for the seasonality component was due to previous successful studies (GURNANI *et al.*, 2017; SILVESTRE; SANTOS; CARVALHO, 2021). At the same time, we characterized the time series $\mathscr{Y}$ using a set of meta-features.

Component forecasts are carried out and combined after modeling by their respective methods. From the forecasting combination, the predictive performance is obtained. The meta-target is the base learner that achieves the best predictive performance. Thus, the problem is cast as a classification task.

In the model recommendation phase, we forecast the time series Y_{new} based on the meta-knowledge obtained from the metadata. To do this, we performed the STL decomposition step as was done for the time series in $\mathscr{Y}$. The metalearner recommends an algorithm for the combined trend and residual components. At the same time, the seasonality component is modeled by the naive seasonal method. MetaFore returns the additive combination of the predictions of the recommended algorithm and the seasonal naive method.

3.6 Materials & Methods

This section describes the datasets and the methods used to implement and evaluate MetaFore.

3.6.1 Time series data

The experiments carried out in this study use the monthly time series from the M4 competition (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2020). There are 48,000 time series from different real applications such as micro and macro economy, industry, finance, demographic, and others. The M4 competition time series, as in, (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2020) has been scaled so that there are neither negative observations nor below value 10 to avoid possible errors in metric calculations. Some evaluation measures, such as the Mean Average Percentage Error (MAPE), do not work when a time series has values equal to zero (WIDODO; BUDI, 2013).

Unlike other benchmarks datasets, such as the dataset used for the M3 competition, the M4 competition dataset has real world data (SPILIOTIS *et al.*, 2020). We used only the monthly frequency because, according to (SPILIOTIS *et al.*, 2020), in most time series forecasting applications, organizations are more interested in forecasting monthly series.

3.6.2 Meta-features

As meta-features are able to characterize time series, the time series features from the `Tsfresh` (CHRIST *et al.*, 2018) package, available in python, have been successfully used in several applications, such as activity recognition from synchronized sensors (KEMPA-LIEHR *et al.*, 2019), volcanic eruption forecasting (DEMPSEY *et al.*, 2020) and sensor anomaly detection (TEH *et al.*, 2021). Their use in these applications show their relevance.

Among others, the meta-features extracted by `Tsfresh` include measures from descriptive statistics, trend, frequency domain, Benford correlation, auto-correlation, and entropy. For this study, we extracted 783 meta-features from this package, avoiding meta-features with high computational cost (with the parameter *default_fc_parameters* set to `efficient`).

3.6.3 Base-Learners

The following regression algorithms were used as base-learners: Light Gradient Boosting Machine (LGBM) (KE *et al.*, 2017), K Nearest Neighbors Time Series Prediction with Invariances (KNN) (PARMEZAN; SOUZA; BATISTA, 2022), Support Vector Regression (SVR) (SMOLA; SCHÖLKOPF, 2004), Random Forest Regressor (RF) (HO, 1995), Bayesian Kernel Ridge Regressor (BKR) (TIPPING, 2001), and Linear Regression (LINREG) (LAI; ROBBINS; WEI, 1979). They were chosen because they are widely used for time series regression and forecasting tasks with very good results reported in the literature (PARMEZAN; SOUZA; BATISTA, 2019a; MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2022; PARMEZAN; SOUZA; BATISTA, 2022).

The `sktime` package was used to support the induction of the regressors for the time series forecasting task. The support includes a reduction procedure to transform the time series into a supervised learning task. The recursive strategy is used for multi-step ahead forecasting. As an example, given the time series $Y = \{1,2,3,4,5,6,7\}$, Figure 20 illustrates the reduction procedure and the recursive forecasting process.

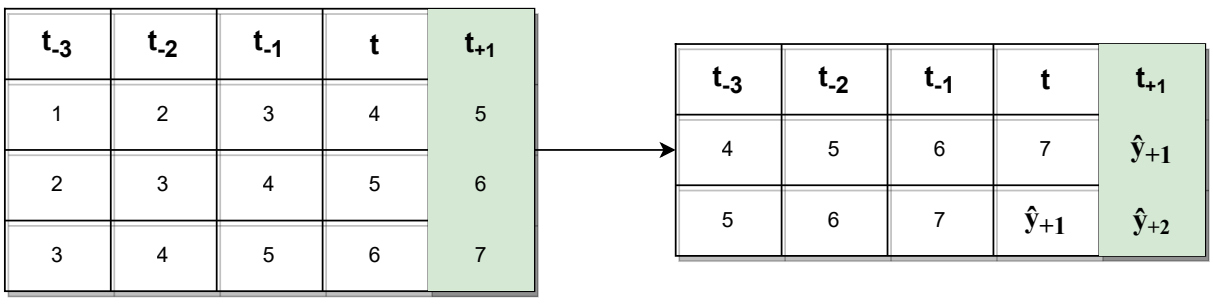


Figure 20 – Example of the reduction Procedure.

Figure 20 illustrates an example of using the reduction procedure to transform the time series Y into a structured dataset (4 predictor variables and 1 target variable). In the right part,

we see the recursive strategy. In the example, we forecast two steps ahead $\hat{y}_{+1}$ and $\hat{y}_{+2}$. Since the data in the time series are collected monthly, we used 12 temporal lags as predictor variables. We forecast 18 steps ahead, using the M4 competition test data.

In the experiments carried out, all base-level algorithms use the default hyperparameter values. For reproducibility, we controlled the seed of the value random generator.

3.6.4 Meta-target definition

For the definition of the meta-target, we used the lowest value of the Mean Squared Error (MSE) Equation 3.4. This metric was chosen due to its high penalty for large errors, a feature that can be beneficial for differentiating algorithm performances.

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \tag{3.4}$$

Table 1 summarizes the main characteristics of metadata. In the meta-target lines, we show the distribution of the meta-target divided among the base learners.

Table 1 – Metadata description

Meta-features	783					
Meta-examples	48000					
Meta-target	**KNN**	**LGBM**	**RF**	**LINREG**	**BKR**	**SVR**
	13035	7809	7504	7267	6203	6182

3.6.5 Metalearner

The metalearner used in this work is the Random Forest (RF) algorithm. The RF algorithm was chosen due to it being successfully used for this task in previous studies, such as (TALAGALA *et al.*, 2018). Besides, RF works well with noisy features and presents good results even without hyperparameter tuning. Only the RF was used as a metalearner because the metalearning algorithm is not an important issue of the proposed framework. The only hyperparameter value defined by the authors was the number of estimators; i.e., 2000. For the other hyperparameters, the default values were used.

3.6.6 Meta level evaluation

This evaluation assesses the overall predictive performance for each base learner and the metalearner. The confusion matrix was used to evaluate the recommendation performance for each base learner. The confusion matrix arranges the results of a classifier in two dimensions. One dimension represents the predicted target variable, and the other represents the true target variable.

The Geometric mean (Gmean) was used to measure overall predictive performance at the meta-level. The Gmean measure is used when the data set has a class imbalance. The Gmean definition for multiclass problems is the root of the product of all the class sensitivity (BARANDELA *et al.*, 2003).

The baseline used for the performance of the recommendation was random classification (Rd). To simulate Rd, 30 random classes were drawn. Then the Gmean was calculated, and the result considered was the average.

3.6.7 Base level evaluation

For the definition of the meta-target, and the subsequent performance evaluation of the regression models, the training and test set division of the competition dataset, the base-dataset, was used, when the monthly time series was separated from the 18 forward test steps, which will not be seen during the training phase. The meta-dataset was divided into 80% for training and 20% for testing. The metric used to assess the difference between the predictive performance of the regressor recommended by the metalearning and the regressor defined by the baselines was the Mean Absolute Percentage Error (MAPE) Equation 3.5.

$$MAPE = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{3.5}$$

MAPE was used because it is a percentage metric, therefore, is independent of the scale of different time series. This behavior is desirable because differences in scales could cause bias in the analysis of results. To analyze the statistical significance of the results, the Friedman statistical test with Post-hoc Nemenyi was used, and the results are shown in the critical difference diagram (DEMŠAR, 2006).

3.6.8 Base level benchmarking

For a base-level validation of the proposal approach, we also compared it with recurrent deep learning algorithms, in addition to the performance of the single models. These algorithms were chosen because they have state-of-the-art results in several applications (SEZER; GUDELEK; OZBAYOGLU, 2020; HEWAMALAGE; BERGMEIR; BANDARA, 2021). However, the training process of recurrent neural networks has a high computational cost and requires a large amount of data (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2018a; PARMEZAN; SOUZA; BATISTA, 2019c). The methods used as base learners in this study are less computationally complex than recurrent neural networks. The recurrent architectures used as benchmarking were the stacked LSTM and the bilateral LSTM neural networks.

3.6.9 Computational cost

In addition to the predictive performance, we analyzed the computational cost when comparing MetaFore with the other methods. The processing time used is equal to the sum of the user and system processing time during the current process, not including the time elapsed during sleep. Entering the sleep state means that the process suspends its access to the processor. The computational structure available can affect the execution time. For reproducibility, the structure used was a server with 2 Intel Xeon Processors E5-2650v4, 2.2 GHz with twelve cores, 128 GB DDR3 1866MHz, and 1 Nvidia Tesla P100 GPU.

3.6.10 Meta-knowledge interpretation

The mean decrease in the impurity (MDI) method (BREIMAN *et al.*, 1984b) was used to obtain the ten most important metafeatures. The MDI can be defined as the total impurity decay of a node weighted by the probability of reaching this node divided by the number of trees in the ensemble.

After obtaining the ten most important meta-features, we used the Partial Dependence Plot (PDP) (HASTIE; TIBSHIRANI; FRIEDMAN, 2009) to extract meta-knowledge and to understand how the model recommendation probability varies according to the value of the meta-feature.

3.7 Experimental Results

In this section, we present and analyze the main experimental results obtained by MetaFore. At the meta level, we compare MetaFore with the baseline, and at the base level we compare the performance of the recommended regression models with the single models and LSTM architectures. At the end, we analyze the meta-knowledge present in the classification model using feature importance and PDP.

3.7.1 Meta level results

We start by analyzing the meta-level predictive performance. In this evaluation, we extract predictive performance measures from the confusion matrix. Figure 21 shows the metalearner confusion matrix for the test data. The predicted algorithms are on the x-axis, and the y-axis are the true algorithms. The confusion matrix is scaled by the total number of time series in the test data.

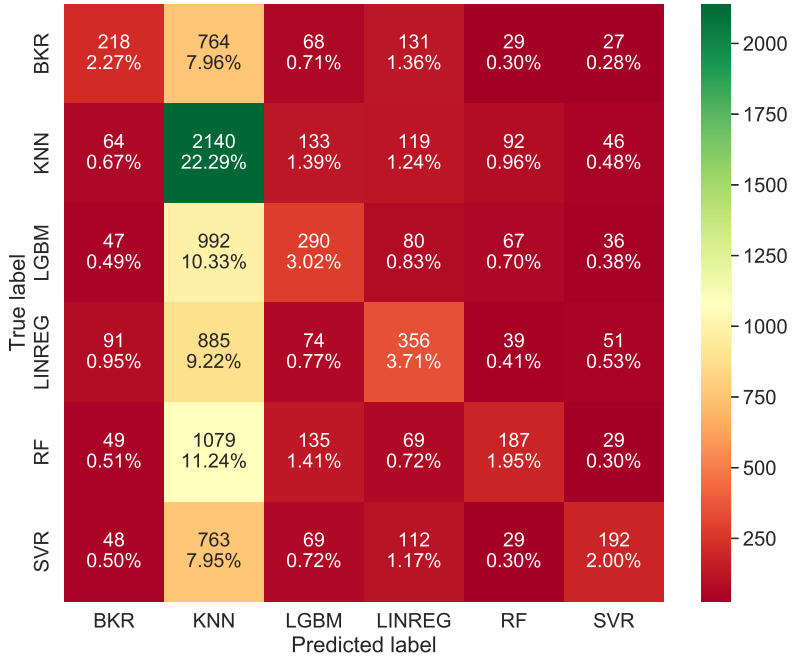


Figure 21 – Recommendation Model Confusion Matrix.

In the confusion matrix, the main diagonal represents the correctly classified labels. In the MetaFore confusion matrix, it can be observed that the percentage of correctly predicted labels was higher than the errors for all classes. There is an imbalance of classes in the dataset. The largest proportion of data refers to the KNN class. Most of the predictions also focused on the KNN class. We will analyze the Gmean measure to understand how the imbalance impacted the overall performance. Table 2 shows the results of the Gmean obtained by MetaFore and for the mean of the random classification (Rd).

Table 2 – Meta Level: Classification Metrics.

Metric	**MetaFore**	**Rd**
Gmean	0.22	0.16

Table 2 indicates that the MetaFore Gmean is greater than the average of the random classifications. The Gmean measure penalizes errors given the class imbalance. The Gmean presented by MetaFore shows that it was difficult to differentiate the classes in many time series. In future work, finding effective strategies to minimize these errors is desirable.

These results provide evidence that the algorithms recommended by MetaFore were correctly classified for a large proportion of the data. In terms of Gmean, the proposed baseline was surpassed. Given the difficulty of classifying six classes, the results were satisfactory.

3.7.2 Base level results

Figure 22 shows the results at the base level, using the predictive performance (MAPE) and computational cost. In this figure, the y-axis shows the MAPE results obtained by the baselines, the state of the art models, LSTM and Bi LSTM, and MetaFore, and the x-axis their computational cost in seconds.

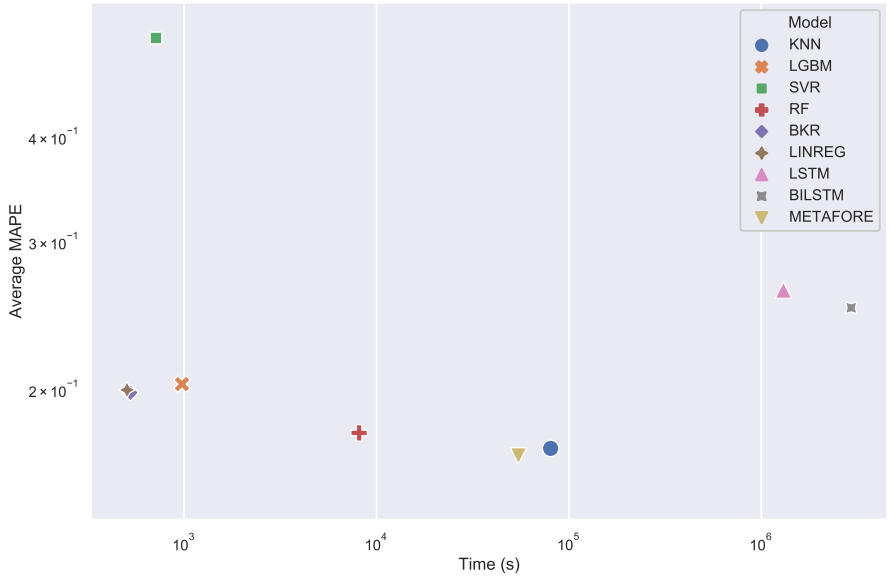


Figure 22 – Base Level: comparison between predictive performance and execution time.

According to these results, algorithms recommended by MetaFore obtained lower error rates than the base-level algorithms. The predictive performance of MetaFore was close to that obtained by KNN, the algorithm with the best single model. Besides, KNN, in its TSPI (PARMEZAN; SOUZA; BATISTA, 2022) version, is the most frequent algorithm in the meta-target. Thus, when MetaFore recommends an algorithm other than KNN, it gains an advantage in predictive performance. It is easy to see that when considering computational time, MetaFore is more efficient than executing all the base-level algorithms, as the latter takes less time than KNN, which is the slowest algorithm. In a statistical significance test, MetaFore was significantly better than the worst single model, the SVR, corroborating the learning potential obtained by the meta-model.

Finally, MetaFore performed much better than the LSTM neural networks, both in predictive performance and computational cost. In more than 70% of the test time series, the recommended algorithm was better than the LSTM architectures. Figure 23 shows the Critical Difference Diagram referring to the Post-hoc Nemenyi statistical test.

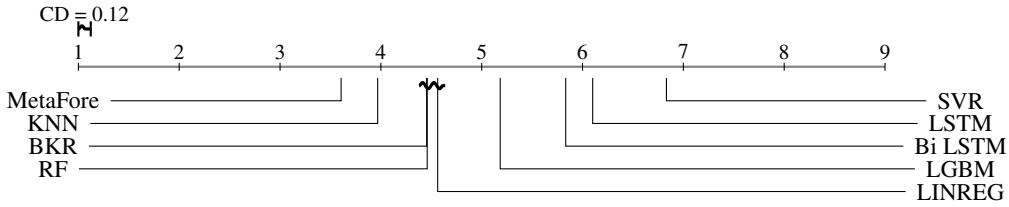


Figure 23 – Base Level: Critical Difference Post-hoc Nemenyi Test.

The Critical Difference Diagram confirms that the result obtained by MetaFore outperformed models induced by single algorithms and LSTM neural networks with statistical significance. The second best alternative was KNN, followed by BKR, RF, and LINREG, with no statistically significant difference. The worst predictive performance was achieved by the model induced by the SVR algorithm.

These results show that the MetaFore approach balances the predictive performance and computational time consumption well. In several applications, approaches with these characteristics are desirable. MetaFore shows good results at the meta and baseline level for a set of time series from different real-world domains. Due to the empirical evaluation of a challenging dataset, MetaFore has a consistent and promising performance.

3.7.3 Feature importance and partial dependence plot

Table 3 shows the ten most important meta-features used by the metalearner for the training data. The first column has the name of the meta-features and the second their importance measured in MDI. This information helps to understand the characteristics considered the most relevant in the MetaFore decision-making process.

Table 3 – Top 10 Metalearner Feature Importance.

Meta-feature	**Feature Importance (MDI)**
Augmented Dickey-Fuller Statistic	0.0026
Augmented Dickey-Fuller P-value	0.0026
Autoregressive Coefficient 0	0.0023
First Location of Maximum	0.0022
Autoregressive Coefficient 4	0.0022
Autoregressive Coefficient 10	0.0022
Last Location of Maximum	0.0022
Partial Autocorrelation Lag 4	0.0022
Autoregressive Coefficient 6	0.0022
Autoregressive Coefficient 7	0.0022

The first two meta-features, Augmented Dickey-Fuller statistic and p-value, are the time series trend-stationarity test results. The Augmented Dickey-Fuller statistic is a negative value.

The more negative, the higher the probability that the time series has a trend. The Augmented Dickey-Fuller p-value is the probability of obtaining the observed time series, assuming that the null hypothesis is true. The null hypothesis of the statistical test is that there is no trend-stationarity in the time series.

Autoregressive Coefficients can show how smooth a time series is. They are defined by the estimation of parameters of an autoregressive process. The explanation of this meta-feature comes from the data order and value. The order comes from how many past temporal lags affect the future values. The coefficient in a given order can be interpreted as the relevance of the lag to the prediction. Regarding the value, the larger the value in higher orders, the smoother the time series.

The first and last Locations of Maximum are the relative positions of the maximum value in the time series. Each one of these values is scaled from 0 to 1. These positions define whether the values in a time series in a particular interval are growing or decreasing.

The Partial Autocorrelation coefficient is a statistical value that shows the order of an autoregressive process. If the Partial Autocorrelation coefficient value of a given lag is statistically relevant, it signals the order of the autoregressive process. The higher the time series seasonality, the higher the lag. For example, the statistically relevant lag in monthly time series is usually 12 (annual seasonality) or 4 (quarterly seasonality).

A meta-knowledge that can be derived from these meta-features to meta-knowledge is that trend, smoothness, and seasonality properties are important in the MetaFore algorithm selection phase. The meta-level model assigned relevance to meta-features directly related to the trend and seasonality components, which are the main focus of the STL decomposition.

In Table 3, we selected four representative meta-features from each category to investigate further with PDP plots. In Figure 24, the values in the vertical axis, Partial dependence, are the probability of the model being chosen. The values in the horizontal axis are assumed by a particular meta-feature.

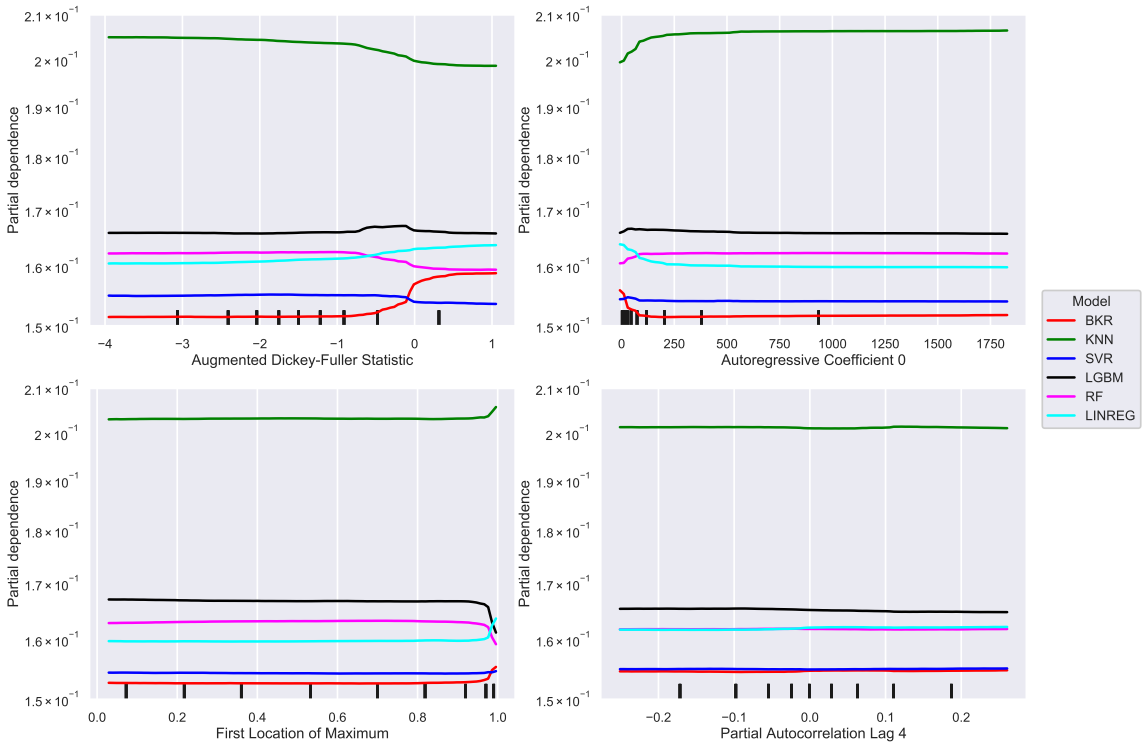


Figure 24 – Partial dependence plot for the metalearner.

The top left graph shows the partial dependence for the Augmented Dickey-Fuller Statistic meta-feature. From -1 onwards, the probability of recommending KNN decreases. The probabilities for the LGBM and SVR models remain practically constant with the variation of this meta-feature. Close to the value 0, the BKR and LINREG models increase the probability of being recommended. The RF model probability decays smoothly when the meta-feature value is close to the value 0. Regarding the meta-knowledge, the results show that the KNN and RF algorithms benefit from trend stationarity, while the opposite happens for the BKR and LINREG algorithms.

The top right graph shows the partial dependence for the Autoregressive Coefficient 0 meta-feature. For all models, the larger variations of the partial dependence are shown in the interval between 0 and 250. While this value increases for KNN and RF, it decreases for LINREG and BKR. For SVR and LGBM, it remains constant. Thus, KNN and RF algorithms benefit from higher coefficient values, and LINREG and BKR from values close to 0.

The bottom left graph shows the probabilities for the First Location of the Maximum meta-feature. The probability values for KNN remain constant for this meta-feature. For LGBM and RF, they sharply decrease when the maximum value approaches the end of the time series. The opposite occurs for BKR, SVR, and LINREG. BKR, SVR, and LINREG gain when the

maximum value is near the end of the time series. LGBM and RF have their highest probabilities before the last decile.

The bottom right graph shows the probabilities for the Partial Autocorrelation Lag 4 meta-feature. In this graph, there are no probability variations with the change of the meta-feature values. An investigation with another technique or with sets of meta-features can extract some meta-knowledge from this meta-feature.

The interpretation of meta-knowledge with PDP provides new insights into the relationships between the meta-features and the selected algorithm. In this analysis, it can be observed how the selection of algorithms depends on the presence of trend-stationarity in the time series. It is also shown that the location of the maximum value at the end of the time series impacts the algorithm selection by the meta-level model.

3.8 Conclusion

This paper proposed and experimentally assessed a new metalearning-based approach for time series forecasting. The proposed approach, MetaFore, uses metalearning for the recommendation of the most suitable ML algorithm for STL decomposition based-forecasting combinations. Experiments were carried out to assess the proposed approach. According to the experimental results, for the datasets used in this study, the predictive performance obtained by MetaFore was superior to the results obtained by literature methods.

MetaFore was implemented and evaluated at meta and base levels for the monthly time series of the M4 Competition. At the meta level, the predictive performance for each base-learner and the metalearner was evaluated. At the base level, the prediction error and computational time were compared with single models and LSTM architectures. The predictive performance presented by the MetaFore was superior to the predictive performance obtained by single models and LSTM architectures, with a lower computational cost.

Furthermore, the most important meta-features for the meta-level model were analyzed. Based on the analysis, the following meta-knowledge found that the most important meta-features are related to trend and seasonality. These properties are the main focus of STL decomposition. The PDP analysis showed that the presence of trend-stationarity and the location of the maximum value in the time series are related to the selection of the algorithm by the meta-level model.

These experimental results showed the benefits of using MetaFore for time series forecasting, which we understand to be important scientific contributions for research in this area. We believe that the automatic selection of models that compose the combination has several good opportunities for further investigation. New research directions include using metalearning to recommend a model for seasonality, assuming that some time series have more complex seasonal patterns that would hardly be modeled by the naive seasonal method. Another possi-

ble improvement would be the investigation of the proposed approach to test new time series frequencies.

Acknowledgement

Research carried out using the computational resources of the Center for Mathematical Sciences Applied to Industry (CeMEAI) funded by FAPESP (grant 2013/07375-0).

Funding: This work was partially funded by grants #2019/10012-2 and #2021/13281-4, São Paulo Research Foundation (FAPESP) and CNPq. This work was partially funded also by the projects ConnectedHealth (no. 46858), supported by Competitiveness and Internationalisation Operational Programme (POCI) and Lisbon Regional Operational Programme (LISBOA 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF), by the project Safe Cities - Inovação para Construir Cidades Seguras, reference POCI-01-0247-FEDER-041435, co-funded by the European Regional Development Fund (ERDF), through the Operational Programme for Competitiveness and Internationalization (COMPETE 2020), under the PORTUGAL 2020 Partnership Agreement, by project NextGenAI - Center for Responsible AI (2022-C05i0102-02), supported by IAPMEI, and also by FCT plurianual funding for 2020-2023 of LIACC (UIDB/00027/2020 UIDP/00027/2020).

CHAPTER

4

SEASONAL-TREND DECOMPOSITION BASED ON LOESS + MACHINE LEARNING

Reference: SILVESTRE, G.D.; SANTOS, M.R.; CARVALHO, A.C.P.L.F. Seasonal-Trend decomposition based on Loess + Machine Learning: Hybrid Forecasting for Monthly Univariate Time Series. In: International Joint Conference on Neural Networks (IJCNN), 2021, Shenzhen, China (Virtual Event). IEEE, 20 September 2021. 1-7.

4.1 Abstract

Recent studies have shown that hybrid forecasting models tend to be a powerful tool to forecast univariate time series. However, most of these models are applied to time series of specific domains and do not report general performance analysis for several time series application domains. In this work, we designed a procedure that uses the Seasonal-Trend decomposition based on Loess as a preprocessing step to model the time series components separately using a machine learning algorithm and a seasonal naive forecaster. Finally, we analyze under which conditions our proposed framework can improve a standard machine learning model's predictive performance. Results have shown that our hybrid forecasting framework achieves a significant advantage in comparison to standard machine learning.

4.2 Introduction

Time series forecasting is a crucial problem in tasks that involve planning and support to decision making. The forecasting task is challenging and is part of the daily life of several areas such as finance, industry, government, demography, epidemiology, among others (MONTGOMERY; JENNINGS; KULAHCI, 2015a; HYNDMAN; ATHANASOPOULOS, 2021).

Literature approaches for time series forecasting is found in statistical and machine

learning (ML) methods. The statistical methods were dominant for several decades. However, in recent work, the machine learning algorithms presented an auspicious performance in the time series forecasting task (MULLER *et al.*, 1999; DEB *et al.*, 2017; PARMEZAN; SOUZA; BATISTA, 2019b; CERQUEIRA; TORGO; SOARES, 2019).

Besides, hybrid methods have appeared in the literature to unite the best of the two areas. In time series forecasting, it is known that model combinations generally have better results than individual models or at least minimize the risk of choosing the worst method (CLEMEN, 1989; HIBON; EVGENIOU, 2005). In the M4 competition (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2020), the best results were obtained by model combinations. The method that presents the best results is a hybrid combination of a ML algorithm with a statistical method.

(REN; SUGANTHAN; SRIKANTH, 2014) aims to perform a comparative analysis of the methods based on Empirical Mode Decomposition (EMD) for wind speed forecasting. The results have shown that the combination of decomposition based on EMD and Support Vector Regression (SVR) has outstanding wind speed forecasting performance.

In some applications, like power load time series, multi-phase decomposition might be more useful to handle non-linearity and non-stationarity. The work (LI *et al.*, 2020b) purpose an approach with two-phase decomposition, Complementary Ensemble Empirical Mode Decomposition (CEEMD) and the Variational Modal Decomposition (VMD) considering the Sample Entropy (SE). This two-phase method is performed to model and separate differents non-linear load series patterns. Support Vector Regression is trained in two-phase approach components. The model achieves the state of the art results due to stable performance for short term power load forecasting.

Complete Ensemble Empirical Mode Decomposition with Adaptive Noise (CEEMDAN) was applied to decompose flow traffic complex and irregular patterns in several more simple components (LU *et al.*, 2020). The Extreme Gradient Boosting (XGBoost) was applied to make accurate traffic flow forecasting. This approach achieves superior performance in a benchmark with state of the art models. The model show robustness for capture flow traffic patterns in different conditions.

For short-term load forecasting, EMD and Seasonal-Trend decomposition based on Loess (STL) are widely found in the literature (LI *et al.*, 2020a). The work (LI *et al.*, 2020a) purpose a hybrid approach using Ensemble Empirical Mode Decomposition (EEMD) to decompose electric load time series in different frequency components with low and high levels. The components with low-frequency were forecasted by Multiple Linear Regression (MLR) and high-frequency components by Long Short Term Memory (LSTM) neural network. This approach was called ELM, an abbreviation for (EEMD-LSTM-MLR). The ELM method showed the Mean Absolute Percentage Error (MAPE) 67% smaller than neural network LSTM.

The paper (LI *et al.*, 2020c) uses the STL for decomposing dam displacement time series

in seasonality, trend, and remainder. The seasonality is model with the Extra Trees algorithm, and the components combination trend and the remainder by a neural network stacked LSTM. This hybrid method shows performance superior in comparison to the state of the art models. A great feature of this approach is the excellent stability besides forecast accuracy.

The M4 competition winner method (SMYL, 2020) is a hybrid combination of exponential smoothing based formulas and neural networks LSTM. The exponential smoothing-based formulas were used to seasonality modeling and normalize time series. After the deseasonalization, the LSTM neural network captures the trend component and obtain cross-learning.

Most of these hybrid methods are applied to a specific domain time series. Because of this, it does not have a clear idea of when the method works in another domain and when it had the same performance of not performing decomposition and combinations. Thus, this work aims to describe a framework based on the STL to separately model the time series's model components by an ML algorithm and a naive seasonal forecaster combined for final forecasting. The conditions under which the proposed framework improves predictions of an ML model are evaluated based on the literature's assumptions.

The proposed hybrid framework was evaluated in 30 monthly time series of the ICMC time series prediction repository (PARMEZAN; BATISTA, 2014) from different applications. The ML models used in this work were Bayesian Ridge Regression, Kernel Ridge Regression, Support Vector Regressor, and Kernel Support Vector Regressor. For validation, the multicriteria performance metric (MPM) (PARMEZAN; LEE; WU, 2017) and Mean absolute error (MAE) were used for the Nemenyi post hoc test.

The paper is organized as follows: Section 2 describes the time series analysis methods that are used during this work; Section 3 presents the proposed hybrid framework; Section 4 presents the experimental setup used to achieve the results of Section 5; discussions and conclusion of the work are set out in Section 6.

4.3 Time-series analysis

A time series $Y_t = \{y_t\}_{t=1}^{T}$ is a sequence of observations ordered in time (BOX *et al.*, 2015), for instance, daily stock prices, monthly meat consumption, number of people accessing a website within an hour.

When the time series consists of only one variable varying in time, it is univariate. If one measures the maximum temperature every month, then the temperature itself is the only quantity considered, which means that this set of observations is a univariate time series. On the other hand, if two or more variables are changing simultaneously, we say that it is a multivariate time series (MONTGOMERY; JENNINGS; KULAHCI, 2015a). For example, the data collected from an accelerometer can consist of three time series. Each one represents the acceleration in a

different axis. Together they make up a multivariate time series.

Consider now that Y_t is a univariate time series, its three main components are: trend (T_t), seasonality (S_t) and residue (R_t) (HYNDMAN; ATHANASOPOULOS, 2021). These can be defined as follow:

- **Trend** is a long-term increase or decrease in the data. It can assume both linear and non-linear patterns. The first one is described using a linear map and the latter using polynomial functions, exponential maps, and others;
- **Seasonality** is a cyclic pattern that repeats itself after a constant period. It is important to note that a time series can carry out multiple seasonal patterns, i.e., there are multiple cycles, and each one has its period;
- **Residue** is the short-term fluctuations, it is expected to be a random component, and therefore, it cannot be estimated properly.

Suppose the magnitude of the seasonal fluctuations or the Trend variation does not depend on the level, mean overall observations ignoring the long-term trend effect. In that case, we can assume an additive decomposition, which means that Y_t can be written as Equation 4.1.

$$Y_t = T_t + S_t + R_t \tag{4.1}$$

In contrast, if there is a relationship of proportionality between the level and the seasonal fluctuations then a multiplicative decomposition is more appropriate, and the time series is written as Equation 4.2.

$$Y_t = T_t \times S_t \times R_t \tag{4.2}$$

4.3.1 STL Decomposition

STL (CLEVELAND; CLEVELAND; MCRAE JEAN E ADN TERPENNING, 1990) is a robust and versatile algorithm to perform time series decomposition. It performs sequential applications of the Loess (local regression) smoother in order to decompose a time series Y_t into $T_t + S_t + R_t$. It has several advantages over other decomposition methods, e.g., SEATS (Seasonal Extraction in ARIMA Time Series) and X_{11}. Unlike those methods, the STL can handle any seasonality, the seasonal component can change over time, the user can control the trend smoothness, and it is robust to outliers. Its disadvantages include the limitation to additive decomposition, which the application of a logarithmic transformation can overcome, and it does not handle a trading day or calendar variation (HYNDMAN; ATHANASOPOULOS, 2021).

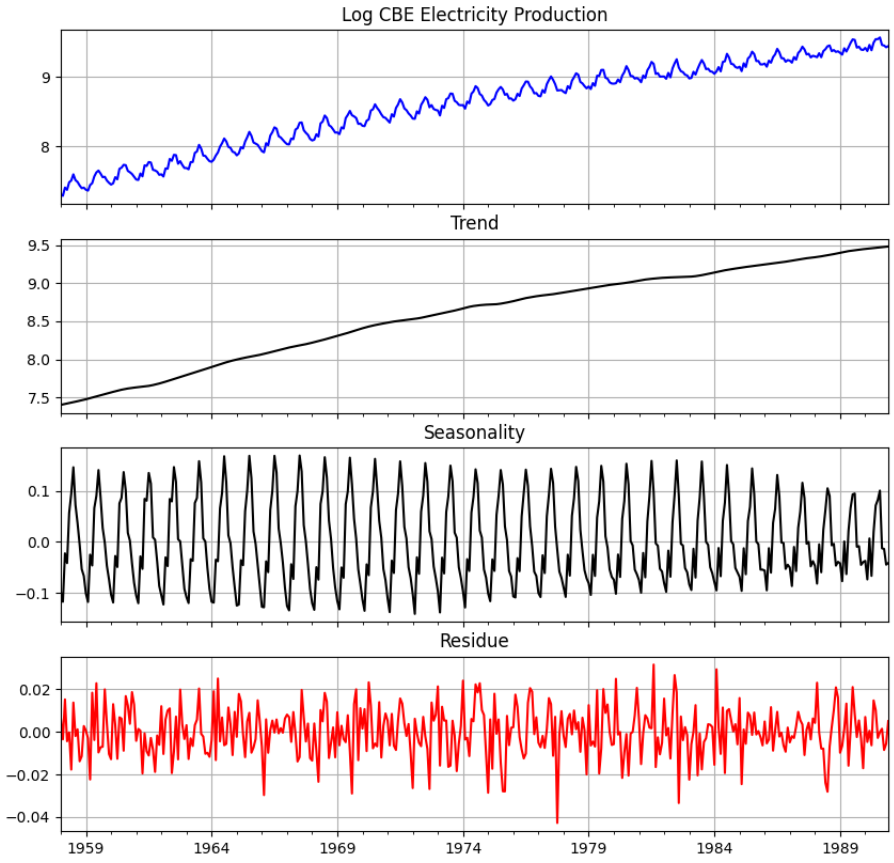


Figure 25 – Components from the CBE: Eletricity production time series

The STL was applied to the monthly CBE: Electricity Production time series available at the ICMC-USP Time Series Prediction Repository (PARMEZAN; BATISTA, 2014). In the figure 25, the first plot shows an increasing linear trend along a strong cycle that repeats itself every 12 months. A visual analysis of the residue plot shows random fluctuations around level 0, and the L-Jung Box test results also give us high confidence that the residuals are noise.

4.3.2 Forecasting

Given a time series $\{y_t\}_{t=1}^{T}$ we would like map it into a function f such that $f(t) = \hat{y}_t \approx y_t$, $\forall t > T$, in other words, it's going to be used the information present in the past values to build a mathematical function that will output predictions for future observations (HYNDMAN; ATHANASOPOULOS, 2021).

The map $\mathscr{A} : \{y_t\}_{t=1}^{T} \to f$ will be referred to as a forecaster or a model, now suppose we want to predict the quantity y_{T+1}, the latter is a random variable. It is possible to assume the existence of a probability measure $\mu(y_{T+1}|\{y_t\}_{t=1}^{T})$ which is what describes the relationship between the available data and the future observation. Every forecaster tries to estimate μ in some sense. In most cases, its output, f, will represent the mean of this distribution. The process of forecasting y_{T+1} given $\{y_t\}_{t=1}^{T}$ is called one-step forecast, on the other hand, if we want to forecast a subset of $\{y_{T+k}\}_{k\geq 1}$ then the process will be called multi-step forecast.

It is essential to evaluate it using out-of-sample observations to compare multiple forecaster's performance(HYNDMAN; ATHANASOPOULOS, 2021), not in the training set. The evaluation can be done by a metric, which tells us if the predictions are close to the actual values. There are several different metrics to choose from, each with its interpretation, and it is up to the user to decide which one is more appropriate for a specific problem.

4.4 Proposed Hybrid Forecasting Framework

The framework sums up the decomposition of the time series, using the STL, into two components: the first one is the seasonality, which will be modeled using a seasonal naive forecaster, and the second component, the trend + residue that a ML algorithm will model. Consequently, it is necessary to make some assumptions before diving into the modeling phase as a consequence of the no free lunch theorem. In this case, we will assume:

1. The seasonal pattern is strong and remains constant over time;
2. The residuals from the STL decomposition are approximately noise.

Each one of them will play an important role during the modelling phase. The assumptions about the seasonal component bring sense to the application of the seasonal naive forecaster. Otherwise, we cannot fit the periodic component properly. Finally, we want to simplify the forecasting task. If one can still spot autocorrelation within the residuals, then we know that the STL failed, and therefore there is no guarantee that the data being fed to the models is indeed what we have been expecting.

Suppose a time series $Y_t = \{y_t\}_{t=1}^{T}$ fulfills our requirements, the sequence of steps we will apply to forecast Y_t is illustrated in the figure 26.

The log transformation ensures that Y_t can be decomposed additively and stabilizes the variance across time (HYNDMAN; ATHANASOPOULOS, 2021), to handle negative values, we added a shift term. Thus the overall procedure is given by Equation 4.3.

$$\begin{cases} log(y_t + 1) & \text{if } min\{y_t\}_{t=1}^{T} > 0 \\ log(y_t + |min\{y_t\}_{t=1}^{T}| + 1) & \text{otherwise} \end{cases} \tag{4.3}$$

The transformed time series will be decomposed by the STL algorithm yielding two components: $Z_t = T_t + R_t$, the trend+residual component, and S_t the seasonal component. The latter is going to be modeled by a seasonal naive forecaster whose predictions are written as Equation 4.4

$$\hat{s_t} = f(T + h, y_1, \ldots, y_T) = y_{T+h-m(k+1)} \tag{4.4}$$

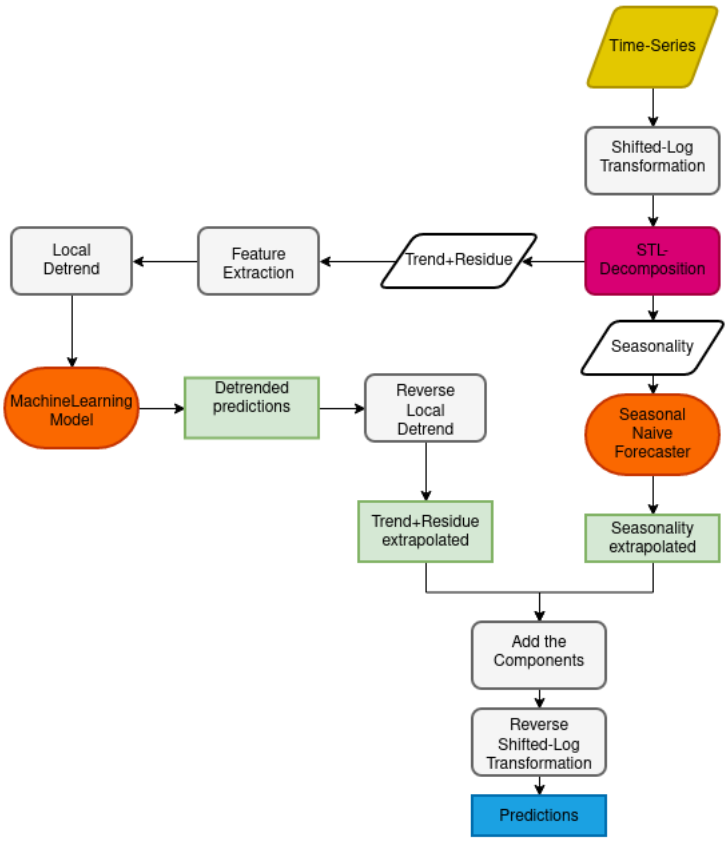


Figure 26 – Proposed framework

Where h is the forecast horizon, m is the seasonal period and k the integer part of $\frac{h-1}{m}$ (HYNDMAN; ATHANASOPOULOS, 2021). The Z_t component cannot be treated by a ML model directly. Firstly it is necessary to extract features from it to turn the time series into a structured dataset. The process is straight-forward and to illustrate it, we will take a time series $Z_t = \{z_1, \ldots, z_{10}\}$ and $p = 5$, Z_t is given by Equation 4.5.

$$\left[\begin{array}{ccccc|c} z_1 & z_2 & z_3 & z_4 & z_5 & z_6 \\ z_2 & z_3 & z_4 & z_5 & z_6 & z_7 \\ z_3 & z_4 & z_5 & z_6 & z_7 & z_8 \\ z_4 & z_5 & z_6 & z_7 & z_8 & z_9 \\ z_5 & z_6 & z_7 & z_8 & z_9 & z_{10} \end{array}\right] \tag{4.5}$$

Note that we end up with a dataset with five samples and five features, more generally with $T - p$ samples and p features, which is precisely the case in which we can apply a classical ML model. However, before training the algorithm, if there is a significant trend in the time series, it will be applied to a local detrend operation (BANDARA; BERGMEIR; SMYL, 2020). It consists of subtracting the last observation from each value in the corresponding input and output. If we apply the local detrend to the previous dataset, we will get the Equation 4.6.

$$\left[\begin{array}{cccc|c} z_1 - z_5 & z_2 - z_5 & z_3 - z_5 & z_4 - z_5 & z_6 - z_5 \\ z_2 - z_6 & z_3 - z_6 & z_4 - z_6 & z_5 - z_6 & z_7 - z_6 \\ z_3 - z_7 & z_4 - z_7 & z_5 - z_7 & z_6 - z_7 & z_8 - z_7 \\ z_4 - z_8 & z_5 - z_8 & z_6 - z_8 & z_7 - z_8 & z_9 - z_8 \\ z_5 - z_9 & z_6 - z_9 & z_7 - z_9 & z_8 - z_9 & z_{10} - z_9 \end{array}\right] \tag{4.6}$$

Once the ML model is fixed and the dataset is constructed, we can start the training process to estimate a function f that will perform the forecasting. The latter is based on an update procedure that uses a queue to store the subsequences and updates it using the predicted values from the model (PARMEZAN; SOUZA; BATISTA, 2019b). Suppose we want to predict z_{11} in the previous example, then we will apply the following steps:

1. Construct the subsequence:

$$\{z_6 - z_{10}, z_7 - z_{10}, z_8 - z_{10}, z_9 - z_{10}\}$$

2. Calculate

$$f(11, z_6 - z_{10}, z_7 - z_{10}, z_8 - z_{10}, z_9 - z_{10}) = \tilde{z_{11}} - \tilde{z_{10}}$$

3. Revert the detrend operation:

$$\hat{z_{11}} = \tilde{z_{11}} - \tilde{z_{10}} + z_{10}$$

To forecast z_{12} it is enough to remove the first element from the subsequence, stack the quantity $z_{10} - \hat{z_{11}}$ to it and repeat the previous steps applying the necessary changes. It is not hard to generalize for any time series and any forecasting horizon.

The predictions $\{\hat{z}_t\}_{t=T+1}^{T+h}$ from the ML model are then added with the predictions from the seasonal naive forecaster, $\{\hat{s}_t\}_{t=T+1}^{T+h}$. Now it is enough to revert the predictions to the original scale by the application of the Equation 4.7.

$$\begin{cases} exp(\hat{z}_t) - 1 & \text{if } min\{y_t\}_{t=1}^{T} > 0 \\ exp(\hat{z}_t) - |min\{y_t\}_{t=1}^{T}| - 1 & \text{otherwise} \end{cases} \tag{4.7}$$

4.5 Experimental setup

The 30 monthly time series from the ICMC-USP Time Series Prediction Repository (PARMEZAN; BATISTA, 2014) were chosen to test the proposed framework, since we believe that they are more likely to fulfill our assumptions. Moreover, the repository contains a wide variety of datasets from different domains which is rich environment to analyse if the hybrid approach will behave as we expect. We will describe the procedure used to compare the approaches starting by the ML models considered.

4.5.1 Machine Learning Models

It was decided to go for models that can be written as $\min_{f \in \mathscr{H}} \hat{\mathscr{E}}(f) + \lambda R(f)$, where $\hat{\mathscr{E}}$ is the empirical error, λR is the regularization term and $\mathscr{H}$ is a reproducing kernel hilbert space (EVGENIOU *et al.*, 2002). They rely on optimization procedures in order to estimate the function f given the training data, in this work, we will apply 4 algorithms:

- **Bayesian Ridge Regression (BRR)**: Given the data matrix X and its corresponding target vector y, the model can be written as follows:

$$\text{prior: } P(\theta) = \mathscr{N}(\theta | m_0, S_0)$$

$$\text{likehood: } P(y|X,\theta) = \mathscr{N}(y|X\theta, \sigma^2 I)$$

  It addresses the problem of overfitting in the least squares formulation by taking the mean over all possible solutions according to a probability distribution instead of trying to estimate θ directly (DEISENROTH; FAISAL; ONG, 2020). To make predictions we are interested on the mean of the distribution $P(\theta|X,y)$ which is exactly the solution of the regularized least squares Problem 4.8.

$$\min_{\theta} ||X\theta - y||^2 + \lambda ||\theta||^2 \tag{4.8}$$

  In other words, it's the maximum a posteriori solution for some convenient λ.

- **Kernel Ridge Regression (KRR)**: In this case, we are also looking for the solution of the regularized least squares problem but instead of assuming the correspondence $f \leftrightarrow \theta$, i.e. f is a linear map, we will allow f to be any element from a reproducing kernel hilbert space $\mathscr{H}$ that will be constructed using some kernel function K (EXTERKATE, 2013). In this case, by the representation theorem the solution is given by Equation 4.9.

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i K(\cdot, x_i) \tag{4.9}$$

  where $x_i, i = 1, \ldots, n$ are the training points, then our problem reduces to find the vector α such that Equations 4.10 and 4.11.

$$\alpha = \underset{\alpha}{\text{argmin}} \; ||\hat{K}\alpha - y||^2 + \lambda \alpha^T \hat{K} \alpha \tag{4.10}$$

$$\hat{K}_{ij} = K(x_i, x_j) \tag{4.11}$$

- **Support Vector Regressor (SVR)**: It allows sparse solutions by restricting the optimization to the points outside a certain decision boundary controlled by the user (BISHOP, 2006). This can be achieved by replacing the squared error in the least-squares problem with the ε-insensitive loss shown the Equation 4.12.

$$max(0, |y - (\theta^T x + b)| - \varepsilon) \tag{4.12}$$

Therefore, if the distance between the prediction and the actual value is less than some ε, the error at that point is completely ignored. Hence the solution depends only on a few data points, and they are called support vectors.

- **Kernel Support Vector Regressor (KSVR)**: If one introduces slack variables that tells whether a point lies within, above or below the decision boundary in the SVR formulation discussed previously and reformulate the whole optimization procedure using Lagrangian Multipliers it's possible to re-write the problem as:

$$\begin{aligned} \min_{\alpha,\hat{\alpha}} \ & \frac{1}{2}(\alpha - \hat{\alpha})XX^T(\alpha - \hat{\alpha}) + \varepsilon \mathbf{1}^T(\alpha + \hat{\alpha}) - y^T(\alpha - \hat{\alpha}) \\ \text{s.t.} \quad & \mathbf{1}^T(\alpha - \hat{\alpha}) = 0 \\ & 0 \leq \alpha_i, \hat{\alpha}_i \leq C, i = 1, \ldots, n \end{aligned}$$

Note that the covariance matrix XX^T can be replaced by $\Phi(X)\Phi(X)^T$, where Φ is some feature map. Finally, by representing the feature map using some kernel K, it is possible to model the relationship between x and y using non-linear maps (BISHOP, 2006).

The implementation was provided by the scikit-learn API (BUITINCK *et al.*, 2013), to isolate the effects from the preprocessing steps, we used the default hyperparameters. The kernel fixed was the Radial Basis Function (RBF), and the autoregressive order p was set to 12, which is the maximum seasonal variation for the time series. Moreover, the STL decomposition implementation is available at (SEABOLD; PERKTOLD, 2010).

4.5.2 Metrics

The metrics chosen to evaluate the models quantify the model's capability to predict the level, the seasonal, and the trend component while bringing high interpretability to the user (HYNDMAN; KOEHLER, 2006). In order to simplify the analysis, they will be combined into a single performance measure that will be used for comparison (PARMEZAN; SOUZA; BATISTA, 2019b).

- **REL naive**: Let $\{\hat{l}_t\}_{t=T}^{T+h}$ the predictions from the naive forecaster, i.e. $\hat{l}_t = y_T, \forall t > T$ and $\{\hat{y}_t\}_{t=T}^{T+h}$ the prediction from some forecaster $\mathscr{A}$, then the REL naive is define as follows the Equation 4.13.

$$\frac{\sum_{t=T}^{T+h} |y_t - \hat{y}_t|}{\sum_{t=T}^{T+h} |y_t - \hat{l}_t|} \tag{4.13}$$

Thus, if this quantity is greater or equal than 1 we conclude that $\mathscr{A}$ is, at least, as worse as the naive forecaster to forecast the level, otherwise, the model is better than the baseline.

- **REL snaive**: Let $\{\hat{s}_t\}_{t=T}^{T+h}$ the predictions from the seasonal naive forecaster, then the REL snaive is define as follows the Equation 4.14.

$$\frac{\sum_{t=T}^{T+h} |y_t - \hat{y}_t|}{\sum_{t=T}^{T+h} |y_t - \hat{s}_t|} \tag{4.14}$$

  The interpretation is similar to the latter, but in this case we are evaluating the capability of extrapolating the seasonal component.

- **POCID**: It measures if the model is able to tell if future values are increasing or decreasing (PARMEZAN; SOUZA; BATISTA, 2019b), it ranges from 0 to 1 and values closer to the latter are desired. We calculate it using the following the Equation 4.15.

$$\sum_{t=T}^{T+h} \mathbb{I}\{(\hat{y}_t - \hat{y}_{t-1})(y_t - y_{t-1}) > 0\} \tag{4.15}$$

  where $\mathbb{I}\{A\}$ is 1 if A holds true and 0 otherwise.

Finally, the multicriteria performance metric (MPM) used was devepoled by (PARMEZAN; LEE; WU, 2017), it consists on calculate the area of the triangle which sides lengths are the metrics chosen. In this case if $a =$ REL naive, $b =$ REL snaive, and $c =$ 1-POCID, then the MPM is given by Equation 4.16.

$$\frac{1}{2}\sin(\frac{2\pi}{3})(a \times b + a \times c + b \times c) \tag{4.16}$$

The model with a lower MPM yields better forecasts.

4.5.3 Evaluation procedure

The time series were split using a temporal holdout (HYNDMAN; ATHANASOPOULOS, 2021), leaving 5% for the testing set. We applied both the hybrid framework proposed and what we call a classical framework, consisting of the same steps described in section 4.4. However, instead of decomposing the time series, it feeds the log-transformed data directly to the feature extractor. Applying the reverse local detrend operation is enough to transform the predictions back to the original scale. Once we have the values, forecast it is enough to calculate the MPM for both frameworks. The steps are illustrated in the figure 27.

We will compare each ML model individually against its hybrid version using the MPM. The goal is to analyze if our assumptions are fulfilled, then the STL decomposition as a preprocessing technique is likely to improve the forecast quality. Furthermore, we will also compare all models along with the baselines, the naive forecaster and the seasonal naive forecaster, using the Nemenyi posthoc test (NEMENYI, 1963), the metric used in this case was the mean absolute error (MAE).

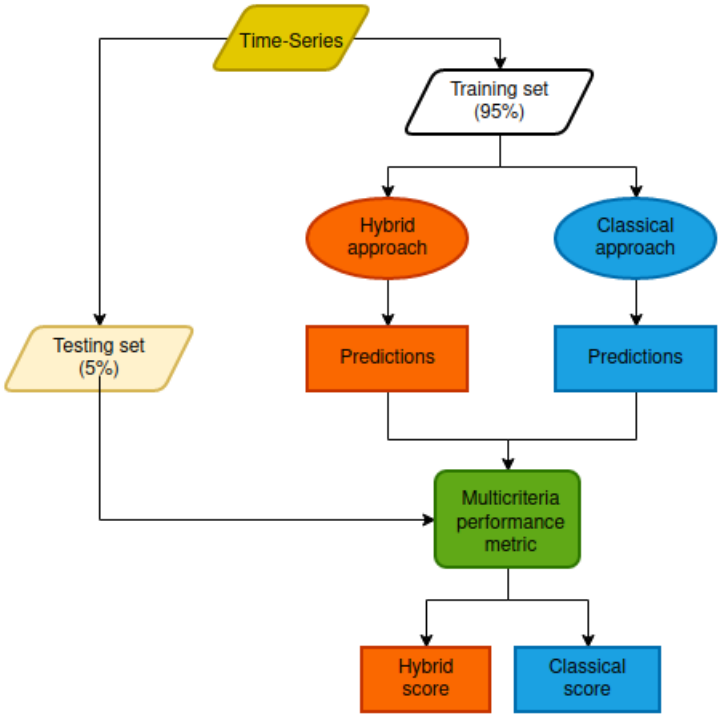


Figure 27 – Evaluation procedure

4.6 Experimental results

The time series chosen were divided into two groups based on the outcome of the STL decomposition. Using time, ACF, and histogram plots along with the L-Jung Box test, we were able to tell apart the time series in which our assumptions held, group A, and the ones where at least one of them failed, group B. The Tables 4 and 5 show the MPM's for the time series, and the letter "H" in front of the name stands for a hybrid.

Table 4 – Group A

Dataset	BRR	H-BRR	KRR	H-KRR	SVR	H-SVR	KSVR	H-KSVR
Bebida	0.84	**0.55**	2.01	**0.78**	0.92	**0.46**	1.24	**0.48**
CBE: Chocolate	0.58	**0.33**	**3.01**	30.24	0.62	**0.36**	1.01	**0.87**
CBE: Elec Prod	0.13	**0.06**	56.72	**55.26**	0.40	**0.13**	**1.55**	1.57
OSVisit	**0.07**	**0.07**	17.88	**14.66**	0.26	**0.04**	**0.23**	0.48
Ozonio	0.42	**0.33**	0.36	**0.32**	0.44	**0.41**	**0.30**	0.54
W: Dry White	0.66	**0.53**	**1.77**	6.31	0.59	**0.34**	1.69	**1.39**
W: Fortified White	0.39	**0.32**	15.05	**12.71**	0.46	**0.35**	0.60	**0.24**
W: Red	0.19	**0.12**	21.76	**16.21**	0.20	**0.01**	0.66	**0.33**
W: Rose	0.92	**0.49**	**5.03**	5.95	0.83	**0.57**	0.69	**0.27**
W: Sparkling	0.31	**0.18**	4.06	**0.20**	0.62	**0.18**	**0.19**	0.22
W: Sweet White	0.70	**0.24**	7.00	**3.25**	0.72	**0.24**	0.46	**0.24**
Latex	**0.44**	0.57	34.24	**31.10**	0.75	**0.59**	**0.54**	0.70
PFI	**0.97**	2.12	**3.03**	4.60	**1.15**	2.25	**0.71**	2.24

When the time series has a strong and constant seasonality and the STL produces uncorrelated residuals, the hybrid approach is likely to give better predictions than the standard ML approach to forecasting univariate time series. The Table 4 also shows us that the linear models tend to benefit more from the proposed framework than the kernel-based models. The

Table 5 – Group B

Dataset	BRR	H-BRR	KRR	H-KRR	SVR	H-SVR	KSVR	H-KSVR
CBE: Beer	**0.23**	0.63	0.80	**0.70**	**0.37**	0.56	**0.37**	0.63
Consumo	**0.39**	0.43	0.30	**0.10**	**0.43**	0.50	**1.32**	1.55
Lavras	**0.33**	1.36	**0.60**	4.41	**0.32**	1.39	**0.28**	1.64
Maine	**0.44**	0.50	**0.66**	0.71	**0.49**	0.66	1.35	**0.84**
Reservoir	**0.54**	2.50	**2.25**	2.58	**1.99**	2.51	**1.94**	2.50
Temp: Ubatuba	**0.63**	0.78	**0.31**	0.65	0.58	**0.60**	1.78	**0.80**
Chicken	1.97	**1.17**	**119.74**	130.86	5.06	**3.65**	**1.37**	1.49
DowJones	1.48	**1.28**	**3.99**	4.13	1.33	**1.29**	**0.68**	0.82
Energia	**0.41**	0.50	10.33	**9.84**	**0.25**	0.36	1.52	**1.50**
Global	**1.44**	5.42	7.21	**5.52**	9.83	**5.63**	6.78	**1.85**
ICV	**0.04**	0.09	**7.94**	**7.94**	**0.21**	0.23	1.73	**1.71**
IPI	0.21	**0.13**	1.24	**0.98**	0.24	**0.18**	**0.30**	0.47
STemp	**4.07**	6.13	10.15	**8.76**	9.12	**6.26**	**16.04**	20.19
Temp: Cananeia	**0.27**	**0.27**	**0.39**	0.56	**0.37**	0.38	0.31	**0.23**
USA	**0.44**	0.64	2.19	**2.07**	**0.45**	0.81	3.80	**3.61**
Darwin	0.51	**0.44**	**0.33**	0.37	0.44	**0.41**	**0.25**	0.37
MPrime	0.49	**0.22**	0.35	**0.18**	0.46	**0.27**	0.43	**0.23**

Nemenyi test, Figure 28 shows a significant difference between the strategies, and the hybrid linear models are within the best ones.

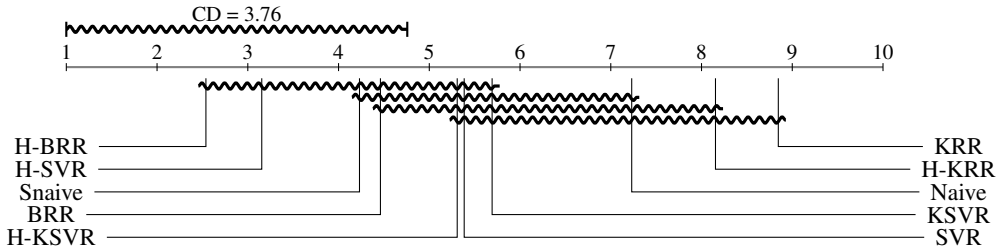


Figure 28 – Nemenyi - Group A

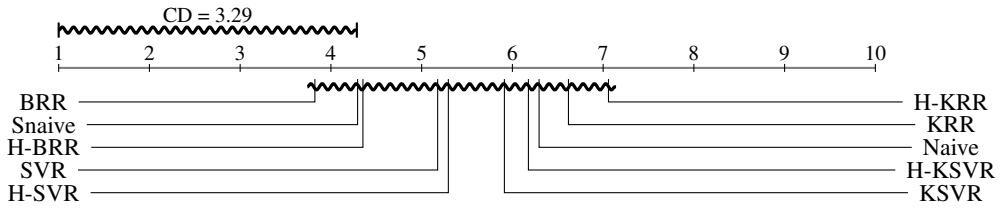


Figure 29 – Nemenyi - Group B

In contrast, the table 5 shows the time series where the residuals are not independent or lacking seasonality. The classical approach outperformed the proposed framework in most of the datasets, as expected. If the STL decomposition failed, we have no guarantee of what component it is being modeled precisely. Thus treating them all as one single measure is, in fact, the best option. In some time series, the hybrid framework improved the predictions we can see in the figure 29. There is no statistical difference between the approaches, and none of the hybrid models could outperform neither its classical version nor the seasonal naive baseline. Again, the kernel models showed themselves as worse as the naive forecaster to predict future observations,

reinforcing the idea that modeling a time series with complex models does not always guarantee better predictions.

4.7 Conclusion

This work showed that when we have a time series that fulfills our assumptions, there is a high probability that using the STL as a preprocessing approach along the seasonal naive forecaster can improve the performance of the ML models. We also saw that when one of those assumptions does not hold, there is a big chance that the model will not be able to perform better than a simple naive forecaster. Hence the application of it ends up adding unnecessary complexity to the pipeline. Finally, we believe that linear models are more likely to be improved by the proposed framework than the kernel-based models. Their simplicity can be a powerful and reliable tool to forecast univariate time series among the ARIMA and exponential smoothing models.

Acknowledgment

This study was partially funded by the Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) - Process 2019/10012-2 and CEPID-CeMEAI Process 2013/07375-0, and CNPq

CHAPTER

5

TSMORPH: GENERATION OF SYNTHETIC TIME SERIES TO UNDERSTAND ALGORITHM PERFORMANCE

Reference: This work was submitted to the "AutoML Conference 2023". The current status is "Under review.".

5.1 Abstract

Time series forecasting is a task of great scientific and industrial interest. Despite the widespread use of forecasting methods, there is limited work on understanding under which conditions they achieve good or bad results. Existing approaches are mostly based on metalearning. The limited number of datasets available for that purpose makes extracting general and reliable knowledge difficult. Synthetic data generation is a promising alternative to obtaining large sets of datasets. However, current methodologies for generating synthetic data for time series are complex and have a high computational cost. We propose *tsMorph*, a simple method for generating synthetic time series based on dataset morphing. Given two datasets, tsMorph creates a sequence of datasets that are a gradual transformation between the two. We illustrate the usefulness of the methodology by analyzing the performance of two forecasting algorithms: Long Short-Term Memory Networks and Support Vector Regression. The time series used in this study are from the NN5 Competition. The results include several interesting observations that illustrate the usefulness of the approach. For example, the performance of the Long Short-Term Memory Networks improves with the increase in the amplitude of variation of the time series values. In general, the experiments showed that tsMorph could be used to obtain knowledge about the behavior of forecasting algorithms.

5.2 Introduction

Forecasting is one of the main tasks in the decision-making process (HYNDMAN; ATHANASOPOULOS, 2018b). Quantitative approaches use historical data such as time series to make forecasts (MONTGOMERY; JENNINGS; KULAHCI, 2015b). Time series forecasting is used in several application domains, such as weather, stock markets, and epidemiology. Several methods for time series forecasting have been proposed in the literature with high predictive performance on diverse domains (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2018b). However, according to Wang *et al.* (2022a), a limited amount of work tries to understand under which conditions we can expect a forecasting method to obtain good (and bad) results.

Datasets are needed to understand the variation in the performance of forecasting algorithms from an empirical perspective. However, benchmarking for time series analysis is limited (KEOGH; KASETTY, 2002). Some approaches have been proposed in the literature for generating synthetic time series with realistic characteristics to generate benchmarking and improve the performance of metalearning. Autoregressive approaches generate time series with specific characteristics with optimization techniques. Furthermore, generative approaches (HYLAND; ESTEBAN; RÄTSCH, 2017; YOON; JARRETT; SCHAAR, 2019) based on Generative Adversarial Networks (GOODFELLOW *et al.*, 2020) learn the temporal dynamics of real time series for the generation of synthetic series. These frameworks are committed to the reality of the generated time series. However, generating datasets to improve understanding of the performance of algorithms has two important challenges: (1) high computational cost related to the optimization process and the training of neural networks; (2) absence of mechanisms for gradual variation data characteristics that lead to variation of the behavior of algorithms.

(CORREIA; SOARES; JORGE, 2019) proposed a simple approach for systematic dataset generation called *dataset morphing*. Dataset morphing consists of gradually transforming a source dataset into a target dataset. The changes in the behavior of learning algorithms in the sequence of manipulated datasets lead to a better understanding of those algorithms. This approach was originally proposed for the evaluation of collaborative filtering algorithms. It is a simple implementation method, and depending on the transformation function, it may have a low computational cost. Here, we extend this work by proposing tsMorph, a simple method for generating synthetic time series from morphing datasets. We propose a specific transformation function for time series that is guaranteed to converge and has linear complexity. We apply tsMorph to the NN5 competition data to understand the difference in the performance of forecasting algorithms. Our results show that tsMorph can generate synthetic time series with gradual variation between two datasets. This variation is reflected in the performance curve and the characteristics of the time series. These properties make our method a support tool for understanding forecasting algorithms.

5.3 Background

5.3.1 Time series forecasting

A time series is a sequence of observations of a variable of interest equally spaced in time (MONTGOMERY; JENNINGS; KULAHCI, 2015b). Let T be periods of historical data for the variable of interest. Then we can denote an observed time series at period t by $Y_t = \{y_t\}_{t=1}^{T}$. Such a time series can represent many phenomena in the real word. For example, the demand for beds for patients in a hospital during period t or the daily closing price of a stock on the stock exchange.

One of the main tasks in time series analysis is forecasting. Time series forecasting consists of using the available observations of the variable of interest to extrapolate the time series into the future. Given Y_t, the goal of the time series forecasting task is to find a function f such that $f(t) = \hat{Y}_{t+h}$ and $\hat{Y}_{t+h} \approx Y_{t+h}$, $\forall t + h > T$. In the forecasting definition, h denotes the number of forecast observations ahead and is called forecast horizon. The mapping $F : Y_t \rightarrow f$ is a forecasting model. Forecasting models assume a direct relationship between the available observations and the future of a variable of interest. In other words, these models assume the existence of a probability measure $\mu(\hat{Y}_{t+h}, Y_t)$ (MONTGOMERY; JENNINGS; KULAHCI, 2015b).

Several statistical methods and machine learning algorithms are capable of inducing time series forecasting models (MAKRIDAKIS; SPILIOTIS; ASSIMAKOPOULOS, 2018b). It is necessary to analyze the forecast error e_t to decide the most suitable one. The forecast error e_t is defined by Equation 5.1.

$$e_t(h) = Y_{t+h} - \hat{Y}_{t+h} \tag{5.1}$$

There are several measures of forecasting performance. Each measure adapts the notion of forecast error for specific purposes. We use the Mean Absolute Scaled Error (MASE) in this work. MASE (HYNDMAN *et al.*, 2006) is a scale-free error measure. Because it is scale-free, this measure can be applied to analyze time series from different scales. This performance measure originates from the scaled error q_t defined by Equation 5.2.

$$q_t = \frac{e_t}{\frac{1}{n-1}\sum_{i=2}^{n} |Y_i - Y_{i-1}|} \tag{5.2}$$

where n is the number of observations in the time series. The denominator of Equation 5.2 is the in-sample Mean Absolute Error of the naive forecast method. The naive forecast is one period ahead by repeating the last observed value $\hat{Y}_{t+1} = Y_T$. Once these terms are defined, the MASE can be seen in Equation 5.3.

$$MASE = mean(|q_t|) \tag{5.3}$$

The interpretation of this error value is that if the MASE is less than 1, the forecasting method is better than the naive forecast and vice versa. An additional interpretation is that the smaller the MASE value, the better the forecasting method.

5.3.2 Understanding the behavior of ML forecasting algorithms

In the literature, some approaches are used to understand ML algorithms. This section discusses some that serve as a basis for the proposed approach–starting with dataset morphing and the meta-knowledge analysis for time series forecasting.

Dataset morphing is the process of generating synthetic data from the gradual transformation of a source dataset into a target dataset. Given a source dataset D_s, a target dataset D_t, and a transformation τ, intermediate synthetic datasets D_j are obtained by the dataset morphing process. Correia, Soares and Jorge (2019) defines a generic dataset morphing process for various tasks in Equation 5.4.

$$\mathscr{D}_{morph} : \{D_j | D_0 = D_s, D_n = D_t, D_j = \tau(D_{j-1})\}, 1 \leq j < n \tag{5.4}$$

where D_{morph} is the total set of datasets, and n is the number of transformations. The dataset morphing process is proposed and used initially to understand the contrasting performance of a pair of algorithms on pairs of datasets.Correia, Soares and Jorge (2019) analyzed the performance curves obtained from the $\mathscr{D}_{morph}$ datasets. Data characteristics related to performance variation, called meta-features, were also analyzed.

Given a set of datasets $D = \{d_0, \ldots, d_n\}$, a meta-feature m can be defined as a function that maps $m : D \rightarrow R^k$ (ALCOBAÇA *et al.*, 2020). Function m returns k values that characterize each dataset in D. Several meta-features have been proposed in the literature for different specific tasks. According to Brazdil *et al.* (2022b), meta-features must have three main properties: performance discrimination power, computationally not very expensive, and suitable dimensionality to the amount of data available.

Another approach that makes use of meta-features is metalearning. Metalearning is the set of methods that uses knowledge extracted from learning tasks, algorithms, or task performance evaluation to improve predictive performance, make it faster, or understand how algorithms work (HUTTER; KOTTHOFF; VANSCHOREN, 2019b). It is also occasionally used to analyze the obtained meta-knowledge, despite being typically used for algorithm selection. Here we focus on the use of metalearning.

Armstrong, Adya and Collopy (2001) analyze meta-features in the pioneering work on metalearning for time series forecasting. The relations between the used meta-features and the

selection of the algorithms were shown to evaluate the gain of the automatic selection about the judgmental selection.

The approach proposed by Lemke and Gabrys (2010) uses decision trees to extract meta-knowledge. Decision tree trained on metadata were forecasting methodologies are the target. After inducing a decision tree, rules were extracted on which methodology to follow according to the value of the meta-features.

The work of Talagala *et al.* (2018) presents an extensive meta-knowledge analysis. The first analysis consists of a probability matrix of the output of a classification model. The rows represent the time series, the columns are the forecasting models, and the matrix values are the probabilities of selecting a model for a time series. The authors extract interpretations based on the hierarchical grouping of the matrix by columns and the characteristics of the time series in the rows. The second analysis is based on feature importance, measured in individual conditional expectation score (ICE). This analysis aims to measure the effect of changing the value of a single time series feature on the probability output of the algorithm.

The following section better explores related work to our proposal. In this work, the focus will be on the dataset morphing approach. Dataset morphing is the basis for the development of the tsMorph method.

5.4 Synthetic time series

The need for datasets is a challenge in many applications. In order to reach the performance discrimination power and obtain more solid conclusions, a representative diversity of these datasets is also necessary. In the context of time series analysis, some approaches are available in the literature to generate synthetic data with realistic characteristics.

The autoregressive approach called GRATIS proposed by Kang, Hyndman and Li (2020) is a method that generates synthetic time series given the desired values of some meta-features. It is an evolutionary approach that searches for the parameters of a data generation method that minimizes the distance between the metafeatures of the generated time series and values defined by the user. The data generation is done using a Gaussian Mixture Autoregressive (MAR) model. The applications proposed for GRATIS were generating representative benchmarks and metadata augmentation. One important characteristic of this method is that the time series generated may be very different from each other. This occurs because the similarity which guides search is calculated in the selected meta-features space and not in the time series space. Additionally, the search optimizes only some meta-features, which means that the values of other meta-features (i.e., other characteristics of the time series) can be very different.

Yoon, Jarrett and Schaar (2019) proposed Time Series Generative Adversarial Networks (TimeGAN), an adaptation of Generative Adversarial Networks (GAN) for time series. GAN is a

framework for the generation of realistic data, and in TimeGAN the focus is on the preservation of the temporal dynamics in the generated time series. This is achieved by adding to the classic unsupervised adversarial loss a term representing the similarity to the original data. The work proposes improving the performance of prediction, forecasting, and classification tasks as direct applications.

GRATIS and TimeGAN frameworks are methods for generating synthetic time series that can be used to understand the behavior of algorithms, which is the goal of our work. Both are committed to the realism and diversity of the data generated. However, they have some limitations. First, the time series that is generated may be too different from each other. By analyzing the performance of algorithms on such time series, it is possible to obtain an overall perspective of their behavior (e.g., on what types of time series algorithm A perform better than algorithm B). However, it may also be important to have a more detailed characterization of their behavior (e.g., how does the relative performance of algorithms A and B evolve as the characteristics of the time series gradually change). The time series generated by autoregressive and the generative processes support this kind of analysis. Secondly, the computational costs of TimeGAN training and the GRATIS optimization process are high.

5.5 Dataset morphing for time series

tsMorph generates synthetic time series for understanding forecasting algorithms. Let a source time series Y_{source_t} and a target time series Y_{target_t} of the same length be observed up to time t. The proposed transformation function φ for the gradual transition between Y_{source_t} and Y_{target_t} is:

$$\varphi(i) = \alpha_i \cdot Y_{target_t} + (1 - \alpha_i) \cdot Y_{source_t} \tag{5.5}$$

where α is a contribution coefficient equals to $\frac{i}{n-1}$, n is the number of time series and i is the index of transformations between Y_{source_t} and Y_{target_t}. The function φ is a linear transformation whose contribution coefficient α spaces the generated synthetic time series equally in the range of values between Y_{source_t} and Y_{target_t} during the morphing process. The function φ is guaranteed to converge. The computational cost of the φ transformation is $O(t \times n)$. For time series with length $t >> n$, the computational cost of the transformation φ is $O(t)$. The dataset morphing for time series developed in the tsMorph method is defined by:

$$Y_{morph} : \{Y_i | Y_0 = Y_{source_t}, Y_n = Y_{target_t}, Y_i = \varphi(i)\}, 1 < i < n \tag{5.6}$$

where Y_{morph} is the set of datasets and n is the number of time series in the set Y_{morph}. Therefore, Y_i are the synthetic time series gradually generated by the tsMorph method. Figure 30 illustrates the application of the tsMorph method on two time series source and target with $n = 5$.

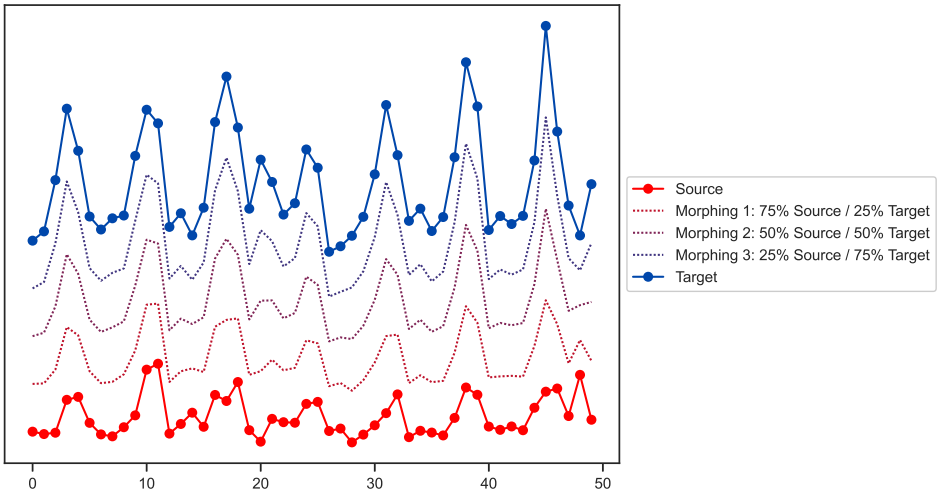


Figure 30 – Illustrative example of the tsMorph method with n=5.

Figure 30 illustrates how tsMorph generates synthetic time series. The value of α for each series can be found in the legend (percentage associated with the source). We can observe that the series are gradually less similar to the source and more similar to the target.

Given Y_{source_t} and Y_{target_t} and one or more forecasting algorithms with contrasting performance on time series. It is possible to understand the behavior of algorithms from the perspective of performance variation and meta-features in the set Y_{morph} using tsMorph. We can also control the metadata augmentation process with the generation of synthetic data based on realistic Y_{source_t} and Y_{target_t} time series.

The advantages of the tsMorph method are the simplicity of the method, which can be easily implemented in any programming language from the formulation. It is a computationally cheap method, as explained earlier. Moreover, it promotes the gradual transformation between the source and target time series. This last characteristic is of great interest in understanding the behavior of forecasting algorithms.

Our method complements other synthetic time series generation methods, such as GRATIS and TimeGAN, rather than replacing them. The main focus will be on investigating whether it is possible to extract interesting meta-knowledge from the time series of the set Y_{morph} generated with tsMorph.

5.6 Empirical validation

5.6.1 Experimental setup

We carried out experiments to illustrate the potential of tsMorph to support a better understanding of the performance of forecasting algorithms. The repository for reproducing the

results of this study is publicly available [1].

We use the 111 time series from the NN5 (CRONE, 2008) Competition. The dataset consists of 2 years of historical data from cash machines located in different places in England. The objective of the competition is to forecast 56 days ahead. The main characteristic of choosing this set of this dataset is that it is a benchmark known in the literature with time series of the same size. The data have missing observations that were linearly interpolated since this work objective is not to evaluate this effect.

The algorithms which we analyze are Long Short-Term Memory Neural Networks (LSTM) and Support Vector Regression (SVR). LSTM is a machine learning algorithm proposed by Hochreiter and Schmidhuber (1997). It is a widely used algorithm for forecasting time series with relevant results in the literature. SVR is an algorithm originating from statistical learning theory (VAPNIK, 1999). It also has good results in domain-varied time series forecasting (PARMEZAN; SOUZA; BATISTA, 2019d). These algorithms were selected because they are widely used for time series forecasting and there is limited understanding of their behavior. LSTM is from the Python package DARTS (HERZEN *et al.*, 2022). SVR is available in the Python packages scikit-learn (PEDREGOSA *et al.*, 2011) and sktime (LÖNING *et al.*, 2019).

To illustrate the usefulness of tsMorph, we selected source and target times series from the 111 in NN5 as follows. For each algorithm, we selected ten source time series and ten target time series for each algorithm. The ten source/target time series are the ones where the algorithm obtained best/worst predictive performance, respectively, according to MASE. We paired each of the sources with a target and applied tsMorph to generate the corresponding datasetoids. The goal is to understand the changes in the properties of the time series as the predictive performance of the algorithm degrades.

The meta-features used in this work are from the Python package Time Series Feature Extraction Library (TSFEL) (BARANDAS *et al.*, 2020). This package implements a set of 60 meta-features with different natures: statistical, temporal, and spectral. It should be noted that the meta-features were extracted only from the training data, and the performance is extracted only from the test data, while the tsMorph is applied to the complete time series.

5.6.2 Distance between synthetic, source, and target data

One of the motivations of this work is the need to generate sequences of time series that are increasingly different from a source and similar to another. Therefore, the first test was to assess if tsMorph can do that. In this experiment, tsMorph was used with $n = 20$. In the plots in Figure 31 the y-axis represents the average Euclidean distance between the synthetic data and the source/target data; the x-axis represents the order of the synthetic series in the set Y_{morph} (i.e. from source to target). Figures 31a and 31b present the results for LSTM and SVR, respectively.

[1] Repository URL: <https://anon-github.automl.cc/r/paper_tsmorph-7C1E/README.md>

The scatter bar is the standard deviation of the Euclidean distances between the time series of Y_{morph} and each algorithm experiment source and target time series.

(a) LSTM (b) SVR

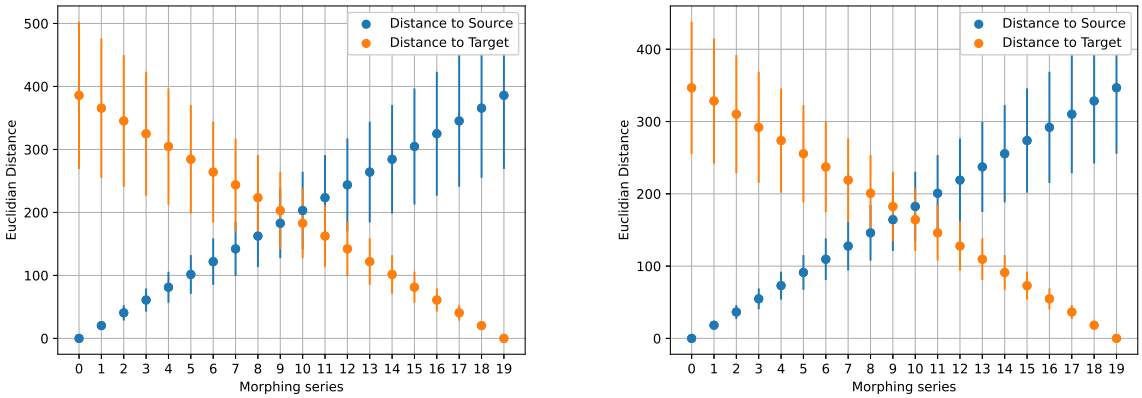


Figure 31 – Euclidean distance between synthetic, source, and target time series.

These results show that tsMorph generates sequences of gradually changing time series: the figure shows that the distance to the source gradually increases as the distance to the target gradually decreases. This is the first indication that the tsMorph method is working as expected.

The gradual variation of the Euclidean distance indicates that the variation of the meta-features that characterize the time series is also gradual (with the reasonable assumption that the functions that compute the meta-features are continuous). This transition of characteristics can reveal patterns for understanding the change in performance. This will be further investigated in the next section.

5.6.3 Understanding the performance of forecasting algorithms

The time series generated by tsMorph can be used to better understand the behavior of forecasting algorithms using different approaches, such as meta-learning. Here, for illustration purposes, we use a very simple method: we analyze the correlation between the value of a single meta-feature and the error of the algorithm.

Given that not all meta-features are expected to contain useful information about the predictive performance of algorithms, we present results for the three meta-features with stronger correlations. Tables 6a and 6b) present the mean and standard deviation of the results for all pairs of source and target time series for each algorithm.

The tables show the correlation results obtained by three meta-features. Before we interpret with an example, we explain each of these meta-features. A brief description of each of them follows:

Table 6 – Pearson correlation between performance and meta-features.

(a) LSTM

	Pearson correlation	
meta-feature	mean	std
median frequency	-0.8220	0.1173
mean absolute difference	-0.7900	0.1217
signal distance	-0.7860	0.1268

(b) SVR

	Pearson correlation	
meta-feature	mean	std
median frequency	-0.8862	0.1133
signal distance	-0.8061	0.2087
mean absolute difference	-0.8078	0.2120

Median frequency: it is the time series median frequency in the Fourier spectrum. The smaller the value, the greater the interval between peaks in the time series.

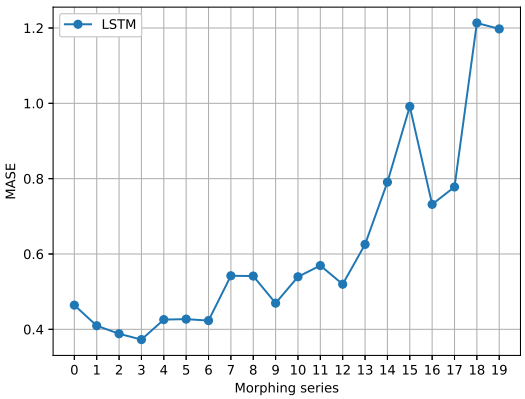
Mean absolute difference: it is the mean absolute difference between consecutive observations in the time series. The smaller the value, the smoother the time series.

Signal distance: it is the total distance traveled by the signal using the hypotenuse between 2 observations. It measures the amplitude of the total variation in time series without considering whether this variation occurs at fixed intervals.

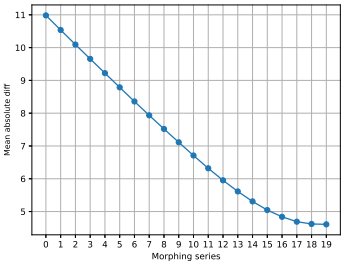
The three meta-features that had a strong negative correlation and consistency with the performance of the two forecasting algorithms are related to the variation of time series values over time. Therefore, in this analysis, we evaluate how the meta-features influence the difference in performance between the forecasting algorithms and the naive forecast. Figures 32 and 33 show the curves of performance of LSTM and SVR, respectively and values of the meta-features, on an illustrative pair of source-target time series. From these figures and the interpretation of the meta-features, it is possible to extract knowledge about the prediction algorithms.

In Figure 32a, we observe that the predictive performance varies gradually with the morphing process, as expected. We see that the meta-features mean absolute difference in Figure 32b and the signal distance in Figure 32d also vary gradually with the morphing process. This indicates that the transformations did not produce unusual values in the generated time series, also as expected. Additionally, the values of the two meta-features decrease with the similarity to the target. Given that the predictive performance of LSTM is degrading, this means that the smaller the variations over time in the time series, the worse the performance of the LSTM is. Concerning the other meta-feature, median frequency, Figure 32c shows that it, in general, also decreases with the morphing process. However, the shape is not as smooth as the other two meta-features. In the interval from index 3 to index 11, the value of the meta-feature remains constant, with a sharp drop from index 12 onwards. Curiously, in the same interval, there is also an increase in MASE (Figure 32a). This means that the predictive performance of LSTM relative to the naive baseline decreases with smoother time series. This is to be expected, not because of the effect on the forecasting error of LSTM but rather because the naive prediction becomes more accurate.

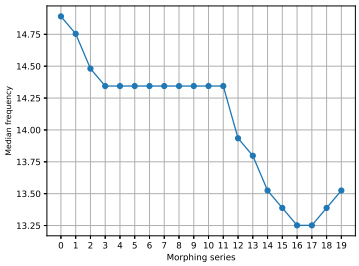
(a) Predictive performance curve from NN5-016 to NN5-026.



(b) Mean absolute difference.



(c) Median frequency.



(d) Signal distance.

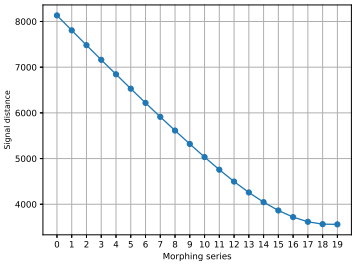
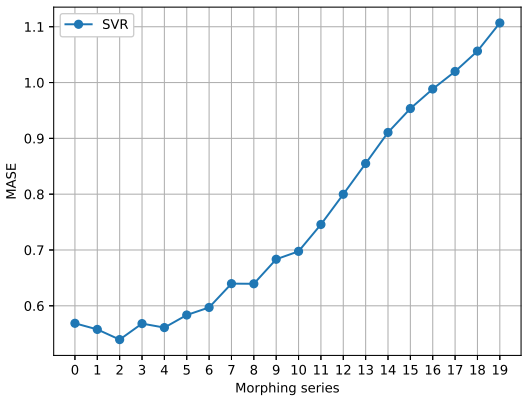


Figure 32 – A example from LSTM experiment: tsMorph from NN5-016 to NN5-026.

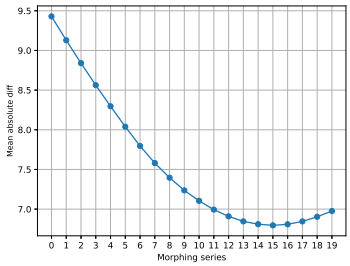
Figure 33a shows that the relation between the forecasting error of SVR and the meta-feature values is generally very similar to what was previously observed for LSTM, even though the source and target datasets are different. We observe that the smaller the amplitude of variation of the time series, the worse the performance. of SVR, relative to the naive forecast. The median frequency meta-feature (Figure 33c). The behavior after index 12 is not as clear. The median frequency increases slightly, indicating an increase in the variation of the time series, but the MASE continues to grow. This means that, even though the higher the variation, the larger the gain in forecasting accuracy obtained with SVR compared to the naive forecast, the results also indicate that further analysis is still necessary, possibly using other meta-features.

In summary, these results show that tsMorph can be used to generate synthetic time series from real ones that can be used to understand the behavior of forecasting algorithms with gradual variations of the data: 1. the generated time series are gradually less similar to the source and more similar to the target, in the data, meta-feature, and forecasting performance spaces; 2. it is possible to extract knowledge about the behavior of forecasting methods, even using simple analysis methods.

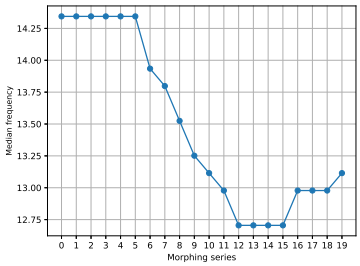
(a) Predictive performance curve from NN5-008 to NN5-011.



(b) Mean absolute difference.



(c) Median frequency.



(d) Signal distance.

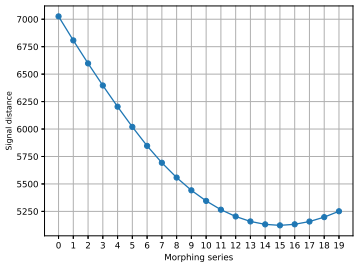


Figure 33 – A example from SVR experiment: tsMorph from NN5-008 to NN5-011.

5.7 Conclusion

In contrast to the extensive work on developing and testing forecasting algorithms, there is limited work on understanding their behavior. In this work, we propose a simple method of generating synthetic time series that can be used for that purpose. The tsMorph method gradually transforms a source time series into a target time series, generating a sequence of semi-synthetic time series. The data generated by tsMorph supports an empirical and systematic approach to understanding the behavior of algorithms.

To illustrate the potential of tsMorph, we analyzed the behavior of two algorithms, LSTM and SVR, using a simple approach based on the correlation between the values of meta-features of the data and the forecasting accuracy of the algorithms. The results showed that, as desired, the tsMorph method generates synthetic time series with Euclidean distance gradually increasing concerning the source time series and decreasing concerning the target time series for both prediction algorithms in the data, meta-feature, and forecasting accuracy spaces. Furthermore, even in such a simple experimental scenario, it was possible to obtain knowledge about the behavior of the algorithms. The results suggest that increasing the amplitude and range of variation of the time series can improve the performance of the LSTM and SVR algorithms, relative to the naive forecast.

One limitation of the work presented here is that the transformation operation is very simple and, most importantly, is only applicable to time series of the same size. However, we plan to develop other transformation options which can deal with time series of varying sizes. Additionally, in the experiments, we focused on analyzing algorithms individually. However, by choosing the target and source time series differently, we can carry out other types of analyses (e.g., comparing the performance of two algorithms). In fact, tsMorph can be used to define and understand the borders of the meta-data space that delimit the areas of expertise of each algorithm.

Finally, the (semi)-synthetic data generated by tsMorph can also be used as training data for autoML and meta-learning approaches, addressing a major limitation of the current work in the area, which is the limited amount of meta-data.

5.8 Broader impact

Understanding the behavior of forecasting algorithms is one of the core challenges for the responsible use of AI and ML technology. tsMorph can be used to improve that understanding. Additionally, morphing can be applied to other types of data analysis (e.g., recommender systems (CORREIA; SOARES; JORGE, 2019)), so the scope of the applicability of this approach goes beyond time series. Additionally, (semi-)synthetic data is important to promote data sharing with a lower risk of violating privacy. Finally, due to its low computational complexity, the tsMorph method can reduce energy consumption in the generation of synthetic data.

CHAPTER

6

CONCLUSION

The selection of algorithms is an essential and time consuming step in time series forecasting. After algorithm selection, understanding the characteristics of the time series related to predictive performance is a differential in evaluating the forecasting process. Although there are some proposals for automatic algorithm selection based on metalearning for time series forecasting, a study of design choices in the framework is necessary. Another direction is related to the selection of a combination of forecasts. Previous studies that deal with forecast combinations, the choice of algorithms is usually made arbitrarily, based on the knowledge of an expert. Understanding under which conditions a forecasting algorithm will have good or bad performance is challenging. Few studies propose empirical and systematic approaches to understanding algorithm performance. In this thesis, some paths were proposed to solve these problems.

We empirically evaluated the use of various performance measures for meta-label definition in selecting forecasting algorithms based on metalearning. The main result we obtained is that the performance measure used for meta-label definition does not significantly influence the performance of metalearning at the meta and base levels. Therefore, using less complex measures such as Mean Absolute Error or Mean Squared Error can reduce computational costs in this algorithm selection method.

For the automatic selection of forecasting algorithms, we propose the MetaFore approach. MetaFore is a forecasting combination approach that uses the STL decomposition as a preprocessing step and metalearning to select forecasting algorithms. After decomposing the time series, the trend and noise components were combined additively and forecasted by a machine learning algorithm. The seasonality component was predicted with the seasonal naive method. We proposed to use metalearning to recommend the machine learning algorithm based on the meta-features of the time series. From the meta-knowledge obtained in the metalearning process, we extracted helpful information about the characteristics of the time series related to the selected algorithm. We obtained competitive predictive performance and computational

cost results. The MetaFore approach outperformed individual hybrid forecasting algorithms and LSTM and bilateral LSTM neural networks.

When using the STL decomposition for time series analysis, testing for the normality of the residual component is common. There is no evidence of cyclic behavior residuals if the residual component follows the normal distribution. We hypothesize that using the STL decomposition as a preprocessing step improves predictive performance when the residual follows the normal distribution. To test this hypothesis, we applied the STL decomposition to the entire set of time series. Based on the result of the normality test, we divided this set into two groups: residuals do not follow/follow the normal distribution. We applied two machine learning algorithm configurations to these groups. For the first configuration we used an algorithms without the STL decomposition. For the second, we adopted the STL decomposition, combined the trend and residual and use a predictive model induced by a machine learning algorithm. The seasonality component is predicted with the seasonal naive method in this second approach. In the analysis of the results, we obtained that when the residual of the STL decomposition follows the normal distribution, hybrid algorithms that use the STL as a preprocessing step have better predictive performance than default algorithms. In the other group, there is no statistically relevant difference. These results validate the hypothesis elaborated in the study.

The tsMorph method was developed to deepen the understanding of time series characteristics related to performance variation. The tsMorph method is a simple approach for generating synthetic time series based on dataset morphing. Dataset morphing is the gradual transformation of a source dataset into a target dataset, generating intermediate datasets. These intermediate datasets can be used to analyze algorithm performance and characteristic variation between the original datasets. The proposed approach yielded the expected results, revealing meta-features strongly correlated with performance variation of LSTM and SVR algorithms.

6.0.1 Limitations and Future Work

The limitations of this study are distributed throughout each path we adopted to solve the problems. In Chapter 2, the limitations include needing a more extensive set of time series to validate better the results obtained. Another line of improvement for this work is the investigation of these properties in different forms of meta-label, such as ranking recommendations or meta-regression. In Chapter 3, a natural extension of the work is the recommendation of algorithms for the seasonal component or each component separately based on metalearning. In Chapter 4, the analysis could be extended to other data frequencies, such as daily, weekly, and quarterly. In Chapter 5, the main limitation is that the method only applies to time series of the same length. More complex transformations may solve this problem. The possibility of applying the synthetic time series generated by the tsMorph method as training data in metalearning and autoML can also be evaluated.

BIBLIOGRAPHY

ALCOBAÇA, E.; SIQUEIRA, F.; RIVOLLI, A.; GARCIA, L. P.; OLIVA, J. T.; CARVALHO, A. C. D. Mfe: Towards reproducible meta-feature extraction. **The Journal of Machine Learning Research**, JMLRORG, v. 21, n. 1, p. 4503–4507, 2020. Citation on page 80.

ALI, A. R.; GABRYS, B.; BUDKA, M. Cross-domain meta-learning for time-series forecasting. **Procedia Computer Science**, Elsevier, v. 126, p. 9–18, 2018. Citations on pages 25, 30, 48, and 49.

ANDERSEN, T. G. Some reflections on analysis of high-frequency data. **Journal of Business & Economic Statistics**, Taylor & Francis Group, v. 18, n. 2, p. 146–153, 2000. Citation on page 45.

ARMSTRONG, J. S. Combining forecasts. In: **Principles of forecasting**. [S.l.]: Springer, 2001. p. 417–439. Citation on page 30.

ARMSTRONG, J. S.; ADYA, M.; COLLOPY, F. Rule-based forecasting: Using judgment in time-series extrapolation. **Principles of forecasting: A handbook for researchers and practitioners**, Springer, p. 259–282, 2001. Citations on pages 24, 26, and 80.

BANDARA, K.; BERGMEIR, C.; SMYL, S. Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. **Expert Systems with Applications**, v. 140, p. 112896, 2020. ISSN 0957-4174. Available: <http://www.sciencedirect.com/science/article/pii/S0957417419306128>. Citation on page 69.

BARAK, S.; NASIRI, M.; ROSTAMZADEH, M. Time series model selection with a meta-learning approach: evidence from a pool of forecasting algorithms. **arXiv preprint arXiv:1908.08489**, 2019. Citations on pages 25, 30, and 49.

BARANDAS, M.; FOLGADO, D.; FERNANDES, L.; SANTOS, S.; ABREU, M.; BOTA, P.; LIU, H.; SCHULTZ, T.; GAMBOA, H. Tsfel: Time series feature extraction library. **SoftwareX**, Elsevier, v. 11, p. 100456, 2020. Citation on page 84.

BARANDELA, R.; SÁNCHEZ, J. S.; GARCIA, V.; RANGEL, E. Strategies for learning in class imbalance problems. **Pattern Recognition**, v. 36, n. 3, p. 849–851, 2003. Citation on page 54.

BERGMEIR, C. N.; SÁNCHEZ, J. M. B. *et al.* Neural networks in r using the stuttgart neural network simulator: Rsnns. **Journal of Statistical Software**, 2012. Citation on page 36.

BISHOP, C. M. **Pattern Recognition and Machine Learning (Information Science and Statistics)**. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN 0387310738. Citations on pages 71 and 72.

BOX, G.; JENKINS, G.; REINSEL, G.; LJUNG, G. **Time Series Analysis: Forecasting and Control**. Wiley, 2015. (Wiley Series in Probability and Statistics). ISBN 9781118674925. Available: <https://books.google.com.br/books?id=rNt5CgAAQBAJ>. Citation on page 65.

BOX, G. E.; JENKINS, G. M.; REINSEL, G. C.; LJUNG, G. M. **Time series analysis: forecasting and control**. [S.l.]: John Wiley & Sons, 1970. Citation on page 23.

____. **Time series analysis: forecasting and control**. [S.l.]: John Wiley & Sons, 2015. Citation on page 31.

BRAZDIL, P.; CARRIER, C. G.; SOARES, C.; VILALTA, R. **Metalearning: Applications to data mining**. [S.l.]: Springer Science & Business Media, 2008. Citations on pages 25, 30, 44, 47, and 48.

BRAZDIL, P.; RIJN, J. N. van; SOARES, C.; VANSCHOREN, J. **Metalearning: Applications to Automated Machine Learning and Data Mining**. [S.l.]: Springer Nature, 2022. Citations on pages 15 and 47.

____. **Metalearning: Applications to Automated Machine Learning and Data Mining**. [S.l.]: Springer Nature, 2022. Citation on page 80.

BREIMAN, L.; FRIEDMAN, J. H.; OLSHEN, R. A.; STONE, C. J. **Classification and Regression Trees**. Monterey, CA: CRC Press Book, 1984. Citation on page 35.

____. **Classification and regression trees**. [S.l.]: Routledge, 1984. Citation on page 55.

BUITINCK, L.; LOUPPE, G.; BLONDEL, M.; PEDREGOSA, F.; MUELLER, A.; GRISEL, O.; NICULAE, V.; PRETTENHOFER, P.; GRAMFORT, A.; GROBLER, J.; LAYTON, R.; VANDERPLAS, J.; JOLY, A.; HOLT, B.; VAROQUAUX, G. API design for machine learning software: experiences from the scikit-learn project. In: **ECML PKDD Workshop: Languages for Data Mining and Machine Learning**. [S.l.: s.n.], 2013. p. 108–122. Citation on page 72.

CERQUEIRA, V.; TORGO, L.; PINTO, F.; SOARES, C. Arbitrated ensemble for time series forecasting. In: SPRINGER. **Joint European conference on machine learning and knowledge discovery in databases**. [S.l.], 2017. p. 478–494. Citation on page 49.

CERQUEIRA, V.; TORGO, L.; SOARES, C. Machine learning vs statistical methods for time series forecasting: Size matters. **arXiv preprint arXiv:1909.13316**, 2019. Citation on page 64.

CHRIST, M.; BRAUN, N.; NEUFFER, J.; KEMPA-LIEHR, A. W. Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). **Neurocomputing**, Elsevier, v. 307, p. 72–77, 2018. Citation on page 52.

CLEMEN, R. T. Combining forecasts: A review and annotated bibliography. **International journal of forecasting**, Elsevier, v. 5, n. 4, p. 559–583, 1989. Citations on pages 25 and 64.

CLEVELAND, R. B.; CLEVELAND, W. S.; MCRAE, J. E.; TERPENNING, I. Stl: A seasonal-trend decomposition. **J. Off. Stat**, v. 6, n. 1, p. 3–73, 1990. Citation on page 46.

CLEVELAND, R. B.; CLEVELAND, W. S.; MCRAE JEAN E ADN TERPENNING, I. Stl: A seasonal-trend decomposition procedure based on loess. **Journal of Official Statistics**, v. 6, n. 1, p. 3–73, 1990. Citation on page 66.

CORREIA, A.; SOARES, C.; JORGE, A. Dataset morphing to analyze the performance of collaborative filtering. In: SPRINGER. **Discovery Science: 22nd International Conference, DS 2019, Split, Croatia, October 28–30, 2019, Proceedings 22**. [S.l.], 2019. p. 29–39. Citations on pages 27, 78, 80, and 89.

CRONE, S. **NN5 Forecasting Competition**. 2008. Accessed on 2023-02-08. Available: <http://www.neural-forecasting-competition.com/NN5/index.htm>. Citation on page 84.

CRYER, J. D.; CHAN, K.-S. **Time Series Analysis: With Applications in R**. 2nd. ed. [S.l.]: Springer, 2008. (Springer Texts in Statistics). Citations on pages 23 and 30.

DAGUM, E. B.; BIANCONCINI, S. **Seasonal adjustment methods and real time trend-cycle estimation**. [S.l.]: Springer, 2016. Citation on page 46.

DEB, C.; ZHANG, F.; YANG, J.; LEE, S. E.; SHAH, K. W. A review on time series forecasting techniques for building energy consumption. **Renewable and Sustainable Energy Reviews**, Elsevier, v. 74, p. 902–924, 2017. Citations on pages 23, 44, and 64.

DEISENROTH, M. P.; FAISAL, A. A.; ONG, C. S. **Mathematics for Machine Learning**. [S.l.]: Cambridge University Press, 2020. Citation on page 71.

DEMPSEY, D.; CRONIN, S. J.; MEI, S.; KEMPA-LIEHR, A. W. Automatic precursor recognition and real-time forecasting of sudden explosive volcanic eruptions at whakaari, new zealand. **Nature communications**, Nature Publishing Group, v. 11, n. 1, p. 1–8, 2020. Citation on page 52.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. **The Journal of Machine learning research**, JMLR. org, v. 7, p. 1–30, 2006. Citation on page 54.

DUDEK, G.; PEŁKA, P.; SMYL, S. A hybrid residual dilated lstm and exponential smoothing model for midterm electric load forecasting. **IEEE Transactions on Neural Networks and Learning Systems**, IEEE, 2021. Citation on page 49.

EVGENIOU, T.; POGGIO, T.; PONTIL, M.; VERRI, A. Regularization and statistical learning theory for data analysis. **Computational Statistics & Data Analysis**, v. 38, n. 4, p. 421 – 432, 2002. ISSN 0167-9473. Nonlinear Methods and Data Mining. Available: <http://www.sciencedirect.com/science/article/pii/S016794730100069X>. Citation on page 71.

EXTERKATE, P. Model selection in kernel ridge regression. **Computational Statistics & Data Analysis**, v. 68, p. 1–16, 2013. ISSN 0167-9473. Available: <http://www.sciencedirect.com/science/article/pii/S0167947313002181>. Citation on page 71.

FINN, C.; ABBEEL, P.; LEVINE, S. Model-agnostic meta-learning for fast adaptation of deep networks. In: JMLR. ORG. **Proceedings of the 34th International Conference on Machine Learning-Volume 70**. [S.l.], 2017. p. 1126–1135. Citations on pages 25 and 47.

GOODFELLOW, I.; POUGET-ABADIE, J.; MIRZA, M.; XU, B.; WARDE-FARLEY, D.; OZAIR, S.; COURVILLE, A.; BENGIO, Y. Generative adversarial networks. **Communications of the ACM**, ACM New York, NY, USA, v. 63, n. 11, p. 139–144, 2020. Citation on page 78.

GURNANI, M.; KORKE, Y.; SHAH, P.; UDMALE, S.; SAMBHE, V.; BHIRUD, S. Forecasting of sales by using fusion of machine learning techniques. In: IEEE. **2017 International Conference on Data Management, Analytics and Innovation (ICDMAI)**. [S.l.], 2017. p. 93–101. Citations on pages 44, 49, and 51.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. H. **The elements of statistical learning: data mining, inference, and prediction**. [S.l.]: Springer, 2009. Citation on page 55.

HAYKIN, S. **Neural networks: a comprehensive foundation**. [S.l.]: Prentice Hall PTR, 1994. Citation on page 32.

HERZEN, J.; LÃ¤SSIG, F.; PIAZZETTA, S. G.; NEUER, T.; TAFTI, L.; RAILLE, G.; POTTELBERGH, T. V.; PASIEKA, M.; SKRODZKI, A.; HUGUENIN, N.; DUMONAL, M.; KOÅ›CISZ, J.; BADER, D.; GUSSET, F.; BENHEDDI, M.; WILLIAMSON, C.; KOSINSKI, M.; PETRIK, M.; GROSCH, G. Darts: User-friendly modern machine learning for time series. **Journal of Machine Learning Research**, v. 23, n. 124, p. 1–6, 2022. Available: <http://jmlr.org/papers/v23/21-1177.html>. Citation on page 84.

HEWAMALAGE, H.; BERGMEIR, C.; BANDARA, K. Recurrent neural networks for time series forecasting: Current status and future directions. **International Journal of Forecasting**, Elsevier, v. 37, n. 1, p. 388–427, 2021. Citation on page 54.

HIBON, M.; EVGENIOU, T. To combine or not to combine: selecting among forecasts and their combinations. **International journal of forecasting**, Elsevier, v. 21, n. 1, p. 15–24, 2005. Citations on pages 25 and 64.

HO, T. K. Random decision forests. In: IEEE. **Proceedings of 3rd international conference on document analysis and recognition**. [S.l.], 1995. v. 1, p. 278–282. Citation on page 52.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, MIT press, v. 9, n. 8, p. 1735–1780, 1997. Citation on page 84.

HUTTER, F.; KOTTHOFF, L.; VANSCHOREN, J. Automatic machine learning: methods, systems, challenges. **Challenges in Machine Learning**, Springer, 2019. Citation on page 47.

____. Automatic machine learning: methods, systems, challenges. **Challenges in Machine Learning**, Springer, 2019. Citation on page 80.

HYLAND, S. L.; ESTEBAN, C.; RÄTSCH, G. Real-valued (medical) time series generation with recurrent conditional gans. **stat**, v. 1050, n. 8, 2017. Citation on page 78.

HYNDMAN, R.; ATHANASOPOULOS, G. **Forecasting: Principles and Practice**. 3rd. ed. Melbourne, Australia: OTexts, 2021. Available: <"OTexts.com/fpp3",Acessedon01-18-2021>. Citations on pages 63, 66, 67, 68, 69, and 73.

HYNDMAN, R.; KOEHLER, A. B.; ORD, J. K.; SNYDER, R. D. **Forecasting with exponential smoothing: the state space approach**. [S.l.]: Springer Science & Business Media, 2008. Citations on pages 31, 32, and 45.

HYNDMAN, R. J.; ATHANASOPOULOS, G. **Forecasting: principles and practice**. [S.l.]: OTexts, 2018. Citations on pages 23, 31, 34, 44, and 46.

____. **Forecasting: principles and practice**. [S.l.]: OTexts, 2018. Citation on page 78.

HYNDMAN, R. J.; KHANDAKAR, Y. *et al.* **Automatic time series for forecasting: the forecast package for R**. [S.l.]: Monash University, Department of Econometrics and Business Statistics ..., 2007. Citation on page 34.

HYNDMAN, R. J.; KOEHLER, A. B. Another look at measures of forecast accuracy. **International Journal of Forecasting**, v. 22, n. 4, p. 679 – 688, 2006. ISSN 0169-2070. Available: <http://www.sciencedirect.com/science/article/pii/S0169207006000239>. Citation on page 72.

HYNDMAN, R. J. *et al.* Another look at forecast-accuracy metrics for intermittent demand. **Foresight: The International Journal of Applied Forecasting**, International Institute of Forecasters, v. 4, n. 4, p. 43–46, 2006. Citations on pages 24, 31, and 79.

KANG, Y.; HYNDMAN, R. J.; LI, F. Gratis: Generating time series with diverse and controllable characteristics. **Statistical Analysis and Data Mining**, John Wiley and Sons Inc., v. 13, p. 354–376, 8 2020. ISSN 19321872. Citation on page 81.

KE, G.; MENG, Q.; FINLEY, T.; WANG, T.; CHEN, W.; MA, W.; YE, Q.; LIU, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. **Advances in neural information processing systems**, v. 30, 2017. Citation on page 52.

KEMPA-LIEHR, A. W.; ORAM, J.; WONG, A.; FINCH, M.; BESIER, T. Feature engineering workflow for activity recognition from synchronized inertial measurement units. In: SPRINGER. **Asian Conference on Pattern Recognition**. [S.l.], 2019. p. 223–231. Citation on page 52.

KEOGH, E.; KASETTY, S. On the need for time series data mining benchmarks: a survey and empirical demonstration. In: **Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.: s.n.], 2002. p. 102–111. Citation on page 78.

KUHN, M. The caret package. **R Foundation for Statistical Computing, Vienna, Austria. URL https://cran. r-project. org/package= caret**, Citeseer, 2012. Citations on pages 35 and 36.

Kück, M.; Crone, S. F.; Freitag, M. Meta-learning with neural networks and landmarking for forecasting model selection an empirical evaluation of different feature sets applied to industry data. In: **2016 International Joint Conference on Neural Networks (IJCNN)**. [S.l.: s.n.], 2016. Citations on pages 25, 30, 33, 34, 48, and 49.

LAI, T. L.; ROBBINS, H.; WEI, C. Z. Strong consistency of least squares estimates in multiple regression ii. **Journal of multivariate analysis**, Academic Press, v. 9, n. 3, p. 343–361, 1979. Citation on page 52.

LEMKE, C.; GABRYS, B. Meta-learning for time series forecasting and forecast combination. **Neurocomputing**, Elsevier, v. 73, n. 10-12, p. 2006–2016, 2010. Citations on pages 25, 26, 30, 48, 49, and 81.

LI, J.; DENG, D.; ZHAO, J.; CAI, D.; HU, W.; ZHANG, M.; HUANG, Q. A novel hybrid short-term load forecasting method of smart grid using mlr and lstm neural network. **IEEE Transactions on Industrial Informatics**, IEEE, 2020. Citation on page 64.

LI, W.; SHI, Q.; SIBTAIN, M.; LI, D.; MBANZE, D. E. A hybrid forecasting model for short-term power load based on sample entropy, two-phase decomposition and whale algorithm optimized support vector regression. **IEEE Access**, IEEE, v. 8, p. 166907–166921, 2020. Citation on page 64.

LI, Y.; BAO, T.; GONG, J.; SHU, X.; ZHANG, K. The prediction of dam displacement time series using stl, extra-trees, and stacked lstm neural network. **IEEE Access**, IEEE, v. 8, p. 94440–94452, 2020. Citations on pages 49 and 64.

LIAW, A.; WIENER, M. Classification and regression by randomforest. **R News**, v. 2, n. 3, p. 18–22, 2002. Available: <https://CRAN.R-project.org/doc/Rnews/>. Citation on page 36.

LÖNING, M.; BAGNALL, A.; GANESH, S.; KAZAKOV, V.; LINES, J.; KIRÁLY, F. J. sktime: A unified interface for machine learning with time series. **arXiv preprint arXiv:1909.07872**, 2019. Citation on page 84.

LU, W.; RUI, Y.; YI, Z.; RAN, B.; GU, Y. A hybrid model for lane-level traffic flow forecasting based on complete ensemble empirical mode decomposition and extreme gradient boosting. **IEEE Access**, IEEE, v. 8, p. 42042–42054, 2020. Citation on page 64.

MA, S.; FILDES, R. Retail sales forecasting with meta-learning. **European Journal of Operational Research**, Elsevier, 2020. Citations on pages 25, 48, and 49.

MAKRIDAKIS, S.; ANDERSEN, A.; CARBONE, R.; FILDES, R.; HIBON, M.; LEWANDOWSKI, R.; NEWTON, J.; PARZEN, E.; WINKLER, R. The accuracy of extrapolation (time series) methods: Results of a forecasting competition. **Journal of forecasting**, Wiley Online Library, v. 1, n. 2, p. 111–153, 1982. Citation on page 48.

MAKRIDAKIS, S.; HIBON, M. The m3-competition: results, conclusions and implications. **International journal of forecasting**, Elsevier, v. 16, n. 4, p. 451–476, 2000. Citation on page 48.

MAKRIDAKIS, S.; SPILIOTIS, E.; ASSIMAKOPOULOS, V. Statistical and machine learning forecasting methods: Concerns and ways forward. **PloS one**, Public Library of Science, v. 13, n. 3, 2018. Citations on pages 31 and 54.

____. Statistical and machine learning forecasting methods: Concerns and ways forward. **PloS one**, Public Library of Science San Francisco, CA USA, v. 13, p. e0194889, 2018. Citations on pages 78 and 79.

____. The m4 competition: 100000 time series and 61 forecasting methods. **International Journal of Forecasting**, Elsevier, n. 1, p. 54–74, 2020. Citations on pages 25, 51, and 64.

____. M5 accuracy competition: Results, findings, and conclusions. **International Journal of Forecasting**, Elsevier, 2022. Citation on page 52.

MAKRIDAKIS, S.; WHEELWRIGHT, S.; HYNDMAN, R. J. **Forecasting: methods and applications**. [S.l.]: John Wiley & Sons, 1998. Citation on page 46.

MANTOVANI, R. G.; ROSSI, A. L. D.; ALCOBAÇA, E.; VANSCHOREN, J.; CARVALHO, A. C. P. L. F. de. A meta-learning recommender system for hyperparameter tuning: Predicting when tuning improves SVM classifiers. **Inf. Sci.**, v. 501, p. 193–221, 2019. Citation on page 30.

MARSLAND, S. **Machine learning: an algorithmic perspective**. [S.l.]: Chapman and Hall/CRC, 2014. Citation on page 31.

MEADE, N. Evidence for the selection of forecasting methods. **Journal of forecasting**, Wiley Online Library, v. 19, n. 6, p. 515–535, 2000. Citations on pages 24, 25, 30, 48, and 49.

MEDEIROS, M. C.; TERÄSVIRTA, T.; RECH, G. Building neural network models for time series: a statistical approach. **Journal of Forecasting**, Wiley Online Library, v. 25, n. 1, p. 49–75, 2006. Citations on pages 31 and 32.

MERSMANN, O. *et al.* microbenchmark: Accurate timing functions. **R package version**, v. 1, n. 4-2, p. 1, 2015. Citation on page 37.

MONTERO-MANSO, P.; ATHANASOPOULOS, G.; HYNDMAN, R. J.; TALAGALA, T. S. Fforma: Feature-based forecast model averaging. **International Journal of Forecasting**, Elsevier, v. 36, n. 1, p. 86–92, 2020. Citations on pages 25 and 49.

MONTGOMERY, D. C.; JENNINGS, C. L.; KULAHCI, M. **Introduction to time series analysis and forecasting**. [S.l.]: John Wiley & Sons, 2015. Citations on pages 23, 45, 63, and 65.

____. **Introduction to time series analysis and forecasting**. [S.l.]: John Wiley & Sons, 2015. Citations on pages 78 and 79.

MONTGOMERY, D. C.; JOHNSON, L. A.; GARDINER, J. S. **Forecasting and time series analysis**. [S.l.]: McGraw-Hill Companies, 1990. Citations on pages 23, 30, and 31.

MULLER, K.-R.; SMOLA, A. J.; RäTSCH, G.; SCHöKOPF, B.; KOHLMORGEN, J.; VAPNIK, V. Using support vector machines for time series prediction. In: ____. **Advances in Kernel Methods: Support Vector Learning**. Cambridge, MA, USA: MIT Press, 1999. p. 243–253. ISBN 0262194163. Citations on pages 23 and 64.

NEMENYI, P. **Distribution-free Multiple Comparisons**. Princeton University, 1963. Available: <https://books.google.com.br/books?id=nhDMtgAACAAJ>. Citation on page 73.

PARMEZAN, A.; LEE, H.; WU, F. Metalearning for choosing feature selection algorithms in data mining: Proposal of a new framework. **Expert Systems with Applications**, v. 75, 01 2017. Citations on pages 65 and 73.

PARMEZAN, A. R.; SOUZA, V. M.; BATISTA, G. E. Time series prediction via similarity search: Exploring invariances, distance measures and ensemble functions. **IEEE Access**, IEEE, v. 10, p. 78022–78043, 2022. Citations on pages 52 and 57.

PARMEZAN, A. R. S.; BATISTA, G. E. A. P. A. **ICMC-USP Time Series Prediction Repository**. São Carlos, Brasil: [s.n.], 2014. Citations on pages 31, 33, 65, 67, and 70.

PARMEZAN, A. R. S.; SOUZA, V. M.; BATISTA, G. E. Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model. **Information sciences**, Elsevier, v. 484, p. 302–337, 2019. Citation on page 52.

____. Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model. **Information Sciences**, v. 484, p. 302 – 337, 2019. ISSN 0020-0255. Available: <http://www.sciencedirect.com/science/article/pii/S0020025519300945>. Citations on pages 64, 70, 72, and 73.

PARMEZAN, A. R. S.; SOUZA, V. M. A.; BATISTA, G. E. A. P. A. Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model. **Information Sciences**, Elsevier, United States of America, v. 484, p. 302–337, May 2019. ISSN 0020-0255. Citations on pages 44, 49, and 54.

____. Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model. **Information Sciences**, Elsevier, v. 484, p. 302–337, 5 2019. ISSN 0020-0255. Citation on page 84.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011. Citation on page 84.

PIMENTEL, B. A.; CARVALHO, A. C. P. L. F. de. A new data characterization for selecting clustering algorithms using meta-learning. **Inf. Sci.**, v. 477, p. 203–219, 2019. Citation on page 30.

PRUDÊNCIO, R. B.; LUDERMIR, T. B. Meta-learning approaches to selecting time series models. **Neurocomputing**, Elsevier, v. 61, p. 121–137, 2004. Citations on pages 24, 25, 30, 33, 34, 48, and 49.

REN, Y.; SUGANTHAN, P.; SRIKANTH, N. A comparative study of empirical mode decomposition-based short-term wind speed forecasting methods. **IEEE Transactions on Sustainable Energy**, IEEE, v. 6, n. 1, p. 236–244, 2014. Citation on page 64.

RICE, J. R. The algorithm selection problem. In: **Advances in computers**. [S.l.]: Elsevier, 1976. v. 15, p. 65–118. Citations on pages 33 and 47.

RIVOLLI, A.; GARCIA, L. P.; SOARES, C.; VANSCHOREN, J.; CARVALHO, A. C. de. Towards reproducible empirical research in meta-learning. **arXiv preprint arXiv:1808.10406**, p. 32–52, 2018. Citation on page 48.

SEABOLD, S.; PERKTOLD, J. statsmodels: Econometric and statistical modeling with python. In: **9th Python in Science Conference**. [S.l.: s.n.], 2010. Citation on page 72.

SEZER, O. B.; GUDELEK, M. U.; OZBAYOGLU, A. M. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. **Applied soft computing**, Elsevier, v. 90, p. 106181, 2020. Citation on page 54.

SHAH, C. Model selection in univariate time series forecasting using discriminant analysis. **International Journal of Forecasting**, Elsevier, v. 13, n. 4, p. 489–500, 1997. Citation on page 36.

SHMUELI, G.; JR, K. C. L. **Practical time series forecasting with r: A hands-on guide**. [S.l.]: Axelrod Schnall Publishers, 2016. Citation on page 23.

SILVESTRE, G. D.; SANTOS, M. R. dos; CARVALHO, A. C. de. Seasonal-trend decomposition based on loess+ machine learning: Hybrid forecasting for monthly univariate time series. In: IEEE. **2021 International Joint Conference on Neural Networks (IJCNN)**. [S.l.], 2021. p. 1–7. Citations on pages 44, 49, and 51.

SMITH-MILES, K.; MUñOZ, M. A. Instance space analysis for algorithm testing: Methodology and software tools. **ACM Computing Surveys**, Association for Computing Machinery (ACM), 11 2022. ISSN 0360-0300. Citation on page 26.

SMOLA, A. J.; SCHÖLKOPF, B. A tutorial on support vector regression. **Statistics and computing**, Springer, v. 14, n. 3, p. 199–222, 2004. Citation on page 52.

SMYL, S. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. **International Journal of Forecasting**, Elsevier, v. 36, n. 1, p. 75–85, 2020. Citations on pages 25, 44, 49, 50, and 65.

SOUZA, B. Feres de; SOARES, C.; CARVALHO, A. C. de. Meta-learning approach to gene expression data classification. **International Journal of Intelligent Computing and Cybernetics**, Emerald Group Publishing Limited, v. 2, n. 2, p. 285–303, 2009. Citation on page 30.

SPILIOTIS, E.; KOULOUMOS, A.; ASSIMAKOPOULOS, V.; MAKRIDAKIS, S. Are forecasting competitions data representative of the reality? **International Journal of Forecasting**, Elsevier, v. 36, n. 1, p. 37–53, 2020. Citation on page 51.

TALAGALA, T. S.; HYNDMAN, R. J.; ATHANASOPOULOS, G. *et al.* Meta-learning how to forecast time series. **Monash Econometrics and Business Statistics Working Papers**, Monash University, Department of Econometrics and Business Statistics, v. 6, n. 18, p. 16, 2018. Citations on pages 25, 26, 49, 53, and 81.

TALAGALA, T. S.; LI, F.; KANG, Y. **Feature-based Forecast-Model Performance Prediction**. [S.l.], 2019. Citations on pages 25 and 49.

TEH, H. Y.; KEVIN, I.; WANG, K.; KEMPA-LIEHR, A. W. Expect the unexpected: unsupervised feature selection for automated sensor anomaly detection. **IEEE Sensors Journal**, IEEE, v. 21, n. 16, p. 18033–18046, 2021. Citation on page 52.

TIPPING, M. E. Sparse bayesian learning and the relevance vector machine. **Journal of machine learning research**, v. 1, n. Jun, p. 211–244, 2001. Citation on page 52.

TSAY, R. S. **Multivariate time series analysis: with R and financial applications**. [S.l.]: John Wiley & Sons, 2013. Citation on page 31.

VAICIUKYNAS, E.; DANENAS, P.; KONTRIMAS, V.; BUTLERIS, R. Two-step meta-learning for time-series forecasting ensemble. **IEEE Access**, IEEE, v. 9, p. 62687–62696, 2021. Citation on page 49.

VAPNIK, V. **The nature of statistical learning theory**. [S.l.]: Springer science & business media, 1999. Citation on page 84.

VILALTA, R.; DRISSI, Y. A perspective view and survey of meta-learning. **Artificial intelligence review**, Springer, v. 18, n. 2, p. 77–95, 2002. Citations on pages 25 and 30.

WANG, X.; HYNDMAN, R. J.; LI, F.; KANG, Y. Forecast combinations: an over 50-year review. **arXiv preprint arXiv:2205.04216**, 2022. Citations on pages 25 and 78.

____. Forecast combinations: an over 50-year review. **arXiv preprint arXiv:2205.04216**, 2022. Citation on page 50.

WIDODO, A.; BUDI, I. Model selection using dimensionality reduction of time series characteristics. In: **International Symposium on Forecasting, Seoul, South Korea**. [S.l.: s.n.], 2013. Citations on pages 25, 30, 35, 48, 49, and 51.

WOLPERT, D. H. The lack of a priori distinctions between learning algorithms. **Neural computation**, MIT Press, v. 8, n. 7, p. 1341–1390, 1996. Citation on page 32.

YOON, J.; JARRETT, D.; SCHAAR, M. Van der. Time-series generative adversarial networks. **Advances in neural information processing systems**, v. 32, 2019. Citations on pages 78 and 81.

ZHOU, Z.-H. **Ensemble methods: foundations and algorithms**. [S.l.]: CRC press, 2012. Citation on page 25.


ICMC
USP
SÃO CARLOS