ULB FACULTÉ DES SCIENCES

UNIVERSITÉ LIBRE DE BRUXELLES

Towards multivariate multi-step-ahead time series forecasting :
A machine learning perspective

Thèse présentée par Jacopo De Stefani
en vue de l'obtention du grade académique de
Docteur en Sciences Informatiques
Année académique 2021-2022

Sous la direction du Professeur Gianluca Bontempi, promoteur
Machine Learning Group - Université Libre de Bruxelles

Jury de thèse :

Prof. Maarten JANSSEN (Université libre de Bruxelles, Président)
Prof. Hugues BERSINI (Université libre de Bruxelles, Secrétaire)
Prof. Luis TORGO (Dalhousie University)
Prof. Souhaib BEN TAIEB (Université de Mons)

MLG MACHINE LEARNING GROUP

JACOPO DE STEFANI

TOWARDS MULTIVARIATE MULTI-STEP-AHEAD TIME SERIES FORECASTING : A MACHINE LEARNING PERSPECTIVE

DECLARATION

This thesis has been written under the supervision of Prof. Gianluca Bontempi (Université Libre de Bruxelles, Belgium). The members of the Jury are:

- Prof. Maarten Jansen (Université Libre de Bruxelles, Belgium)
- Prof. Hugues Bersini (Université Libre de Bruxelles, Belgium)
- Prof. Luis Torgo (Dalhousie University, Canada)
- Prof. Souhaib Ben Taieb (Université de Mons, Belgium)
- Prof. Gianluca Bontempi (Université Libre de Bruxelles, Belgium)

I hereby declare that this thesis contains no material which has been accepted for the award of any other degree or diploma in any university or equivalent institution, and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Bruxelles, 09 December 2021

Jacopo De Stefani

ABSTRACT

Time series forecasting deals with the prediction of future values of time-dependent quantities (e.g. stock price, energy load, city traffic) on the basis of their historical observations. In its simplest form, forecasting concerns a single variable (i. e., univariate problem) and deals with the prediction of a single future value (i. e., one-step-ahead).

Existing studies in the literature focused on extending the univariate time series framework to address multiple future predictions (also called multiple-step-ahead) and multiple variables (multivariate approaches) accounting for their interdependencies. However, most of the approaches deal either with the multiple-step-ahead aspect or the multivariate one, rarely with both. Moreover state-of-the-art multivariate forecasting methods are restricted to low dimensional problems, linear dependencies among the values and short forecasting horizons.

The recent technological advances (notably the Big Data revolution) are instead shifting the focus to problems characterized by a large number of variables, non-linear dependencies and long forecasting horizons. Those forecasting tasks are recently more and more addressed with a representation learning approach, by feeding the data into large scale deep neural networks and letting the model learn the most suitable data representation for the task at hand. This approach, despite its success and effectiveness, often requires considerable computational power, intensive model calibration and lacks interpretability of the learned model.

The motivation of this thesis is that the potential of more interpretable approaches to multi-variate and multi-step-ahead tasks has not been sufficiently explored and that the use of complex neural methods is often neither necessary nor advantageous. In this perspective we explore two multivariate and multiple-step-ahead forecasting strategies based on dimensionality reduction :

- The first strategy, called Dynamic Factor Machine Learning, is a machine learning extension of a famous technique in econometrics: it transforms the original high-dimension multivariate forecasting problem by extracting first a (small) set of latent variables (also called factors) and forecasting them independently in a multi-step-ahead yet univariate manner. Once the multi-step-ahead forecast of factors is computed, the predictions are transformed back to the original space.

- The second strategy, called Selective Multivariate to Univariate Reduction through Feature Engineering and Selection, addresses the dimensionality issue in the original space and deals with the combinatorial explosion of possible spatial and temporal dependencies by feature selection. The resulting strategies combines expert-based feature engineering, effective feature selection strategies (based on filters), and ensembles of simple models in order to develop a set of computationally inexpensive yet effective models.

The thesis is structured as follows. After the introduction, we present a description of the fundamentals of time series analysis and a review of the state-of-the-art in the domain of multivariate, multiple-step-ahead forecasting. Then, we provide a theoretical description of the two original contributions, along with their positioning in the current scientific literature. The final part of thesis is devoted to their empirical assessment on several synthetic and real data benchmarks (notably from the domain of finance, traffic and wind forecasting) and discusses their strengths and weaknesses.

The experimental results show that the proposed strategies are a promising alternative to state-of-the-art models, overcoming their limitations in terms of problem size (in case of statistical models) and interpretability (in case of large scale black-box machine learning models, such as Deep Learning techniques).

Moreover, the findings show a potential for implementation of these strategies on large-scale ($> 10^2$ variables and $> 10^3$ samples) real forecasting tasks, providing competitive results both in terms of computational efficiency and forecasting accuracy with respect to state-of-the-art and deep learning strategies.

ABSTRACT

La prévision des séries temporelles consiste à prédire les valeurs futures de certaines quantités dépendantes du temps (par exemple, le prix des cours boursiers, la production énergétique, le trafic urbain) sur base de l'historique de leurs observations. Dans la forme plus simple du problème, la prévision ne concerne qu'une seule variable (problème univarié) et produit la prédiction d'une seule valeur future (one-step-ahead).

Les études existantes dans la littérature se sont concentrées sur l'extension du cadre des séries temporelles univariées pour traiter des prédictions de plusieurs variables (approches multivariées), à plusieurs étapes dans le futur (également appelées multiple-step-ahead), en tenant compte de leurs interdépendances. Cependant, la plupart des approches traitent soit l'aspect "multiple-step-ahead", soit l'aspect "multivarié", mais rarement les deux. En outre, les méthodes de prévision multivariées les plus récentes se sont limitées à des problèmes de dimensionalité réduite, avec dépendances linéaires entre les valeurs et un court horizon de prévision.

Les récentes avancées technologiques (notamment la révolution du Big Data) déplacent l'attention vers des problèmes caractérisés par un plus grand nombre de variables, des dépendances non linéaires et des longs horizons de prédiction. Ces tâches de prédiction sont de plus en plus abordées avec une approche d'apprentissage de représentation, en alimentant les données dans des réseaux neuronaux profonds à grande échelle et en laissant le modèle apprendre la représentation des données la plus appropriée pour la tâche à accomplir. Cette approche, malgré son succès et son efficacité, nécessite souvent une puissance de calcul considérable, une calibration intensive du modèle et un manque d'interprétabilité du modèle appris.

La motivation de cette thèse est que le potentiel d'approches plus interprétables pour les tâches multi-variables et multi-étapes n'a pas été suffisamment exploré et que l'utilisation de méthodes neuronales complexes n'est souvent ni nécessaire ni avantageuse. Dans cette perspective, nous explorons deux stratégies de prédiction à plusieurs variables et à plusieurs étapes basées sur la réduction de la dimensionnalité :

- La première stratégie, appelée Dynamic Factor Machine Learning, est une extension de l'apprentissage automatique d'une technique célèbre en économétrie : elle transforme le problème original de prévision multivariée à haute dimension en extrayant d'abord un (petit) ensemble de variables latentes (également appelées facteurs) et en les prévoyant indépendamment de manière univariée mais à plusieurs étapes dans le futur. Une fois que la prévision multi-étapes des facteurs est calculée, les prédictions sont retransformées dans l'espace original.

- La deuxième stratégie, appelée Selective Multivariate to Univariate Reduction through Feature Engineering and Selection, aborde le problème de la dimensionnalité dans l'espace original et traite l'explosion combinatoire des dépendances spatiales et temporelles possibles par la sélection de variables. Les stratégies qui en résultent combinent un processus de feature engineering basée sur la connaissance des experts dans le domaine, des stratégies efficaces de sélection des caractéristiques (basées sur des filtres) et des ensembles de modèles simples afin de développer un ensemble de modèles efficaces mais peu coûteux en termes de calcul.

La thèse est structurée comme suit. Après l'introduction, nous présentons une description des principes fondamentaux de l'analyse des séries temporelles et une revue de l'état de l'art dans le domaine de la prévision multivariée à plusieurs étapes dans le futur. Ensuite,

nous fournissons une description théorique des deux contributions originales, ainsi que leur positionnement dans la littérature scientifique actuelle. La dernière partie de la thèse est consacrée à leur évaluation empirique sur plusieurs jeux de données synthétiques et réelles (notamment dans le domaine de la finance, du trafic et de la prédiction de la production d'énergie éolienne) et discute de leurs forces et faiblesses.

Les résultats expérimentaux montrent que les stratégies proposées constituent une alternative prometteuse aux modèles de l'état de l'art, en surmontant leurs limites en termes de taille du problème (dans le cas des modèles statistiques) et d'interprétabilité (dans le cas des modèles d'apprentissage automatique de type boîte noire à grande échelle, tels que les techniques d'apprentissage profond).

En outre, les résultats montrent un potentiel pour la mise en œuvre de ces stratégies sur des tâches de prédiction réelles à très grande échelle ($> 10^2$ variables et $> 10^3$ échantillons), fournissant des résultats compétitifs à la fois en termes d'efficacité de calcul et de précision de la prévision par rapport aux stratégies de pointe et d'apprentissage profond.

ACKNOWLEDGMENTS

The journey towards a Ph.D. is without any doubt a challenging experience but at the same time a once-in-a-lifetime opportunity for both personal and professional growth. As in many journeys, the most rewarding elements are the journey itself, rather than the destination and the people met along the way. I am grateful for all the opportunities I had during these six years, as well as the people that accompanied me during this journey, whose presence had inspired me, challenged me and overall helped me improve both as a person and as a researcher. Some of these people deserve a special mention for their significant contribution my Ph.D.

First and foremost, a heartful thank to Gianluca Bontempi for his continuous supervision and patience throughout all the journey. Even when my research was slowing down due to teaching duties, Gianluca's trust and support helped me to keep focused. His in-depth statistical knowledge, combined with his pragmatic attitude always gave me new and interesting research directions as well as taught me a structured approach to problem solving.

For the first part of my journey, during the collaboration with Worldline (which contributed to the funding of my research through the ULB-Worldline Collaboration Agreement 2015-2018), I would like to thank Dr. Fréderic Oblé for securing the funding and for the relevant feedback, Dr. Dalila Hattab, for the insightful case studies and the knowledge transfer concerning the financial domain, as well as Dr. Olivier Caelen, whose extensive machine learning knowledge, combined with his domain expertise has greatly facilitated the development of relevant projects for both the academic and industrial side.

For the second part of my journey, Dr. Fabrizio De Caro deserves a special mention. A friend first, but at the same time a skilled and tireless researcher, whose expertise in the power system domain, combined with its eagerness to learn, allowed for a fruitful collaboration, which I am looking forward to continuing in the future.

In addition, I would also thank the different mentors I had in ULB for both teaching and research. For the teaching activities, I would like to thank prof. Gwenael Joret, for his continous support and attention to the well-being of the assistants, prof. Frederic Pluquet, for its introduction to Agile methodologies, which definitely improved my coding quality as well as my time management skills and prof. Yves Roggeman, for his informative discussions at the "Thursday Seminars". From the research side, a big thank to dr. Yann-Aël Le Borgne, whose extensive knowledge in the computing field, combined with great communication capabilities always helped to find solutions to technical and research problems. Many thanks to all the colleagues at MLG-ULB that helped proofread this manuscript and provided valuable feedback to improve its quality (in alphabetical order): Charlotte, Elias, Gian Marco, Robin, Theo. I owe you one! My sincere thanks also go to the PhD jury members (in alphabetic order: Prof. Souhaib Ben Taieb, Prof. Hugues Bersini, Prof. Maarten Jansen and Prof. Luis Torgo.) for the time spent in the revision of the thesis and for their insightful comments during the private defense.

Un pensiero speciale ai miei genitori: Corinna e Patrizio, senza i quali non avrei potuto raggiungere questo traguardo. Grazie per il vostro supporto, in presenza e virtuale! Senza la vostra guida (nonostante la distanza), i vostri insegnamenti e la vostra fiducia continua non avrei potuto raggiungere questo traguardo.

A special mention also for my friends Andi, Eleonora, Thomas & Francesca, Philip, Bob & Friends for their cheerful attitude and the highly-needed relax moments throughout all the thesis, in the form of nice meals, football matches and gaming sessions!

Enfin, mes remerciements à ma chère Charlotte, mon 2P (qui comme on sait bien gagne toujours!), pour sa présence constante, pour me supporter (dans tous les sens du terme) et pour nos discussions toujours enrichissantes que ça soit du point de vue personnel ou scientifique.

CONTENTS

ACRONYMS

ANN Artifical Neural Network

AR Auto-Regressive

ARIMA Auto-Regressive Integrated Moving Average

CNN Convolutional Neural Network

DAF-E Dynamic Adaptive Feature-based Ensemble

DAFT-E Dynamic Adaptive Feature-based Temporal Ensemble

DFM Dynamic Factor Model

DFML Dynamic Factor Machine Learner

DGP Data Generating Process

DL Deep Learning

ES Exponential Smoothing

GBM Gradient Boosting Machine

GRU Gated Recurrent Unit

IoT Internet of Things

k-NN k Nearest Neighbors

LL Lazy Learning

LSTM Long Short Term Memory

MISO Multiple Input Single Output

MIMO Multiple Input Single Output

ML Machine Learning

mRMR minimum Redundancy Maximum Relevance

MSE Mean Squared Error

NAR Non-linear Auto-Regressive

NVAR Non-linear Vector Auto-Regressive

NNMSE Naive-Normalized Mean Squared Error

OHLC Opening High Low Closing (Price Data)

PCA Principal Component Analysis

RF Random Forest

RNN Recurrent Neural Network

SCADA Supervisory Control And Data Acquisition

SISO Single Input Single Output

SMURF-ES Selective Multivariate to Univariate Reduction through Feature Engineering and Selection

SVM Support Vector Machine

SVR Support Vector Regression

TSO Transmission System Operator

VAR Vector Auto-Regressive

WPF Wind Power Forecasting

NOTATION TABLE

General

$a, b, c, \cdots$ Scalar variables

$\mathbf{a}, \mathbf{b}, \mathbf{c}, \cdots$ 1-dimensional vector

$\mathbf{A}, \mathbf{B}, \mathbf{C}, \cdots$ 2-dimensional matrix

$\mathbf{A}_{(x \times y)}$ 2-dimensional matrix with dimensions $(x \times y)$

$\mathbb{R}$ Real numbers

Time series forecasting

y_t Value of a univariate time series y at time t

d Delay

m Autoregressive lag (also Embedding order, Model order)

H Forecasting horizon

n Number of time series, dimensionality of the multivariate vector

N Number of time series samples/observations

$\mathbf{Y}$ Multivariate time series $\mathbf{Y}$ $(N \times n)$

$\hat{y}_t$ Forecast of a univariate time series y at time t

$\mathbf{Y}_t$ Value of a multivariate time series $\mathbf{Y}$ at time t $(1 \times n)$

$\mathbf{Y}_{[t,t+h-1]}$ Value of a multivariate time series $\mathbf{Y}$ from time t until time $t+h-1$ $(h \times n$ matrix)

$\mathbf{Y}^i_:, Y[i]$ Value of the univariate time series $\mathbf{Y^i}$ across N timesteps $(N \times 1$ matrix)

$\mathbf{Y}^{[i,i+k-1]}_:$ Value of a subset of k multivariate time series $\mathbf{Y}$ across N timesteps $(N \times k$ matrix)

Y^i_t Value of the univariate time series $\mathbf{Y^i}$ at time t (1×1)

$\hat{\mathbf{Y}}_t$ Forecast of a multivariate time series $\mathbf{Y}$ at time t $(1 \times n)$

Volatility

$P^{(o)}_t$ Opening price of trading day t

$P^{(h)}_t$ Maximum price of trading day t

$P^{(l)}_t$ Minimum price of trading day t

$P^{(c)}_t$ Closing price of trading day t

R_t Return of trading day t

r_t Continuously compound return of trading day t

h_t	Conditional variance of the return distribution at time t
$\hat{\sigma}^{(SD)}$	Volatility proxy based on standard deviation
$\hat{\sigma}_i^2$	i^{th} volatility proxy based on coarse grained volatility data
n	Time frame (in days) for volatility computation

Part I

OVERVIEW

"If I have seen further it is by standing on the shoulders of Giants."

– Isaac Newton

1 INTRODUCTION

1.1 TIME SERIES FORECASTING AND MACHINE LEARNING

Forecasting, in its simplest form, deals with the prediction of a given quantity of interest in the future, given its available historical data. Forecasts are employed in several scientific and applied domains, ranging from finance, where they are used to support investors in selecting the most profitable (or less risky) investment, to the energy sector, in order to support transmission operators to control the contribution of renewable energy production (Figure 1.1). These are only a few of the numerous examples that highlight the importance of forecasting as a tool to support decision-making and perform effective planning.

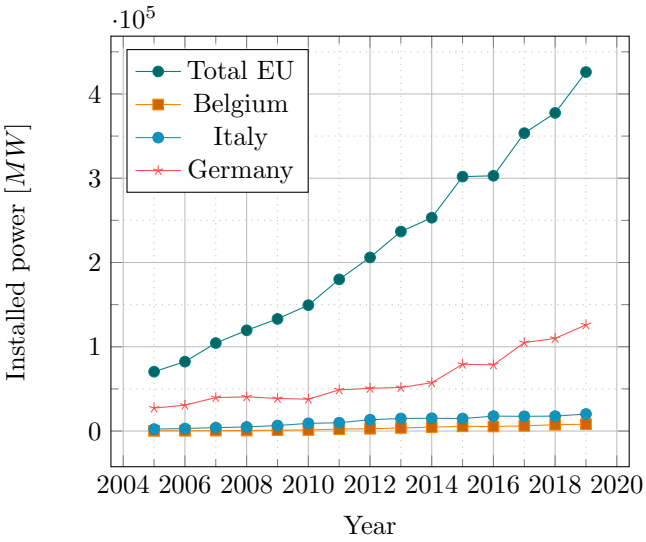


Figure 1.1: Evolution of the cumulative power generated by wind farms in Europe in *MW* - Source: Eurostat except 2017 (EurObserv'ER)

According to the problem at hand, one could be interested in forecasting the immediate next value in the future (one-step-ahead forecasting) or be concerned with the estimation of a sequence of future values (multi-step-ahead forecasting). In addition, forecasting can focus on a single quantity (univariate forecasting), as well as several quantities at once (multivariate forecasting), in order to exploit their cross-dependencies.

This thesis will address the problem of multivariate forecasting, with the increased difficulty of focusing on multiple-step-ahead predictions. The problem of multivariate forecasting has been widely studied in econometrics (Tsay, 2000) since the 1950s, but researchers mostly focused on linear approaches to problems with a reduced number of variables and short forecasting horizons (rarely bigger than one). Example of such approaches are the Vector Auto-Regressive (VAR) model (Lütkepohl, 2005), the multivariate extension of the well-known univariate Auto-Regressive Integrated Moving Average (ARIMA) model (Box et al., 2015) and Vector Error Correction models (Engle and Granger, 1987). Larger dimensional settings have subsequently been addressed by Dynamic Factor Model (DFM) models (Escribano, Peña, and Ruiz, 2021), where a dimensionality reduction process is used to cope with the increased dimensionality before performing a linear forecasting approach.

Nowadays, technological advancements such as Internet of Things (IoT) and the Big Data revolution are increasing data availability (Figure 1.2) and changing the nature of the multivariate forecasting problems by introducing large dimensional streams of nonlinear time series, potentially with strong spatio-temporal dependencies, and by requiring longer prediction horizons. These factors are becoming increasingly important in a growing number of scientific and applied domains, going from environmental science, meteorology, industrial processes to electric grids (Galicia et al., 2017; Perez-Chacon et al., 2016) and finance.

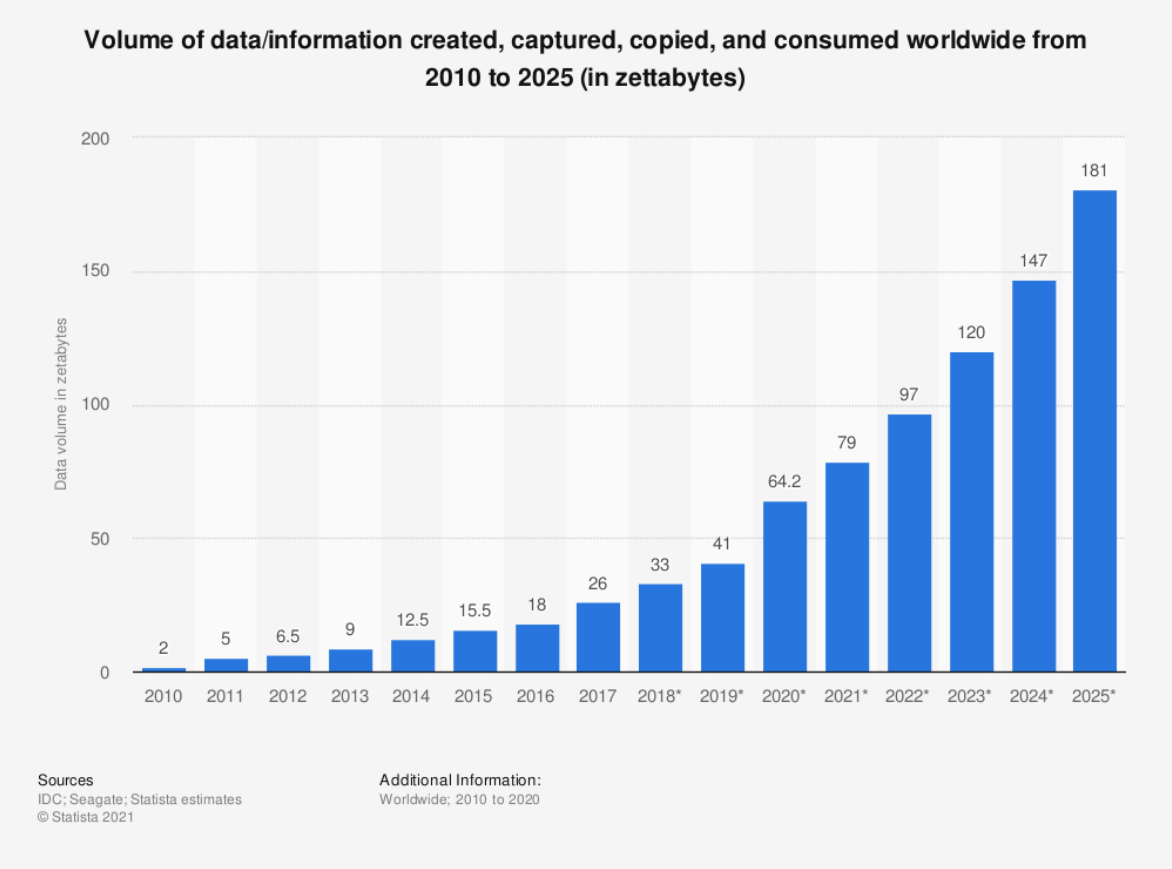


Figure 1.2: Evolution of the worldwide data production across the years since 2010, with projections up to 2025.

As linear econometrics models began to show their limitation in scaling up to larger dimensionality and longer horizons, machine learning based techniques (Friedman, Hastie, Tibshirani, et al., 2001) have started to be considered as alternative solutions to the problem. Machine learning techniques, also called data-driven (Januschowski et al., 2020), are a category of nonlinear and nonparametric models (i. e., with reduced assumptions on the statistical properties of the data generating process), which estimate the stochastic dependency between historical and to-be-predicted data based only on the available data samples. Their black-box approach to forecasting, together with a widespread availability of model implementations (e. g., `Scikit-learn` by Pedregosa et al., 2011, `caret` by Kuhn, 2008 for general-purpose machine learning, `Keras` by Chollet et al., 2015, `PyTorch` by Paszke et al., 2019 for Deep Learning (DL), among others) simplified the adoption of machine learning models by an increasing number of practitioners.

At the same time, the rise of forecasting competitions (e. g., M4, M5 and the upcoming M6) and the corresponding hosting platforms (e. g., Kaggle, DrivenData, Zindi, among others), allowed the practitioners to easily access real datasets and to benchmark their models against each other, further improving the performances of machine learning techniques. As shown by the results of the M4 (Makridakis, Spiliotis, and Assimakopoulos, 2020b) and

M5 (Makridakis, Spiliotis, and Assimakopoulos, 2020a) forecasting competitions, machine learning techniques are becoming more accurate than statistical ones over a variety of multivariate forecasting tasks.

However, the two competitions gave different indications about the choice of the machine learning technique: the winning method of the M4 competition (Smyl, 2020) relies on a deep learning approach, whereas the top-ranked method in the M5 competitions employs a forecast combination of gradient boosting techniques. In particular, deep learning models, a sub-field of machine learning concerned with the development of forecasting models inspired by the human neural system (i. e., deep, interconnected networks of simple processing units), has become more and more popular (Figure 1.3) for the forecasting of multivariate time series (Hewamalage, Bergmeir, and Bandara, 2021; Torres et al., 2021), given its capability to learn from a large amount of data, without any prior assumptions on the corresponding data generating process, and easily re-use the model to produce multiple-step-ahead forecasts, usually with good forecasting accuracy. Nevertheless, this flexibility comes at the price of a computationally intensive training process, a need for extensive parameter tuning as well as a lack of interpretability of the learned model. Even though interpretability, computational cost and ease-of-use are particularly relevant aspects for practitioners of time series forecasting, especially in an industrial setting, those aspects are often neglected in the scientific literature.

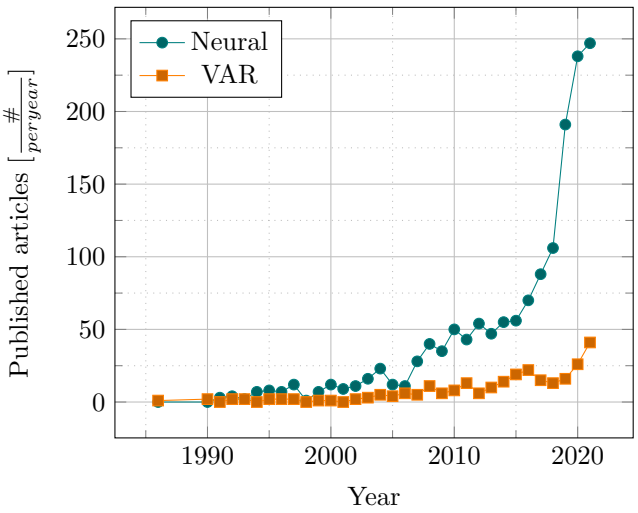


Figure 1.3: Number of published articles concerning multivariate forecasting with different forecasting techniques - Source: Scopus Queries `(TITLE-ABS-KEY(multivariate forecasting X))` with `X` ∈ {Neural,VAR}

Given the state-of-the art of multivariate, multi-step-ahead forecasting, we decided to orient our research towards machine learning approaches, with a particular attention to the aforementioned aspects of interest for forecasting practitioners.

1.2 MOTIVATION AND AIMS

This thesis will address the problem of multivariate and multiple-step-ahead forecasting, currently regarded as the hardest problem in the field, with a particular focus on machine learning solutions. In fact, if we model the prediction task as an autoregressive process (i. e., in which the future can be expressed as a function of the past), such task presents several formidable challenges for any learning machine, notably the large dimensionality of the input and the output data, cross-sectional and temporal dependencies (inducing both non-linear multivariate dependencies in the inputs and a non-linear structure in the outputs) and last but not least the risk of error propagation across several forecasting horizons.

For these reasons, even though several real-life forecasting problems (e. g., wind power, mobility, and stock market forecasting) are inherently multivariate, they are often approached as a set of independent univariate problems. This approach allows to tackle an easier problem at the cost of ignoring the cross-dependencies among the series and potentially losing informative content for the prediction. When we consider multivariate approaches, the majority of proposed forecasting techniques belongs either to the linear family or to the DL-based techniques.

On one hand, linear statistical approaches (such as the VAR family - Section 2.3.4.3) are characterized by an easily interpretable model (due to their linear structure). Their structure allows for closed-form solutions and detailed modeling of the interdependencies, which limits their ability to scale up when the dimensionality of the problem increases and is often limited to short forecasting horizons.

On the other hand, large scale DL models (such as Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) deep networks - Section 2.3.5.1) allow to easily scale up the model when the size of the problem increases and longer forecasting horizons are required, but lack in interpretability. Moreover, their flexibility and the popularity of ready-to-use frameworks led to a pervasive adoption in the field of forecasting. Given their success on several tasks, practitioners tend to assume an a priori superiority of the modeling capabilities of the DL approaches. This assumption, in combination with the ease-of-use and black-box approach to modeling, often results in employing this family of techniques without a critical assessment of their capabilities. Limited attention is given to understanding the actual causes of their good forecasting performance and new models are often created by recombining or extending existing techniques, resulting in a continuous increase in model complexity (Figure 1.4). Last but not least, their assessment is often lacking a comparison against more conventional strategies.

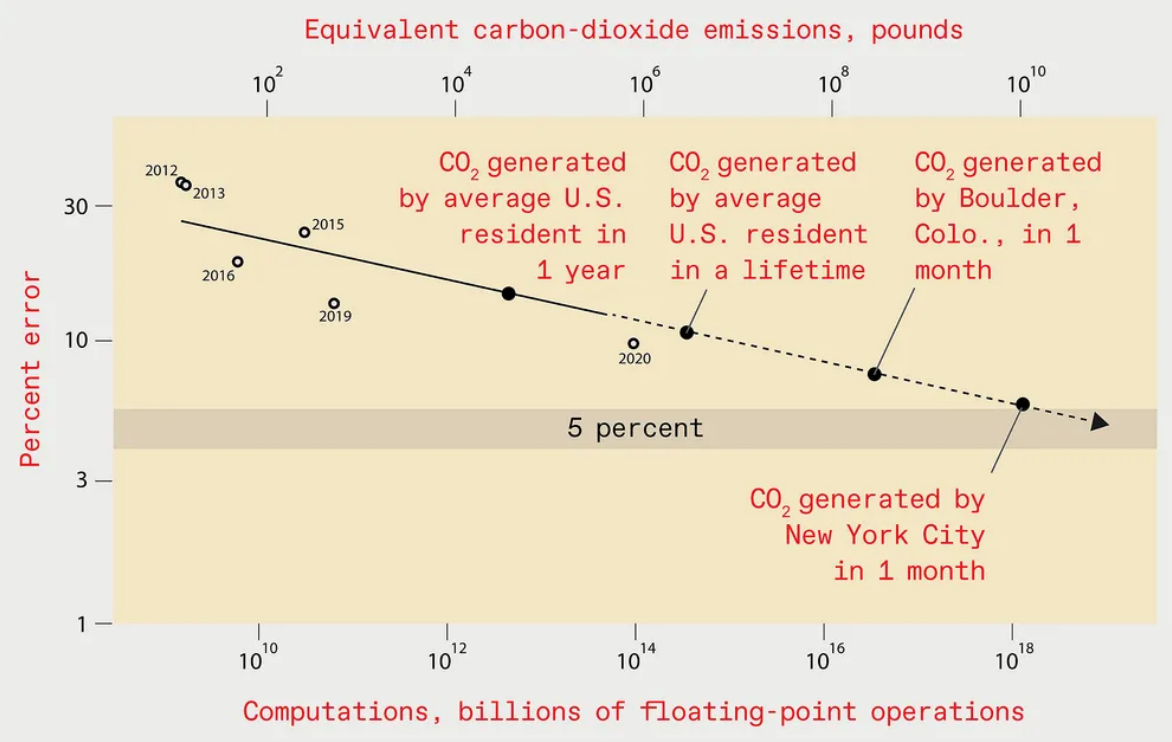


Figure 1.4: Forecasting error of Deep Learning technique as a function of the number of floating point operations and corresponding CO_2 emissions. The white dots represent model sizes and performances of state-of-the-art models, with the corresponding years, while the black dots indicates potential future scenarios. - Source: (*Deep Learning's Diminishing Returns* 2021)

The motivation for this thesis comes from the idea of building a best-of-both-worlds solution, combining scalability and interpretability, and exploring the potential of conventional techniques in the multivariate multiple-step-ahead forecasting. In order to do so, we propose two modular forecasting strategies based on machine learning procedures, tailored for multiple-step-ahead forecasting. To the best of our knowledge, no existing work addresses both the multivariate and the multiple-step-ahead issues. Most of the published works focus on a single subproblem: for instance: (Kirchgassner and Wolters, 2007) studies the problem of one-step-ahead prediction of linear multivariate time series, (Ben Taieb et al., 2012; Bontempi and Ben Taieb, 2011) address the multiple-step-ahead forecasting of a univariate time series, while recent textbooks (Tsay, 2014) consider only multivariate, linear one-step-ahead approaches.

Our first strategy focus on large dimensionality problems ($n > 10^2$ variables and $N > 10^3$ samples), where a reduction of the dimensionality is required in order to reduce noise in the data, as well as to improve the interpretability of the considered models. The proposed strategy, named Dynamic Factor Machine Learner (DFML) (Figure 1.5), inspired by the DFM literature (Escribano, Peña, and Ruiz, 2021), explores the predictive capabilities of both linear and non-linear techniques for both dimensionality reduction and forecasting in this large scale setting. DFML still maintains the key assumption of DFM models: the available time series data are multivariate observation of some latent temporal variable (e. g., stock market intrisic behaviour or atmospheric weather), while relaxing some modeling constraints imposed to estimate conventional DFM models. The proposed strategy is supported by an in-depth assessment over several real-world datasets, allowing the definition of guidelines for tuning the most relevant parameters (i. e., the number of dynamic factors, the choice of dimensionality reduction and forecasting technique, and the multi-step-ahead strategy).

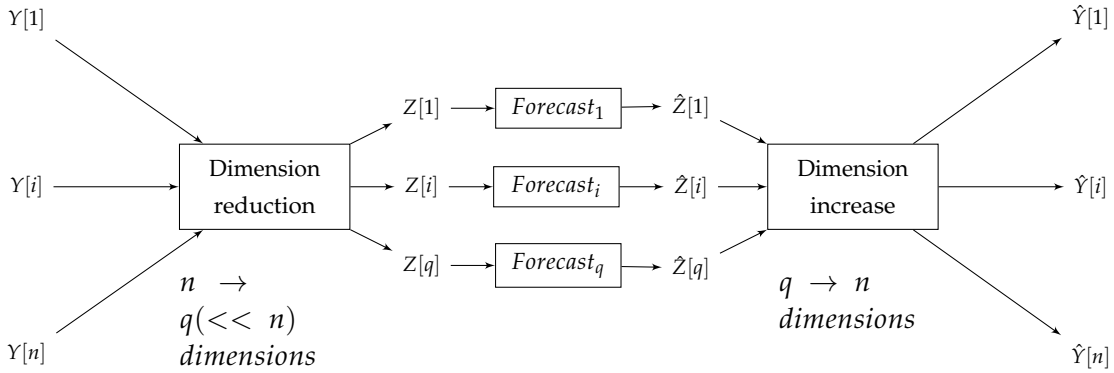


Figure 1.5: Overview of the DFML strategy for multivariate time series forecasting.

The assessment of the DFML strategy on additional datasets highlighted a decrease in performance when the problem size is reduced ($n \sim 30$) and there is little evidence of latent structure in the data, leading us to rethink the approach for smaller problems, while still retaining the interpretability of the previous strategy. This led to the development of the second strategy, named Selective Multivariate to Univariate Reduction through Feature Engineering and Selection (SMURF-ES) (Figure 1.6), based on a process of feature engineering (in order to maximize the informative content for the prediction) and selection (to reduce the computational complexity and only retain the relevant variables), followed by a forecast combination of both statistical and machine learning techniques.

Even though this approach would have been computationally prohibitive on large-scale datasets, here, the reduced problem size allows the execution of the complete pipeline of feature engineering, selection, model fitting (of multiple models) and combination, while still maintaining a reduced complexity. Two implementations of the strategy, employing different feature engineering and forecast combination techniques, are proposed and assessed on

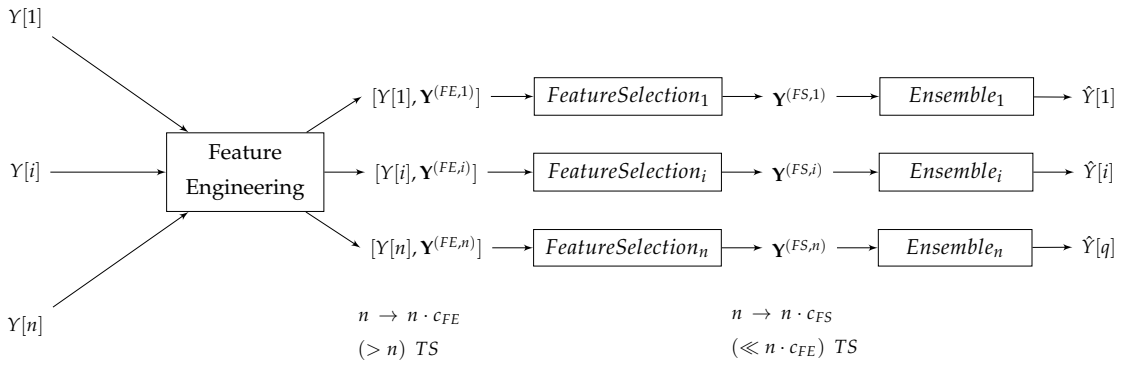


Figure 1.6: Overview of the SMURF-ES strategy for multivariate time series forecasting. $c_{FE} * n$ and $c_{FS} * n$ represent the number of variables after feature engineering and feature selection, respectively.

real-life datasets. An additional contribution is represented by the introduction of a statistical analysis of the spatio-temporal distribution of the forecast error. This procedure aims to improve the interpretability of the results, with a particular focus on the detection of inconsistent forecasting performances.

Overall, the aim of this thesis is twofold. On one hand, we explore a family of solutions oriented to real-life practical applications, based on traditional machine learning procedures, a family of approaches often neglected in the literature, in favor of either statistical techniques, or DL-based solutions. On the other hand, we employ statistical analysis to characterize which components of our forecasting strategies are beneficial for a good predictive performance.

1.3 THESIS CONTRIBUTIONS

To summarize, the main contributions of this manuscript are:

- An overview of the state-of-the-art of the forecasting literature for multivariate problems (Chapter 2 - Sections 2.3.4 and 2.3.5).
- The formalization of the multivariate and multi-step-ahead forecasting problem (Chapter 3).
- The design of two novel forecasting multivariate and multi-step-ahead strategies: DFML and SMURF-ES (Chapter 3), based on traditional machine learning approaches.
- The empirical assessment of the DFML (Chapter 4) and SMURF-ES (Chapter 5) strategies on several real-life challenging forecasting tasks (i. e., wind power forecasting) against state-of-the-art approaches from both the statistical and DL literature.
- The design of model assessment procedures based on the bias-variance principle, to assess the spatial distribution of multivariate forecasting errors (Chapter 5).

1.4 ACTIVITIES SUMMARY

The following publications have been authored/co-authored during the fulfillment of the requirements for the Doctor of Philosophy degree. The publications are presented according to the chapter in which they occur.

1.4.1 *Publications*

- Chapter 4:
  - Gianluca Bontempi, Yann-Aël Le Borgne, and Jacopo De Stefani (2017). "A Dynamic Factor Machine Learning Method for Multi-Variate and Multi-Step-Ahead Forecasting." In: *Proceedings of DSAA 2017, the 4th IEEE International Conference on Data Science and Advanced Analytics 2017*
  - Jacopo De Stefani, Yann-Aël Le Borgne, Olivier Caelen, Dalila Hattab, and Gianluca Bontempi (Aug. 31, 2018). "Batch and Incremental Dynamic Factor Machine Learning for Multivariate and Multi-Step-Ahead Forecasting." In: *International Journal of Data Science and Analytics*. ISSN: 2364-4168. DOI: 10.1007/s41060-018-0150-x. URL: https://doi.org/10.1007/s41060-018-0150-x (visited on 09/12/2018)
  - Jacopo De Stefani and Gianluca Bontempi (2021b). "Factor-Based Framework for Multivariate and Multi-Step-Ahead Forecasting of Large Scale Time Series." In: *Frontiers in Big Data* 4, p. 75. ISSN: 2624-909X. DOI: 10.3389/fdata.2021.690267. URL: https://www.frontiersin.org/article/10.3389/fdata.2021.690267 (visited on 09/10/2021)
- Chapter 5
  - Fabrizio De Caro, Jacopo De Stefani, Gianluca Bontempi, Alfredo Vaccaro, and Domenico Villacci (Oct. 18, 2020). "Robust Assessment of Short-Term Wind Power Forecasting Models on Multiple Time Horizons." In: *Technology and Economics of Smart Grids and Sustainable Energy* 5.1, p. 19. ISSN: 2199-4706. DOI: 10.1007/s40866-020-00090-8. URL: https://doi.org/10.1007/s40866-020-00090-8 (visited on 07/27/2021)
  - Fabrizio De Caro, Jacopo De Stefani, Alfredo Vaccaro, and Gianluca Bontempi (2021). "DAFT-E : Feature-based Multivariate and Multi-step-ahead Wind Power Forecasting." In: *IEEE Transactions on Sustainable Energy*, pp. 1–1. DOI: 10.1109/TSTE.2021.3130949
- Appendix A
  - Jacopo De Stefani, Olivier Caelen, Dalila Hattab, and Gianluca Bontempi (n.d.). "Machine Learning for Multi-Step Ahead Forecasting of Volatility Proxies." In: *ECML PKDD 2017 Workshops - MIning DAta for financial applicationS (MIDAS 2017)*
  - Jacopo De Stefani, Olivier Caelen, Dalila Hattab, Yann-Aël Le Borgne, and Gianluca Bontempi (2019b). "A Multivariate and Multi-Step Ahead Machine Learning Approach to Traditional and Cryptocurrencies Volatility Forecasting." In: *ECML PKDD 2018 Workshops - MIning DAta for financial applicationS (MIDAS 2018)*. Ed. by Carlos Alzate, Anna Monreale, Livio Bioglio, Valerio Bitetta, Ilaria Bordino, Guido Caldarelli, Andrea Ferretti, Riccardo Guidotti, Francesco Gullo, Stefano Pascolutti, Ruggero G. Pensa, Celine Robardet, and Tiziano Squartini. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 7–22. ISBN: 978-3-030-13463-1. DOI: 10.1007/978-3-030-13463-1_1

In addition, the following awards have been granted to the aforementioned publications:

- **MIDAS 2017 - Best Paper Award:** Jacopo De Stefani, Olivier Caelen, Dalila Hattab, and Gianluca Bontempi (n.d.). "Machine Learning for Multi-Step Ahead Forecasting of Volatility Proxies." In: *ECML PKDD 2017 Workshops - MIning DAta for financial applicationS (MIDAS 2017)*

- **DSAA 2017 - Honorable Mention Research Paper Award:** Gianluca Bontempi, Yann-Aël Le Borgne, and Jacopo De Stefani (2017). "A Dynamic Factor Machine Learning Method for Multi-Variate and Multi-Step-Ahead Forecasting." In: *Proceedings of DSAA 2017, the 4th IEEE International Conference on Data Science and Advanced Analytics 2017*

Moreover, an empirical evaluation of the performance of automatic machine learning techniques versus naive techniques for time series forecasting (albeit on univariate series), has been performed in:

- Gian Marco Paldino, Jacopo De Stefani, Fabrizio De Caro, and Gianluca Bontempi (2021). "Does AutoML Outperform Naive Forecasting?" In: *Engineering Proceedings* 5.1 (1), p. 36. DOI: 10.3390/engproc2021005036. URL: https://www.mdpi.com/2673-4591/5/1/36 (visited on 07/27/2021)

Last but not least, one European international patent has been granted on the topics covered by the dissertation:

Jacopo De Stefani, Gianluca Bontempi, Olivier Caelen, and Dalila Hattab (Jan. 3, 2019a). "System and Method for Managing Risks in a Process." Pat. WO2019002582 (A1). Worldline. **Classifications: IPC:** G06Q10/04; G06Q50/04, **CPC:** G06Q10/04 (EP); G06Q50/04 (EP); Y02P90/30 (EP). URL: https://worldwide.espacenet.com/publicationDetails/biblio?FT=D&date=20190103&DB=&locale=en_EP&CC=WO&NR=2019002582A1&KC=A1&ND=5 (visited on 07/27/2021)

1.4.2 *Presentations*

The work included in this thesis has been presented at the following international conferences/workshop:

- *Benelearn 2017*, Eindhoven (the Netherlands), 9-10 June 2017
- *MIDAS @ ECML 2017, The 2nd Workshop on MIning DAta for financial applicationS at the European Conference of Machine Learning*, Skopje, Macedonia, 18-22 September 2017
- *DSAA 2017: International Conference of Data Science and Advanced Analytics*, Tokyo, Japan, 19-21 October 2017
- *MIDAS @ ECML 2018, The 3rd Workshop on MIning DAta for financial applicationS at the European Conference of Machine Learning*, Dublin, Ireland, 10-14 September, 2018
- *ISF 2020, 40th International Symposium on Forecasting*, Virtual, 25-30 October, 2020

Moreover, the topics covered in this thesis have been presented as invited speaker at the following industrial/academic institutes:

- Workshop Volatility forecasting: From standard to ML-based multistep ahead models (Equens-Worldline R&D) (08/01/2018 - 31/01/2018)
- Machine Learning for Multi-step Ahead Forecasting of Volatility Proxies, INESC-ID Research institute, Lisbon, Portugal 02/11/2017
- Methods for multivariate and multi-step-ahead time series forecasting, Scisports, Amersfoort, Netherlands, 01/11/2018

1.4.3 *Research activities*

Part of the work included in this thesis has been developed during an industrial partnership with *Atos-Worldline*, and funded through the *ULB-Worldline* agreement.

In addition I attended the following summer school:

- DeepLearn - International summer school on Deep Learning 2018 – 23-27/07/2018

1.4.4 *Software development*

The work presented in this thesis has led to the development of multiple R packages, publicly available on Github:

- `UEMTS`: Univariate Error Measures for Time Series forecasting - https://www.github.com/jdestefani/UEMTS, implementing the forecasting error measures analyzed in (Hyndman and Koehler, 2006).
- `MEMTS`: Multivariate Error Measures for Time Series forecasting - https://www.github.com/jdestefani/MEMTS, proving the multivariate extensions of the forecasting error measures analyzed in (Hyndman and Koehler, 2006), employed for the experimental assessments in Chapter 4 and 5.
- `MM4Benchmark`: Multivariate extension of the Benchmarks used for the M4 competition - https://github.com/jdestefani/MM4Benchmark, implementing the most common multivariate benchmark, employed in the experimental assessments in Chapter 4 and 5.
- `ExtendedDFML`: Extended DFML framework - https://www.github.com/jdestefani/ExtendedDFML, providing an implementation of the DFML strategy with both linear/non-linear factor estimation and model-driven/data-driven factor forecasting techniques.

2 BACKGROUND

The research questions addressed in this thesis are related the to problem of time series forecasting, tackled both with a traditional approach, rooted in statistical analysis and signal processing (also called model-based), as well as a data-driven approaches from the machine learning domain. Both machine learning and time series forecasting are large fields of research in continuous evolution, hence, for the sake of clarity, we will focus only on the most relevant concepts and techniques that we will employ in our proposed solutions, presented in the following chapters.

More precisely, we start by introducing the concept of statistical learning and the corresponding learning pipeline, together with a set of general techniques commonly employed in both the machine learning and time series analysis fields. We continue by delving deep into the time series forecasting literature, focusing on the adaptations that are required to tackle the forecasting problem as a learning problem. For a broader introduction to time series forecasting, we suggests the following references: (Hyndman and Athanasopoulos, 2018), (Petropoulos et al., 2021), (Brockwell et al., 2016), (Terasvirta, Tjostheim, and Granger, 2010) for a more general introduction, and (Lütkepohl, 2005), (Tsay, 2014) for a focus on multivariate approaches. For machine learning, key references include (Hastie, Tibshirani, and Friedman, 2009), (James et al., 2013) and (Bontempi, 2013) for statistical learning theory, and (Aggarwal et al., 2018) for a specific focus on neural-based techniques.

2.1 MACHINE LEARNING PROCEDURE

According to (Mitchell, 1997), each machine learning problem can be precisely defined as the problem of improving some measure of performance P when executing some task T, through some type of training experience E.

A key phase in the machine learning procedure is represented by the problem formulation, in which the practitioner formalizes the task T to be analyzed and the performance measure P to be considered for the problem at hand. With the knowledge of these elements, an experimental design is defined in order to collect suitable data (i. e., the training experience E to be fed to the learning machine). Then, the practitioner proceeds with the data collection step and the eventual preprocessing step, to prepare the available data in a suitable format for the selected learning machine.

Once this preliminary phase is completed, the learning phase starts by defining a model (sometimes also called hypothesis) for the task T at hand, followed by a model identification phase. Generally speaking, this phase involves dividing the available training experience E into two parts: the first part is used for the identification of the parameters of the model (the actual learning), while the second is employed to evaluate the performance of the model according to the selected performance measure P. The learning phase often involves a feedback loop in which the model is adapted/modified if the performance P on the task at hand is not sufficiently good.

The structure of a generic machine learning procedure is summarized in Figure 2.1.

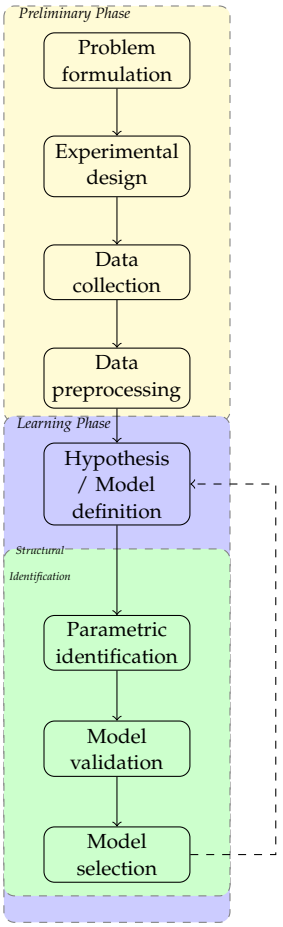


Figure 2.1: Structure of a traditional machine learning procedure according to (Bontempi, 2013). The feedback loop is represented by the dashed line.

2.1.1 *Problem formulation*

In the problem formulation phase, the practitioner selects a phenomenon to be modeled (from a specific application domain) and formulates a hypothesis on the existence of an unknown dependency (i. e., the model) which has to be estimated from experimental data.

Two major problem formulations exist in the domain of machine learning: unsupervised and supervised machine learning. Here, the notion of supervision is related to the presence of a human in the learning loop.

In unsupervised learning, there is no human intervention in the loop, and the objective is to extract patterns of similarities/dissimilarities in the data, whereas supervised learning assumes the presence of humans labeling the available data.

In supervised learning, it is assumed that a black-box generating process (Breiman, 2001b) is producing the data, controlled by some input variables $\mathbf{x}$ and generating an output variable y. In this context, the goal of the human is to provide the labels for the output variable, so that the learning machine can learn the mapping from the input to the output variable (i. e., the labeling criterion). According to the nature of this variable, the problem is defined as a classification problem if the output variable y belongs to a discrete, finite set of classes $\{C_1, \cdots, C_k\}$ and as a regression problem if y is a continuous variable. Other problem formulations exist (e. g., reinforcement learning), but their discussion would be outside the scope of this thesis.

Given this theoretical framework, the appropriate problem definition for the task at hand requires domain-specific knowledge, analytical skills, intuition, and experience from the practitioner. Considering its positioning in the machine learning procedure, a well-

defined problem (and appropriate data collection for the task at hand) is a crucial aspect for the success of a machine learning procedure, in practice often more relevant than the determination of the optimal model for the task at hand.

2.1.2 *Experimental design*

Along with an accurate problem definition, it is of paramount importance to appropriately define an experimental protocol to collect the data. In a data-driven domain such as machine learning, no matter how powerful a learning machine could be, the successive steps in the pipeline would be ineffective if the data are not informative enough. Especially in the case of supervised learning (a form of input/output modeling), it is essential that the learning data is a representative sample of the phenomenon and covers adequately the input space. Experimental design (Fedorov, 2013) is a discipline concerned with determining the optimal data collection strategy in order to maximize the performance of the learning process. It should be noted that, although in theory the problem definition and the experimental design phase should come before the data collecting, in practice, the practitioner often has no control over these steps and needs to deal with the available data, rather than following an optimal design process.

2.1.3 *Data collection*

Once the experimental design is defined, data collection can be performed in different ways. For instance, for some well-defined and well-studied problems, a practitioner can retrieve pre-cleaned, structured datasets (such as educational, academic resources) where the data have already been treated to ensure a certain level of data quality. On the other hand, the practitioner might be required to engineer its own data, for example by retrieving it from publicly accessible sources via automated tools (such as crawlers and scrapers) or by extracting it from private data sources (e. g., databases).

Although at a first glance the process of data collection might seem quite straightforward (especially in presence of well-defined problem definitions and experimental designs), some common problems can be encountered during this step such as missing/inaccurate, imbalanced and biased data.

The problem of missing or inaccurate data is often related to faults in the processes/devices employed in the data collection process, causing data losses, or inaccuracy in the corrected values, requiring a further preprocessing step to be correctly dealt with. On the other hand, even in the absence of missing data, the collected data might be imbalanced (i. e., one or more states of the output variable could be over-/underrepresented with respect to the others) or biased (i. e., including implicit biases over one or more input variables) providing a somewhat distorted representation of the underlying phenomenon. While the problem of imbalanced data has been extensively studied and can be corrected through rebalancing techniques (Tantithamthavorn, Hassan, and Matsumoto, 2018), the problem of detecting biases in data is more complex to tackle and still remains an open research topic (Turner Lee, 2018).

2.1.4 *Data preprocessing*

After the collection, a crucial step to improve data quality is the preprocessing step. In the preprocessing step, the raw collected data is analyzed and transformed in order to improve the learning performance of the model that will be subsequently fitted on the data. For a detailed review concerning the different applicable preprocessing techniques, we refer the

interested reader to (Fan et al., 2021). Here, we will be presenting only the most relevant techniques for our domain of interest (time series forecasting).

2.1.4.1 *Missing value handling*

Many learning procedures are unable to natively handle missing values. To cope with this problem, a practitioner has two main approaches in an operational setting.

If the available data are sufficiently large, missing data can simply be discarded. Conversely, if discarding values is not an option (either due to the size of the available data or to external constraint), the practitioner needs to perform missing value imputation to replace the missing values with meaningful substitutes. Missing value imputation can have varying degrees of complexity, ranging from simple imputation of null values to univariate and multivariate techniques based on the statistical properties of the data.

Note that any imputation technique makes different assumptions about the statistical properties of the process that caused the missing observation, as well as on the distribution of replacement values (e. g., normality). One should carefully consider such assumptions before selecting the most adapted technique for the problem at hand.

2.1.4.2 *Feature Selection*

The majority of traditional supervised learning algorithms have been designed to deal with learning tasks in which the dimension of the input space is small and most of the input variables $\mathbf{x}$ are relevant with respect to the output variable y to model. As a consequence, their learning performance could rapidly degrade when they are employed in problems with few available data and a huge number of input variables. As such, it has become increasingly common to employ a feature selection approach in order to remove irrelevant and unnecessary features.

According to (Bontempi, 2013), the approaches to feature selection can be classified into three main categories:

- Filter methods: Assessing the relevance of features solely from the data, ignoring the effects of the selected features subset on the performance of the learning algorithm.
- Wrapper methods: Assessing subsets of variables according to their usefulness to a given learning technique. Wrapper methods search the most relevant variable subset using the learning algorithm itself as part of the evaluation function.
- Embedded methods: Variable selection is performed as part of the learning procedure and is usually specific to given learning machines (for instance model regularization techniques, e. g., LASSO).

For more information about the empirical performances of the different families of feature selection techniques, we refer the interested reader to (Bolón-Canedo, Sánchez-Maroño, and Alonso-Betanzos, 2013).

Among the proposed feature selection techniques, we focus here on two filter techniques: minimum Redundancy Maximum Relevance (mRMR) and Principal Component Analysis (PCA). In order to better discuss the implications of the latter on temporal dependent data, PCA will be discussed in more detail in Section 2.4.1.

MRMR mRMR (Eq. 2.1) (Jain, Duin, and Jianchang Mao, 2000), is an information-theoretic feature selection technique, extracting a subset of variables expected to be as relevant as possible for the prediction target, while minimizing the redundance within the subset.

To be more precise, mRMR returns a subset of $n_S << n$ relevant features by using a forward procedure which at the k-th step ($k = 1, \ldots, n_S$) selects the least redundant and most informative predictor variable according to the following formula:

$$\underset{\mathbf{X}_\lambda \in X - \Phi_{k-1}}{\arg\max} \left[I(\mathbf{X}_\lambda, \mathbf{y}) - \frac{1}{k-1} \sum_{\mathbf{X}_\psi \in \Phi_{k-1}} I(\mathbf{X}_\psi, \mathbf{X}_\lambda) \right] \tag{2.1}$$

where Φ_{k-1} is the set of $k-1$ previously selected variables and $I(\mathbf{X}_i, \mathbf{y})$ denotes the mutual information between the variables $\mathbf{X}_i$ and $\mathbf{y}$ (De Jay et al., 2013). Note that the mutual information term can be efficiently estimated by $I(x, y) = \frac{1}{2} \ln(1 - \rho(x, y)^2)$ where ρ is the Pearson correlation coefficient under an assumption of normality. A specific advantage of mRMR with respect to compression techniques is that it does not transform the original features, allowing an easier data interpretation. Moreover, the technique has a reduced computational complexity given its simple heuristic to determine feature relevance. This makes the procedure particularly efficient even on tasks with a large number of variables (e. g., bioinformatics - Rego-Fernández, Bolón-Canedo, and Alonso-Betanzos, 2014, Meyer, Lafitte, and Bontempi, 2008, De Jay et al., 2013).

2.1.4.3 *Feature Engineering*

Sometimes, when the informative content of the available data is too limited, a process of feature engineering is performed to construct new features from the available data. Feature engineering creates $n' = n \cdot c_{FE}$ new features through the combination of the existing n variables via both linear and non-linear approaches. Although statistical approaches, based on the creation of new features through the computation of spatio-temporal statistics on the available data (Zheng and Casari, 2018), are generally a good solution, designing efficient feature engineering techniques requires a great amount of domain-specific knowledge, analytical skills and experience.

2.1.4.4 *Data scaling*

In presence of input variables having different orders of magnitude, a rescaling process is required to perform a meaningful learning process.

The rescaling process might aim to maintain the original distribution of the considered variable while aligning its order of magnitude to those of the other considered variables. In this case, a min-max scaling (Eq. 2.2) or an interquantile scaling (Eq. 2.3) might be the best choice.

$$x_{minmax} = \frac{x - min(x)}{max(x) - min(x)} \tag{2.2}$$

$$x_{Q_1 Q_3} = \frac{x - Q_1(x)}{Q_3(x) - Q_1(x)} \tag{2.3}$$

where $min(x)$, $max(x)$, $Q_1(x)$ and $Q_3(x)$ represent the minimum, maximum, 1^{st} and 3^{rd} quartile of the x variable, respectively.

On the other hand, if we can assume that the x variable follows a normal distribution, the z-score rescaling (Eq. 2.4) can be employed in order to ensure that the variable has its mean equal to zero and unit variance.

$$x_z = \frac{x - \mu_x}{\sigma_x} \tag{2.4}$$

where μ_x and σ_x represent the mean and the standard deviation of the x variable, respectively.

2.1.5 *Learning phase*

In this context, the outcome of the preliminary phase is a structured dataset $\mathcal{D}_N$, composed of N pairs $< \mathbf{x}_i, y_i >$ (also named samples) of observations $\mathbf{x}_i$ and the corresponding targets y_i. Figure 2.2 represents a dataset in a tabular form. Note that, without loss of generality, the dataset $\mathcal{D}_N$ can be represented as a matrix $\mathbf{D_N}_{(N\times(n'+1))}$.

Moreover, as part of the learning procedure, the dataset will be split into a part dedicated to the actual learning of the model, named the training set, and a part dedicated to the validation of the model, named the testing or validation set.

	Features/Variables					Target	
	$\mathbf{X}_1$	$\cdots$	$\mathbf{X}_i$	$\cdots$	$\mathbf{X}_{n'}$	$\mathbf{y}$	
Sample	x_{11}	$\cdots$	x_{1i}	$\cdots$	$x_{1n'}$	y_1	Training set $\mathcal{D}_{train}$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
	x_{k1}	$\cdots$	x_{ki}	$\cdots$	$x_{kn'}$	y_k	
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	Testing set $\mathcal{D}_{test}$
	x_{N1}	$\cdots$	x_{Ni}	$\cdots$	$x_{Nn'}$	y_N	

Figure 2.2: Example of generic dataset D_N of N samples, $n' = n \cdot c_{FE}$ variables, resulting of feature engineering of the original n variables, split into a training set $\mathcal{D}_{train}$ of k samples and a testing set $\mathcal{D}_{test}$ of $N - k$ samples.

The key assumption of a supervised learning process is that there exists an unknown stochastic dependency $f : \mathbb{R}^{n'} \mapsto \mathbb{R}$, describing the relation between the input variables $\mathbf{x}$ and the output (target) variable y:

$$y = f(\mathbf{x}) + w \tag{2.5}$$

where w denotes the noise term, which is supposed to have null mean and constant variance, and that should account for all the unmeasured contributions to the variability of y. In addition, we assume that each of the samples composing the observed data D_N are independent and identically distributed, generated from the stochastic process described by f.

As the dependency f between input and output variable is unknown, we will need to find an approximation of this dependency whose behavior is as close as possible to the real dependency f. The approximation is generally referred in the literature as *hypothesis* or *model* $h : \mathbb{R}^{n'} \mapsto \mathbb{R}$, and it is defined in a parametric form $h(\mathbf{x}, \boldsymbol{\theta})$, with a generic parameter vector $\boldsymbol{\theta}$.

Given this definition, finding the best model for a given task entails two sub-problems: the definition of a measure of discrepancy between the model and the real dependency, and the determination of the best model according to this measure.

The measure of discrepancy (also called loss function in the literature) $L : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$ compares the true value y produced by f, with the outcome of the parametric model $h(\mathbf{x}, \boldsymbol{\theta})$. Given the task at hand, we can define a quantity, named the empirical risk R_{emp}, which measures the discrepancy between the considered hypothesis h and true dependency f on the available data $\mathcal{D}_N$ as the average of the loss function across the N samples:

$$R_{\text{emp}}(h, \boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} L\left(y_i, h\left(x_i, \boldsymbol{\theta}\right)\right) \tag{2.6}$$

Note that the empirical risk is a function of both the chosen hypothesis h and the corresponding parameters $\boldsymbol{\theta}$. Hence, determining the best model for a given task requires the solution of two additional sub-problems: selecting the best family of hypotheses Λ^* from different hypotheses class $\Lambda_i \in \{1, \cdots, s\}$ (i. e., the problem of *structural identification*) and selecting the best set of parameters $\boldsymbol{\theta}_N$ for the selected hypothesis class $h(\cdot, \boldsymbol{\theta}) \in \Lambda_i$ (i. e., the problem of *parametric identification*).

The problem of parametric identification is tackled first, via a learning algorithm $\mathcal{L}$ that takes as input a given hypothesis $h(\cdot, \boldsymbol{\theta})$ and the available dataset $\mathcal{D}_N$ to produce the hypothesis with the optimal set of parameters $h(\cdot, \boldsymbol{\theta}_N)$. The optimization of the parameters is done through the Empirical Risk Minimization procedure (Vapnik, 1999), which, given the hypothesis class Λ_i, determines the set of parameters $\boldsymbol{\theta}_N$ that minimizes the empirical risk over the dataset D_N:

$$\boldsymbol{\theta}_N^i = \arg\min_{\boldsymbol{\theta} \in \Lambda_i} R_{\text{emp}}(h, \boldsymbol{\theta}) \tag{2.7}$$

After the parametric identification is performed for all the different hypotheses class Λ_i, the optimal models in each class are compared against each other in terms of empirical risk R_{emp}. The best model is selected as the output of the learning procedure (summarized in Figure 2.3).

2.1.5.1 *Hypothesis definition*

A crucial aspect in the learning process is the choice of an appropriate class of models (also named model structure). A relevant theoretical result in this domain is often referred as *No Free Lunch Theorem* (Wolpert, 2002). The theorem states that all machine learning algorithms are equally effective across all possible prediction problems. In other words, one should not assume an a priori superiority of one technique over the other. With that in mind, a heuristic for the choice of a class of hypotheses for a given problem might be to look at some problem-specific features, for instance: is the phenomenon to model linear or non-linear? Is it required to be able to interpret the learned model or the final objective is simply to get the best possible predictions?

Once the broader class of model is selected, some additional architectural decisions still need to be made (e. g., the size of the input of the model, the number of free parameters, the presence of a technique to limit the complexity of the model). Careful attention should be paid to these design decisions as the chosen model structure directly influences the generalisation capabilities of the model itself.

It should also be noted that the choice of the best model architecture for a given task can be seen as a learning problem itself, yielding to a meta-learning problem (Hospedales et al., 2020; Thrun and Pratt, 1998), an actively studied research subject. Recent developments in the field yielded to the so-called AutoML approaches (Hutter, Kotthoff, and Vanschoren, 2019), where the choice of a class of hypotheses and the corresponding meta-parameters is framed as an optimization problem and automatically solve to determine the best configuration for a given task.

2.1.5.2 *Structural identification*

In the remainder of this thesis, we will focus on supervised learning problems, and more specifically, a regression setting.

In practice, to perform structural identification, we need to define three aspects: a procedure to generate alternative model structures (i. e., model generation), a method for assessing the alternatives (i. e., model validation) and a technique to select the optimal model among the candidates (i. e., model selection).

Concerning model generation, two main approaches can be found in the literature (Maron and Moore, 1997): an exhaustive approach (also called brute force or grid search), which consists of generating all the possible model structures, testing every different combination for the different degrees of freedom in the model architecture, and a local search approach, evaluating only a limited number of configurations through a metric-based research in the model space.

2.1.5.3 *Parametric identification*

Once the model has been generated, the next phase in the structural identification procedure is the determination of the optimal parameters through empirical risk minimization (Equation 2.12). The choice of an optimization algorithm depends on the form of the empirical risk function $R_{\text{emp}}(h, \boldsymbol{\theta})$ and consequently, by the nature of its composing terms: the model $h(\cdot, \boldsymbol{\theta})$ and the loss function $L(h(\mathbf{x}, \boldsymbol{\theta}), y)$. For regression problems, a common choice is a quadratic error function:

$$L(h(\mathbf{x}, \boldsymbol{\theta}), y) = (y - h(\mathbf{x}, \boldsymbol{\theta}))^2 = (y - \hat{y})^2 \tag{2.8}$$

which in turns gives the empirical risk R_{emp} the form of a Mean Squared Error (MSE) function:

$$R_{\text{emp}}(h, \boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} L\left(y_i, h\left(x_i, \boldsymbol{\theta}\right)\right) = \frac{1}{N} \sum_{i=1}^{N} (y_i - h(x_i, \boldsymbol{\theta}))^2 = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{2.9}$$

With this sum of squares format, in the case of a linear model, the optimization problem can be tackled in an analytical, closed form, through OLS estimation. In case of non-linear models, or non-quadratic loss functions, where a closed form is not obtainable, the only approach is to employ iterative search methods, where the optimal values of the parameters are incrementally determined in an iterative process:

$$\boldsymbol{\theta}^{\tau+1} = \boldsymbol{\theta}^{\tau+1} + \Delta\boldsymbol{\theta}^{\tau} \tag{2.10}$$

Examples of iterative approaches are gradient-based methods where the modification to the solution $\Delta\boldsymbol{\theta}^{\tau}$ is derived from the gradient or higher-order derivatives (e. g., gradient descent, Newton method and Levenberg-Marquardt algorithms), or non-gradient techniques, where $\Delta\boldsymbol{\theta}^{\tau}$ is determined via an heuristic approach (e. g., random search, genetic algorithms, local search approaches).

BIAS-VARIANCE TRADEOFF AND REGULARIZATION So far, we only considered a parameter optimization criterion based on empirical risk minimization. However, if the model is miss-specified, or over-specified, empirical risk minimization might force the model to specialize on the specific dataset $\mathcal{D}_N$ it has been training on, reducing its generalization capabilities in a phenomenon called overfitting. The problem of overfitting can also be

explained by the bias-variance tradeoff (Equation 2.11) principle: the empirical risk can be decomposed into three components: the bias, the variance and the intrinsic (irreducible) error (Bontempi, 2013):

$$R_{\text{emp}}(h,\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \underbrace{(\mathbb{E}[\hat{y}] - y)^2}_{\text{Bias}^2} + \underbrace{\mathbb{E}[\hat{y} - \mathbb{E}[\hat{y}]]^2}_{\text{Variance}} + \underbrace{\sigma_e^2}_{\text{Error}} \tag{2.11}$$

The intuition behind this decomposition is that the bias term represent how much the average of the predictions $\hat{y}$ differ from the true values y, while the variance term represents how variable are the predictions across different realizations of the model. An overfitted model minimizes the bias terms, but at the same time increases the variance term, given that their sum should be constant. An increased variance term will then explain the difference in performance between training set and testing set.

In order to contrast this phenomenon, one possible solution is to constrain the model to discard unnecessary parameters and to prefer simpler structures over more complex ones. This approach will allow to reduce the variance term, at the expense of increasing the bias term. This solution, called regularization, is implemented by adding to the empirical risk function an additional term $S_{\text{emp}}(h,\boldsymbol{\theta})$ controlling the complexity of the model, regulated by the hyperparameter [1] λ:

$$\boldsymbol{\theta}_N^i = \arg\min_{\boldsymbol{\theta}\in\Lambda_i} R_{\text{emp}}(h,\boldsymbol{\theta}) + \lambda S_{\text{emp}}(h,\boldsymbol{\theta}) \tag{2.12}$$

Common forms of $S_{\text{emp}}(h,\boldsymbol{\theta})$ involve having a sum of the absolute values of the parameters ($\sum|\boldsymbol{\theta}|$ as in ridge regression, (Hoerl and Kennard, 1970)) or a sum of the squared values $\sum\boldsymbol{\theta}^2$ (as in LASSO, (Tibshirani, 1996)).

2.1.5.4 *Model validation*

After having performed parametric identification, the learned models need to be assessed in order to determine the best one for the task at hand. Several criteria can be employed to determine the goodness of a model for a practical application, for instance: simplicity of use, interpretability, computational efficiency or possibility to integrate expert knowledge. Although increasing attention is being given to alternative criteria, such as interpretability (Molnar, n.d.), in practice, the main criterion employed to validate models is predictive accuracy.

As previously discussed, the process of machine learning basically consists in learning from the available data $\mathcal{D}_N$ the best possible approximation of the unknown data generating process f, from the considered model classes Λ_i for parametric identification. However, employing the same data $\mathcal{D}_N$ to validate the model might yield overly optimistic conclusions over the model performance with respect to newly available data.

For this reason, a holdout approach is performed, in which the available data $\mathcal{D}_N$ is split into two mutually exclusive sets: $\mathcal{D}_{train}$, only employed for model identification (or training) and $\mathcal{D}_{test}$, only employed for model validation (also called testing) (Figure 2.2). Moreover, for a more robust statistical assessment of the model performance, this procedure is repeated for an arbitrary amount of times k, in a process named k-fold-cross-validation. In a k-fold-cross-validation, $\mathcal{D}_N$ is divided in k partitions. For each of the k folds, a different partition (i. e., $\frac{N}{k}$ samples) is used as test set, whereas the remaining $k-1$ partitions are used for model identification. ($N - \frac{N}{k}$ samples). In each fold, the empirical risk over the

1 In machine learning, a hyperparameter is a parameter controlling the learning process, to distinguish them from the values of other parameters (specific to the learning model) which are obtained through training.

testing set (also called out-of-sample-error) is computed and the cross-validated error is obtained by taking the average out-of-sample-error across k folds. To further reduce bias in the data, another approach used in practice is nested cross-validation (Cawley, 2012), in which the data are first split into two parts: one for model identification and one to test model generalization. The first part is further split into a part used for model training, and another one for validation. The training and validations sets are employed to determine the best model, which is then tested on the remaining set (testing set), initially hold out from the learning process.

2.1.5.5 *Model selection*

The last step of the machine learning procedure consists in selecting the optimal model. As previously discussed, several criteria can be considered in the optimization process of the model, but in practice, the optimal model is the one minimizing the out-of-sample error (i. e., the one displaying the best generalization capabilities). This process is also called *winner-takes-it-all*, referring to the fact that all the other sub-optimal models are discarded in favor of the optimal one.

However, the discarded models can still have some relevance for the learning task at hand, for example by making different assumptions on the relationship between input and output data (e. g., linear vs non-linear). In addition, in statistical terms, combining the different discarded models is equivalent to perform a combination of statistical estimators, which has been proven to improve the estimation capabilities with respect to the individual estimators (Graybill and Deal, 1959). For these reasons, the *combination of estimators* approach (commonly referred as *ensemble* of estimators) have become increasingly studied in the scientific literature and applied in practice.

Given their practical relevance and their central role with respect to the strategies proposed within this thesis, a detailed discussion of the ensemble approach will be performed in Section 2.5.

Require: Model classes Λ_i, $i \in \{1, \cdots, S\}$, Learning algorithms $\mathcal{L}_i$
Ensure: Optimal model $h^*(\cdot, \boldsymbol{\theta}_N)$

1: **for** $i \in \{1, \ldots, S\}$ **do** ▷ Structural identification
2: **for** $h_j \in \Lambda_i$ **do** ▷ Parametric identification
3: $\boldsymbol{\theta}_N^{i,j} \leftarrow \mathcal{L}_i(h_j, \mathcal{D}_N)$ ▷ Identification of optimal $\boldsymbol{\theta}_N^{i,j}$ via learning algorithm $\mathcal{L}_i$ for hypothesis h_j
4: **end for**
5: $h_N^i \leftarrow \arg\min_{h_j \in \Lambda_i} R_{\text{emp}}(h_j, \boldsymbol{\theta}_N^{i,j})$ ▷ Optimal model h_N^i in class Λ_i
6: **end for**
7: $h_N^* \leftarrow \arg\min_{i \in \{1,\ldots,S\}} R_{\text{emp}}(h_N^i, \boldsymbol{\theta}_N^{i,j}) + \lambda S_{\text{emp}}(h_N^i, \boldsymbol{\theta}_N^{i,j})$ ▷ Optimal model h_N^* across classes

Figure 2.3: Pseudo-code summarizing the model selection procedure, highlighting the nested structure of structural and parametric identification.

2.2 TIME SERIES FORECASTING AS A MACHINE LEARNING PROCEDURE

A time series is defined as sequence of ordered observations of a given phenomenon, indexed by time. Formally, we describe the time series associated to the observed quantity y as y_t, with t being the temporal index, whose definition could come from standard time definitions (SI, ISO: e. g., seconds, hours, days) or by domain-dependent measures (e. g., trading days,

quarters, business cycles for the economy domain). Figure 2.4 displays an example of time series in graphical form (on the left, with the value of y being the dependent variable and time being the independent variable) as well as in tabular form (on the right, with each column being a vector).

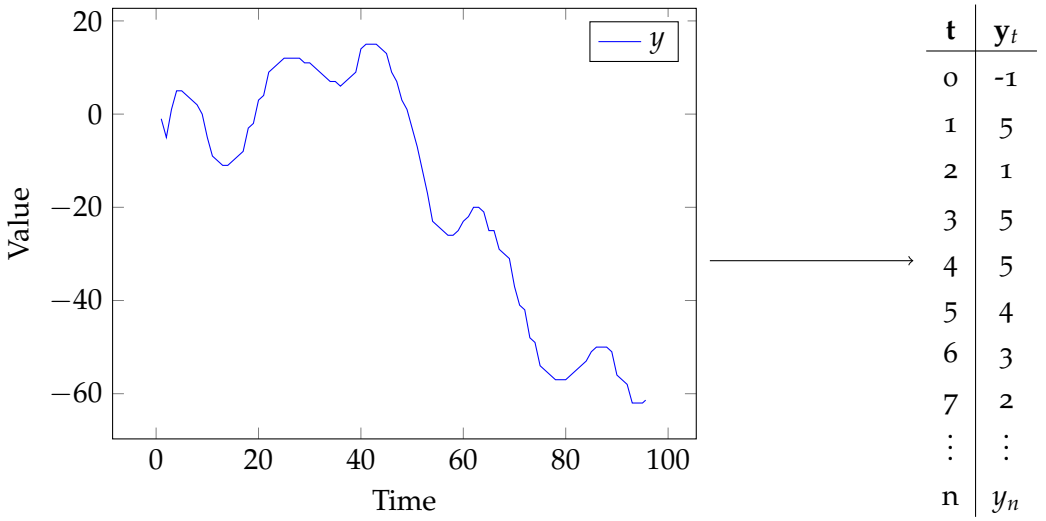


Figure 2.4: Visual and tabular representation of a time series

The only aspect that hasn't been covered yet by these definitions is the anticipation of future events (or prediction/forecasting), which will be the main focus of this thesis, and will be discussed in the following section.

2.2.1 *Problem formulation*

The problem of time series forecasting deals with the prediction of the future values of a given quantity of interest (i. e., the time series $\mathbf{y}$), given a set of N historical observations.

In order to produce meaningful forecasts, some assumptions about the informative content of the time series and its underlying dynamics need to be made. The key assumption on which the time series domains rests is that, if we denote with t the last available sample for the quantity y, the observed data (up to time t) contains relevant information that could be employed to provide predictions about the future (from time $t+1$ onward). In fact, if the future is independent of the past observations then there is no hope to produce good forecasts based on historical observations.

In more formal terms, we assume the existence of an unknown Data Generating Process (DGP), which is responsible for generating both past and future data. In practice, a common model of the data generating process is the autoregressive model (or process):

$$\{y_{t+H}, \cdots, y_{t+1}\} = F(y_{t-d}, \cdots, y_{t-d-m+1}) + \mathbf{e}_t \tag{2.13}$$

where the unknown process F produces the future values of the time series $\{y_{t+H}, \cdots, y_{t+1}\}$, based on the values of the previous m time steps, with an optional delay term d. [2] Note that the parameters of this model are the form of the unknown function F (linear or non-linear), the model order (or lag) m and the noise term $\mathbf{e}_t$, which is considered to be a stochastic independently and identically distributed process with null mean and fixed variance σ^2.

In this form, the problem of time series forecasting can be easily cast as a learning problem and more specifically a regression problem (cf. Equation 2.5), which can be tackled with the same statistical learning framework we presented in Section 2.1.

2 In the following of the thesis we will assume $d = 0$ for the sake of simplicity.

A comprehensive overview of the different time series forecasting tasks is presented in Figure 2.5. Note that *univariate* and *multivariate* refers to the structure of the input data to the problem, whereas *SISO*, *SIMO*, *MISO*, *MIMO*, refer to the structure of the model, respectively Single/Multiple Input, Single/Multiple Output, where Single/Multiple refer to the number of time series considered as input/output of the model. The details of the different formulations are discussed in the following sections.

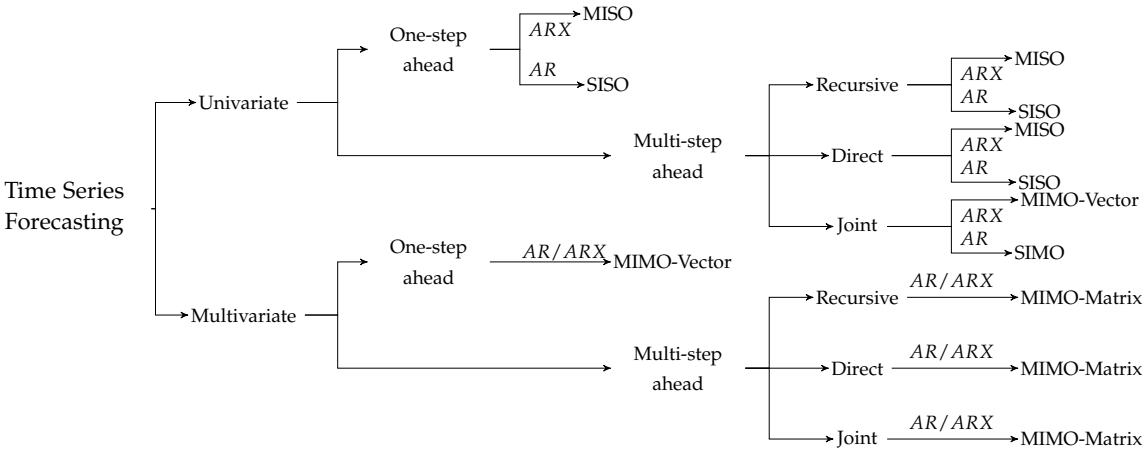


Figure 2.5: Summary of the different time series forecasting problems and the corresponding tasks. *AR* indicates an autoregressive hypothesis (i.e. the forecast uses only the information from the past of the considered time serie(s)), while *ARX* indicates the presence of external regressor(s). Single/Multiple refer to the number of time series considered as input/output of the model.

2.2.1.1 *Univariate time series forecasting techniques - One step ahead*

The simplest time series forecasting problem deals with the forecasting of a single time series (i. e., univariate) for a single step in the future (one-step-ahead). The problem is formulated as the estimation of a Single-Input, Single-Output (SISO) auto-regressive mapping $f : \mathbb{R}^m \mapsto \mathbb{R}$:

$$y_{t+1} = f(y_{t-d}, \cdots, y_{t-d-m+1}) + e_{t+1} \tag{2.14}$$

where e is the noise term (stochastic i.i.d process with null mean and fixed variance), d is the delay and $m > 0$ is called the model order (or embedding lag). This formulation is general since it can be employed for estimating both a linear (Auto-Regressive (AR)) and a nonlinear mapping (Non-linear Auto-Regressive (NAR)), and enables the adoption of supervised machine learning algorithms (Bontempi, Taieb, and Le Borgne, 2012).

In a linear autoregressive formulation (AR) the function f is a linear combination of the previous m values of the time series:

$$f(y_{t-d}, \cdots, y_{t-d-m+1}) = \sum_{i=t-d-m+1}^{t-d} a_i y_i \tag{2.15}$$

In a non-linear autoregressive (NAR) formulation, the function f is a non-linear, non parametric function. This flexibility in modeling allows the use of machine learning techniques for one-step-ahead time series forecasting (Bontempi, Ben Taieb, and Le Borgne, 2013), after a specific preprocessing phase (Section 3.3.2). This phase transforms the original univariate time series forecasting problem into the problem of learning the unknown input-output mapping f through supervised learning. Once a model of the mapping f is learned, it can be used for returning one-step-ahead forecasts.

2.2.1.2 *Multi-step-ahead univariate forecasting*

If the one-step forecasting of a time series is already a challenging task, performing multi-step forecasting is even more difficult because of additional complications, such as accumulation of errors and increased uncertainty.

Multi-step-ahead univariate forecasting consists in predicting the next $H > 1$ values of a time series.

Strategies for predicting univariate time series multi-step-ahead have been extensively discussed in (Ben Taieb, Sorjamaa, and Bontempi, 2010; Ben Taieb et al., 2012; Bontempi, Ben Taieb, and Le Borgne, 2013) and can be summarised into two main classes: single output and multiple output strategies.

Instances of the first class are the *Iterated* and the *Direct* strategies. The *Iterated* (or Recursive) strategy (Cheng et al., 2006; Sorjamaa et al., 2007; Weigend and Gershenfeld, 1994) learns a one-step-ahead model $f_{REC} : \mathbb{R}^m \mapsto \mathbb{R}$

$$y_{t+1} = f_{REC}(y_t, \ldots, y_{t-m+1}) + e_{t+1} \tag{2.16}$$

and then uses it recursively H times to return a multi-step-ahead prediction. Though the iterated method is highly sensitive to the estimation error, it has been often used to forecast real-world time series (Bontempi, Birattari, and Bersini, 1999b; McNames, 1998; Saad, Prokhorov, and Wunsch, 1998). The noise term e is a stochastic i.i.d process with null mean and fixed variance.

The *Direct* strategy (Cheng et al., 2006; Sorjamaa et al., 2007; Weigend and Gershenfeld, 1994) learns independently H models $f_h : \mathbb{R}^m \mapsto \mathbb{R}$, $h = 1, \ldots, H$

$$y_{t+h} = f_h(y_t, \ldots, y_{t-m+1}) + e_{t+h} \tag{2.17}$$

and returns a multi-step-ahead forecast by concatenating the H predictions. Since the *Direct* strategy does not use any estimated value as input, it is not prone to the accumulation of one-step-ahead errors. Notwithstanding, no conditional dependency between the predictions (Bontempi, 2008; Bontempi and Ben Taieb, 2011; Kline, 2004) is considered and these methods often require higher functional complexity (Tong, 1983) than iterated ones in order to model the dependency between two distant instants (Guo, Bai, and An, 1999). The noise term e is a stochastic i.i.d process with mean equal to zero and constant variance.

The *Multi-Input Multi-Output* (MIMO) strategy (Bontempi, 2008; Bontempi and Ben Taieb, 2011) (also known as Joint strategy (Kline, 2004)) avoids the simplistic assumption of conditional independence between future values made by the Direct strategy by learning a single multiple-output model:

$$[y_{t+H}, \ldots, y_{t+1}] = F_J(y_t, \ldots, y_{t-m+1}) + \mathbf{E} \tag{2.18}$$

where $F_J : \mathbb{R}^m \mapsto \mathbb{R}^H$ is a vector-valued function (Micchelli and Pontil, 2005) and $\mathbf{E}$ is a noise vector whose covariance is not necessarily diagonal (Matías, 2005). So far, this strategy has been successfully applied to several real-world multi-step time series forecasting tasks (Ben Taieb et al., 2009; Ben Taieb, Sorjamaa, and Bontempi, 2010; Bontempi, 2008; Bontempi and Ben Taieb, 2011).

2.2.2 *Data preprocessing*

This section presents some time-series-specific preprocessing techniques, aiming to improve the predictive performance of the forecasting model. It is worth noting that some of the general-purpose techniques presented in Section 2.1.4, such as data normalization and feature selection still apply in this context.

2.2.2.1 *Missing value handling*

A common assumption in the domain of statistical learning is that all the samples in the dataset are sampled from the same data generating process (i. e., identically distributed) and independent of each other. However, in the case of time series forecasting, this assumption neglects the existence of an underlying temporal dependence among the values (i. e., the unknown mapping f) to be estimated. For these reasons, a general-purpose missing value imputation technique should be avoided and specific techniques oriented to temporal data should be preferred. A complete overview of the different techniques available in the literature and the corresponding implementations can be found in (Moritz and Bartz-Beielstein, 2017). The techniques range from basic replacement methods such as: Last Observation Carried Forward or Next Observation Carried Backwards, consisting of repeating the last available value; to model-based techniques, relying on forecasts made via an autoregessive model, Kalman filtering or exponential smoothing; and statistical techniques, relying on interpolation or rolling statistics.

2.2.2.2 *Data transformation*

A common practice in forecasting models is to transform the variable of interest y_t in order to ensure that the statistical properties of the available data match the underlying assumptions of the employed forecasting model.

BOX-COX TRANSFORMATION The parametric transformation initially proposed by (Box and Cox, 1964) (known as the Box-Cox transformation), for instance, was developed to ensure assumptions of normality and constant error variance, commonly required for performing inference statistical models. The Box-Cox transformation of the time series y_t, controlled by the parameters λ_1 and λ_2 is defined as:

$$y_t^{(\lambda_1,\lambda_2)} = \begin{cases} \frac{\left((y_t+\lambda_2)^{\lambda_1}-1\right)}{\lambda_1} & \lambda_1 \neq 0 \\ \log(y_t+\lambda_2) & \lambda_1 = 0 \end{cases} \tag{2.19}$$

For $\lambda_1 = 0$, the transformation is logarithmic, whereas for $\lambda_1 = 1$, the transformation corresponds to a negative shift of one unit of the time series y_t. λ_2 appears as an additive constant in both the formulations, having a multiplicative effect on the original time series when it appears as an argument of the logarithm. By tuning the values of λ_1 and λ_2, this transformation can represent a continuum between multiplicative and additive models.

DIFFERENTIATION A relevant property of DGP process underlying the generation of the time series y_t is stationarity. Stationarity can be loosely described as the lack of dependence of the statistical properties of the DGP on time shift. More formally, two forms of stationarity can be defined: weak and strict. A process $\{y_t\}$ is said to be weakly stationary if (i) its mean $\mathbb{E}[y_t]$ is a constant value, independent of t, and (ii) its autocovariance function $\mathrm{Cov}(y_t, y_{t+\tau})$ does not depend on the time t itself, only on time-shift τ. A process is said to be strictly stationary if its unconditional joint probability distribution does not change when shifted in time.

A large part of the statistical forecasting theory is based on stationary models, with a focus on weak stationarity. However, real-life time series are often non-stationary due to the presence of a consistent trend (an increase/decrease in the mean over time) or to changes in the dynamic of the process (and consequently in the autocovariance function).

A common approach to ensure stationarity in time series is to apply differencing. First order differencing consists in taking the difference between two consecutive values in the time series:

$$y'_t = y_t - y_{t-1} \tag{2.20}$$

In practice, the application of a first-order difference is sufficient to ensure that y'_t is weakly stationary in the majority of the cases. However, higher-order differencing might be required to ensure stationarity. In that case, second order differencing consists in taking the difference of the first order difference series:

$$y''_t = y'_t - y'_{t-1} = y_t - 2y_{t-1} + y_{t-2} \tag{2.21}$$

In addition, for series presenting a cyclic behavior (i. e., a seasonality), it might be worth considering seasonal differencing (Hyndman and Athanasopoulos, 2018). Instead of taking the difference from the previous value, the difference is computed with the corresponding value in the previous cycle (with s being the length of the cycle).

$$y'^{s}_t = y_t - y_{t-s} \tag{2.22}$$

It should be noted that, since the proposed differencing techniques require a lookup of up to s samples in the past, the length of the differenced time series is reduced with respect to the original one. In other words, for a time series of length N, the differenced time series will have length $N - s$, with s being the number of samples in the lookup.

Last but not least, statistical testing can be applied to determine whether differencing is required or not. Statistical tests for stationarity are also called unit root tests (referring to the tested statistical property of the underlying DGP). Different unit root tests are based on different assumptions and have different null hypotheses. The most used tests are the Augmented Dickey-Fuller (ADF), and the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test.

DOMAIN-SPECIFIC Different transformations act on different components of the time series. As shown in (Hyndman and Athanasopoulos, 2018), differencing can help stabilize the mean of a time series by reducing (or eliminating) the effects of changes in trend and seasonality, and could be used as a general-purpose transformation.

However, specific applicative domains might require domain-specific transformations in order to exploit specific properties.

We will focus here on a domain-specific transformation, from the financial domain, relevant for the remainder of the thesis: compounding return.

In the framework of stock markets, the price of a stock at the end of the day is named closing price $P_t^{(c)}$, or simply price (A.6) at time t (t being the considered trading day).

$$P_t = P_t^{(c)} \tag{2.23}$$

Here, instead of considering the simple difference between the price of two consecutive trading days, we define an additional quantity, the return R_t, as the ratio between two consecutive prices (minus a constant), corresponding to the net relative increase/decrease occurred during the day:

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}} = \frac{P_t}{P_{t-1}} - 1 \tag{2.24}$$

This allows us to define, in turn, the notions of continuously compounded return (A.7).

$$r_t = \ln\left(\frac{P_t}{P_{t-1}}\right) = \ln(P_t) - \ln(P_{t-1}) \tag{2.25}$$

Here, the logarithmic transformation, in combination with the ratio, helps to smooth the time series dynamic and consequently could help to stabilize the variance of a time series (Hyndman and Athanasopoulos, 2018).

2.2.2.3 *Seasonal decomposition*

In the previous sections, we mentioned the existence of alterations to the stationary behavior of the time series, namely shifts in the trend, and cyclical behavior.

Through seasonal decomposition (Petropoulos et al., 2021), it is possible to represent the original time series as a composition of different elements, each of them representing a specific dynamic: the seasonal component S_t represents the repeating patterns, the trend T_t captures the underlying mean and the remainder component Re_t includes all the dynamic that hasn't been captured by the previous two components.

Two main families of decomposition exist: additive (Equation 2.26) and multiplicative (Equation 2.27). Their difference is in the way the composition of the elements is modeled, in the first case, as a sum, while in the second case, as a multiplication.

$$y_t = S_t + T_t + Re_t \tag{2.26}$$

$$y_t = S_t \cdot T_t \cdot Re_t \tag{2.27}$$

It should be noted that an additive decomposition of a log-transformed series is equivalent to a multiplicative decomposition.

For a detailed analysis of the different decomposition techniques, we refer the interested reader to the following references: (Hyndman and Athanasopoulos, 2018; Petropoulos et al., 2021).

2.2.2.4 *Feature engineering*

General-purpose feature engineering (such as the techniques discussed in Section 2.1.4.3) could be applied without loss of generality to the time series problem. However, the majority of these techniques do not take into account the temporal dependence that characterizes the time series forecasting problem. The following sections discuss some techniques for feature extraction that are specific to the time series domain, namely: date-based, summary-based, window-based. It is worth noting that all aforementioned categories can include domain-specific features, manually created according to the expertise of the practitioner. Moreover, automated feature extraction represents a promising research direction (Mierswa, 2005).

DATE-BASED FEATURES The majority of the feature engineering techniques works directly on the value of the time series, neglecting the information contained in the time index itself. However, according to the type of available data, the time index can be a full timestamp (i. e., including both date and time), a partial timestamp (including only either the date or the time component) or a generic increasing index. When timestamps (both full and partial) are available, they can be used to extract relevant information such as the day of the week, the week in the year or an indicator variable stating if the corresponding date is an holiday/working day/weekend. With this approach, every sample can be augmented by including the extracted information, thus generating a complete time series of length N. When this information is relevant for the forecasting task at hand (i. e., mobility or sales forecasting), their extraction can greatly improve the forecasting performance of the model.

SUMMARY-BASED FEATURES An additional approach for extracting features from a time series is to compute a metric over the N available data points and provide a single value summarizing a specific property of the series. The `tsfeatures` library (Hyndman et al., 2020) contains several examples of such statistics such as the *Shannon's entropy*, or *stability* (i. e., the variance of the means of tiled, non-overlapping windows) and *lumpiness* (i. e., the variance of the variances of tiled, non-overlapping windows). Additional summary features can be computed based on the results of the STL decomposition (Section 2.2.2.3) or by fitting a statistical model (e. g., Holt-Winters or ARCH) and extracting the coefficients corresponding to the series.

WINDOW-BASED FEATURES If there is a need for richer temporal features, a better approach could be to compute rolling metrics, using either a *rolling origin* or a *sliding window* approach (Tashman, 2000). Common choices for window-based features are rolling averages, rolling counts and rolling standard deviations. In this case, if we consider the size of the time window to be s_w, the resulting feature is a time series of length $N - s_w$ as opposed to a summary-based statistic, which will yield a single value.

2.2.3 *Learning phase*

The native format of time series data is not suitable to approach the forecasting problem as a statistical (supervised) learning problem. For this reason, an embedding procedure is required to restructure the available data in order to ensure that a statistical learning procedure could be implemented and that the mapping f can be learned through the procedure corresponding to the desired forecasting problem. Once embedding is performed, the theoretical framework presented in Section 2.1.5 can be applied with minor modifications, in order to account for the temporal correlation between samples. For this reason, in this section, we will only present the specific adaptations related to the time series domain, namely those concerning the structural identification process in case of multiple-step-ahead forecasts, the evaluation procedures and corresponding validation metrics. The different learning algorithms to learn the mapping f will be presented in detail in Section 2.3.

2.2.3.1 *Embedding*

Once the process of feature engineering has been completed, the last necessary step before the statistical learning process is the embedding one. In the embedding phase, the time series, originally a one-dimensional vector, is embedded into an higher-dimensional format, in order to be suitable for the learning process. As shown in Figure 2.6, the same time series can be embedded into different formats, each of them more suitable for a certain category of models.

The *matrix embedding* is commonly employed for shallow machine learning models, as it allows to frame the forecasting problem as the learning of the non-linear one-step-ahead autoregressive mapping f (Equation 3.3) between the past m values of the time series $\{y_{t-m}, \cdots, y_t\}$ and the future value y_{t+1}.

The *tensor embedding* is commonly employed for deep learning models, especially for sequence-based ones (such as RNN), where the data is rearranged in a tensor format, with an additional window size parameter s_w. In this format, the sequence of s_w samples is passed simultaneously to the model which is able to learn the temporal dependencies between the samples. In addition, the tensor format is better suited for parallel processing.

With the focus of the thesis being on traditional machine learning models, in the remainder of the thesis, we will mostly focus on the *matrix embedding*. Figure 2.7 presents the alternative formats of the matrix embedding in order to fit a one-step-ahead model (also employed in

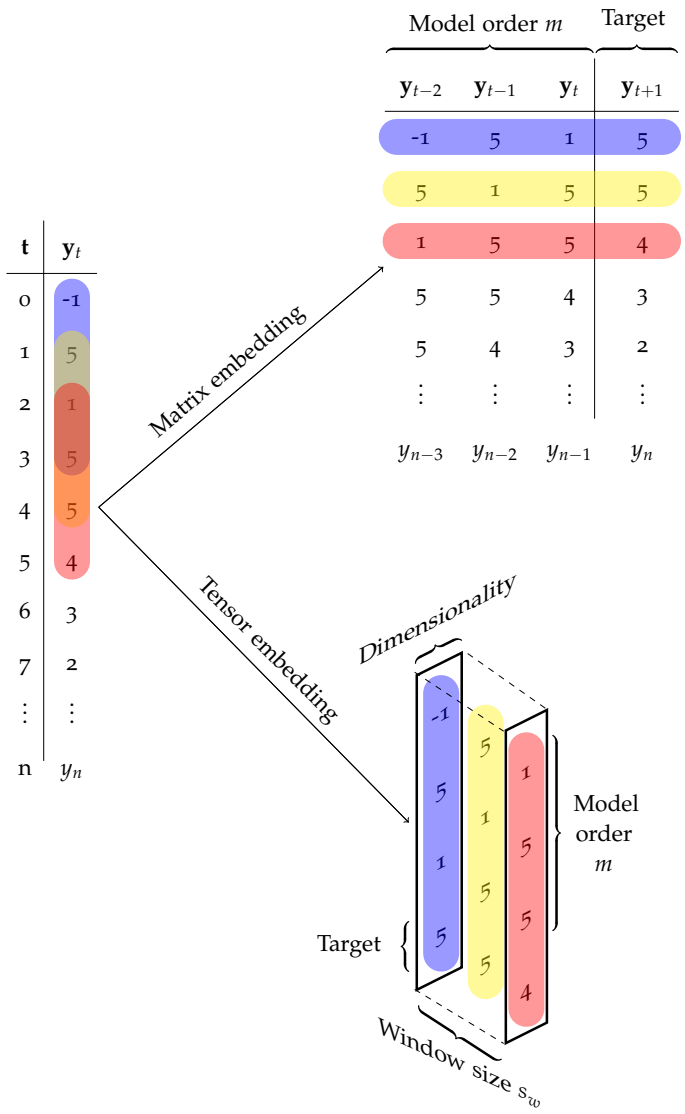


Figure 2.6: Embedding of a univariate time series with an embedding order $m = 3$ for a one-step-ahead forecasting as a matrix (suitable for traditional Machine Learning (ML) models), and a tridimensional tensor (suitable for DL models).

the *Recursive* multi-step-ahead strategy) and a h-step-ahead model (employed in the *Direct* strategy).

2.2.3.2 *Multiple-step-ahead forecasting strategies*

The traditional supervised learning approach is well-suited to solve the problem of one-step-ahead forecasting (Section 2.2.1.1), as this problem can be naturally framed as an input-output mapping between the past values $\{y_t, \cdots, y_{t-m+1}\}$, and the single value to forecast y_{t+1}. However, when the problem requires multiple-step-ahead forecasting (Section 2.2.1.2), the target variable to predict is not a scalar anymore but indeed a vector of H elements, with H being the forecast horizon.

When facing a multiple-step-ahead forecasting problem, we typically have a choice between the recursive and the direct forecasting strategy. With the *Recursive* strategy, forecasts are generated using a one-step-ahead model and iteratively fed back into the model until the desired number of steps is reached. With the *Direct* strategy, a horizon-specific model is estimated and forecasts are computed directly by the corresponding model for each forecast horizon. The strategies and their application in a Machine Learning context are detailed

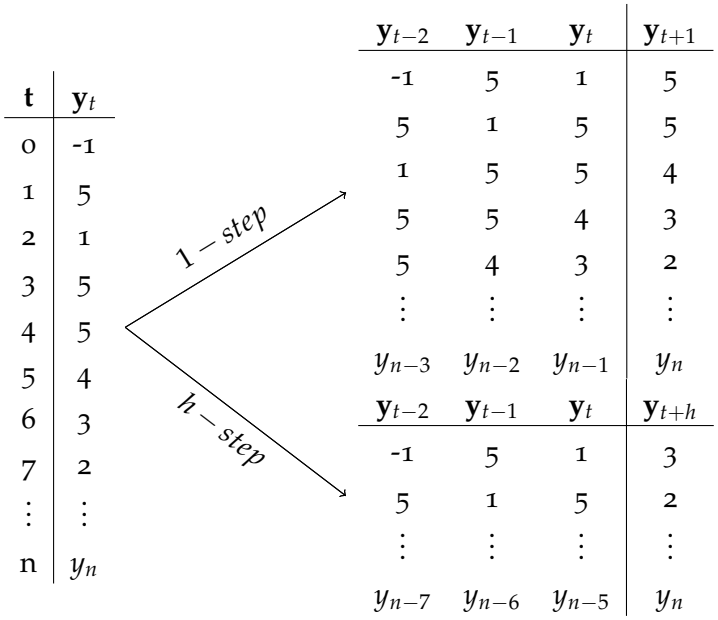


Figure 2.7: Embedding of a univariate time series with an embedding order $m = 3$ for a 1-step and h-step ahead forecast ($h = 4$).

in sections 2.2.3.2 and 2.2.3.2 respectively. An overview of the two strategies is provided in Figure 2.8, while Table 2.1 contains some guidelines for the practical application of the strategies.

Recursive

- A single model for all the horizons.
- Forecast are made recursively (i.e. forecast at step h based on forecast at step $h-1$).

X			y
y_3	y_2	y_1	y_4
y_4	y_3	y_2	y_5
...	...	...	...
y_{T-1}	y_{T-2}	y_{T-3}	y_T

Direct

- h independent models, one for each horizon h.
- Forecast at h step is made using h^{th} model.

X				y
y_4	y_3	y_2	y_1	y_6
y_5	y_4	y_3	y_2	y_7
...	...	...	...	...
y_{T-4}	y_{T-5}	y_{T-6}	y_{T-7}	y_{T-2}

Figure 2.8: Comparison of the dataset structure and model identification for direct and recursive forecasting strategies as described in Figure 2.9 and Figure 2.10.

RECURSIVE STRATEGY The first step of the recursive strategy is a data preprocessing step that transforms the time series (i.e. a time indexed sequence of points) in an embedded input-output form.

After this step, one disposes of a dataset $\mathcal{D}$ of $T - m - 1$ points in the form $(\mathbf{x}; y)$, where y corresponds to the value of the time series at time t, while $\mathbf{x}$ is composed of the m values preceding the one at time t, where p is one of the hyperparameters of the model, representing the estimated lag order d. The dataset is then split into a training subset $\mathcal{D}_{\text{train}}$ and a validation subset $\mathcal{D}_{\text{test}}$ according to the standard cross-validation schemes.

Then, the model is estimated through a two-levels nested procedure that first aims to perform a parameter estimation of the models parameters vector $\boldsymbol{\theta}^{i,j}_{\text{train}}$ on the training set

	Linear DGP	**Non-linear DGP**
Linear model	*Recursive* > *Direct* with short time series and long h, otherwise equivalent.	*Recursive* > *Direct* with short time series and long h. Both biased due to model misspecification.
Non-linear model	*Recursive* $\approx$ *Direct* if the model cannot switch to linear	*Direct* > *Recursive* for long time series.

Table 2.1: Guidelines for the application of direct and recursive forecasting strategies according to (Ben Taieb, Hyndman, and Bontempi, 2014).

Require: Model classes Λ_i $i \in \{1, \cdots, S\}$, Learning algorithms $\mathcal{L}_i$, Time series $\{y_1, \cdots, y_T\}$
Ensure: h-step ahead forecasts $\{\hat{y}_{T+1}, \cdots, \hat{y}_{T+H}\}$

1: $\{y_1, \cdots, y_T\} \rightarrow \mathcal{D} = \{(y_{t-1}, ..., y_{t-m}; y_t)\}_{t=m+1}^{T}$ ▷ Construct dataset $\mathcal{D}$ from available data via *matrix embedding*
2: $\mathcal{D} \rightarrow \mathcal{D}_{\text{train}} = \{(\mathbf{x}_{t-1}, y_t)\}_{t=1}^{j} \vee \mathcal{D}_{\text{test}} = \{(\mathbf{x}_{t-1}, y_t)\}_{t=j+1}^{T}$ ▷ Cross-validation split
3: **for** $i \in \{1, \ldots, S\}$ **do** ▷ Structural identification on $\mathcal{D}_{\text{test}}$
4: **for** $h_j \in \Lambda_i$ **do** ▷ Parametric identification on $\mathcal{D}_{\text{train}}$
5: $\boldsymbol{\theta}^{i,j}_{\text{train}} \leftarrow \mathcal{L}_i(h_j, \mathcal{D}_{\text{train}})$ ▷ Identification of optimal $\boldsymbol{\theta}^{i,j}_{\text{train}}$ via learning algorithm $\mathcal{L}_i$ for hypothesis h_j
6: **end for**
7: $h^i_{\text{train}} \leftarrow \arg\min_{h_j \in \Lambda_i} R_{\text{emp}}(h_j, \boldsymbol{\theta}^{i,j}_{\text{train}})$ ▷ Optimal model h^i_{train} in class Λ_i on $\mathcal{D}_{\text{train}}$
8: **end for**
9: $h^*_{\text{test}} \leftarrow \arg\min_{i \in \{1,\ldots,S\}} R_{\text{emp}}(h^i_{\text{train}}, \boldsymbol{\theta}^{i,j}_{\text{train}})$ ▷ Optimal model h^*_{val} across classes based on $\mathcal{D}_{\text{test}}$
10: $m(\mathbf{x}_t, \boldsymbol{\theta}^{i,j}_{\text{train}}) \leftarrow h^*_{\text{test}}$ ▷ A *single* model $m(\mathbf{x}_t, \boldsymbol{\theta}^{i,j}_{\text{train}})$ for h-step forecast is produced
11: **for** $k \in \{1, \cdots, H\}$ **do** ▷ *Recursive* h-step ahead forecasting
12: $\hat{y}_{T+k} \leftarrow m(\mathbf{x}_{T+k}, \boldsymbol{\theta}^{i}_{\text{train}})$
13: **end for**

Figure 2.9: Pseudo-code summarizing the h-step-ahead *Recursive* strategy, including both model identification (Lines 1-10) and forecasting (Lines 11-13).

$\mathcal{D}_{\text{train}}$ and then select the best model $m(\mathbf{x}_t, \boldsymbol{\theta}^{i,j}_{\text{train}})$, (i. e., the one that minimizes the one step ahead forecasting error on the validation set $\mathcal{D}_{\text{test}}$). Nevertheless, this one-step-ahead-oriented selection does not offer any guarantee of the minimization of the h-step-ahead forecasting error.

Once this estimation has been completed, the optimal values of these parameters are used to perform one step ahead forecasts. Every time a new forecast is produced, it is concatenated to the existing values and fed again into the model as an estimation of the corresponding missing future value, in order to determine the successive prediction, until the desired forecasting horizon H. An overview of the complete procedure is presented in Figure 2.9.

DIRECT STRATEGY The *Direct* strategy shares several aspects of the model identification algorithm, namely data preprocessing and parameters optimization procedures with the *Recursive* (Figure 2.10). Nevertheless, its main difference is that a different model $m^{(k)}(\mathbf{x}_t, \boldsymbol{\theta}^{i,j,k}_{\text{train}})$

Require: Model classes Λ_i $i \in \{1, \cdots, S\}$, Learning algorithms $\mathcal{L}_i$, Time series $\{y_1, \cdots, y_T\}$
Ensure: h-step ahead forecasts $\{\hat{y}_{T+1}, \cdots, \hat{y}_{T+H}\}$

1: **for** $k \in \{1, \cdots, H\}$ **do** ▷ H different models are fit
2: $\{y_1, \cdots, y_T\}$ $\{y_1, \cdots, y_T\} \rightarrow \mathcal{D}^k = \{(y_{t-k}, ..., y_{t-k-m}; y_t)\}_{t=k+m+1}^{T-k}$ ▷ Construct dataset $\mathcal{D}^{\|}$ from available data via *matrix embedding*
3: $\mathcal{D}^k \rightarrow \mathcal{D}^k_{\text{train}} = \{(\mathbf{x}_{t-k}, y_t)\}_{t=1}^{j} \vee \mathcal{D}^k_{\text{test}} = \{\{(\mathbf{x}_{t-k}, y_t)\}_{t=j+1}^{T}$ ▷ Cross-validation split
4: **for** $i \in \{1, \ldots, S\}$ **do** ▷ Structural identification on $\mathcal{D}^k_{\text{val}}$
5: **for** $h_j \in \Lambda_i$ **do** ▷ Parametric identification on $\mathcal{D}^k_{\text{train}}$
6: $\boldsymbol{\theta}^{i,j,k}_{\text{train}} \leftarrow \mathcal{L}_i(h_j, \mathcal{D}^k_{\text{train}})$ ▷ Identification of optimal $\boldsymbol{\theta}^{i,j,k}_{\text{train}}$ via learning algorithm $\mathcal{L}_i$
7: **end for**
8: $h^{(k),i}_{\text{train}} \leftarrow \arg\min_{h_j \in \Lambda_i} R_{\text{emp}}(h_j, \boldsymbol{\theta}^{i,j,k}_{\text{train}})$ ▷ Optimal model $h^{(k),i}_{\text{train}}$ in class Λ_i on $\mathcal{D}^k_{\text{train}}$
9: **end for**
10: $h^{(k)*}_{\text{val}} \leftarrow \arg\min_{i \in \{1,\ldots,S\}} R_{\text{emp}}(h^i_{\text{train}}, \boldsymbol{\theta}^{i,j,k}_{\text{train}})$ ▷ Optimal model h^*_{test} across classes based on $\mathcal{D}^k_{\text{test}}$
11: $m^{(k)}(\mathbf{x}_t, \boldsymbol{\theta}^{i,j,k}_{\text{train}}) \leftarrow h^{(k)*}_{\text{test}}$ ▷ A model $m^{(k)}(\mathbf{x}_t, \boldsymbol{\theta}^{i,j,k}_{\text{train}})$ is produced for each k^{th} forecasting step
12: **end for**
13: **for** $k \in \{1, \cdots, H\}$ **do** ▷ *Direct* h-step ahead forecasting
14: $\hat{y}_{T+k} \leftarrow m^{(k)}(\mathbf{x}_{T+k}, \boldsymbol{\theta}^{i,j,k}_{\text{train}})$ ▷ For k-step-ahead forecasting, the k^{th} model is employed.
15: **end for**

Figure 2.10: Pseudo-code summarizing the h-step-ahead *Direct* strategy, including both model identification (Lines 1-12) and forecasting (Lines 13-15).

is estimated for each forecasting horizon $k \in \{1, \cdots, H\}$. As a consequence, a different dataset $\mathcal{D}$ and different parameter vector $\boldsymbol{\theta}^{i,j,k}_{\text{train}}$ have to be estimated for each forecasting horizon $h \in \{1, \cdots, H\}$, resulting in a greater computational complexity. In this case, at every forecasting horizon k the parameter estimation is performed by minimizing the forecasting error on the k-step-ahead forecast. Then, each of the H estimated models is used to compute the forecast at the corresponding k-step-ahead forecasting horizon.

2.2.3.3 *Model validation*

As shown in Figures 2.9 and 2.10, model validation plays an important role in determining the best model for the successive forecast phase. However, the choice of an appropriate cross-validation technique, as well the choice of the adequate empirical risk R_{emp} are crucial design choices, with a strong impact on the final selected model. Finally, the choice of the adequate statistical test is an additional relevant step for an appropriate validation of the model. In the following sections, we will briefly provide some guidelines for the aforementioned design choices.

CROSS-VALIDATION The process of cross-validation (Section 2.1.5.4) still remains relevant for the problem of time series forecasting. As noted by the author of (Tashman, 2000), validating the model on the same data it has been trained on (i. e., in-sample assessment) tends to give overly optimistic results, which more often than not cannot be repeated on the validation sets. For these reasons, the author proposed several cross-validation procedures, with a specific focus on temporal-dependent data. We present here the two most employed in practice with contiguous training and validation sets: rolling (forecast) origin (Figure

2.11), where a training period of increasing size is generated for every cross-validation fold, and rolling window (Figure 2.12), where a fixed size training period is shifted across time.

In (Tashman, 2000), the author advocates the use of a rolling origin technique, more specifically in combination with a parameter re-estimation technique.

More recently, the authors of (Cerqueira, Torgo, and Mozetic, 2019) performed an extensive empirical study of performance evaluation techniques, with a specific focus on time series problem, including the techniques proposed by (Tashman, 2000) and some of their variations, as well as traditional cross-validation techniques. The variations involved the introduction of a gap between the training and validation set (Figure 2.13), whereas the traditional cross-validation techniques involved standard cross-validation (Figure 2.14), cross-validation with removal of adjacent samples between training and validation set (Figure 2.15) and cross-validation with random removal of samples (Figure 2.16).

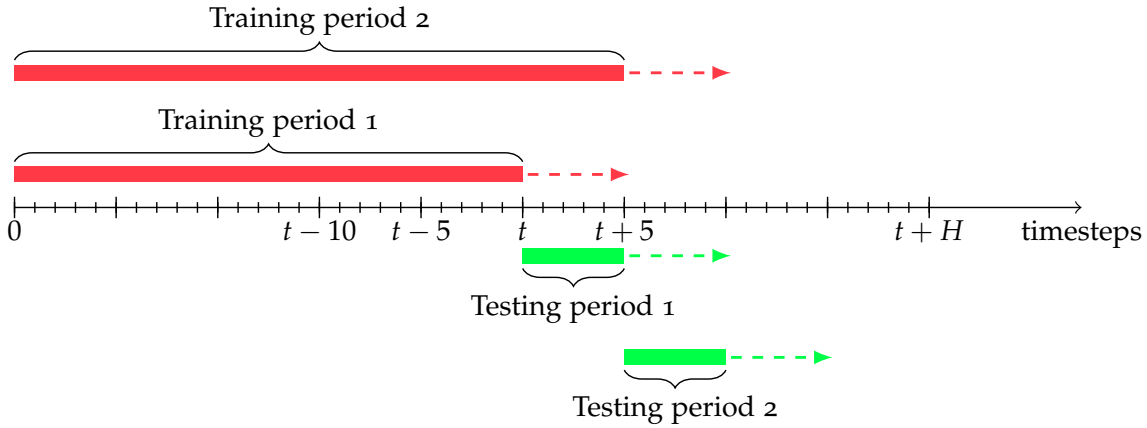


Figure 2.11: Example of rolling origin (Tashman, 2000) (also called Prequential growing window `Preq-Bls` in (Cerqueira, Torgo, and Mozetic, 2019)) 2-fold cross-validation.

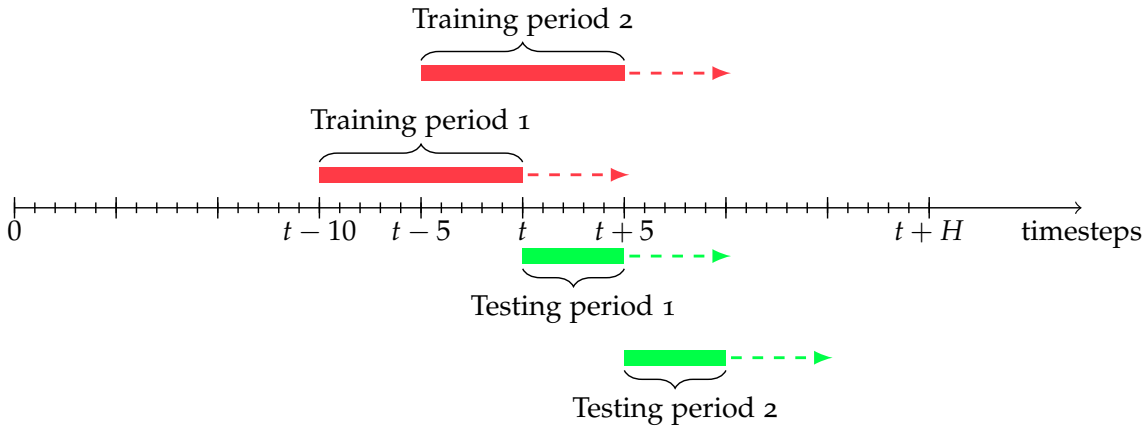


Figure 2.12: Example of rolling window (Tashman, 2000) (also called Sliding window - `Preq-Sld-Bls` in (Cerqueira, Torgo, and Mozetic, 2019)) 2-fold cross-validation.

After their empirical assessment, the authors of (Cerqueira, Torgo, and Mozetic, 2019) concluded that, based on the previous literature, cross-validation techniques are more appropriated when the considered time series is stationary, whereas holdout techniques, such as those presented in (Tashman, 2000) should be preferred for real-life time series.

UNIVARIATE ERROR MEASURES As shown in Section 2.1.5.3, the choice of an empirical risk R_{emp} plays a crucial role in the model identification procedure. A common choice, both for general-purpose machine learning and for time series oriented machine learning, is to use

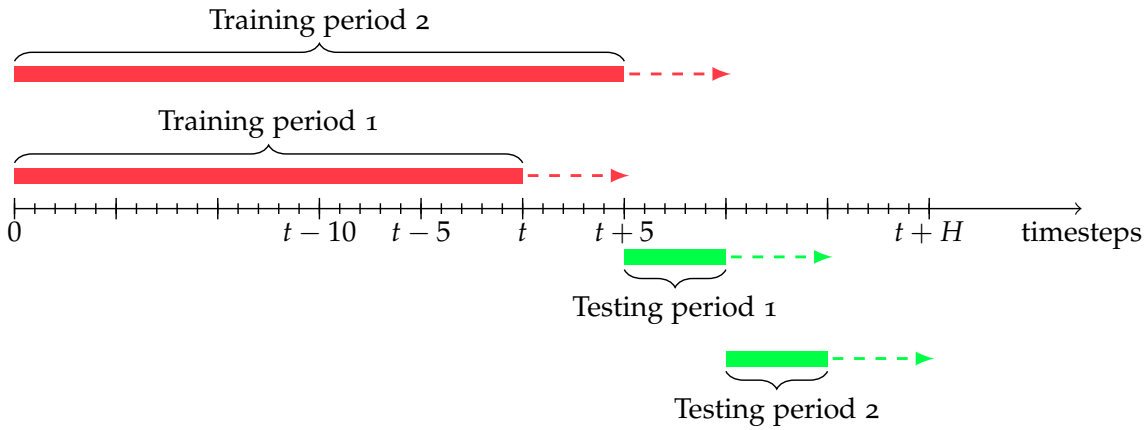


Figure 2.13: Example of Prequential growing window with gap - `Preq-Bls-Gap` (Cerqueira, Torgo, and Mozetic, 2019) 2-fold cross-validation.

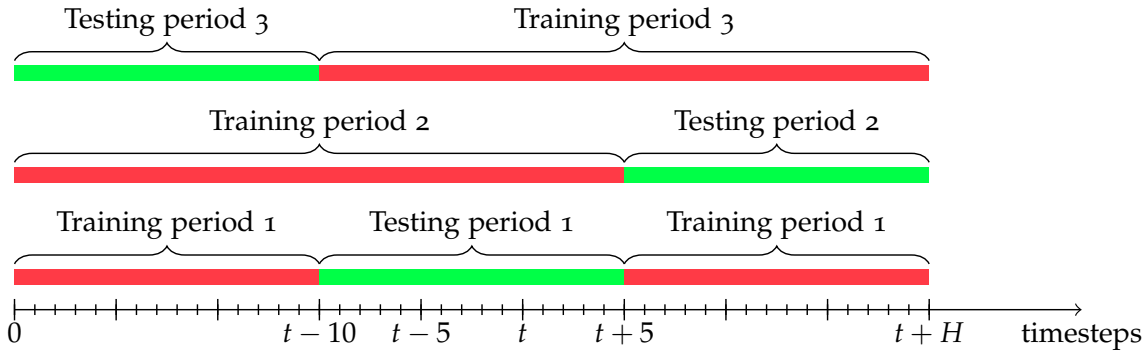


Figure 2.14: Example of Block cross-validation - `CV-Bl` (Cerqueira, Torgo, and Mozetic, 2019) 3-fold cross-validation.

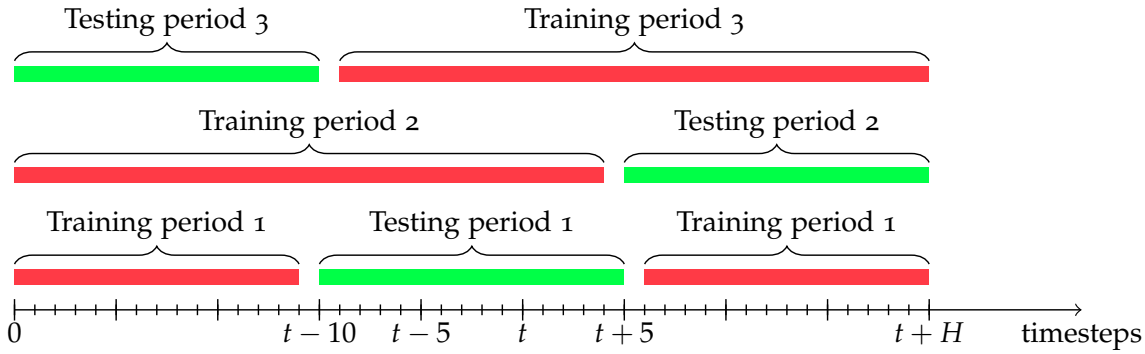


Figure 2.15: Example of hv Block cross-validation - `CV-hvBl` (Cerqueira, Torgo, and Mozetic, 2019) 3-fold cross-validation.

the mean squared error (MSE). However, as shown by the authors of (Hyndman and Koehler, 2006), this error measure is scale-dependent and might lead to incorrect conclusions about the forecasting performance, especially on problems displaying time series with different orders of magnitude. For this reason, they analyzed the forecasting literature, collecting and classifying the different error measures, as summarized in Table 2.2. Generally speaking, the authors suggest employing measures that are normalized with respect to a naive forecasting method (such as the MASE), due to their simplicity of interpretation: values of a naive normalized error measure greater than one indicate that the forecasts are worse, on average, than in-sample one-step forecasts from the naïve method. However, if all series have the same scale, the MAE should be preferable, due to its easier interpretation. Conversely, if

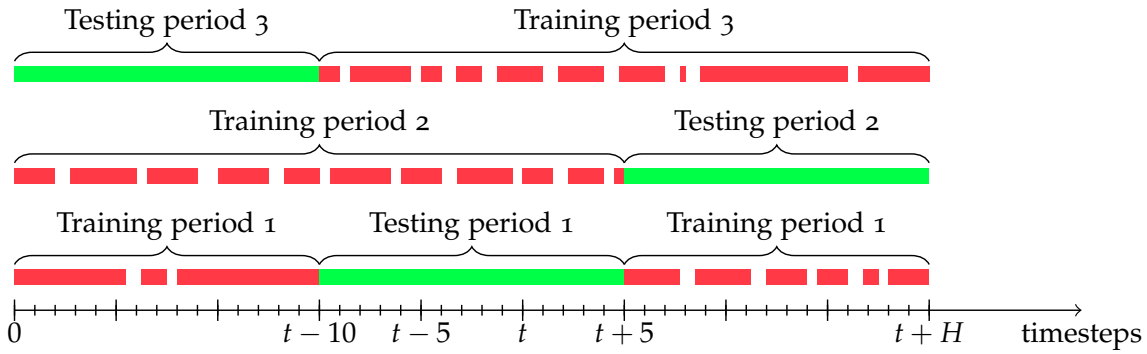


Figure 2.16: Example of Modified cross-validation - `CV-Mod` (Cerqueira, Torgo, and Mozetic, 2019) 3-fold cross-validation.

all data are positive and much greater than zero, the MAPE should be preferred. For a more detailed discussion, we direct the interested reader to (Hyndman and Koehler, 2006). It is worth mentioning that, in the framework of this thesis, we have implemented all the presented error measures in a publicly available R package: `UEMTS`.

MULTIVARIATE ERROR MEASURES The aforementioned univariate error measures can be extended without loss of generality to the multivariate case, simply by computing the chosen error measure for each of the n individual time series $Y[i]$ to obtain a distribution of n error measures. Finally, the individual measures can be aggregated via traditional statistics (such as mean or median). Following this principle and the recommendations from the authors of (Hyndman and Koehler, 2006), we propose the multivariate Naive-Normalized Mean Squared Error (NNMSE) for the empirical assessments in this thesis:

$$\text{NNMSE} = \frac{1}{n}\sum_{j=1}^{n}\text{NNMSE}[j] \tag{2.28}$$

$$\text{NNMSE}[j] = \frac{\frac{1}{H}\sum_{h=1}^{H}(\mathbf{Y}_{T+h}[j]-\hat{\mathbf{Y}}_{T+h}[j])^2}{\frac{1}{H}\sum_{h=2}^{H}(\mathbf{Y}_{T+h}[j]-\mathbf{Y}_{T+h-1}[j])^2} \tag{2.29}$$

The NNMSE is defined as the mean of the individual NNMSEs NNMSE$[j]$. For each of the n individual time series, NNMSE$[j]$ corresponds to the MSE of the considered forecasting method divided by the MSE of the Naive benchmark(Section 2.3.2.1). As for the univariate techniques, it is worth mentioning that, in the framework of this thesis, we implemented a collection of multivariate error measures in the publicly available R package `MEMTS`.

2.2.3.4 *Statistical testing*

Last but not least, once the model has been identified and validated, a statistical assessment procedure is performed in order to determine whether the difference in performance between two (or more) forecasting techniques is statistically significant. Statistical hypothesis testing can also be employed directly on the input data, in order to assess the validity of the hypotheses underlying parametric forecasting techniques (such as Augmented Dickey-Fuller test for stationarity). Due to the non-parametric nature of the proposed methods, here we will focus only on tests assessing the significance of the difference in performance between techniques. Among the different techniques existing in the literature, we focus on the Friedman test (with Tukey's and Nemenyi post-hoc analysis), whose appropriateness has been discussed in (Demšar, 2006).

FRIEDMAN-TUKEY TEST Friedman's test is a non-parametric randomized block analysis of variance whose null hypothesis H_0 is that the error distributions are the same across repeated measures (Friedman, 1937). If the test rejects the H_0 hypothesis, a post-hoc analysis

Type	Error measure	Formulation
Scale-dependent	$MSE(y_t, \hat{y}_t)$	$\frac{1}{T}\sum_{t=0}^{T}(y_t - \hat{y}_t)^2$
	$RMSE(y_t, \hat{y}_t)$	$\sqrt{\frac{1}{T}\sum_{t=0}^{T}(y_t - \hat{y}_t)^2}$
	$MAE(y_t, \hat{y}_t)$	$\frac{1}{T}\sum_{t=0}^{T}\lvert y_t - \hat{y}_t\rvert$
	$MdAE(y_t, \hat{y}_t)$	$Md_{t\in\{1\cdots T\}}(\lvert y_t - \hat{y}_t\rvert)$
Scale-independent	$MAPE(y_t, \hat{y}_t)$	$\frac{1}{T}\sum_{t=0}^{T}\mid 100\cdot\frac{y_t-\hat{y}_t}{y_t}\mid$
	$MdAPE(y_t, \hat{y}_t)$	$Md_{t\in\{1\cdots T\}}(\left\lvert 100\cdot\frac{y_t-\hat{y}_t}{y_t}\right\rvert)$
	$RMSPE(y_t, \hat{y}_t)$	$\sqrt{\frac{1}{T}\sum_{t=0}^{T}(100\cdot\frac{y_t-\hat{y}_t}{y_t})^2}$
	$RMdSPE(y_t, \hat{y}_t)$	$\sqrt{Md_{t\in\{1\cdots T\}}((100\cdot\frac{y_t-\hat{y}_t}{y_t})^2)}$
	$sMAPE(y_t, \hat{y}_t)$	$\frac{1}{T}\sum_{t=0}^{T}200\cdot\frac{\lvert y_t-\hat{y}_t\rvert}{y_t+\hat{y}_t}$
	$sMdAPE(y_t, \hat{y}_t)$	$Md_{t\in\{1\cdots T\}}(200\cdot\frac{\lvert y_t-\hat{y}_t\rvert}{y_t+\hat{y}_t})$
Relative errors	$MRAE(y_t, \hat{y}_t)$	$\frac{1}{T}\sum_{t=0}^{T}\left\lvert\frac{y_t-\hat{y}_t}{y_t-\hat{y}_t^{\text{bench}}}\right\rvert$
	$MdRAE(y_t, \hat{y}_t)$	$Md_{t\in\{1\cdots T\}}(\left\lvert\frac{y_t-\hat{y}_t}{y_t-\hat{y}_t^{\text{bench}}}\right\rvert)$
	$GMRAE(y_t, \hat{y}_t)$	$\sqrt[n]{\frac{1}{T}\prod_{t=0}^{T}\left\lvert\frac{y_t-\hat{y}_t}{y_t-\hat{y}_t^{\text{bench}}}\right\rvert}$
	$MASE(y_t, \hat{y}_t)$	$\frac{1}{T}\sum_{t=1}^{T}\left(\frac{\lvert y_t-\hat{y}_t\rvert}{\frac{1}{T-1}\sum_{i=2}^{T}\lvert Y_i-Y_{i-1}\rvert}\right)$
	$NMSE(y_t, \hat{y}_t)$	$\frac{1}{T}\frac{\sum_{t=0}^{T}(y_t-\hat{y}_t)^2}{var(y_t)}$
Relative measures	$RelX(X, X^{\text{bench}})$	$\frac{X}{X^{\text{bench}}}$
	Percent Better $PB(X, X^{\text{bench}})$	$100\cdot\frac{1}{T}\sum_{\text{forecasts}} I(X < X^{\text{bench}})$

Table 2.2: Overview of the different univariate performance evaluation techniques found in the literature (Hyndman and Koehler, 2006). In the *Relative errors* category, $\hat{y}_t^{\text{bench}}$ indicates the forecast of the benchmark method at time t. In the *Relative measures* category, X represents the error measure of the analyzed method, while X^{bench} indicates the error measure of the benchmark.

is run to find which pairs of methods are significantly different (Pereira, Afonso, and Medeiros, 2015).

The analysis is based on Tukey's test and supplies an upper diagonal square matrix of size equal to the number of methods, where the column elements are sorted by their rank. Each cell of the matrix can contain two possible colors: grey if the forecasting model corresponding to the row is significantly worse than the one in the column, orange otherwise. By looking at the colors (for instance Figure 5.7), this representation allows to visualize groups of predictions whose forecasting performances are not significantly different from each other.

FRIEDMAN-NEMENYI The Friedman-Nemenyi test combines the Friedman ranking statistical test (to determine the ranking of the different forecasting techniques) with a post-hoc Nemenyi test (to assess the significance of the differences in ranking of the methods). For each time series in the multivariate dataset, the considered forecasting techniques are ranked according to the values of their corresponding error measure. Then, the average rank across time series is computed for every forecasting technique and employed as input for the Friedman test (Demšar, 2006). Finally, the post-hoc Nemenyi test is employed to assess the statistical significance of the results of the Friedman test by determining the value of the critical difference (CD). Two forecasting techniques are considered to not have a statistically

significant difference if the difference between their average ranks is smaller than the critical difference.

In the results visualization (for instance, see Figure 4.2), the methods are ordered according to their performance from left to right (the leftmost the best), while the black bar connects methods that are not significantly different (at $p = 0.05$). The top-left bar labeled *CD* gives a visual indication of the value of the critical difference.

2.3 FORECASTING METHODS

In Section 2.2, we have seen how the problem of time series forecasting can be framed as a statistical learning problem and consequently how to learn the underlying unknown mapping f between past and future data, in order to be able to perform predictions. The goal of this section is to present the different categories of forecasting techniques that can be found in the literature and to briefly discuss their properties and domains of application.

2.3.1 *Forecasting method classification*

According to (Januschowski et al., 2020), several criteria, both objective and subjective, can be employed to classify forecasting methods (cf. Table 2.3).

Category	**Dimension**
Objective	Global vs. Local Methods
	Probabilistic vs. Point Forecasts
	Computational Complexity
	Linearity & Convexity
Subjective	Data-driven vs. Model-driven
	Ensemble vs. Single Models
	Discriminative vs. Generative
	Statistical Guarantees
	Explanatory/Interpretable vs. Predictive

Table 2.3: Criteria to classify forecasting methods, according to (Januschowski et al., 2020)

For our purposes, we will focus on the data versus model driven approach to present the different methods.

We will consider as model-driven approaches the methods that are characterized by a closed, analytic formulation of the forecasting technique relying on well-defined hypotheses on the statistical properties of the input data.

On the other hand, we will consider as data-driven approaches the methods that are based on pattern extraction and hypothesis formulation based on the input data, instead of a pre-defined analytical formulation.

Moreover, we will distinguish between univariate techniques (i. e., dealing with the prediction of a single time series, for a SISO or MISO forecasting task, cf. Figure 2.5) and multivariate technqiues (i. e., dealing with multiple time series at a time, for MIMO forecasting tasks).

It is worth noting that, without loss of generality, the considered forecasting models can be used within the *Recursive* and *Direct* strategies (Section 2.2.3.2) to produce multiple-step-ahead forecasts.

2.3.2 *Univariate - Model-driven*

This family of models assume that the historical values of the considered time series are either readily available or could be estimated (via imputation techniques, for instance). In other words $y_{t-\tau}$ is available for any $\tau > 0$. In the following, we will denote with y_t the true value of the considered time series at time t while $\hat{y}_t$ will indicate the forecast for the corresponding time step.

2.3.2.1 *Naive/Persistence/Random Walk*

The simplest benchmark model, assumes that the current forecast for the upcoming value of a time series corresponds to the latest available historical value.

$$\hat{y}_t = y_{t-1} \tag{2.30}$$

This benchmark can be found with different names in the literature: *random walk* (referring to the fact that it corresponds to the optimal forecasting strategy, if the analyzed time series follows a random walk data generating process (Hyndman and Athanasopoulos, 2018)), *persistence* (related to the fact that the assumption underlying the technique is that the past value will persist in the future) and *naive* (referring to its simplicity).

Despite its trivial nature, the naive method still is a strong baseline to be employed as a benchmark, often outperforming more complex techniques in highly dynamical settings. (Paldino et al., 2021).

2.3.2.2 *Average-based models*

The basic random model can be improved by extending the lookback period in the past, in order to better exploit the historical information at hand. All the considered methods propose different solutions to the tradeoff between the number of observations and the possibility to sample closer to t.

HISTORICAL AVERAGE (HA) The historical average method simply consists in forecasting the future time series value as the historical average of all the previous N available values.

$$\hat{y}_t = \frac{1}{N}\sum_{i=1}^{N} y_{t-i} \tag{2.31}$$

MOVING AVERAGE (MA) In order to limit the dependency on old historical values, the moving average model employs a sliding window of τ past observations. Anytime a new forecast for the time series is required, the oldest value is discarded from the average and substituted with the most recent value.

$$\hat{y}_t = \frac{1}{\tau}\sum_{i=1}^{\tau} y_{t-i} \tag{2.32}$$

2.3.2.3 *Exponential smoothing*

Exponential smoothing is a family of SISO forecasting methods, originally introduced in (Holt, 2004), in which the forecasts are computed as a weighted average of the past values with weights decaying exponentially with time. As opposed to standard (moving) average,

this non-uniform weighting scheme favors more recent observations over older ones. In simple exponential smoothing (SES), the forecasts are produced by:

$$\hat{y}_t = (1-\alpha)y_{t-1} + \alpha\hat{y}_{t-1} = \sum_{i=1}^{t} \alpha(1-\alpha)^{i-1} y_{t-i} \tag{2.33}$$

$$0 \leq \alpha \leq 1 \tag{2.34}$$

where y_t and $\hat{y}_t$ are the value and the forecast of the time series y, respectively. The basic method has been extended to include the historical trend of the time series as well as the presence of seasonality, in both an additive and multiplicative form, leading to the Holt-Winters and the Holt-Winters Damped techniques (Gardner Jr, 1985, 2006).

2.3.2.4 *Ensemble-based methods*

THETA The Theta method (Assimakopoulos and Nikolopoulos, 2000) is based on the combination of multiple one-step ahead SISO individual forecasters, called Theta-lines. Each Theta-line $y''_{t,\vartheta}$ is constructed by taking a second-order approximation of the original time series, with a specific coefficient ϑ. The final forecast is returned by averaging the forecasts produced by the different Theta-lines.

$$y''_{t,\vartheta_i} = \vartheta_i y''_t = \vartheta_i(y_t - 2y_{t-1} + y_{t-2}) \quad \text{with } 0 \leq \vartheta_i \leq 1, t \geq 2 \tag{2.35}$$

$$\hat{y}_t = \tfrac{1}{N_\vartheta} \textstyle\sum_i y''_{t,\vartheta_i} \tag{2.36}$$

The Theta ensemble is composed by two lines, with the ϑ_i coefficients being equal to 0 for the first line and 2 for the second. Despite its simplicity, this technique outperformed all the competitors in the M3 real-world forecasting competition (Hyndman, 2020) and has been selected as benchmark method for the M4 forecasting competition.

$$\hat{y}_t = \frac{1}{2}(y''_{t,0} + y''_{t,2}) \tag{2.37}$$

In (Hyndman and Billah, 2003), the authors demonstrated the equivalence of the Theta method to a specific form of the exponential smoothing format, in which the drift parameter is half the slope of a linear regression fitted to the data.

COMBINED Combined (also called Comb) is an ensemble method based on the combination of three SISO one-step-ahead exponential techniques (Gardner Jr, 1985): Single Exponential Smoothing for capturing the level, Holt-Winters to linearly extrapolate and Holt-Winters Damped to dampen the linear trend. The combination of the models is obtained by averaging the three outputs as follows:

$$\hat{y}_t = \frac{1}{3}(\hat{y}_{t,SES} + \hat{y}_{t,Holt} + \hat{y}_{t,DAMPED}) \tag{2.38}$$

For a detailed description of the methods, we refer the interested reader to the relevant reviews (Gardner Jr, 1985, 2006). Unlike Theta, Combined implements an ensemble of heterogeneous forecasting methods, each capturing a different characteristic of the original time series. Like Theta, Combined has been used as benchmark during the different M competitions (Makridakis, Spiliotis, and Assimakopoulos, 2020b).

2.3.2.5 *Simple Regression (SR-AR, SR-TAR, SR-ARMA)*

The simple regression model assumes the existence of a linear dependence of the future values of the time series on its past values. The unknown model parameters in this dependence are estimated from the data through the minimization of a loss function, in a linear regression fashion.

SR-AR In the case of a SR-AR, the time series is considered to be autoregressive (i.e. linearly dependent on its previous values). Besides the actual model parameters $\gamma_{i,t-1}$, the depth of the dependence on past value τ is another important structural parameter to be estimated.

$$\hat{y}_t = \sum_{i=1}^{\tau} \gamma_{i,t-1} y_{t-i} \tag{2.39}$$

SR-ARMA The SR-ARMA model, on the other hand, specifies an autoregressive dependence on the past values of the time series along with a linear dependence on the current and past values of a stochastic (imperfectly predictable) term (i.e. the forecasting error ε_{t-i}).

$$\hat{y}_t = \sum_{i=1}^{\tau_{AR}} \gamma_{i,t-1} y_{t-i} + \sum_{i=1}^{\tau_{MA}} \lambda_{i,t-1} \varepsilon_{t-i} \tag{2.40}$$

As it can be noted in equation 2.40, the autoregressive and moving average component are allowed to have different lag orders, respectively expressed by the parameters τ_{AR} and τ_{MA}.

2.3.3 *Univariate - Data-driven*

2.3.3.1 *Artificial Neural Networks*

In machine learning and cognitive science, an Artifical Neural Network (ANN) is a network of interconnected processing elements, called neurons, which are used to estimate or approximate functions that can depend on a large number of inputs that are generally unknown. The concept of artificial neural networks is inspired by the structure of the central nervous systems of animals, in particular the brain. In such biological neural networks, a set of cells (i.e. the neurons) are interconnected with each other to form a data-processing network. The processing capability of the network depends on the strength of the connections between the neurons, which can be dynamically modified across time in response to external stimuli that the network is subject to. This dynamic adaptation of the connections gives this system the possibility to learn from the experiences it is subject to.

Both artificial and neural networks are characterized by three features (MacKay, 2003) : *architecture*, *activity* and *learning rules*.

The *architecture* is a specification of which input variables are involved in the network and what are the topological relationships between the nodes of the network. The *activity rule* defines how the activities of the neurons change in response to each other (usually with a short time-scale dynamics). The *learning rule* specifies the way in which the neural network's weights needs to be adapted with time. This learning is usually viewed as taking place on a longer time scale than the time scale of the dynamics under the activity rule. Usually the learning rule will depend on the activities of the neurons. It may also depend on the values of the targets supplied by a teacher.

For our task, we will focus on a specific family of artificial neural networks, the multi-layer perceptron (MLP). The *architecture* of a multi-layer perceptron is organized in layers, with

each layer being fully connected to the following. The first layer, also called input layer, is composed of the input variables. After it, there are one or more intermediate layers, named hidden layers, yielding to an output layer with one output variable. In this type of network, information moves from the input nodes, through the hidden nodes, to the output node. Moreover, every connection between nodes has an associated weight.

In the following, we will focus on a standard one-hidden layer network (cf. Figure 2.17b), described by the equation :

$$m(\mathbf{x}) = f_o\left(b_o + \sum_{j=1}^{|H|} w_{jo} \cdot f_h\left(\sum_{i=1}^{|I|} w_{ij} y_{t-i} + b_j\right)\right) \tag{2.41}$$

w_{ij} is the weight of the connection between the i^{th} input node and the j^{th} hidden node, w_{jo} are the weights of the connections between hidden node j and the output node and $|H|$ is the number of hidden nodes. The number of hidden nodes ($|H|$) controls the complexity of the model. The *activation rule* of such network is divided into two steps. First, each node j determines its activation a_j, by collecting the output of its input nodes:

$$a_j = \sum_i w_{ij} x_i \tag{2.42}$$

Then, the activity of the neuron is computed as a function of the value of the activation a_j. In the case of the considered network, we have two different activity functions: $f_h(\cdot)$ for the hidden layer and $f_o(\cdot)$ for the output node. Common choices for activity functions are:

$$f(x) = x \qquad \text{Linear} \tag{2.43}$$

$$f(x) = \frac{1}{1+e^{-x}} \qquad \text{Logistic} \tag{2.44}$$

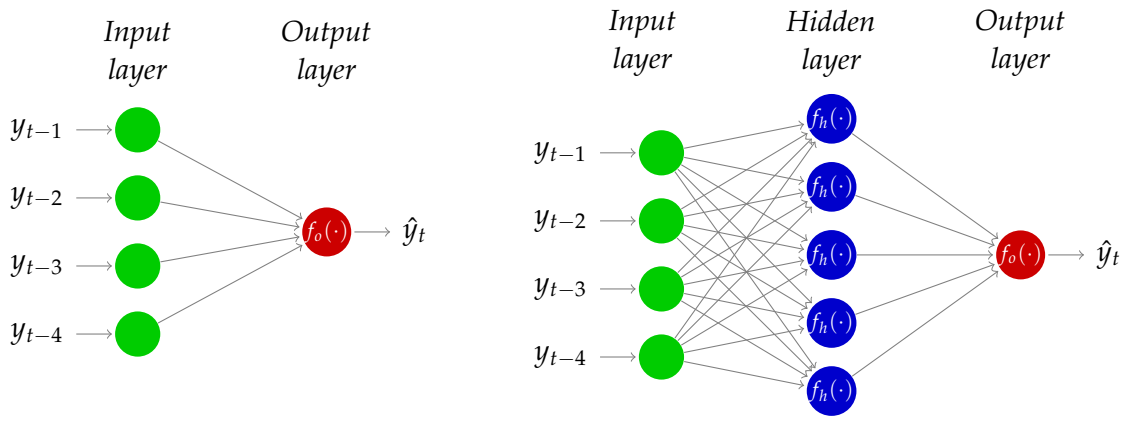
$$f(x) = \tanh(x) = \frac{e^{2x}-1}{e^{2x}+1} \qquad \text{Hyperbolic tangent} \tag{2.45}$$

$$f(x) = \begin{cases} 1 & x > 0 \\ -1 & x \leq 0 \end{cases} \qquad \text{Threshold} \tag{2.46}$$

Concerning the *learning rule*, the weights are generally estimated using some specific optimization procedure, the most popular one being the backpropagation procedure (Rumelhart, Hinton, and Williams, 1988). Usually, at the beginning, the weights are chosen to be random values near zero and the backpropagation procedure updates the weights in order to minimize the prediction errors. The backpropagation procedure could be done using all the available data in a single session (batch training) or providing the network one training example at a time (online training). The error function minimized by neural networks is nonconvex and so can have multiple local minima. In consequence, the final solution will depend on the value chosen as starting point. Because of this randomness, neural networks are often trained multiple times using different random starting values, and the outputs of the different networks are averaged to obtain the final predictions.

In the following, we will briefly discuss the three most relevant architecture for the problem of time series forecasting. The *activation rule* and *learning rule* are the same as those of the multilayer perceptron.

It is worth noting that multi-step-ahead forecasting can be performed with the same architecture by employing either the *Recursive* (i.e. using a single model, and re-injecting the one-step-ahead forecast as input value) or the *Direct* (that is, having multiple models, each one modeling a h-step-ahead dependency) strategies (Ben Taieb et al., 2012). Alternatively, a multioutput network (Zhang, Eddy Patuwo, and Y. Hu, 1998) with h output neurons one



$$\hat{y}_t = f_o\left(b_o + \sum_{i=1}^{|I|} w_{io} y_{t-i}\right) \tag{2.47}$$

$$\hat{y}_t = f_o\left(b_o + \sum_{j=1}^{|H|} w_{jo} \cdot f_h\left(\sum_{i=1}^{|I|} w_{ij} y_{t-i} + b_j\right)\right) \tag{2.48}$$

(a) Linear model (i.e. degenerate multi-layer perceptron architecture with zero hidden nodes) and corresponding output equation.

(b) Standard feed forward multi-layer perceptron architecture (one hidden layer) and corresponding output equation.

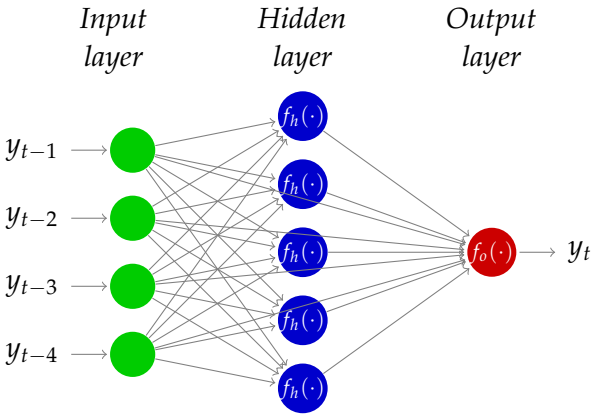
Figure 2.17: Comparison between the standard ANN architectures for time series for time series forecasting.

for each of the h-step ahead predictions to perform), can be employed to model a MIMO forecasting strategy. A multioutput network could be also used to perform one-step-ahead multivariate prediction (Chakraborty et al., 1992a).

LINEAR MODEL If the network does not have any hidden nodes (cf. Figure 2.17a), the function approximated by the neural network reduces to a non-linear function $f_o(\cdot)$ of the linear combination of the inputs of the network (i.e. the past values of the time series).

STANDARD FEED-FORWARD MODEL Figure 2.17b represents the standard architecture of a single-layer function approximator. In the case of a regression problem, the activity functions $f_h(\cdot)$ is a logistic function while the $f_o(\cdot)$ is a linear function. On the other hand, in the case of a classification problem $f_o(\cdot)$ is also a sigmoid function. Such architecture is generally considered to be a black-box model, in the sense that the model learned by the neural network does not have a direct interpretation.

FEED FORWARD MODEL WITH SKIP-LAYER CONNECTIONS If we extend the network of Figure 2.17b, allowing for skip-layer connections (i.e. direct connection between the input and output layers), we obtain an architecture with a direct interpretation. To be more precise, as shown in Equation 2.49, the model can be decomposed into a linear autoregressive component of order $|I|$ and a nonlinear component. In this case, both activity functions $f(\cdot)$ and $g(\cdot)$ are sigmoids (logistic or hyperbolic tangent) functions.



$$y_t = f_o\left(\underbrace{b_o + \sum_{i=1}^{|I|} w_{io} y_{t-i}}_{\text{Linear AR(|I|)}} + \underbrace{\sum_{j=1}^{|H|} w_{jo} \cdot f_h\left(\sum_{i=1}^{|I|} w_{ij} y_{t-i} + b_j\right)}_{\text{Non-linear component}}\right) \tag{2.49}$$

Figure 2.18: Feed forward multi-layer perceptron architecture allowing skip-layer connections and corresponding output equation.

2.3.3.2 *Lazy learning / k-Nearest Neighbors*

A Lazy Learning (LL) technique (Aha, 1997) delays the learning phase until the prediction time. In other words, these techniques perform a fit of the model only when a prediction is required, by using a computationally efficient technique (e.g. linear). This entails a considerable reduction in the computational cost of the model, while still preserving a good accuracy.

The lazy learning technique we will consider for time series forecasting is also referred as k Nearest Neighbors (k-NN) in the literature. A k-NN model is a local nonlinear model used for classification and regression. In the case of regression, the prediction for a given input vector $\mathbf{x}^*$ is obtained through local learning (Atkeson, Moore, and Schaal, 1997), a method that produces predictions by fitting a simple local model in the neighborhood of the point to be predicted. The neighborhood of a point is defined by taking the k values having the minimal values for a chosen distance metric defined on the space of the input vector (Altman, 1992).

To be more precise, every data point can be represented in the form $(\mathbf{x}, y)$ where $\mathbf{x}$ represents the vector of input values and y the corresponding output value. Then the prediction is computed as follows:

$$y(\mathbf{x}^*) = \frac{1}{k} \sum_{i \in k\text{NN}} y_{[i]}(\mathbf{x}) \tag{2.50}$$

where $y_i(\mathbf{x})$ is the output of the i^{th} nearest neighbor of the input vector $\mathbf{x}$ in the data set.

In the case of time series forecasting we will have for the one-step-ahead forecast for the point y_{t+1}, $(\mathbf{x}, y) = (\{y_{t-p+1}, \cdots, y_t\}, \hat{y}_{t+1})$.

Given a new input value $\mathbf{x}^*$, also called query point, the prediction is computed in three steps. First, the available points are ranked according to their distance to the query point $\mathbf{x}^*$. Then, the k closest points are selected and finally used for the computation of the prediction. Given these characteristics, the model is considered to be lazy (i.e. deferring the generation of the model to the moment when a new query is performed) and non-parametric, since the model has no parameters besides the meta-parameter k. The key parameter k has to

be selected with care since it is controlling the bias/variance trade-off of the estimates. A large k will lead to a smoother fit and therefore a lower variance (at the expense of a higher bias), and vice versa for a small k. Any metric can be used to compute the nearest neighbors of the input vector $\mathbf{x}^*$ but the most common choice is the standard Euclidean distance. Despite its simplicity, the k-NN model yields good results in practice and is often used as a benchmark against more complex models such as neural networks. Instead of computing a simple average as in Equation 2.51, we can also compute a weighted average with a weight inversely proportional to the distance from the point.

K-NN MULTIOUTPUT (MIMO) The k-NN method can be extended in order to produce multiple outputs within the same iteration of the algorithm. In this case, every data point is represented in the form $(\mathbf{x}, \mathbf{y})$ where $\mathbf{x}$ represents the vector of input values and $\mathbf{y}$ the corresponding output vector. Then the prediction is computed as follows:

$$\hat{\mathbf{y}}(\mathbf{x}^*) = \frac{1}{k} \sum_{i \in k\text{NN}} \mathbf{y}_i(\mathbf{x}) \tag{2.51}$$

where $\mathbf{y}_{[i]}(\mathbf{x})$ is the output vector of the i^{th} nearest neighbor of the input vector $\mathbf{x}$ in the data set. In the case of time series forecasting we will have for the H-step forecast for point y_{t+1}, $(\mathbf{x}, \mathbf{y}) = (\{y_{t-p+1}, \cdots, y_t\}, \{\hat{y}_{t+1}, \cdots, \hat{y}_{t+H}\})$.

OPTIMAL VALUE OF K - CROSS-VALIDATION BASED METHOD One of the main advantages of the k-NN method is its simplicity in terms of a reduced number of parameters. To be more precise, the only relevant parameter for such method is the size k of the neighborhood.

The optimal value of k can be estimated through a cross-validation procedure (Section 2.1.5.4). In the case of the leave-one-out cross validation, the testing set is constituted of a single value, while the training set contains all the remaining values. The procedure is then repeated until all the available values have been tested. For this case, the PRESS statistics a well-founded and computationally efficient way to perform cross-validation has been developed by (Allen, 1974).

The work of (Bontempi, 1999) extends the same concept to multiple step forecasts by computing an average error E_k across all the forecasting horizons H, as a function of the number of neighbors k.

$$E_k = \frac{1}{H} \sum_{h=1}^{H} e^h \tag{2.52}$$

At each horizon h, the leave-one-out error is computed according to:

$$e^h = \sum_{j=1}^{k} k \frac{y^h_{[j]}(\mathbf{x}) - \hat{y}^h_k(\mathbf{x})}{k-1} \tag{2.53}$$

where $y^h_{[j]}(\mathbf{x})$ corresponds to the h^{th} component of the output vector of the j^{th} closest neighbor of the query point x in the training set, while $\hat{y}^h_k(\mathbf{x})$ represents the h^{th} component of forecast output vector with k neighbors.

The optimal number of neighbors corresponds to the value k^* that minimizes the average error E_k.

$$k^* = \arg\min_k E_k \tag{2.54}$$

Due to its computational efficiency, this optimization strategy is often employed in practical applications, both for one-step-ahead forecasting and multiple-step-ahead with

both *Recursive* and *Direct* strategies (Section 2.2.3.2) as well as with a the aforementioned MIMO implementation.

Other strategies to determine the optimal number of neighbors k^* (based on autocorrelation functions) can be found in (Bontempi and Ben Taieb, 2011).

2.3.3.3 *Support Vector Regression*

Support Vector Regression (SVR) is a regression methodology based on the Support Vector Machine (SVM) theoretical framework (Cortes and Vapnik, 1995). The key idea behind SVR is that the regression model can be expressed using a subset of the input training examples, called the support vectors. In more formal terms, the model (Equation 2.55) is a linear combination over all the n_{SVM} support vectors of a bivariate kernel function $k(\cdot,\cdot)$ taking as inputs the data point x whose forecast is required and the i^{th} support vector $\mathbf{x}_i$. The coefficients α_i, α_i^* are determined through the minimization of an empirical risk function (Sapankevych and Sankar, 2009), solved as a continuous optimization problem.

$$\mathbf{y} = \sum_{i=1}^{n_{SVM}} (\alpha_i - \alpha_i^*)\, k(\mathbf{x}, \mathbf{x}_i) \tag{2.55}$$

$$k(\mathbf{x}, \mathbf{x}_i) = e^{\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2\gamma^2}} \tag{2.56}$$

Among the different available kernel functions we consider here the radial-basis one (Equation 2.56), for which the optimal value of the γ parameter is determined through grid search.

2.3.3.4 *Gradient boosting*

Boosting aims to create an accurate forecaster by combining several "weak learners" models (i.e. models characterized by a high bias and a low variance (Schapire, 1990)).

A boosted ensemble is constructed in a sequential manner, employing a weighting scheme of the samples of the dataset. The first model of the ensemble is defined as a simple average of the available samples $m^{[0]}(\mathbf{z}) = \frac{1}{N}\sum_{i=1}^{N} z_i$ with the weights for all the samples are initialized to the same values. Then, the model is updated via a linear combination, between the learner $l^{[j]}(\mathbf{z})$ estimated at iteration j and the model constructed at the previous iteration $m^{[j-1]}(\mathbf{z})$, weighted by the coefficient $\nu \in [0, 1]$:

$$m^{[j]}(\mathbf{z}) = m^{[j-1]}(\mathbf{z}) + \nu l^{[j]}(\mathbf{z}) \tag{2.57}$$

The weights associated with each sample are adapted in a way to increase the weights to those values that have been wrongly predicted. The process is repeated for the desired number of iterations (J in this case), and then the final prediction $\hat{z}_t$ is computed as a weighted sum of the different learners.

$$\hat{z}_t = m^{[J]}(\mathbf{z}) = m^{[0]}(\mathbf{z}) + \sum_{j=0}^{J} \nu l^{[j]}(\mathbf{z}), \tag{2.58}$$

The gradient aspect of a Gradient Boosting Machine (GBM) method is related to the fact that the sample weight update procedure is performed via a minimization procedure of a given error metric, performed via gradient descent. Moreover, when the chosen error metric

is the mean squared error (also called quadratic loss), the gradient boosting procedure is equivalent to training each subsequent model in the ensemble on the residuals of the previous model. For more details concerning the inner workings for the model, we refer the interested reader to (Taieb, 2014).

LightGBM (Ke et al., 2017) is an example of gradient-boosted based algorithm specifically optimized to deal with a large number of data instances and a large number of features, implemented with both a *Direct* and *Recursive* strategy.

2.3.3.5 *Random Forest*

An additional application of the boosting approach can be seen in the Random Forest (RF) family of models, where the considered base learner is a decision tree.

A decision tree is a model that partitions the input space into mutually exclusive regions, based on the values of the different features. In case of a regression tree, after the partitioning, each region will have a dedicated local regression model (Breiman et al., 2017).

According to (Hastie, Tibshirani, and Friedman, 2009), a decision tree is a good weak learner due to its combination of an high bias in forecasting and advantageous properties such as: invariance with respect to feature scaling and various other transformations, robustness to inclusion of irrelevant features, and intrepretable nature.

In order to improve the forecasting performances of the decision tree, the authors of (Breiman, 2001a) proposed to perform bagging (a portmanteau of boosting and aggregating), by training several different decision trees via bootstrapping (i. e., training each model on a different subset, uniformly sampled with replacement from the original dataset) and then combining their predictions.

The success of this approach among practitioners can be explained by a combination of its effectiveness on several regression tasks (Segal, 2004) and its interpretability, stemming from both its tree-based nature and the presence of informative metrics about the selected features, such as Variable Importance (Grömping, 2009).

2.3.4 *Multivariate - Model-driven*

The following section will discuss the traditional, statistical-based techniques for multivariate and multi-step ahead time series forecasting. We will start by introducing two methods, Naive and Average, which are often employed as baseline methods for comparison against more advanced techniques, to move further towards methods explicitly modelling the autoregressive dependence between the time steps (VAR) and the noise (VARMA).

2.3.4.1 *Naive*

The *Naive* method simply consist in setting the forecast of the future h values for all the k considered series, as the last available value, that is $y_{j,t-1}\ \forall j \in \{1, \cdots, k\}$, that is:

$$\begin{bmatrix} \hat{y}_{1,t+h} & \hat{y}_{1,t+h-1} & \cdots & \hat{y}_{1,t} \\ \hat{y}_{2,t+h} & \hat{y}_{2,t+h-1} & \cdots & \hat{y}_{2,t} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{y}_{n,t+h} & \hat{y}_{n,t+h-1} & \cdots & \hat{y}_{n,t} \end{bmatrix} = \begin{bmatrix} y_{1,t-1} & y_{1,t-1} & \cdots & y_{1,t-1} \\ y_{2,t-1} & y_{2,t-1} & \cdots & y_{2,t-1} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n,t-1} & y_{n,t-1} & \cdots & y_{n,t-1} \end{bmatrix} \tag{2.59}$$

Or, in a more compact form:

$$\bar{\mathbf{Y}}_{t-1} = \begin{bmatrix} y_{1,t-1} \\ y_{2,t-1} \\ \cdots \\ y_{n,t-1} \end{bmatrix} \tag{2.60}$$

$$\hat{\mathbf{Y}}_{(n\times h)} = \begin{bmatrix} \bar{\mathbf{Y}}_{t-1} & \bar{\mathbf{Y}}_{t-1} & \cdots & \bar{\mathbf{Y}}_{t-1} \end{bmatrix} \tag{2.61}$$

The *Naive* approach can be easily extended in other to account for the presence of a seasonal effect in the data by using as a forecast the most recent observation from the corresponding season:

$$\bar{\mathbf{Y}}_{t-L+[(h-1)modL]+1} = \begin{bmatrix} y_{1,t-L+[(h-1)modL]+1} \\ y_{2,t-L+[(h-1)modL]+1} \\ \cdots \\ y_{k,t-L+[(h-1)modL]+1} \end{bmatrix} \tag{2.62}$$

$$\hat{\mathbf{Y}}_{(n\times h)} = \begin{bmatrix} \mathbf{Y}_{t-L+[(h-1)modL]+1} & \mathbf{Y}_{t-L+[(h-1)modL]+1} & \cdots & \mathbf{Y}_{t-L+[(h-1)modL]+1} \end{bmatrix} \tag{2.63}$$

2.3.4.2 *Average*

The average method can be constructed with a similar approach as the Naive one, this time replacing the last value available value with an average of the previous T_a values yielding to:

$$\bar{\mathbf{Y}}_t = \frac{1}{T_a} \begin{bmatrix} \sum_{i=1}^{T_a} y_{1,t-i} \\ \sum_{i=1}^{T_a} y_{2,t-i} \\ \vdots \\ \sum_{i=1}^{T_a} y_{n,t-i} \end{bmatrix} \tag{2.64}$$

$$\hat{\mathbf{Y}}_{(n\times h)} = \begin{bmatrix} \bar{\mathbf{Y}}_t & \bar{\mathbf{Y}}_t & \cdots & \bar{\mathbf{Y}}_t \end{bmatrix} \tag{2.65}$$

2.3.4.3 *VAR(p)*

Another traditional mathematical model that has been proven successful in modelling linear dependencies among multiple univariate is the VAR (Tuarob et al., 2017). Analogously to its univariate counterpart (Section 2.3.2.5), the current value of the multivariate time series Y_t is expressed as a linear combination of the past values Y_{t-i} of the series with time invariant coefficient matrices A_i plus an error vector term $\mathbf{e}_t$ with zero mean and diagonal covariance matrix. The number m of considered autoregressive dependencies is also called order of the VAR model. VAR and state space models have been shown to be equivalent and their equivalence is discussed in (Gilbert., 1993). The model is represented here in a compact form:

$$\hat{\mathbf{Y}}_t = \mathbf{c} + \sum_{i=1}^{p} \mathbf{A_i} \mathbf{Y}_{t-i} + \mathbf{e}_t \tag{2.66}$$

And here in the corresponding matrix form:

$$\begin{bmatrix} y_{1,t} \\ y_{2,t} \\ \vdots \\ y_{n,t} \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} + \begin{bmatrix} a^1_{1,1} & a^1_{1,2} & \cdots & a^1_{1,n} \\ a^1_{2,1} & a^1_{2,2} & \cdots & a^1_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a^1_{n,1} & a^1_{n,2} & \cdots & a^1_{n,n} \end{bmatrix} \begin{bmatrix} y_{1,t-1} \\ y_{2,t-1} \\ \vdots \\ y_{n,t-1} \end{bmatrix} + \cdots + \begin{bmatrix} a^m_{1,1} & a^m_{1,2} & \cdots & a^m_{1,n} \\ a^m_{2,1} & a^m_{2,2} & \cdots & a^m_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a^m_{n,1} & a^m_{n,2} & \cdots & a^m_{n,n} \end{bmatrix} \begin{bmatrix} y_{1,t-m} \\ y_{2,t-m} \\ \vdots \\ y_{n,t-m} \end{bmatrix} + \begin{bmatrix} e_{1,t} \\ e_{2,t} \\ \vdots \\ e_{n,t} \end{bmatrix} \tag{2.67}$$

In order to use the model to perform multivariate forecast, one has to perform the estimation of all the the parameter matrices. Several methods exist in the literature, often stemming from the multivariate adaptation of well known univariate techniques (i. e., Ordinary least squares, Yule-Walker equations and Maximum likelihood estimation, (Lütkepohl, 2005)) Here, we will here focus on the multivariate ordinary least squares method, which will also allow a multiple step ahead forecast with a single optimization. Equation 2.66 can be rewritten in the following regression form:

$$\hat{\mathbf{Y}} = \mathbf{BZ} + \mathbf{U} \tag{2.68}$$

where:

$$\hat{\mathbf{Y}} = \begin{bmatrix} \hat{\mathbf{Y}}_t & \hat{\mathbf{Y}}_{t+1} & \cdots & \hat{\mathbf{Y}}_{t+p} \end{bmatrix} \tag{2.69}$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{c} & \mathbf{A}_1 & \mathbf{A}_2 & \cdots & \mathbf{A}_m \end{bmatrix} \tag{2.70}$$

$$\mathbf{U} = \begin{bmatrix} \mathbf{e}_t & \mathbf{e}_{t+1} & \cdots & \mathbf{e}_{t+m} \end{bmatrix} \tag{2.71}$$

$$\mathbf{Z} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \mathbf{Y}_{t-1} & \mathbf{Y}_t & \cdots & \mathbf{Y}_{t+m-1} \\ \mathbf{Y}_{t-2} & \mathbf{Y}_{t-1} & \cdots & \mathbf{Y}_{t+m-2} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{Y}_0 & \mathbf{Y}_1 & \cdots & \mathbf{Y}_t \end{bmatrix} \tag{2.72}$$

VAR models have some well known limitations: their linear nature doesn't allow to capture nonlinear dependencies among the variables, and, for a correct result of the estimation process, data must meet some conventional requirements (e.g. null mean and stationarity). Moreover, the number of parameters to estimate is quadratic in the number of multivariate series and linear in the model order (i.e. $O(n^2m)$, m matrices $\mathbf{A}_i$ of size $n \times n$). This number of parameters can be handled in the case of small problems which involve only a moderate number of variables (i.e. n smaller than 20).

The authors of (De Gooijer and Hyndman, 2006) provide a nice summary of the historical evolution of VAR models, its strengths and limitations. Among the presented applications, several different domains are discussed: (Downs and Rocke, 1983) for yearly municipal budget data, (Edlund and Karlsson, 1993) for quarterly unemployment rate data, (Heuts and Bronckers, 1988) for monthly truck sales data, (Hillmer, Larcker, and Schroeder, 1983) for monthly accounting data and (Lin, 1989) monthly hospital patient movements. It should be noted that the common characteristics of the different applications are the reduced dimensionality of the forecasting problem $n < 10$ and the reduced number of available samples $N < 10^2$.

REGULARIZED VAR In order to deal with the increase in the number of variables, optimized version of a VAR(m) model have been developed. Among the different variants, we cite here the work of (Messner and Pinson, 2019), where regularization on the number of free parameters to estimated is performed via LASSO regularization (Tibshirani, 1996). In addition, the proposed variant allows for an incremental update of the model parameters further reducing the computational cost required to fit the model.

2.3.4.4 *Partial least squares*

The use of linear combinations for reducing the dimensionality of a multivariate time series is not restricted to static PCA (Section 2.4.1). In (Box and Tiao, 1977) the combination aims to maximize the predictability of the series. This is obtained by canonical correlation between Y_{t+1} and its lags $Y_t, \ldots, Y_{t-m}$, i.e. finding the coefficients that maximize the correlation between the present and the past. Similar approaches rely on reduced rank models.

The extension of this principle to multi-step-ahead forecasting led to the use of partial least squares (Franses and Legerstee, 2010). PLS allows the joint forecasting of the H steps ahead of the multivariate time series on the basis of the lagged vectors $Y_t, \ldots, Y_{t-m}$. This is a multi-input multi-output regression task where the number of inputs amounts to nm and the number of outputs to Hn respectively. The benefit of PLS is that it allows at the same time a dimensionality reduction of the inputs and a joint prediction of the outputs, taking then into consideration the dependency between the future steps.

2.3.5 *Multivariate - Data-driven*

By looking to the scientific literature one can observe that two categories of models consistently appear among the most used approaches for multivariate forecasting. On one side, we can find closed-form linear models (such as the VAR family, presented in Section 2.3.4.3), whose applicability is limited to problems with a moderate number of variables (i.e. n smaller than 20), since their model structure, albeit interpretable, encounters numerical estimation problems as the number of variables increase, hindering their applicability. On the other side, to cope with the increase of dimensionality, large scale representation-based models (such as RNN and CNN deep network) have been developed. Instead of having a well-defined mathematical model of the dependencies among the variables, this category of techniques is able to learn its own internal representation of the dependencies from the data. Thanks to their modular architecture, this family of models can be easily adapted to scale up to forecast a number of series $n > 10^2$ with a large number of available samples $N > 10^3$. This scaling up can still ensure good forecasting performance, but the increase of the size of the problem, combined with the complex structure of the network greatly limits the interpretability of this family of models, (hence their classification as black-box models). Despite their lack of robustness in highly dynamic settings, as well as the need for specialized architecture for efficient computational performance and extensive fine tuning (Guo et al., 2016), representation-based techniques have become increasingly popular in the scientific literature. In the following sections we will give a brief overview of the most relevant representation-based techniques for multivariate time series forecasting.

2.3.5.1 *Deep Learning*

Deep feed-forward neural networks have been developed in order to overcome the limitations of single-layer neural networks, namely their inability to learn approximate more complex dependencies (such as spatio-temporal dependencies). The architecture of a deep feed-forward neural network is closely related to those presented in Section 2.3.3.1, with the

only differences being in the number of hidden layers, and potentially in the choices of the activation functions for the hidden layers. The depth of the network is given by the number of hidden layers L. For instance, the model for a network with L layers, each layer having M_l, $l \in \{1, \cdots, L\}$ neurons, can be summarized as follows:

For the input layer:

$$a_i^1 = \sum_{j=1}^{M_1} w_{ij} y_{t-j} + b_i^1 \tag{2.73}$$

$$f_i^1 = h(a_i^1) \tag{2.74}$$

For the $L-1$ intermediate layers:

$$f_i^k = h\left(\sum_{j=1}^{M_k} w_{ij}^k f_j^{k-1} + b_i^k\right) \tag{2.75}$$

For the output layer:

$$y_{t+1} = h\left(\sum_{j=1}^{M_L} w_{ij}^L f_j^{L-1} + b_i^L\right) \tag{2.76}$$

where a_i^k represents the activation for neuron $i \in \{1, \cdots, M_k\}$ in layer k and $h(\cdot)$ is the nonlinear, differentiable activation function (i.e. the *activity rule*). $\mathbf{W}^k \in \mathbb{R}^{M_{k-1} \times M_k}$ represents the weight matrix for layer l, while $\mathbf{b}^k \in \mathbb{R}^{M_k \times 1}$ represent the bias vector. The empirical assessment of these architectures on the time series domain (Lara-Benítez, Carranza-García, and Riquelme, 2021) has shown that despite the increase in predictive power given by the additional hidden layers, and an extensive grid search of the optimal configuration of the network, the forecasting performances of deep feed-forward neural networks still remains far from optimal. For these reasons alternative architectures such as CNN and RNN have been analyzed to exploit the spatio-temporal dependencies in the multivariate time series data.

CONVOLUTIONAL NEURAL NETWORKS In order to account for local temporal dependencies among the input values, convolutional neural networks can be used as an alternative architecture to perform time series forecasting. A convolution between a signal f and a filter g is theoretically defined as:

$$(f * g)(t) = \sum_{j=-\infty}^{\infty} f(j) g(t-j) \tag{2.77}$$

In practice, a convolution at point t is computed by shifting the filter g over the signal f along the j axis and computing the weighted sum of the two. Repeating this operation for every available t generates a third signal $(f * g)$ which represents the impact of the filter g on the original signal f.

In the context of CNN, the filter g is not pre-defined but rather learned from the available data by minimizing a certain metric (namely the forecasting error). Once the filter is learned, it can be used to extract translation-invariant features from the input f. The intuition behind a convolutional neural network is thus to learn in each layer a different filter and to stack several layers together to combine the different low-level features extracted in antecedent layers in a more powerful feature extractor.

In the domain of time series, convolution can be applied in both a one-dimensional and two-dimensional fashion. One-dimensional convolution tries to extract features either in the temporal dimension (i. e., within a single time series, over time) or in the spatial dimension (i. e., across different time series at a fixed time). Conversely, two-dimensional convolution works on both dimensions at the same time. Moreover, a dilated convolution (also called temporal convolution) (Yu and Koltun, 2015) (i. e., including in the convolution points that are a distance d from the current input) can be also used to model multi-scale temporal dependencies.

In (Borovykh, Bohte, and Oosterlee, 2017), CNN are used for univariate, recurrent multiple-step-ahead forecasting both unconditional, and conditional on a second time series. In order to avoid degradation problems in training the network, the authors propose to perform residual learning (that is, training the network to approximate the error function, rather than the target value).

On the other hand, (Sezer and Ozbayoglu, 2018) and (Liu, Hou, and Liu, 2017) propose a time series to image conversion approach, in order to be able to reuse state-of-the-art CNN architecture for image recognition/prediction tasks, for financial forecasting.

In (Sezer and Ozbayoglu, 2018), images are created by plotting the values of 15 standard technical indicators over 15 time steps. These values are then fed into the network in order to output the most likely investment position (e.g. buy, hold, sell), and the quality of the prediction is evaluated by computing the returns yielded by employing the forecast strategy.

In (Liu, Hou, and Liu, 2017), the image is created by stacking the stocks data in a 2D format such that on one axis represents the hourly evolution of a stock, while the other represent the daily evolution of the stock over a period of 24 days. The network is than used to perform a univariate, recurrent multiple step ahead forecasting of a given stock. Finally, the recent work by (Liu et al., 2021) proposed a dedicated architecture (SCINet), derived from the aforementioned temporal convolution architecture (Bai, Kolter, and Koltun, 2018), working natively with numerical data, without dedicated spatio-temporal modeling components, achieving state-of-the-art performances on multivariate problems.

RECURRENT NEURAL NETWORKS An alternative approach to better model dynamic temporal dependencies among the input data is the introduction of recurrent connections in the hidden layers of a standard feedforward network, creating a recurrent neural network. In their simple form (also known as simple RNN or Elman RNN) (Graves, 2012; Lipton, Berkowitz, and Elkan, 2015), the recurrent connections come from a *hidden state* H_t, which is also used for predicting future values Y_t:

$$\mathbf{H}_t = \sigma(\mathbf{W}_{HY}\mathbf{Y}_{t-1} + \mathbf{W}_{HH}\mathbf{H}_{t-1} + \mathbf{B}_H), \tag{2.78}$$

$$\mathbf{Y}_t = \mathbf{W}_{YH}\mathbf{H}_t + \mathbf{B}_Y \tag{2.79}$$

The matrices $\mathbf{W}_{HY}$, $\mathbf{W}_{HH}$, $\mathbf{W}_{YH}$, $\mathbf{B}_H$ and $\mathbf{B}_Y$ are the learnable parameters (weights and biases) of the network. Given the recurrent nature of the network, the network cannot be trained with the traditional backpropagation algorithm, but requires a specific adaptation of the algorithm, called backpropagation through time (Graves, 2012), that basically performs an unfolding of the recurrent structure through time (Figure 2.19). A sigmoid activation function σ allows the modelling of nonlinear dependencies, while the recurrent connections allow the modelling of long-term temporal dependencies.

A recurring problem in deep neural networks is the vanishing/exploding gradient (Hochreiter et al., 2011). This problem is due to the limited domain of the activation functions employed in the network, in combination with the derivative chain rule employed in backpropagation, which yields to longer chains of multiplications of small numbers, as the number of layers in the network increase. To solve this problem, several approaches

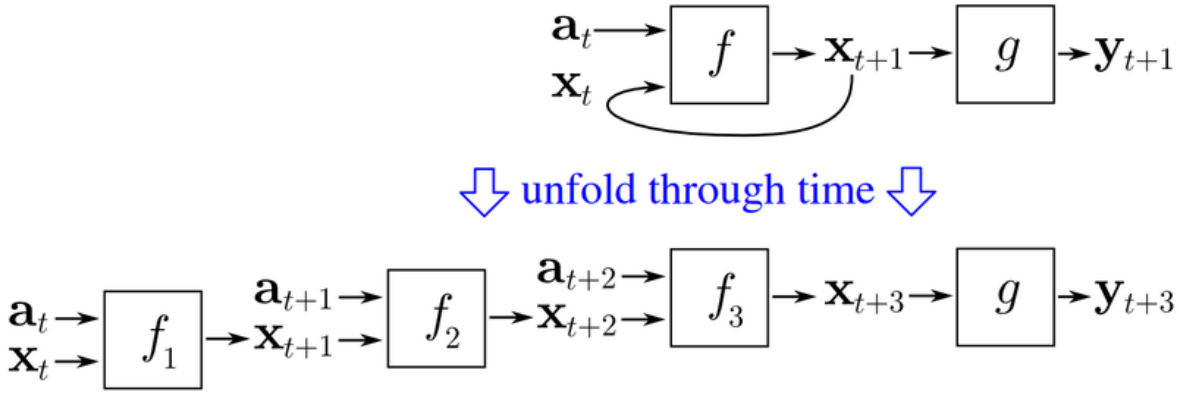
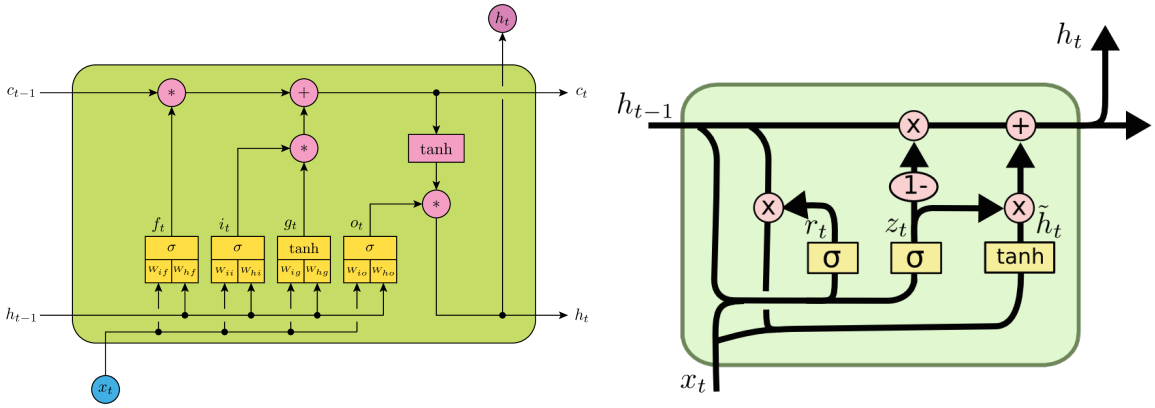


Figure 2.19: Unfolding of a recurrent neural network through time. Public domain schema by Olexa Riznyk.

have been proposed in the literature, but we will focus on those that have been empirically proven to be more effective: LSTM and GRU (Figure 2.20).



(a) Long Short Term Memory (LSTM), by Andre Holzner (b) Gated Recurrent Unit (GRU) by Cristopher Olah

Figure 2.20: Comparison between the LSTM and GRU cells internal architectures.

LSTM The key idea of LSTM networks (Hochreiter and Schmidhuber, 1997) is to augment the neuron of a simple RNN with a time varying internal state $\mathbf{c}_t$, and to have a 3 step mechanism for updating such state (Figure 2.20a). First, the previous output of the neuron $\mathbf{h}_{t-1}$ is combined with the new input values $\mathbf{y}_t$ and fed in both an input gate, to determine the new candidate values $\mathbf{i}_t$ for the update of the internal state and to determine the activation of the forgetting gate $\mathbf{f}_t$, indicating with values should be removed from the internal state. Then, the two values are combined to update the internal state. Finally, the updated state $\mathbf{c}_t$ is combined with the output activation vector $\mathbf{o}_t$ to produce the new neuron output $\mathbf{h}_t$.

LSTM networks are often employed in multivariate and multistep ahead forecasting within an hybrid architecture, in combination with a linear model. For instance, in (Lai et al., 2017), LSTM networks with skip-layer connections are employed as non-linear component in an architecture where the linear forecast is produced through a simple autoregressive model. (Goel, Melnyk, and Banerjee, 2017) also employs a similar approach, where first a linear model is fitted to the available data, and then an LSTM network is fitted to the residual of the first regression.

GRU A GRU (Cho et al., 2014b) is another type of improved recurrent cell, without any explicitly modeled internal state as LSTM, but with a similar multi-step forgetting

mechanism, here the update gate vector $\mathbf{z}_t$ and the reset gate vector are computed in a similar fashion, both depending on input data $\mathbf{y}_t$ and previous cell output $\mathbf{h}_t$, each one with specific weight matrices (Figure 2.20b). Then, the internal state $\mathbf{h}'_t$ is computed by combining the current input and the application of the reset gate to the previous output. Finally the new output is produced $\mathbf{h}_t$ by combining the previous output and the current internal state $\mathbf{h}'_t$.

GRU networks have been directly employed in (Zhang, Wu, and Chang, 2018) for a multivariate and multiscale (short term, long tem and cycle), one-step-ahead forecasting of electricity load forecasting in a two layer architecture. In (Chang et al., 2018), GRUs are employed as non-linear forecasting component in a more complex architecture, where a convolutional network is used as a feature extractor (encoder) for the non-linear component, whose prediction is then combined with the linear forecast provided by traditional ARMA model for both univariate and multivariate multiple step ahead forecasting.

RELATED LITERATURE For an overview of the most effective techniques, the authors of (Hewamalage, Bergmeir, and Bandara, 2021) propose an extensive study RNN testing different data preprocessing methods, hyperparameter configurations, architectures, type recurrent units and optimizers, comparing across datasets with diverse characteristics, concluding that LSTM cells, in combination with seasonal decomposition of the input is a competitive model configuration compared to state-of-the-art techniques. In addition, the authors of (Bianchi et al., 2017) provide an additional empirical study, comparing the performance of RNN with different cell types against non-recurrent architectures such as Nonlinear AutoRegressive with eXogenous inputs or Echo State Networks. In their empirical assessment, on highly dynamic time series, they confirmed the effectiveness of gated units such as LSTM and GRU, although they advise the use of non-gated units for series presenting long-term dependencies with smoother dynamics.

Last but not least, the work of (Lara-Benítez, Carranza-García, and Riquelme, 2021) provides a complementary extensive empirical evaluation, including feed-forward, recurrent and convolutional network on several real datasets. Their conclusions are aligned with the other empirical assessments, confirming the outperformance of LSTM cells over different types of gated cells as well as different architectures (e. g., CNN or temporal convolution networks).

TRANSFORMERS Transformers are a specific neural network architecture, initially developed for natural language sequence prediction (Vaswani et al., 2017), based on the encoder-decoder principle: the encoder part of the network tries to compress the informative content of the input data into a latent representation, which is then used by the decoder to forecast the future data. In addition, a specific algorithm (i. e., Scaled Dot-Product Attention) is employed to direct the attention of the network to the most relevant elements of the input sequence to learn (in this case, the time series). Transformers have shown their abilities to have state-of-the-art performances on multivariate and long-term forecasting (Eisenach, Patel, and Madeka, 2020), as well as on spatio-temporal forecasting tasks (Grigsby, Wang, and Qi, 2021). However, as shown by (Lara-Benítez et al., 2021), although their performances are comparable with respect to other neural techniques (such as CNN and RNN), their computational cost for both model optimization and inference is generally higher.

2.3.5.2 *Multi-task learning*

Multitask Learning (MTL) is defined as "an inductive transfer mechanism whose principle goal is to improve generalization performance, by leveraging the domain-specific information contained in the training signals of related tasks" (Caruana, 1997). In this paper, the author

proposes a methodology to transform a set of independent neural networks trained for single task learning, into a a single network able to learning the multiple related tasks in parallel while using a shared representation (i.e. a common hidden layer). The rationale behind this idea is that the shared representation should contain domain specific information which should be common to all the tasks, Through an empirical evaluation on both classification and regression tasks, the author proceed to show the effectiveness of the approach. The paper is concluded by an extension of the methodology to K-nearest neighbors and kernel regression. More recently, (Ruder, 2017) contains an extension of Caruana's work to deep neural network, highlighting different deep architectures for multi-task forecasting. While the previous articles focus on parallel multi-task learning, (Laptev, Yu, and Rajagopal, 2018) can be seen an example of sequential multi-task learning or transfer learning, where first a general model is trained for univariate, multistep ahead time series forecasting on a given data subset, then up to n layers of the network are frozen and the same network is reused for a similar forecasting task on a different dataset, showing an improvement in forecasting accuracy especially with a reduced amount of data.

2.4 DIMENSIONALITY REDUCTION

If we consider the aforementioned learning process, the number of input features for the learning algorithms may be extremely high due to several reasons, ranging from the simple multivariate nature of the problem to the number of newly generated features through feature selection. However, a large number of features in learning may negatively affect generalization performance, especially in the presence of irrelevant or redundant features.

For these reasons, dimensionality reduction is a crucial technique in improving the learning performance of the considered forecasting techniques.

Dimensionality reduction techniques transform the considered dataset $\mathbf{Y}$ with dimensionality n into a new dataset $\mathbf{Z}$ with dimensionality $q < n$, while trying to preserve the geometry of the original data as much as possible. In practice, neither the geometry of the data manifold, nor the real intrinsic dimensionality of the original dataset $\mathbf{X}$ are known. For these reasons, dimensionality reduction is often approached as an optimization problem, aiming to determine the optimal dimensionality q and the best low-dimensional representation, according to a defined metric. Several techniques for dimensionality reduction have been proposed in the literature throughout the years, see (Cunningham and Ghahramani, 2015) for a specific focus on linear techniques and (Van Der Maaten, Postma, and Herik, 2009) for an extensive review on both linear and non-linear techniques.

Regardless of the type of approach for dimensionality reduction, a common hypothesis is to assume the independence among the samples composing the original dataset $\mathbf{Y}$. Although this hypothesis might seem counterproductive for time-dependent data, some of these techniques (such as PCA, from the linear domain and autoencoder, from the non-linear domain) have been successfully employed in the time series domain, and will be discussed and employed throughout this thesis (addressed in Sections 2.4.1 and 2.4.2, respectively). In addition, we will also consider non-linear techniques specifically tailored to temporally dependent data, such as recurrent auto-encoder (Section 2.4.3).

Concerning PCA (Section 2.4.1), two approaches for the estimation of principal components will be discussed: the *batch* approach which uses all the available data at once, and the *incremental* approach, which performs sequential updates of the estimation.

Regarding autoencoder-based techniques (Sections 2.4.2 and 2.4.3), it is worth mentioning that, even though it will not be discussed in the thesis, it is possible to easily implement an incremental approach given the possibility to easily retrain the underlying neural networks.

2.4.1 *PCA and time series*

PCA performs dimensional reduction in multivariate statistical analysis by doing orthogonal rotations of the observed coordinates. PCA transforms the n original variables $Y[1], \ldots, Y[n]$ into q new variables $Z[1], \ldots, Z[q]$, called *principal components*, such that the new variables are uncorrelated with each other and account for decreasing portions of the variance of the original variables. The q principal components (Equation 2.80 are defined as weighted sums of the elements of Y with maximal variance, under the constraints that the weights are normalized and the principal components are uncorrelated with each other.

$$Z[p] = \sum_{j=1}^{n} w_{jp} Y[j], \qquad p = 1, \ldots, q \tag{2.80}$$

It is well-known from basic linear algebra that the solution to the PCA problem is given in terms of the unit-length eigenvectors of the correlation matrix of Y. Let us order the eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ and the corresponding eigenvector in the matrix W of size $n \times q$. Given the multivariate time series matrix Y, $Z[p] = YW[, p]$ represents the projection of the series on the pth principal component.

Though PCA has been developed originally for independent Gaussian observations, it has been found to be useful also in time series, the most common type of non-independent data. In (Tsay, 2014) PCA is applied directly to a four dimensional economic time series as well as to the residuals of fitted VAR models to show that it is able to find some stable relationships between variables. In (Jolliffe, 2002) it is explicitly stated that when the objective of PCA is not inferential, non-independence of data should not prevent from using PCA. The main difference is that, while in conventional PCA covariance is computed between variables measured at the same time, in time series it is possible to compute also covariances to model dependencies between variables at different times. A nice example of use of PCA for summarizing multivariate time series is in (Papadimitriou, Sun, and Faloutsos, 2005). Note also that in empirical sciences PCA can be found with alternative denominations, like EOF (empirical orthogonal function), SSA and MSSA (singular spectrum analysis for univariate and multivariate case, respectively), or Karhunen-Loeve method (more for continuous time series).

In SSA and MSSA (Golyandina, Nekrutkin, and Zhigljavsky, 2001), the series is decomposed in a weighted sum of decorrelated components, each being a moving average of the original time series. This allows to discover dominant periodicities (SSA) or oscillatory spatial patterns. SSA has also been used for forecasting: in this case the approach, after the decomposition step, creates a reconstructed series based on a subset of components. This series is then forecast by using a one-step-ahead linear approach (Linear Recurrent Formula).

2.4.1.1 *Batch and incremental PCA*

The conventional approach to PCA computation requires a sequence of matrix operations whose computational complexity is a function of the number of data rows N, the number of variables n and the desired number of components q, (i.e. $\mathcal{O}(N(n + n^2 + nq))$). In an online setting, where the principal components should be recomputed from scratch at each observation step, this approach is not affordable.

For this reason, observation-incremental approaches have been proposed in (Arora et al., 2012; Hegde et al., 2006; Oja, 1992; Sanger, 1989; Weng, Zhang, and Hwang, 2003). These approaches can be classified in two main categories: stochastic optimization and neural network based.

Stochastic optimization approaches tackle the PCA estimation problem as a continuous optimization problem, whose objective is to maximize the variance explained by the principal components (with additional orthogonality constraints among the components). These approaches derive the equations for the incremental update of the components, either as a gradient ascent step (Arora et al., 2012) or as a matrix perturbation (Hegde et al., 2006). The method by Weng et al. (Weng, Zhang, and Hwang, 2003) derives a similar update rule, while additionally proving the statistical efficiency of the estimation.

Neural-network-based approaches use a specifically designed single layer neural network, together with a weight optimization algorithm, in order to estimate the principal components of the input dataset. These methods allow to produce the eigenvalues of the covariance matrix as output of the network, while the network weights converge to the eigenvectors. (Oja, 1992) and (Sanger, 1989) propose two different variations on the update rules of the network weights, with the second one introducing an asymmetry in the weights update to improve the convergence rate.

In addition, there exists also component incremental PCA methods (e.g. NIPALS-PCA and GS-PCA), where the estimation of the principal components is made one factor at a time, (Andrecut, 2009) and iterative stochastic block-incremental method (Mitliagkas, Caramanis, and Jain, 2013), which performs a block decomposition of the original data matrix followed by an estimation of the principal components.

2.4.2 *Feed-forward Autoencoders*

Feed-forward autoencoders are a specific category of neural networks trained to learn identity mapping from inputs to outputs (Vincent et al., 2010). Their architecture is characterized by having an input and output layer with the same number of nodes (corresponding to the number of original dimensions n), and by the composition of two symmetrical sub-networks: an *encoder*

$$\mathbf{Z}_t = f_\theta(\mathbf{Y}_t) \tag{2.81}$$

that transforms n-dimensional inputs $\mathbf{Y}_t$ into some latent (encoded) q-dimensional representation $\mathbf{Z}_t$, and a *decoder*

$$\hat{\mathbf{Y}}_t = g_{\theta'}(\mathbf{Z}_t) \tag{2.82}$$

that reconstructs an n-dimensional approximation $\hat{\mathbf{Y}}_t$ of the input $\mathbf{Y}_t$ on the basis of the latent q-dimensional feature $\mathbf{Z}_t$. The two sub-networks are composed solely of feed-forward connections among the layers and might be composed of one or more hidden layers. The networks are usually trained as a single joint network (i.e. the output of the encoder is used as input of the decoder) using gradient descent techniques, such as backpropagation, with the objective of minimizing the mean-squared error between the input and the output (Vincent et al., 2010). In their simplest form, the mappings f_θ and $g_{\theta'}$ are linear functions of the inputs and the encoded features $\mathbf{Z}_t$ closely related to the PCA principal components (Bourlard and Kamp, 1988). If the hidden layers are non-linear, autoencoders behave very differently from PCA, with the ability to capture multi-modal aspects of the input distribution (Bengio, 2009; Vincent et al., 2010). In the context of this thesis, we will consider two types of autoencoder: the base version, having only one hidden layer in both the encoder and the decoder (henceforth *base*), and a version having two hidden layers in both the networks (called *deep*).

2.4.3 *Recurrent Autoencoders*

Recurrent Neural Networks (RNN) is a state-of-the-art neural network approach (Section 2.3.5.1 and (Hewamalage, Bergmeir, and Bandara, 2021) for a detailed review) where the presence of recurrent connections (i.e. allowing loops in the connection graph between nodes), allows the modeling of dynamic temporal dependencies. Without loss of generality, the encoder-decoder architecture presented in Section 2.4.2, can be applied with recurrent neural networks:

$$\mathbf{H}_t = \sigma(\mathbf{W}_{HY}\mathbf{Y}_{t-1} + \mathbf{W}_{HH}\mathbf{H}_{t-1} + \mathbf{B}_H) \tag{2.83}$$
$$\mathbf{Z}_t = f_\theta(\mathbf{W}_{ZH}\mathbf{H}_t + \mathbf{B}_Z) \tag{2.84}$$

$$\mathbf{H'}_t = \sigma(\mathbf{W}_{H'Z}\mathbf{Z}_{t-1} + \mathbf{W}_{H'H'}\mathbf{H'}_{t-1} + \mathbf{B}'_H) \tag{2.85}$$
$$\hat{\mathbf{Y}}_t = g_{\theta'}(\mathbf{W}_{YH}\mathbf{H}_t + \mathbf{B}_Y) \tag{2.86}$$

where the encoder (Equations 2.83, 2.84) and decoder network (Equations 2.85, 2.86) will have independent matrices for weights $\mathbf{W}_{HY}$, $\mathbf{W}_{ZH}$,$\mathbf{W}_{HH}$, $\mathbf{W}_{H'Z}$, $\mathbf{W}_{H'H'}$, $\mathbf{W}_{YH}$ and biases $\mathbf{B}_H$,$\mathbf{B}_Z$,$\mathbf{B}'_H$,$\mathbf{B}_Y$. This encoder-decoder architecture is often referred as a sequence-to-sequence (S2S) (Sutskever, Vinyals, and Le, 2014) model in the literature (Hewamalage, Bergmeir, and Bandara, 2021). Variations of this architecture (with multiple hidden layers and specific attention mechanisms) have been effectively used in the framework of time series forecasting, see (Du et al., 2020) and (Bianchi et al., 2017). Additionally, a theoretical study of the sequence-to-sequence framework for time series forecasting, allowing to determine theoretical bounds have been performed by (Kuznetsov and Mariet, 2018). Last but not least, recurrent encoder-decoder architectures have been effectively employed for dimensionality reduction in the signal processing field (Yang et al., 2020), (Susik, 2020).

For these reasons, we consider two recurrent autoencoders based on LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014a) units, respectively. The choice is motivated by the fact that in the extensive study (Bianchi et al., 2017) concluded that gated units (such as LSTM and GRU) outperform other recurrent methods when the temporal dependencies can be non-linear and abrupt, and that there is no clear outperformance of LSTM over GRU, or vice versa. Both the LSTM and the GRU autoencoder are implemented as a three layer network: input, hidden and output layers. Both the input and output layer have a number of neurons equal to the number of input time series, a sigmoid activation function and are fully connected to the hidden layer. The hidden layer is constituted of a number of recurrent cells (LSTM or GRU) equal to the number of factors to estimate.

2.5 FORECAST COMBINATION

In the previous sections, we presented different families of models (namely data-driven and model driven) making different assumptions on the nature of the data and on the relationship between input and output.

A traditional model selection procedure will assess the different models on the available data and then select the best-performing model, according to some performance criteria (see Section 2.2.3.3), discarding the others (sometimes also referred as winner-takes-it-all approach). However, the discarded models can still have some relevance for the forecasting task at hand, for example by considering alternative input variables, or by making different assumptions on the relationship between input and output data (e. g., linear vs non-linear).

2.5.1 *Problem formulation*

Starting from this idea, the authors of (Bates and Granger, 1969) first proposed the concept of forecast combination (also referred as ensembling in the ML literature), initially limited to two methods and one-step-ahead forecasting. The authors showed that, if the considered models employ independent information, the combined forecast improves over the individual ones.

Moreover, the authors of (Newbold and Granger, 1974) extended the initial idea to more than two methods, and showed that the method combination also reduces the variance of the forecasting error (a similar result, albeit from a different derivation, can be found in (Bontempi, 2013)).

The one-step-ahead formulation of the forecast combination procedure can be described as follows:

$$\hat{y}_{(T+1)} = \sum_{k=1}^{K} \hat{f}^{(k)}_{(T+1)} w^{(k)}_T \tag{2.87}$$

where K is the number of considered machine learning models, $\hat{f}^{(k)}$ is the output of the k-th considered model and $w^{(k)}$ is the corresponding combination weight.

The seminal papers (Bates and Granger, 1969; Newbold and Granger, 1974) propose several time-dependent configurations, in order to adapt the forecast combination according to the forecasting performance of the different composing models $\hat{f}^{(k)}$. The combination can be based either on the forecasting error of the individual model $e_{i,t} = y_t - \hat{f}^{(i)}_t$ or an estimation of the covariance of the forecasting error between different models.

For the forecasting error based techniques, the authors propose a basic form (Equation 2.88) based on past squared error, on a time window of size ν.

$$w_{i,T} = \frac{\left(\sum_{t=T-\nu}^{T-1} e^2_{i,t}\right)^{-1}}{\left\{\sum_{j=1}^{M}\left(\sum_{t=T-\nu}^{T-1} e^2_{j,t}\right)^{-1}\right\}} \tag{2.88}$$

In (Newbold and Granger, 1974), they also propose variations employing an exponential decaying scheme controlled an hyperparameter or constant weighting scheme combination. In addition, they proposed both a weighted and an unweighted schemed based on the estimation of the covariance matrix on the error. In their experiments on economic data, the authors conclude that, while employing statistical models (i. e., ARIMA and Holt-Winters), the error-based weighting scheme of Equation 2.88 appears to be consistently among the top performers.

Without loss of generality the formulation proposed for one-step-ahead forecast (Equation 2.87) can be extended to the multi-step-ahead forecasting as follows:

$$\hat{y}_{(T+i)} = \sum_{k=1}^{K} \hat{f}^{(k)}_{(T+i)} w^{(k)}_T, \quad \forall i \in [1, H] \tag{2.89}$$

where the H-step-ahead can be obtained by employing a multi-step-ahead forecasting strategy (e. g., *Direct*, *Recursive* - Section 2.2.3.2) as in (Koprinska, Rana, and Rahman, 2019).

In a similar fashion, the one-step-ahead univariate formulation can be extended to multivariate one-step-ahead forecasting:

$$\hat{Y}_{(T+1)} = \sum_{k=1}^{K} \hat{F}^{(k)}_{(T+1)} w^{(k)}_T \tag{2.90}$$

where K is the number of considered machine learning models, $\hat{F}^{(k)}$ is the output of the k-th considered multivariate model and $w^{(k)}$ is the corresponding combination weight, as proposed in (Liu et al., 2019) and (Rana, Koprinska, and Agelidis, 2016)

At the time of writing, to the best of our knowledge, we are not aware of any forecasting combination technique tackling both the multivariate and multi-step-ahead aspects.

The forecasting combination framework presented in this section focuses on a fixed (i. e., the number and the type of the underlying forecasting methods does not change over time) combination of heterogeneous forecasting techniques (as no hypothesis on the form of the underlying models is made), with time-varying weights. The following sections will focus on presenting a brief overview of the available techniques in terms of these three dimensions: type of combination (fixed vs variable), type of underlying models (homogeneous vs heterogeneous) and type of combination weights (fixed vs time varying).

2.5.2 *Fixed versus variable combination*

The basic combination rule proposed a combination of a fixed number of models. However, as proposed by the author of (Kuncheva, 2004), several modifications can be applied to the model combination in order to improve its performances.

On one hand, one can act on the input data for the models by updating the available data in an online fashion (either with or without modification of the ensemble itself) as well as by working on the input features of the model. For instance, the authors of (Wang et al., 2018) propose a fixed combination of models, where the composing models are constructed differently by performing either data or feature resampling.

On the other hand, one can work on the models involved in the combination, by either updating/retraining the model as required or by structurally changing the composition of the forecast combination, for example by removing/replacing the worst performing individual forecasting techniques. In (Cerqueira et al., 2017b), for example, the authors propose to form an adaptive combination of a subset of the best performing models, according to their performance on fixed-size windows of the most recent data. In (Saadallah, Priebe, and Morik, 2020), the authors propose to adapt the forecast combination via a two step process, in which the first detects the performance drift (i. e., the moment when a change in the combination is required), while the second step selects the best performing models. Finally, the authors of (Boulegane, Bifet, and Madhusudan, 2019) propose a dynamic selection of the components of the combination, testing two solutions: based on a manually set performance threshold and based on a probabilistic selection, with a selection probability inversely proportional to the forecasting error.

2.5.3 *Homogeneous versus heterogeneous combination*

The presented combination framework doesn't set any hypothesis or the type of the models composing the combination. However, as pointed out by the authors of (Cerqueira et al., 2017a) learning models of the same type (i. e., with similar assumptions concerning the model structure) are prone to behave similarly across similar data-spaces. Moreover, the authors of (Clements and Hendry, 1998) showed that if the predictors are based on the same input data, it is difficult for a combined forecast to outperform each of the individual predictors. For these reasons, it is generally preferred to introduce some forms of diversity in the ensembles in order to improve its forecasting performance, as shown by the (Priebe, 2019) through bias-variance analysis of the forecasting error of forecast combination.

A large part of the techniques analyzed in the literature achieves this goal by combining different model families (e. g., statistical and machine-learning based; naive and more

complex methods as in (Cerqueira et al., 2017b; Saadallah, Priebe, and Morik, 2020; Sánchez, 2008), among others). Nevertheless, employing a set of homogeneous underlying models can still produce competitive results via the diversity introduced by techniques like bagging (Inoue and Kilian, 2004) (i. e., combining homogeneous models trained on different input datasets obtained via bootstrapping), as in (Bergmeir, Hyndman, and Benítez, 2016). The drivers behind the bagging effectiveness are further investigated in (Petropoulos, Hyndman, and Bergmeir, 2018), where the authors conclude that combining different variations of the same model is more beneficial in improving forecasting accuracy than extensively focusing on parameters optimization. In addition, they propose a combination rule proportional to the frequency with which the considered model has proven to be optimal for the considered dataset.

2.5.4 *Time invariant versus time varying combination weights*

A naive approach for forecasting combination involves setting equal weights for all the models in the combination: $w_t^{(k)} = \frac{1}{K} \; \forall k, \forall t$. In this configuration, the forecast combination corresponds to a time-invariant average of the forecasts of the independent models (henceforth referred as *Average* or *Av.*). Despite its simplicity, such approach often yields competitive results, and it is commonly employed as a reference baseline to compare the performance of more advanced methods against. However, this combination proposal shows its limitation in highly dynamical settings, where the time invariant approach fails to properly adapt to the change in dynamics. As discussed in (Kuncheva, 2004), dynamic combiners (or "horse racing" algorithms) are proposed to cope with this issue by adapting the forecast combination through a change in the combination rule, instead of the number and type of underlying models, as discussed in Section 2.5.2.

Examples of *Dynamic combiners* have been proposed by the authors of (Sánchez, 2008) where forecasters are combined using an exponential re-weighting strategy based on their past performance, in combination with a forgetting factor to favour more recent observations over older ones. Another example is given by the authors of (Cerqueira et al., 2017b), where a set of meta-learners are trained to predict the forecasting error of the underlying models, whose prediction are employed to weight the components of the forecasting combination. Additionally, the article also includes dynamic weight adaptation with dynamic ensemble composition.

A similar idea is employed in (Wang et al., 2018), where meta-learning and dynamic combinations are employed together with random sampling of the input data and random feature sampling.

2.5.5 *Stacking*

Last but not least, an alternative approach to forecast combination is implemented through model stacking:

$$\hat{y}_{(T+1)} = F_{STACK}(\hat{f}^{(1)}_{(T+1)}, \cdots, \hat{f}^{(K)}_{(T+1)}) \tag{2.91}$$

Instead of performing a linear combination of the model, in a stacking approach, an additional predictor F_{STACK} is used. The predictor takes as input the forecasts of the underlying models trained on the available data and produces the combined forecast. The standard stacking architecture is composed of two layers: the first being the underlying models, and the second being the meta-model combining them. An example of such architecture can be found in (Pavlyshenko, 2020), where a probabilistic (bayesian) approach is employed to combine predictive models, allowing to quantitatively measure the uncertainty

in the predictions (and consequently the predictive risk). In principle, the mapping F_{STACK} can involve multiple layers of stacking, provided that: the first layer combines the underlying models, each i-th layer employs the predictions from layer $i-1$ and the final layer provide the output prediction.

Part II

CONTRIBUTIONS

"It is difficult to make predictions, especially about the future."

– Danish proverb [3]

3 https://quoteinvestigator.com/2013/10/20/no-predict/

3

METHODOLOGICAL CONTRIBUTIONS

This chapter introduces two original strategies for multivariate and multiple step-ahead time series forecasting:

- The first strategy (called Dynamic Factor Machine Learning) is a machine learning extension of a famous technique in econometrics: it transforms the original high-dimension multivariate forecasting problem by first extracting a (small) set of latent variables (also called factors) and forecasting them independently in a multi-step-ahead yet univariate manner. Once the multi-step-ahead forecast of factors is computed, the predictions are transformed back to the original space.
- The second strategy (called Selective Multivariate to Univariate Reduction through Feature Engineering and Selection) addresses the dimensionality issue in the original space and deals with the combinatorial explosion of possible spatial and temporal dependencies by feature selection. The resulting strategy combines expert-based feature engineering, effective feature selection strategies (based on filters), and ensembles of simple models in order to develop a set of computationally inexpensive yet effective models.

Before moving to the actual contributions, in this chapter we formalize the problem of multivariate forecasting, and briefly present the existing theoretical framework around the problem (Section 3.1)

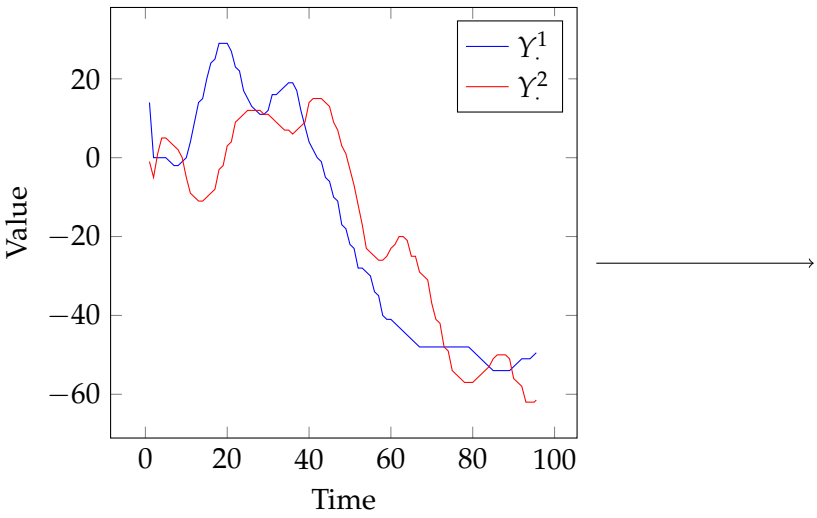
3.1 MULTIVARIATE FORECASTING

A multivariate time series having n observed variables and N observations is conventionally represented via a matrix $\mathbf{Y_t}$. In the scientific literature, two main conventions exist for the representation of $\mathbf{Y_t}$: row-wise and column wise. In the row-wise representation, the different rows of the matrix represent the individual time series, while the column represents the different time steps, yielding to a matrix of size $n \times N$. Conversely, in the column-wise representation, the time series is arranged in the different columns, while the row represents the different time steps, in a $N \times n$ matrix. In the following of the thesis, we will employ the column-wise representation as shown in Figure 3.1.

According to (Januschowski et al., 2020), three main approaches exist to deal with a multivariate forecasting problem: local modeling, global modeling and hybrid modeling (Table 2.3).

3.1.1 *Local modeling*

In local modeling (Januschowski et al., 2020), a single model is estimated independently for each time series to be forecast. In other words, the multivariate forecasting task is decomposed into a set of n Single Input Single Output (SISO) or Multiple Input Single Output (MISO) tasks. In the case of SISO tasks, each of the n forecasting tasks is treated as an independent problem, thus ignoring the cross dependencies with the other series. Here, $Y_t[i]$ denotes the t-th time step of the i-th series of the multivariate series $\mathbf{Y}$.



t	Y_t^1	Y_t^2
1	14	-1
2	0	-5
3	0	1
4	0	5
5	0	5
6	-1	4
7	-2	3
⋮	⋮	⋮
N	Y_N^1	Y_N^2

Figure 3.1: Representation of a multivariate time series in a graphical format, and the corresponding matrix format.

$$\begin{cases} Y_{t+1}[1] & = f_1^{(NAR)}(Y_t[1], \ldots, Y_{t-m+1}[1]) + e_{t+1}[1] \\ \vdots & \\ Y_{t+1}[n] & = f_n^{(NAR)}(Y_t[n], \ldots, Y_{t-m+1}[n]) + e_{t+1}[n] \end{cases} \tag{3.1}$$

where each function $f_i^{(NAR)} : \mathbb{R}^m \mapsto \mathbb{R}$, $i = 1, \ldots, n$, represents a non-linear autoregressive model NAR with model order m.

In the case of MISO tasks, multiple series can be used as input covariates to forecast a single time series.

$$\begin{cases} Y_{t+1}[1] & = f_1^{(NARX)}(Y_t[1], \ldots, Y_{t-m+1}[1], \ldots, \\ & \quad Y_t[n], \ldots, Y_{t-m+1}[n_i]) + e_{t+1}[1] \\ \vdots & \\ Y_{t+1}[n] & = f_n^{(NARX)}(Y_t[1], \ldots, Y_{t-m+1}[1], \ldots, \\ & \quad Y_t[n], \ldots, Y_{t-m+1}[n]) + e_{t+1}[n] \end{cases} \tag{3.2}$$

where each function $f_i^{(NARX)} : \mathbb{R}^{m \times n} \mapsto \mathbb{R}$, $i = 1, \ldots, n$, represents a non-linear autoregressive model with model order m and up to n external regressors.

Although the choice of ignoring cross dependencies (partially, in the case of a MISO decomposition or totally, in the case of a SISO decompostion) might seem disadvantageous at first glance, it allows to greatly reduce the model complexity, thus reducing the variance of the model and its computational learning time. Moreover, the local models could potentially be trained in parallel, thus improving even more the efficiency of the training. Due to this reduced computational complexity, local models are often used as benchmarks, outperforming more complex techniques in some practical cases (such as in the M4 Competition (Makridakis, Spiliotis, and Assimakopoulos, 2020b)).

By employing the forecasting strategies introduced in Chapter 2, the local modeling approach can be easily augmented in order to perform multiple-step-ahead forecasting with the *Direct*, *Iterated* and *Multiple Input Single Output (MIMO)* strategies.

3.1.2 *Global modeling*

In global modeling, the multivariate forecasting problem is tackled as a single MIMO problem, employing a Non-linear Vector Auto-Regressive (NVAR) formulation. In other words, the model $F : \mathbb{R}^{n \times m} \mapsto \mathbb{R}^n$ takes the embedding vectors of size m of the n time series as input and produces the one-step-ahead forecasts as follows:

$$\mathbf{Y}_{t+1} = F\left(\mathbf{Y}_{t-d}, \mathbf{Y}_{t-d-1}, \ldots, \mathbf{Y}_{t-d-m+1}\right) + \mathbf{E}_{t+1} \tag{3.3}$$

where m denotes the lag (or embedding order), d is the delay and $\mathbf{E}$ stands for a multivariate zero-mean noise term, with diagonal covariance matrix.

Analogously to the univariate case, multiple-step-ahead forecasting (H-step ahead here) can be performed by extending the established *Direct*, *Iterated* and MIMO strategies (Ben Taieb et al., 2012) to the multivariate case.

Similarly to the univariate case (Section 2.2.3.1), an input-output embedding of the multivariate data (Figure 3.2) can be performed to allow the application of the *Recursive* and *Direct* strategies. In both cases, the multi-step-ahead forecasting problem is reduced to a one-step-ahead one (at time $t+1$ and $t+H$, for *Recursive* and *Direct* respectively) as follows:

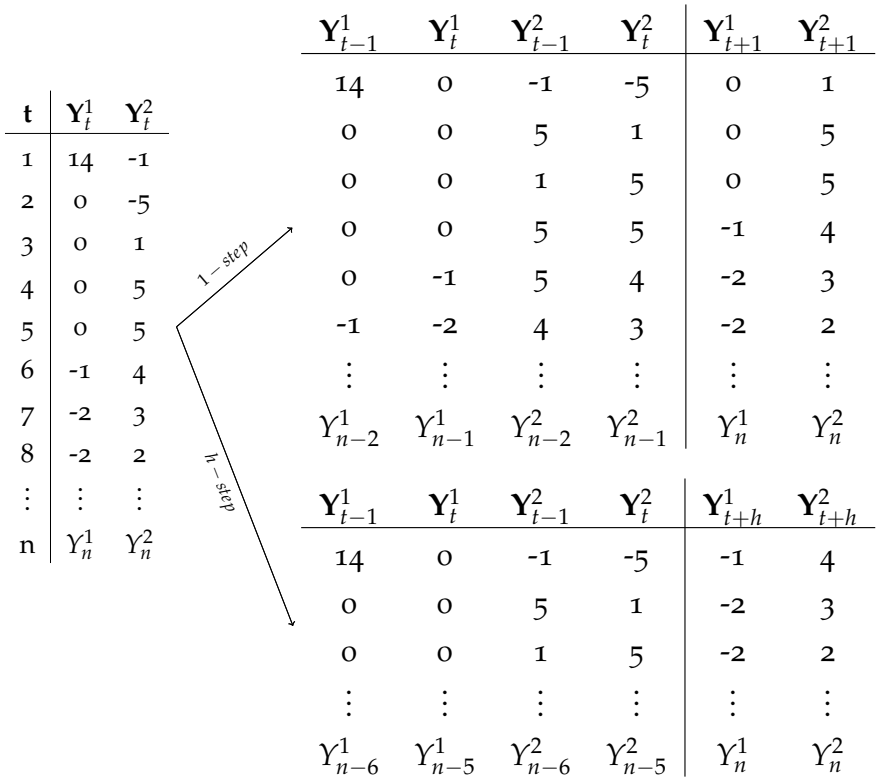


Figure 3.2: Embedding of a multivariate time series with an embedding order $m = 2$ for a 1-step ahead forecast and h-step ahead forecast ($h = 4$), with zero delay ($d = 0$).

With the *Iterated* multiple-step-ahead strategy a single one-step-ahead MIMO model $F_I : \mathbb{R}^{n \times m} \mapsto \mathbb{R}^n$ is iteratively re-used H times to produce the desired forecasts:

$$\mathbf{Y}_{t+1} = F_I(\mathbf{Y}_t, \ldots, \mathbf{Y}_{t-m+1}) + \mathbf{E}_{t+1} \tag{3.4}$$

On the other hand, the *Direct* H-step-ahead strategy employs H separate MIMO models $F_h : \mathbb{R}^{n \times m} \mapsto \mathbb{R}^n$, each one producing the h-step ahead forecasts $\forall h \in \{1, \cdots, H\}$:

$$\mathbf{Y}_{t+h} = F_h(\mathbf{Y}_t, \ldots, \mathbf{Y}_{t-m+1}) + \mathbf{E}_{t+h} \tag{3.5}$$

Finally, the MIMO strategy can be extended to the multivariate case as well:

$$\begin{bmatrix} \mathbf{Y}_{t+H} & \cdots & \mathbf{Y}_{t+1} \end{bmatrix} = F_{JM}(\mathbf{Y}_t, \ldots, \mathbf{Y}_{t-m+1}) + \mathbf{E} \tag{3.6}$$

with $F_{JM} : \mathbb{R}^{n \times m} \mapsto \mathbb{R}^{H \times n}$ being the model jointly providing H-step ahead forecasts for all the n time series.

The global model category allows to properly model the cross-dependencies between the different time series, by increasing the complexity of the functional mappings that have to be estimated (cf. equations 3.4, 3.5, 3.6). The number of parameters to be estimated usually grows quadratically ($O(n^2)$) with respect to the number n of input time series, increasing the computational complexity of the estimation process, and limiting their applicability as the number of time series increases.

3.1.3 *Hybrid models*

In order to exploit the advantages, and limit the drawbacks of both categories, hybrid approaches have been developed, where both the global and the local approaches coexist in different forms.

Hierarchical forecasting models are an example of hybrid forecasting models (Athanasopoulos et al., 2017; Taieb, Taylor, and Hyndman, 2017b; Wickramasuriya, Athanasopoulos, Hyndman, et al., 2015) assuming the existence of a hierarchical structure among the individual time series constituting the multivariate panel. According to (Athanasopoulos et al., 2017), three main approaches exist for hierarchical time series forecasting: top-down, bottom-up and middle-out, each of them referring to the way the hierarchy is defined. Figure 3.3 shows an example of bottom-up specification, with the corresponding formulation either as a system of equations (Eq. 3.7) or, equivalently in a matrix form (Eq. 3.8).

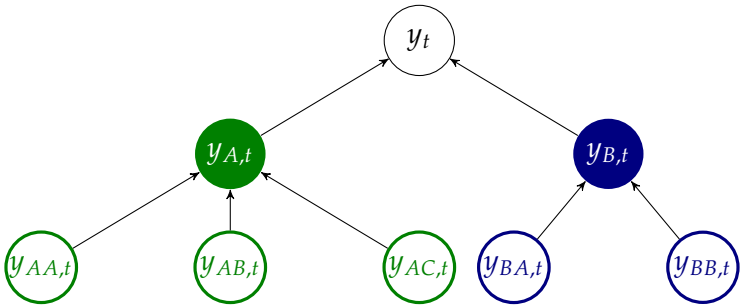


Figure 3.3: Example of bottom-up hierachical structure among time series. Each $y_{X,t}$ represent a univariate time series, whereas the graph structure defines the hierarchical composition of the series (see Eq. 3.7 and 3.8 for the corresponding mathematical formulation).

$$\begin{cases} y_t &= y_{AA,t} + y_{AB,t} + y_{AC,t} + y_{BA,t} + y_{BB,t} \\ y_{A,t} &= y_{AA,t} + y_{AB,t} + y_{AC,t} \\ y_{B,t} &= y_{BA,t} + y_{BB,t} \end{cases} \tag{3.7}$$

$$\begin{bmatrix} y_t \\ y_{A,t} \\ y_{B,t} \\ y_{AA,t} \\ y_{AB,t} \\ y_{AC,t} \\ y_{BA,t} \\ y_{BB,t} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_{AA,t} \\ y_{AB,t} \\ y_{AC,t} \\ y_{BA,t} \\ y_{BB,t} \end{bmatrix} \tag{3.8}$$

All the different hierarchical forecasting techniques can be employed with a two step procedure: forecast and reconciliation. During the forecast phase, the predictions for each individual series in the hierarchy $\widehat{\mathbf{Y}}_t$ are produced. During the reconciliation phase, the forecasts are transformed via the summation $\mathbf{S}$ and grouping $\mathbf{G}$ matrices in order to obtain coherent (i. e., correctly adding up across the hierarchy) forecasts $\widetilde{\mathbf{Y}}_t$:

$$\widetilde{\mathbf{Y}}_t = \mathbf{S}\mathbf{G}\widehat{\mathbf{Y}}_t \tag{3.9}$$

Additional approaches adopt local kernel-based methods (Hwang, Tong, and Choi, 2016) with a global feature selection approach based on memetic algorithm, as well as neural models integrating both local components and global components to perform the global forecast (Sen, Yu, and Dhillon, 2019), or where the output of local models is used as input for the global models (Smyl, 2020).

Finally, a well-known hybrid model category is constituted by dynamic factor models (Stock and Watson, 2010), where the global forecasting problem is reduced to a set of local forecasting problems, through dimensionality reduction. Dimensionality reduction ensures that as much informative content as possible is transferred from the original multivariate problem to the local ones.

3.2 THE DYNAMIC FACTOR MACHINE LEARNING FRAMEWORK

A large part of the research in the domain of dynamic factor models focuses on linear factor estimation techniques combined with model-based forecasting approaches. On one hand, this family of models ensures strong theoretical guarantees, deriving from their analytical formulation. On the other hand, such guarantees are often limited by strong hypotheses defined on the model, and often bound to problems with low-dimensionality (in order to be analytically tractable). To cope with these limitations, we propose our first contribution: the DFML, a hybrid multivariate forecasting strategy, extending the Generalized Dynamic Factor Model framework. Unlike the DFM, the DFML strategy integrates both linear and non-linear factor estimation techniques, in combination with both model-driven and data-driven factor forecasting techniques.

The Generalized Dynamic Factor Model (DFM) is a technique for multivariate forecasting originating in econometrics (Forni et al., 2005) (for a detailed review see (Stock and Watson, 2010)). The basic idea of DFM is that a small number of series (the factors) can account for the dynamics of a much larger number of variables. Such factors are latent, i.e. not directly observable and have to be estimated from the original data. Once estimated, they can be can be forecast instead of the original series, reducing the complexity of the multivariate forecasting process.

In more formal terms:

$$\mathbf{Y}_{t+1} = \mathbf{W}\mathbf{Z}_{t+1} + \mathbf{E}_{Y,t+1} \tag{3.10}$$

$$\mathbf{Z}_{t+1} = \sum_{i=0}^{m-1} (\mathbf{A}_{t-i}\mathbf{Z}_{t-i}) + \mathbf{E}_{Z,t+1} \tag{3.11}$$

where $\mathbf{Z}_t$ is the vector of unobserved factors of size q ($q << n$), $\mathbf{A}_i$ are $q \times q$ coefficient matrices, $\mathbf{W}$ is the matrix $(n \times q)$ of dynamic factor loadings and the disturbances terms $\mathbf{E}_Y, \mathbf{E}_Z$ (also called idiosyncratic disturbances) are assumed to be uncorrelated. In the original formulation, the latent factors follow a vector autoregressive time series process and usually do not have a direct interpretation with respect to the original time series. Note that though the seminal work on DFM adopted a frequency domain approach, we will limit ourselves to consider here the time domain only.

The practical implementation of DFMs demands to address two main issues: the estimation of the factors (including their number) and the forecasting of their evolution. According to (Stock and Watson, 2010) there are three main ways to estimate dynamic factors in literature: the first employs parametric estimation (e.g. maximum likelihood), the second makes use of non-parametric methods (e.g. PCA) and the third relies on Bayesian estimation. It should be noted that, when parametric estimation is employed, some identifying assumptions (i. e., uncorrelated error terms, and specific structures of the factor matrix) need to be made in order to ensure a consistent estimation (Lütkepohl, 2005). As far as forecasting is concerned, both one-step-ahead and multi-step ahead forecasting based on VAR have been proposed in the econometric literature. For an extended study on the use of DFM and PCA for the forecasting of 149 monthly macroeconomic variables we refer the reader to (Stock and Watson, 2002).

3.2.1 *The DFML framework*

From a theoretical standpoint, the two key concepts in the DFM framework are problem decomposition and complexity reduction.

The problem decomposition is achieved by transforming a single multivariate forecasting problem into two independent subproblems of factor estimation and forecasting.

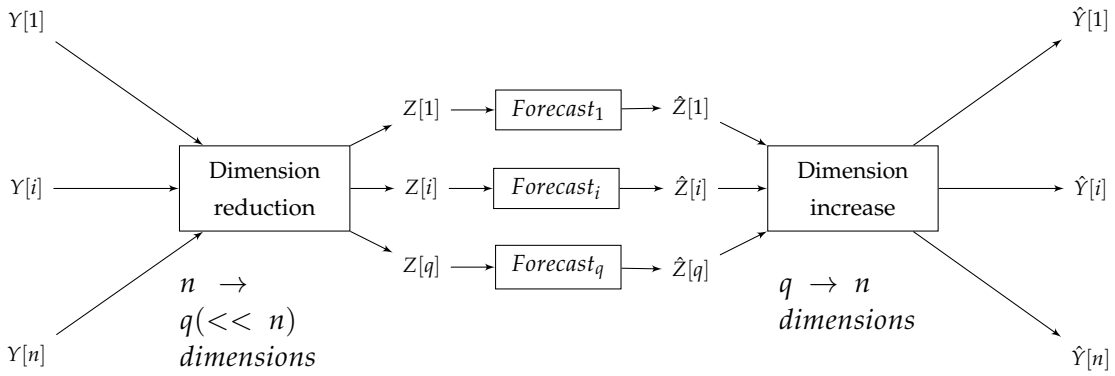
Complexity reduction is achieved via factor estimation, effectively determining a reduced number of factors that can account for the majority of the dynamics in the original time series.

The architecture of our proposed strategy, the DFML, implements the same key concepts (Figure 3.4), while extending the conventional DFM approach both in terms of factor estimation and forecasting.

For factor estimation, the linear method (PCA, for which estimation consistency was proved (Stock and Watson, 2010)) implemented in the original DFM, is complemented by the usage of non-linear and non-parametric techniques, namely feed-forward neural networks (both shallow and deep) and recurrent neural networks encoder-decoder architectures based on LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014a) units.

For the forecasting component, we propose a two-fold improvement.

The first improvement is related to the available forecasting techniques. Conventional DFM only includes a multivariate, statistical technique (i. e., VAR - Section 2.3.4.3). In addition to VAR, in the DFML we implemented well-known statistical forecasting techniques, employed as benchmarks for the M4 competition (Makridakis, Spiliotis, and Assimakopoulos, 2020b), namely Exponential Smoothing (Holt, 2004), Theta method (Assimakopoulos and Nikolopoulos, 2000), and a statistical ensemble technique of these benchmarks, as well as machine learning based techniques, such as MIMO lazy-learning (Bontempi and Ben Taieb,



	Dimensionality reduction	Forecast
DFM	PCA	VAR
DFML	PCA/NN/RNN	Statistical/ML-based

Figure 3.4: Schema of the DFML architecture with a table summarizing the different components as implemented in the different methods.

2011) and gradient boosting based methods (such as LightGBM (Ke et al., 2017), among the top performers in the M5 competition (Makridakis, Spiliotis, and Assimakopoulos, 2020a).

The second improvement addresses the production of multiple-step-ahead forecasts. Conventional DFM techniques employ a one-step-ahead forecasting technique, where multi-step-ahead forecasting requires the re-use of the same forecasting model recursively, with the drawback of propagating the forecasting error. In the DFML, we implemented dedicated, state-of-the-art forecasting strategies such as Iterated, Direct or MIMO (Section 2.2.1.2) to address the multiple-step-ahead problem.

Overall, several compositions of linear/non-linear factor estimation and linear/non-linear forecasting are implemented and assessed. The following sections briefly introduce the different factor estimation and factor forecasting techniques, while a thorough assessment is performed in Chapter 4.

It is worth noting that the factor estimation and the factor forecasting modules (i) follow an encoder-decoder like structure (Sutskever, Vinyals, and Le, 2014), (ii) the two components are decoupled from one another, easily allowing to further extend the architecture by plugging in new components and (iii) the complexity of the forecasting step is made independent of n.

3.2.2 *Factor estimation*

The problem of factor estimation involves the determination of a number of factors q, smaller than the original number of time series n, such that these factors give a good approximation of the dynamics of the original data. A common approach to produce an estimation of the factor adopts dimensionality reduction procedures. A dimensionality reduction procedure assumes that the original multivariate $N \times n$ time series $\mathbf{Y}$ can be represented in a $q < n$ dimensional space while retaining as much informative content as possible about the original dynamics. The lower dimension data is then represented by the Z matrix, having dimensions $N \times q$. Multiple techniques have been developed throughout the years, concerning dimensionality reduction, making either a linear assumption about the structure of the lower dimension subspace (e.g. PCA (Hotelling, 1933)), or non-linear assumptions (e.g kernel PCA (Schölkopf, Smola, and Müller, 1998), autoencoders (DeMers and Cottrell, 1993)) (for a detailed review see (Van Der Maaten, Postma, and Herik, 2009)).

In our strategy we implement three families of dimensionality reduction methods: techniques that do not take into account temporal dependencies, both linear (Section 2.4.1) and non-linear (Section 2.4.2), and techniques that take into account temporal dependencies (Section 2.4.3).

3.2.3 *Factor forecasting*

Once the factors are estimated, a forecasting of $\mathbf{Z}_t$ is required in order to produce the forecasts $\widehat{\mathbf{Y}}$ for the original series. It should be noted that when a factor estimation technique (cf. Section 2.4.1) produces decorrelated factors, the original MIMO forecasting task is transformed into q independent SISO forecasting problems, whereas other factor estimation techniques do not guarantee an independent decomposition. For this reason, in our strategy we include both univariate and multivariate factor forecasting techniques, considered as state-of-the-art approaches in both the statistical and the machine learning domain (Januschowski et al., 2020).

3.2.3.1 *Statistical techniques*

Statistical techniques (also model-driven techniques (Januschowski et al., 2020)) usually define a series of assumptions on the available data, in order to provide a closed-form formulation of the dependency between input and output. We consider here Exponential Smoothing (Section 2.3.2.3), Theta (Section 2.3.2.4) and Combined (Section 2.3.2.4) methods, SISO techniques for one-step-ahead forecasting as well as VAR, a MIMO technique for one-step-ahead forecasting. All those models can be adapted for multi-step-ahead forecasting by implementing a recursive strategy (Section 2.2). The rationale for considering statistical techniques is that in several forecasting competitions on real-world data (Hyndman, 2020) simple forecasting techniques tend to outperform more complex methods.

3.2.3.2 *Machine learning based techniques*

Machine learning techniques (or data-driven in (Januschowski et al., 2020)) do not make any parametric assumptions on the data distribution. In this category, we consider lazy learning (i.e. a single model technique) (Ben Taieb et al., 2012) (Section 2.3.3.2) and gradient boosting (an ensemble technique) (Ke et al., 2017) (Section 2.3.3.4). Those models can be used for multi-step-ahead forecasting both via a *Recursive* and a *Iterative* strategy (Section 2.2). Additionally, we consider a SIMO implementation of the lazy learning model *Joint* (Bontempi and Ben Taieb, 2011), in which all the h steps to be forecast are returned by a single model.

3.2.4 *Dynamic Factor Machine Learner extensions*

This section focuses on introducing two extensions to the basic DFML framework: iterative factor estimation and automatic hyperparameter selection.

The first extension tries to address the computational issues related to the large dimensionality (both in terms of series and samples of the multivariate input time series) by introducing an incremental approach to factor estimation. The main idea of this approach is to replace the computationally intensive process of factor estimation of the whole available data with a reduced complexity alternative, allowing for incremental updates with smaller chunks of data (e. g., a single sample or a set of samples). An additional advantage of this approach is that it extends the capabilities of the DFML framework to deal with streaming

data, a situation where the input data is continuously incremented by the arrival of new samples.

The second extension focuses on improving the factor forecasting component by implementing a grid search for the optimal values of most relevant hyperparameters: number of factors, multi-step-ahead strategy, forecasting technique. The optimal hyperparameter combination is determined via minimization of the out-of-sample prediction error. The rationale of this approach is to simplify the model tuning procedure by automatically providing to the end-user the best framework configuration given the available data.

3.2.4.1 *Incremental factor estimation*

The first improvement introduces an incremental (also referred as online) approach for factor estimation, with a focus on linear techniques. The conventional approach to PCA computation (i. e., batch) requires a sequence of matrix operations whose computational complexity increases linearly with the length N of the time series and quadratically with the number of dimensions n (Section 2.4.1.1). In a large-scale setting, where the number of dimensions is large $n > 100$, and the length of the time series is continuously increasing, recomputing the factors from scratch at each observation step becomes quickly unfeasible. For this reason, we integrated several state-of-the-art linear incremental factor estimation techniques (Section 2.4.1.1). These incremental techniques allow a considerable reduction in terms of computational cost, all while being consistent with respect to their batch counterparts.

3.2.4.2 *Automatic hyperparameter search strategy*

The second improvement introduces a joint selection of the hyperparameters (number of factors, predictor, multi-step-ahead strategy) based on an out-of-sample accuracy criterion (computed after reconstruction) related to the specific forecasting task.

A pseudo-code of the search strategy is available in Fig. 3.5 [1] . Lines 1-2 show that DFML searches in a space of models characterised by the pair (p, a) where p denotes the number of factors and $a \in \mathcal{A}$ the univariate multi-step ahead forecasting strategy (e.g. direct, iterated, or MIMO). For each candidate pair DFML assesses after reconstruction (Line 7) the out-of-sample multi-step-ahead accuracy (Line 9) and returns (Line 12) the most promising one.

Once the selection is performed, each factor (since uncorrelated from the others) is forecast independently by using the chosen multi-step-ahead strategy a^*. An interesting aspect of this approach is that the complexity of the forecasting step is made independent of n.

1 In *batch* PCA the components are recomputed at each instant t while in *incremental* they are updated sequentially (after initialization with a small data subset).

Require: Observed matrix $\mathbf{Y}$, family $\mathcal{A}$ of multi-step-ahead strategies, maximum number of principal components $q \geq 1$, $H \geq 1$, set $\mathcal{T}$ of test set starting points such that $\forall t \in \mathcal{T}, \frac{2}{3}N < t < N - H$

for $p = 1$ to q **do**
  for $a \in \mathcal{A}$ **do**
    for $t \in \mathcal{T}$ **do**
      $\mathbf{W} =$ `FactorEstimation`(Y) ▷ *Computation latent factors*
      $\mathbf{Z} = \mathbf{YW}$ ▷ *Factors observation matrix*
      ▷ `FactorForecast`(*series*,*a*,*H*) return the H next steps of a *series* according to the strategy *a*
      $\{\hat{Z}[p]^a_{t+1}, \ldots, \hat{Z}[p]^a_{t+H}\} =$`FactorForecast`$(\{Z[p]_1, \ldots, Z[p]_t\}, a, H)$
      $\{\hat{Y}^a_{t+1}, \ldots, \hat{Y}^a_{t+H}\} = \hat{Z}^a[, 1\colon p]W^T[, 1\colon p]$ ▷ *Reconstruction of the original series*
    end for
    $\text{MSE}[p, a]=\text{avg}_{t\in\mathcal{T}}$ [`MSE`$(\{Y_{t+1}, \ldots\}, \{\hat{Y}^a_{t+1}, \ldots\})$] ▷ *average forecasting error over all out-of-sample test sets*
  end for
end for
  return $\{k^*, a^*\} = \arg\min_{k\in\{1,\ldots,q\}, a\in\mathcal{A}} \text{MSE}[k, a]$
▷ k^* best number of factors, a^* best multi-step-ahead forecasting strategy

Figure 3.5: Pseudocode of the DFML automatic hyperparameter search strategy.

3.3 SMURF-ES : SELECTIVE MULTIVARIATE TO UNIVARIATE REDUCTION THROUGH FEATURE ENGINEERING AND SELECTION

This section presents the second contribution of the thesis, denoted SMURF-ES. This strategy adapts the conventional machine learning pipeline to the problem of multivariate and multi-step-ahead forecasting. The SMURF-ES pipeline is composed of three main phases: feature engineering, feature selection and forecasting.

In our solution (Figure 3.6), we employ a global approach. Whereas the DFML strategy employs a dimensionality reduction approach, SMURF-ES relies on feature engineering and feature selection to address the high-dimension issue.

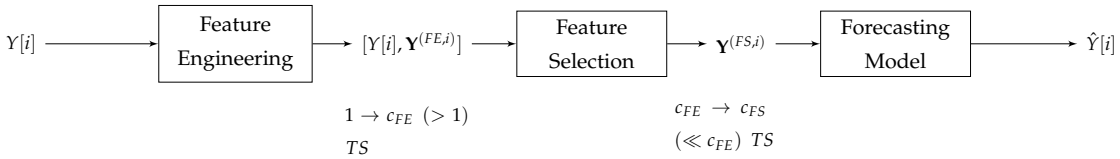


Figure 3.6: Summary of the traditional machine learning pipeline for univariate time series forecasting. Feature extraction allows to produce a multivariate series $\mathbf{Y}^{(FE,i)}$ from univariate time series $Y[i]$, from which a subset of series $\mathbf{Y}^{(FS,i)}$ is extracted via feature selection.

Unlike the DFML strategy, SMURF-ES relies on feature engineering and feature selection to address the high-dimensionality issue (Figure 3.7). The input multivariate time series is augmented by computing rolling statistics (rolling means, standard deviations, quantiles, extremes) as well as expert-based, domain specific features (Section 3.3.1).

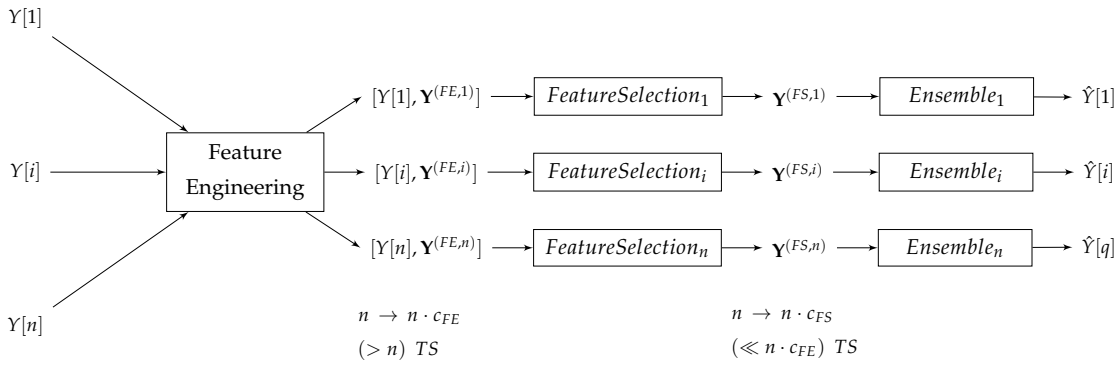


Figure 3.7: Summary of the SMURF-ES strategy for multivariate time series forecasting. A global feature engineering process is followed by a set of local and independent pipelines composed of *Feature Selection* and *Ensemble Forecasting*.

After this augmentation, the dimension of the input data for the forecasting model passes from the original n variables to $c_{FE} * n$, where c_{FE} is an integer coefficient representing the number of augmented features computed for each univariate time series. Even with $c_{FE} \sim 10$, if the dimension n of the multivariate time series is large enough, this augmentation increases considerably the size of the problem.

For this reason, we propose a feature selection approach based on filtering, in order to select only the relevant variables for the forecasting task at hand. This approach is implemented locally and independently for each univariate time series, taking into account the original input data and the corresponding augmented features. It should be noted that, when a number $c_{FS} * n$ of features is selected, with $1 < c_{FS} < c_{FE}$, the dimensionality is still increased from the original number of time series, but is considerably reduced with respect to the output of the feature engineering step.

After this process, the selected features are given as input to the selected forecasting model. For the forecasting step, we propose to employ an ensemble of heterogeneous forecasting model (namely a combination of statistical and machine-learning based models).

Ensemble techniques have been proven to be effective in the domain of time series forecasting (Oliveira and Torgo, 2015), while heterogeneity aspect has shown to further improve the forecasting accuracy, especially in the case of highly dynamical settings (Cerqueira et al., 2017a; Cerqueira et al., 2017b). Moreover, (Cerqueira, Torgo, and Soares, 2019) shows that statistical and machine-learning techniques have a complementary behavior with respect to the size of the training set employed for the model, further highlighting the advantages of heterogeneity.

Last but not least, it should be noted that the proposed strategy could easily employed for both one-step-ahead forecasting, as presented in Figure 3.7, as well as for multiple-step-ahead forecasting, by implementing the *Direct*, *Recursive* and MIMO strategies presented in Section 2.2.1.2.

The following sections discuss the details of the different components of the forecasting pipeline, namely:

1. Feature Engineering: this step augments the representation space by constructing a number of additional input features capturing either the temporal dynamics of the signal or the occurrence of specific events (e.g. curtailment).

2. Embedding strategy: the forecasting task is formulated as a supervised learning problem where the nature of the output and the dimension of the input space depend on the horizon H, temporal lag L and the cross-series dependencies taken into account (Taieb et al., 2012).

3. Dimensionality reduction: this step aims to reduce the number of features, by compression (via Principal Components Analysis - PCA) or by feature selection (Section 2.4). This process aims to contrast the curse of dimensionality (Friedman, Hastie, Tibshirani, et al., 2001) and reduce the computational burden in model training (Bermingham et al., 2015), by removing redundant or correlated features.

4. Model estimation: this step estimates from the available data the input-output relationship defined in the previous steps. State-of-the-art approaches are discussed in Sections 5.3.2 and 5.4.2.

3.3.1 *Feature Engineering*

The feature engineering step consists in augmenting the original multivariate time series with a set of derived time series. Among the techniques presented in Section 2.2.2.4 we employ window-based statistics, computing conventional statistics across a time window of the past w values. In particular, we consider:

Moving Average
$$\overline{y}_{t,[0,w]} = \frac{1}{w+1}\sum_{q=0}^{w} y_{t-q} \tag{3.12}$$

Maximum Value
$$y_t^+ = \max_{q\in\{0,\cdots,w\}} y_{t-q} \tag{3.13}$$

Minimum Value
$$y_t^- = \min_{q\in\{0,\cdots,w\}} y_{t-q} \tag{3.14}$$

p-quantile
$$\begin{aligned} y_{p,t} &= \inf\{z : \widehat{F}_w(z) \geq p\} \\ z &\in \{y_{t-0},\cdots,y_{t-q}\} \\ \widehat{F}_w(z) &= \tfrac{1}{w}\textstyle\sum_{q=0}^{w} \mathbf{1}_{y_{t-q}\leq z} \end{aligned} \tag{3.15}$$

1st order difference
$$\begin{aligned} \Delta y_t &= y_t - y_{t-1} \\ y_t &\in \{y_{t-0},\cdots,y_{t-q}\} \end{aligned} \tag{3.16}$$

Moving Average Incremental Variation
$$\begin{aligned} y_{\Delta_{y_t},t} &= \frac{1}{Q}\left(\overline{y}_{t,[0,Q-1]} - \overline{y}_{t,[Q,2(Q-1)]}\right) \\ &\forall t \in [1,N] \end{aligned} \tag{3.17}$$

Additional features are created by computing parametric, expert-defined functions to detect abrupt changes in the time series, e. g., the sudden change from a dynamical trend to a constant one.

These features are constructed, respectively, using a first order difference based method, discarding all signal variability smaller than σ (3.18), and a Run Length Encoding (RLE) based detection, employing the auxiliary indicator function $\mathbf{1}^S(\cdot)$ (3.19), discarding all the sequences of constant values shorter than a given parameter v (3.20). Both the parameters σ and v are externally specified.

$$\mathbf{1}_\sigma^{FOD}(y_t) = \begin{cases} 1 & |\Delta y_t| < \sigma \\ 0 & otherwise \end{cases} \tag{3.18}$$

$$\mathbf{1}^S(y_t) = \begin{cases} 1 & \Delta y_t = 0 \\ 0 & otherwise \end{cases} \tag{3.19}$$

$$\mathbf{1}_v^{RLE}(y_t) = \begin{cases} 1 & \exists t_s > t_0, t_s < t_e < t \text{ s.t.} \\ & t_e - t_s > v \wedge \forall i \in \{t_s \cdots, t_e\} \mathbf{1}^S(y_i) = 1 \\ 0 & otherwise \end{cases} \tag{3.20}$$

The feature engineering process augments the size of $\mathbf{Y}_{(N\times n)}$, where N is the available number of samples and n is the number of time series. In particular, the total number of time series is $n' = n \cdot c_{FE}$ where $c_{FE} = (1 + n_{stat} \cdot s_q + n_f)$, with n_{stat} being the number of statistics (3.12)-(3.15) computed for s_q different lags, and n_f being the number of features (3.16)-(3.20).

Finally, the raw time series and the augmented ones are rearranged into a matrix $\mathbf{D}_{(N\times n')}$, where N and n' denote the number of samples and variables respectively. This matrix can be decomposed into the predictor submatrix $\mathbf{P}_{(N\times\phi)}$ containing the $\phi = n \cdot (n_{stat} \cdot s_q + n_f)$ predictor time series obtained through the feature engineering procedure as columns, and the target submatrix $\mathbf{Y}[N, n]$, containing the n original time series, one by column.

$$\mathbf{D} = \begin{bmatrix} \mathbf{Y} & \mathbf{P} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} y_{11} & \cdots & y_{1n} \\ \vdots & \cdots & \vdots \\ y_{N1} & \cdots & y_{Nn} \end{bmatrix} & \begin{bmatrix} p_{11} & \cdots & p_{1\phi} \\ \vdots & \cdots & \vdots \\ p_{N1} & \cdots & p_{N\phi} \end{bmatrix} \end{bmatrix} \tag{3.21}$$

This decomposition is performed in order to prepare the available data for the subsequent embedding procedure.

3.3.2 *Embedding Procedure*

This step rearranges the sub-matrices $\mathbf{Y}$ and $\mathbf{P}$ in an embedded input/output form depending on the horizon H and time lag L. The step is performed in order to frame the forecasting problem as a supervised learning problem, and consequently be able to apply forecasting techniques based on supervised learning (i. e., data-driven methods). More precisely, the step produces two matrices: the input matrix $\mathbf{X_{EMB}}$ and the output matrix $\mathbf{Y_{EMB}}$, with the objective of learning a multi-variate and multi-temporal model $F : \mathbf{X_{EMB}} \mapsto \mathbf{Y_{EMB}}$ defining the relation among input and output matrix.

The structure of the matrices $\mathbf{X_{EMB}}$ and $\mathbf{Y_{EMB}}$, whose dimensions are $[N - L - H, \phi \cdot L]$ and $[N - L - H, n \cdot H]$, respectively, is shown in Eq. 3.22 and 3.23.

Note that the increment in size of the two matrices is related to the values of the lag L and the forecasting horizon H respectively, as the procedure requires to generate, for each time series, a new variable for each considered time step, from t until $t - L + 1$ in the input matrix $\mathbf{X_{EMB}}$, and from $t + 1$ until $t + H$ in the output matrix $\mathbf{Y_{EMB}}$.

$$\mathbf{X_{EMB}} = \begin{array}{c|ccccc:c:ccccc} & \overbrace{\quad\quad\quad\quad\quad}^{p_1} & & & & & & \overbrace{\quad\quad\quad\quad\quad}^{p_\phi} & & & & \\ & t-0 & t-1 & t-2 & \cdots & t-L+1 & \cdots & t-0 & t-1 & t-2 & \cdots & t-L+1 \\ \hline & p_{11} & - & - & \cdots & - & \cdots & p_{1\phi} & - & - & \cdots & - \\ & p_{21} & p_{11} & - & \cdots & - & \cdots & p_{2\phi} & p_{1\phi} & - & \cdots & - \\ & p_{31} & p_{21} & p_{11} & \cdots & - & \cdots & p_{3\phi} & p_{2\phi} & p_{1\phi} & \cdots & - \\ & p_{41} & p_{31} & p_{21} & \cdots & \vdots & \cdots & p_{4\phi} & p_{3\phi} & p_{2\phi} & \cdots & - \\ & \vdots & \vdots & \vdots & \ddots & \vdots & \cdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ & p_{N1} & p_{(N-1)1} & p_{(N-2)1} & \cdots & p_{(N+1-L)1} & \cdots & p_{N\phi} & p_{(N-1)\phi} & p_{(N-2)\phi} & \cdots & p_{(N+1-L)\phi} \end{array} \tag{3.22}$$

$$\mathbf{Y_{EMB}} = \begin{pmatrix} \overbrace{\begin{matrix} t+1 & t+2 & t+3 & \cdots & t+H \end{matrix}}^{y_1} & \cdots & \overbrace{\begin{matrix} t+1 & t+2 & t+3 & \cdots & t+H \end{matrix}}^{y_n} \\ \hline \begin{matrix} y_{21} & y_{31} & y_{41} & \cdots & y_{(t+H)1} \\ y_{31} & y_{41} & y_{51} & \cdots & \vdots \\ y_{41} & y_{51} & y_{61} & \cdots & \vdots \\ y_{51} & y_{61} & y_{71} & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{(N-H)1} & - & - & \cdots & - \end{matrix} & \cdots & \begin{matrix} y_{2n} & y_{3n} & y_{4n} & \cdots & y_{(t+H)n} \\ y_{3n} & y_{4n} & y_{5n} & \cdots & \vdots \\ y_{4n} & y_{5n} & y_{6n} & \cdots & \vdots \\ y_{5n} & y_{6n} & y_{7n} & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{(N-H)n} & - & - & \cdots & - \end{matrix} \end{pmatrix} \tag{3.23}$$

Once the matrices $\mathbf{X_{EMB}}$ and $\mathbf{Y_{EMB}}$ are available, a number V of training and test sets are defined in order to implement a forecast evaluation procedure based on cross-validation (Cerqueira, Torgo, and Mozetic, 2019).

In particular, for a generic case test v, the matrices $\mathbf{X}^{(v)}$ and $\mathbf{Y}^{(v)}$ are used to derive the training matrices $\mathbf{X}^{(v)}_{trn}$, $\mathbf{Y}^{(v)}_{trn}$ and the validation matrices $\mathbf{X}^{(v)}_{val}$, $\mathbf{Y}^{(v)}_{val}$, which are used to assess the prediction models.

3.3.3 *Dimensionality reduction*

Though feature engineering and data embedding generate useful information for forecasting, they cause a large increase in data dimension and a consequent number of drawbacks such as curse of dimensionality, high demand of computational resources and ill-conditioning in data analysis (Lian and Chen, 2009).

Hence, it is recommended to adopt dimensionality reduction or feature selection techniques (Rong, Gong, and Gao, 2019). Dimensionality reduction techniques combine the original features to provide a smaller number of features with enhanced predictive power (e. g., PCA Section 2.4.1)

Feature selection aims to select a small subset of relevant features to the forecasting task at hand in order avoid variance and instability issues in model learning (e. g., mRMR Section 2.1.4.2).

For the SMURF-ES strategy, we compared PCA and mRMR on the considered prediction task (wind-power forecasting). Empirical tests over different datasets (Table 5.1) highlighted a better effectiveness of mRMR over PCA (in terms of final prediction error), hence the choice of the former as dimensionality reduction technique. Moreover, the advantage of filter-based techniques (e. g., mRMR) compared to compression-based techniques such as PCA, is that they do not transform the original features, but rather select a subset of them, allowing for easier feature interpretation.

3.3.4 *Model estimation*

The considered preprocessing steps (i. e., feature engineering, embedding and feature selection) allow to estimate different forecasting models, coming from both model-driven and data-driven families (Section 2.3). It should be noted that, whereas model-driven techniques work directly with the raw data, data-driven techniques can fully exploit the preprocessing phase, especially profiting from the feature selection one.

In the framework of the SMURF-ES strategy, the forecasting phase is performed by using heterogeneous ensembles (i. e., forecast combinations with models coming from the different families). As discussed in Section 2.5, by combining models with different assumptions

on the available data and the model to estimate, we expect to improve the forecasting accuracy. Table 3.1 contains a summary of the different families employed for the SMURF-ES forecasting strategies.

Model-driven	Data-driven	SMURF-ES Ensembles
Naive (Persistence)	Feed-forward ANN	Av-GBM-RF
Average	GBM	Av-Average-RF
	Random Forest (RF)	Av-GBM-Average
	Lazy Learning (LL)	Av-SVM-GBM
	SVM	Av-SVM-Average
		Ad-ANN-Average
		Ad-SVM-Average
		Ad-GBM-Average
		Ad-RF-Average
		Ad-Average-RF-SVM-GBM
		DAFT-E (Ad.RF-LL-Naive)

Table 3.1: Overview of the assessed models for the SMURF-ES. *Av* stands for an ensemble of the proposed models based on the averaging of their forecasts, whereas *Ad* denotes a weighted combination of the forecasts, inversely proportional to their forecasting error (Section 2.5).

3.3.4.1 *Model-driven techniques*

As for the model-driven techniques, we consider the Naive model (Section 2.3.2.1), returning the last available value as forecast, the Average model (Section 2.3.2.2) returning a simple moving average on the past observed values. Given their simplicity and reduced computational complexity, these techniques are employed both as ensemble components as well as univariate reference benchmark.

3.3.4.2 *Data-driven techniques*

The embedding procedure (Section 3.3.2) allows to transform the forecasting task into a supervised learning problem and consequently to adopt any kind of supervised learning algorithm. As components of the forecast combination we consider both traditional benchmark techniques like Lazy Learning (LL) (Section 2.3.3.2), Support Vector Machines (SVM) 2.3.3.3 and Artificial Neural Networks (ANN) (Section 2.3.3.1) as well as ensemble techniques such as Random Forest (RF) (Section 2.3.3.5), Gradient Boosting Machine (GBM) (Section 2.3.3.4), that have consistently appeared as top performers in several forecasting competitions (Makridakis, Spiliotis, and Assimakopoulos, 2020a; Makridakis, Spiliotis, and Assimakopoulos, 2020b).

3.3.4.3 *SMURF-ES ensembles*

State-of-the-art approaches such as RF and GBM already rely on the concept of ensemble forecasting, where many low-correlated predictors (also called weak learners) are trained, providing the final prediction by aggregating all model outputs. In our strategy a similar idea is applied at a higher level by combining the forecasts of different models according to a given combination rule (see Section 2.5).

More precisely, our proposed ensemble includes a fixed number of heterogenous forecasting methods (Mendes-Moreira et al., 2012) operating in parallel, both in terms of training and prediction phases. Their heterogeneity is motivated by the fact that each forecasting

model relies on different base assumptions and reacts differently to changes in the input data, such as differences in the distributions of training and validation data sets. These differences, in turn, can dramatically change the forecasting performances of the individual components over time.

The intuition behind our choice to employ forecast combination, is that, by choosing an adequate and adaptive combination rule, one could be able to tune the forecast combination in order to favor the best performing components, while discarding the worse performing one, as time goes by.

3.3.5 *Proposed implementations*

3.3.5.1 *DAF-E : Dynamic Adaptive Feature-based Ensemble*

The first implementation of the SMURF-ES multivariate forecasting strategy employs a forecast combination with a dynamic combination rule, based on the past forecasting error, of a set of models employing feature engineering and selection, hence its name Dynamic Adaptive Feature-based Ensemble (DAF-E). An overview of its main features is presented in Figure 3.8, and is briefly discussed in the following sections.

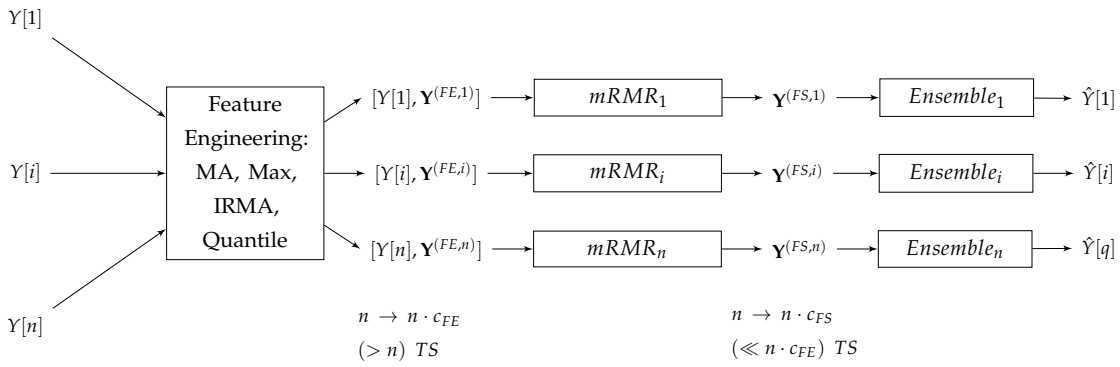


Figure 3.8: Overview of DAF-E model, based on the SMURF-ES strategy.

FEATURE ENGINEERING Following the procedure described in Section 3.3.1 we introduce the following additional predictors: Moving Average (Eq. 3.12), Maximum Value (Eq. 3.13), Incremental Variation of the Moving Average (Eq. 3.17) and p-quantiles (Eq. 3.15), with $p \in \{10\%, 50\%, 90\%\}$.

The feature creation leads to the creation of a matrix $\mathbf{P}_{(N \times \phi)}$, where $\phi = n * (n_{stat} \cdot s_q)$, where $n_{stat} = 6$ corresponds to the number of considered statistics and $s_q = 4$ corresponds the number of values considered for Q (here $Q \in \{6, 12, 18, 24\}$).

DIMENSIONALITY REDUCTION As we can observe from Table 3.2, the feature engineering and consequent embedding process cause a considerable increase in the number of input features to the forecasting models.

As discussed in Section 3.3.3, we employ two techniques for reducing the number of input features: PCA (Section 2.4.1) and mRMR (Section 2.1.4.2).

For this implementation of the SMURF-ES strategy, we selected mRMR after some preliminary experiments, which have highlighted its greater predictive performance with respect to PCA, with a number of selected variables equal to 15.

ENSEMBLE FORECASTING - DAF-E One of the core aspects of the SMURF-ES strategy is the usage of an ensemble technique to combine the multiple-step-ahead forecasts from

Forecasting horizon			Feature Engineering			
hour	H (number of steps ahead)	L(H) Lag	n	n_{stat}	s_q	ϕ
1	6	6	15	6	4	360
2	12	12				720
3	18	18				1080
4	24	24				1400
5	30	30				1720
6	36	36				2040

Table 3.2: Overview of the number of features in the matrices $\mathbf{X_{EMB}}$,$\mathbf{Y_{EMB}}$ as a function of the forecasting horizon H. It is worth noting that, for all the considered forecasting horizons H, the lag (or model order) L is identical to the forecasting horizon.

different forecasting models. Following the taxonomy presented in Section 2.5, we propose two combinations of a fixed number of heterogeneous forecasting models.

In the DAF-E, we consider two combination rules sharing the same general formulation (Equation 3.24): a static simple averaging (Av) and a dynamic, adaptive weighted averaging (Ad).

$$\hat{y}_{(t+i)} = \sum_{k=1}^{K} w_t^{(k)} f_{(t+i)}^{k}, \quad \forall i \in [1, H] \tag{3.24}$$

where K is the number of considered machine learning models and $f_{(t+i)}^{j}$ is the output of the j-th model of the forecasting ensemble for the i-th step-ahead.

The simple averaging combination (Av) rule (Equation 3.25) gives the same weight to every method composing the forecasting ensemble. Moreover, the weights do not change over time. Despite its simplicity, such combination rule is often used as a baseline to test the predictive performance of the forecast combination

$$w_t^{(j)} = \frac{1}{K}, \quad \forall t, \forall j \tag{3.25}$$

The proposed adaptive weighted averaging (Ad), on the other hand, considers dynamical weights, regularly adapted over time. The combination weights are computed based on the forecasting error, more precisely the RMSE (Equation 3.26), in order to be inversely proportional to the forecasting error (3.27). In other words, the smaller the forecasting error, the better the performance of the ensemble component, and consequently, the higher its weight in the ensemble. In addition, the weights are normalized, in order to be constrained in the $[0, 1]$ range.

$$\text{RMSE}_t^j = \sqrt{\sum_{i=0}^{t} (f_{(t)}^{j} - y_i)^2} \tag{3.26}$$

$$w_t^{(j)} = \frac{\left(\text{RMSE}_t^{(j)}\right)^{-1}}{\sum_{k=1}^{K} \left(\text{RMSE}_t^{(k)}\right)^{-1}}, \quad w_t^{(j)} \in [0, 1] \tag{3.27}$$

3.3.5.2 *DAFT-E: Dynamic Adaptive Feature-based Temporal Ensemble*

The second implementation of the SMURF-ES multivariate forecasting strategy improves over the first (DAF-E Section 3.3.5.1). Firstly, the feature engineering process is extended, including also expert-based features. Secondly, the forecasting process is improved with a temporal combination of different ensembles, on top of the one based on the previous forecast error, hence its name Dynamic Adaptive Feature-based Temporal Ensemble (DAFT-E). An overview of its main design choices is presented in Figure 3.9, and is briefly discussed in the following sections.

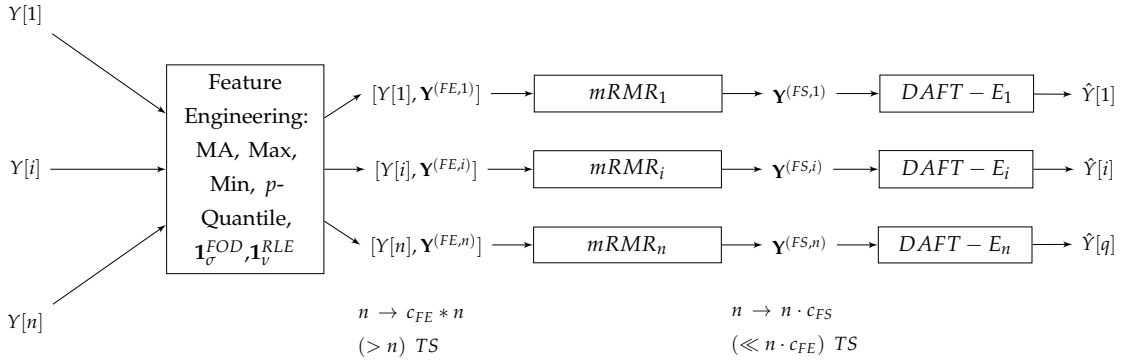


Figure 3.9: Overview of the DAFT-E forecasting model, based on the SMURF-ES strategy.

FEATURE ENGINEERING In this implementation of the strategy, the feature space is augmented in a two steps procedure.

First, this step augments the input space by computing window-based features (Section 2.2.2.4) across a time window of the past w values: Moving average (Eq. 3.12) Maximum value (Eq. 3.13), Minimum value (Eq. 3.14), p-quantiles (Eq. 3.15), First order difference (Eq. 3.16)

Second, it introduces parametric, expert-based features, for example to detect the switch from a highly dynamical trend to a constant one.

These features are constructed using a method based on first order differences discarding all signal variability smaller than σ (3.18), and a method based on Run Length Encoding (RLE), employing the auxiliary indicator function $\mathbf{1}^{S}(\cdot)$ (3.19), discarding all the sequences of constant values shorter than a given parameter v (3.20). The parameters σ and v of the two techniques are externally specified.

DIMENSIONALITY REDUCTION As in the previous implementation of the SMURF-ES strategy, we also consider here two techniques: a filter-based feature selection technique mRMR (Section 2.1.4.2) and a dimensionality reduction technique PCA (Section 2.4.1). Since filter-based techniques do not transform the original features, for the sake of interpretability, we selected mRMR as the feature selection technique within the DAFT-E approach, whereas PCA has been considered as a benchmark technique against which to compare our proposed approach.

ENSEMBLE FORECASTING - DAFT-E We propose an original method, called Dynamic Adaptive Feature-based Temporal Ensemble (DAFT-E), based on the weighted average of M forecasting models, whose weights evolution depends on their forecasting errors over a sliding window of size Γ and a forgetting strategy. The pseudo-code of the method is detailed in the Algorithm 1.

The multivariate multi-step-ahead problem is decomposed in a set of $F = N \cdot H$ multi-input single-output tasks by applying a direct strategy (Taieb et al., 2012) for each n-th

Algorithm 1 DAFT-E algorithm for the k-th trial having V weight update cycles with FF vector $\mathbf{\Lambda}$ and window size Γ

Input: M Algorithms, $\mathbf{X}_{trn}^{(k)}, \mathbf{Y}_{trn}^{(k)}, \mathbf{X}_{test}^{(k)}, \mathbf{\Lambda}, \Gamma$
Output: $\hat{\mathbf{Y}}_{DAFT-E}$ (DAFT-E prediction)

1: $N_c \leftarrow \lfloor N_{test}/\Gamma \rfloor$
▷ *Update the weights over the time span* Γ
2: **for** $c \leftarrow V$ to N_c **do**
▷ *For* $c < V$ *the* $\boldsymbol{w}_{norm}$ *are initialized*
3: $t_{start} \leftarrow \Gamma \cdot (c-1) + 1$
4: $t_{end} \leftarrow \min(N_{test}, \Gamma \cdot c)$
▷ *Compute the error for each of the M algorithms*
5: **for** $i \leftarrow 1$ to M **do**
6: $\hat{\mathbf{Y}}_{test}^{(c,i)} \leftarrow m^{(k,i)}(\mathbf{X}_{trn}^{(k)}, \mathbf{Y}_{trn}^{(k)}, \mathbf{X}_{test}^{(k)}[t_{start} : t_{end}, \ ])$
▷ $m^{(k,i)}$ *produces the outputs of the* i^{th} *algorithm, trained with input* $\mathbf{X}_{trn}^{(k)}$ *and output* $\mathbf{Y}_{trn}^{(k)}$ *on the testing set* $\mathbf{X}_{test}^{(k)}$
7: $\mathbf{E}^{(c,i)} \leftarrow \hat{\mathbf{Y}}_{test}^{(c,i)} - \mathbf{Y}_{test}^{(k)}[t_{start} : t_{end}, \quad]$
8: $\mathbf{E}^{(c,i)} \leftarrow \langle \mathbf{E}^{(c,i)}, \mathbf{E}^{(c,i)} \rangle$
▷ *Update the weights for each of the* $F = N * H$ *maps*
9: **for** $j \leftarrow 1$ to F **do**
10: $\mathbf{w}^{(c,i)}[j] \leftarrow 1/mean(\mathbf{E}^{(c,i)}, j)$
▷ $mean(\mathbf{E}, j)$ *computes the mean of the* j^{th} *column of the* $\mathbf{E}$ *matrix*
11: **end for**
12: **end for**

▷ *Dynamic Adaptive Algorithm Combination*
13: **for** $j \leftarrow 1$ to F **do**
14: **for** $i \leftarrow 1$ to M **do**
▷ *Normalize weights for the* j^{th} *variable*
15: $\mathbf{w}_{norm}^{(c,i)}[j] \leftarrow \mathbf{w}^{(c,i)}[j] / \sum_i^M \mathbf{w}^{(c,i)}[j]$
▷ *Combine normalized weights using FF vector* $\mathbf{\Lambda}$
16: $\mathbf{w}_{norm}^{(c,i)}[j] \leftarrow \sum_{v=1}^{V} \mathbf{\Lambda}[v] \mathbf{w}_{norm}^{(c-v,i)}[j]$
17: **end for**
18: $\hat{\mathbf{Y}}_{DAFT-E}[t_{start} : t_{end}, j] \leftarrow \sum_{i=1}^{M} \mathbf{w}_{norm}^{(c-1,i)}[j] \hat{\mathbf{Y}}_{test}^{(c,i)}[\quad, j]$
19: **end for**

▷ *Sliding window approach to keep last* V *weight matrices* $\mathbf{w}$ *and discard the oldest one*
20: **for** $i \leftarrow 1$ to M **do**
21: **for** $v \leftarrow 1$ to V **do**
22: $\mathbf{w}_{norm}^{(c-v,i)} \leftarrow \mathbf{w}_{norm}^{(c-v+1,i)}$
23: **end for**
24: **end for**
25: **end for**

column of $\mathbf{Y}$. Each prediction task $f_{n,h}$ (line 6) is addressed by M algorithms and the M predictions are combined by weighted averaging (line 18).

Every Γ steps, the weights are returned by the inverse of the mean of the latest Γ squared forecasting errors (line 10), then normalized (line 15) and eventually regularized (line 16).

Note that, after performing the decomposition, each map can be learned independently of each other, allowing for a parallel and computationally efficient learning process. Hence, the aggregate output for each $f_{n,h}$ map is the weighted sum of the M internal algorithms, where the weights are incrementally updated over the time by adopting algebraic operations and forgetting factors.

The regularization uses a vector of V forgetting factors $\Lambda_1, \ldots, \Lambda_V$ satisfying $0 < \Lambda_v < 1, \sum_{v=1}^{V} \Lambda_v = 1$, which quantifies the contribution of the V previous cycles.

DAFT-E controls the bias/variance trade-off of the adaptive algorithm thanks to the hyperparameters Γ and $\Lambda_v, v = 1, \ldots, V$. The larger Γ and the more similar the Λ_v values, the higher the smoothness (and consequently the bias) of the forecast estimation. Such dynamic regularization process allows better robustness (and then accuracy) in front of cyclic regime changes (ramps, power generation curtailments). Given the large degree of uncertainty of highly dynamic processes, the memory-based weight update process reduces the sensitivity to recent noise values and decreases variance and instability. In practice, Γ and Λ_v values are set by considering a grid search procedure over a training portion of the historical series.

The *DAFT-E* model includes Random Forest (*RF*) (Liaw, Wiener, et al., 2002), Lazy Learning (*Lazy*) (Bontempi, Birattari, and Bersini, 1999c), and Persistence (*Naive it*). These models have been selected after a preliminary analysis due to both their low computational burden and the good generalization performances, which were shown in different real-world time series forecasting problems (Parmezan, Souza, and Batista, 2019). Furthermore, the ML models are trained with different feature sets for enhancing the generalization capability of the ensemble model. The first set, called *raw*, considers only features obtained from the raw original time series (i. e., lagged historical values) whereas the second set, named *all* also include the features obtained through feature engineering (and the corresponding lagged values).

This choice reflects the well known consideration about the ML model performance, which often depends on the considered feature sets (Domingos, 2012), even in the presence of a feature selection technique, since the original set itself may affect the result of the feature extraction.

Furthermore, in the presence of a high volatility and nonlinear time series sets, the decision maker may decide to produce a smooth feature set to better catch hidden aspects in the time series behavior. This often improves the results, but as observed from the experience, in some cases the smooth features increase the generalization capability at the detriment of an increasing bias. This happens because a smooth feature may result in more correlation to a target variable, but having lesser predictive power than raw features, since a correlated feature is not always synonym of a good predictor. In addition, the persistence model adoption supports the other models in tail event conditions, such a quasi-constant profiles.

We expect that this increases the model generalization capability, since the final forecasting is obtained by MSE-based weighted average considering the dynamic model performance over time. Particularly, the weights are updated at the end of a whole forecasting horizon span $\Gamma = H$, where the time series are processed in parallel by mRMR (De Jay et al., 2013) by extracting a feature subsets of $N_S = 5$ from a $N' \approx 2000$ feature set. The *DAFT-E* employs $V = 3$ FFs, where the weight values are $\Lambda_1 = 0.5$, $\Lambda_2 = 0.35$, and $\Lambda_3 = 0.15$. The optimal parameters are chosen after a grid search analysis on a subset of the original data.

3.4 CONCLUDING REMARKS

High variate multi-step forecasting is one of the most challenging tasks in data science and requires an extremely careful management of the bias/variance trade-off by exploring several alternatives in series encoding and forecasting. There has been a limited focus on multivariate

and multi-step-ahead forecasting solutions for time series problem in the scientific literature. Most of the available solutions either tackle the multivariate aspect (e. g., Vector Auto-Regressive (Wang, 2018) or neural-networks, both feed-forward (Chakraborty et al., 1992b) and recurrent (Hewamalage, Bergmeir, and Bandara, 2021)), with a one-step-ahead approach, or they focus on the multiple-step-ahead approach, but tackling one time series at the time, in a univariate fashion (Taieb et al., 2012). At the time of writing, to the best of our knowledge, we are not aware of any studies analyzing the performances of the different multiple-step-ahead forecasting strategies in the multivariate case.

To bridge this gap in the literature, we propose two forecasting strategies (Figure 3.10) based on the reduction/decomposition of a multivariate problem into simpler problems. Given their modular nature, the two strategies can be applied, without loss of generality, using both statistical and machine learning forecasting models.

(a) DFML strategy pseudocode. According to the algorithm chosen for the *DimensionalityReduction* step, the q factors can be forecast independently (as presented) as well as jointly.

Require: Observed matrix $\mathbf{Y}$, Number of factors q
Ensure: Forecast matrix $\widehat{\mathbf{Y}}$

1: $\mathbf{Z}$ =DimensionalityReduction($\mathbf{Y}$,q)
2: **for** $i \in \{1, \ldots, q\}$ **do**
3: $\quad \widehat{\mathbf{Z}}^{i}_{\cdot}$ =Forecast($\mathbf{Z}^{i}_{\cdot}$)
4: **end for**
5: $\widehat{\mathbf{Y}}$ =DimensionalityIncrease($\widehat{\mathbf{Z}}$)

(b) SMURF-ES strategy pseudocode. While the *FeatureEngineering* step is performed jointly for all the input time series, the following steps in the procedures are performed independently for each univariate time series $Y[i]$.

Require: Observed matrix $\mathbf{Y}$
Ensure: Forecast matrix $\widehat{\mathbf{Y}}$

1: $[\mathbf{Y}, \mathbf{Y}^{(FE)}]$ =FeatureEngineering($\mathbf{Y}$)
2: **for** $i \in \{1, \ldots, n\}$ **do**
3: $\quad \mathbf{Y}^{(FS,i)}$ =FeatureSelection($[Y[i], \mathbf{Y}^{(FE,i)}]$)
4: $\quad \widehat{Y}[i]$=Forecast($\mathbf{Y}^{(FS,i)}$)
5: **end for**

Figure 3.10: Comparison between the two proposed methodological contributions: DFML and SMURF-ES

The first strategy (DFML - Figure 3.10a) reduces the original large scale multivariate problem to a smaller scale multivariate problem, via a dimensionality reduction technique. The dimensionality reduction technique estimates a reduced number of time series (called factors), whose dynamics is representative of that of the original forecasting problem. The forecasting step is then performed on the reduced dimensional space, and eventually the forecasts are transformed back into the original higher dimensional space, through a dimensionality increase technique (inverse of the dimensionality reduction). Given its reliance on dimensionality reduction techniques, a key assumption for the DFML strategy is that the multivariate input time series should present a high correlation (e. g., spatial) among the underlying individual series.

The second strategy (SMURF-ES - Figure 3.10b), on the other hand, starts with a feature engineering process on the multivariate data to augment the original feature space. After this step, each time series to forecast is treated as an independent forecasting problem. First, a feature selection process is performed in order to retain only the most relevant predictors for the forecasting task at hand. Then, the selected features are then given as input to an ensemble model, producing the multi-step-ahead forecast for the considered time series. Finally, the independent univariate forecasts are put together in order to provide the desired multivariate forecasts.

Given their characteristics we can observe that the two strategies are complementary. The DFML is well suited for very high-dimension problems ($n > 10^2$), since the dimensionality reduction component can effectively reduce the original problem to a low-dimensional problem ($q < 10$), thus reducing of several orders of magnitude the computational cost of the forecasting process, as well as solving some computational issues arising for certain statistical techniques (e. g., VAR, SSA), preventing their applications. Moreover, the choice of heterogeneous factor estimation and forecasting techniques can help addressing the bias/variance trade-off. For instance, a non-linear recurrent factor estimation technique could reduce bias (yet increasing variance) in case of nonlinear low noise dynamics while more conventional statistical techniques may be effective in guaranteeing a lower variance (at the cost of a bias increase) in noisy settings with small number of samples.

Conversely, the SMURF-ES strategy is well-suited for low-dimensional multivariate problems ($n < 30$), since the application of the pipeline of feature engineering, selection and ensemble forecasting can improve the forecasting performance at the individual series level, at the price of an increase in computational complexity (due to the feature engineering step, increasing dimensionality and the ensembling step, requiring the training of multiple models, potentially in parallel).

Note that in this chapter we presented only the rationale and the main components of the two strategies. In the application of these strategies in real settings, many degrees of freedom need to be considered. To address this issue, we analyze in detail some specific implementations of the two strategies in the following chapters.

In Chapter 4 we assess the performance of the DFML strategy for different combinations of dimensionality reduction and forecasting techniques (Section 4.3) and on ill-conditioned problems (Section 4.4). We conclude the chapter with the assessment of extensions to the basic DFML strategy (Section 4.5), on several real and synthetic datasets.

In Chapter 5 we assess the performance of two forecasting methods derived from SMURF-ES strategy: the first one proposing an ensemble technique with an adaptive combination rule based on past forecasting error (Section 5.3) and the second one combining an error-based combination rule with a forgetting factor to favor most-recent values (Section 5.4), with a focus on wind power forecasting problem on real datasets.

4

DYNAMIC FACTOR MACHINE LEARNER - EXPERIMENTAL ASSESSMENT

Results presented in this chapter have been published in the following papers:

- Gianluca Bontempi, Yann-Aël Le Borgne, and Jacopo De Stefani (2017). "A Dynamic Factor Machine Learning Method for Multi-Variate and Multi-Step-Ahead Forecasting." In: *Proceedings of DSAA 2017, the 4th IEEE International Conference on Data Science and Advanced Analytics 2017*
- Jacopo De Stefani, Yann-Aël Le Borgne, Olivier Caelen, Dalila Hattab, and Gianluca Bontempi (Aug. 31, 2018). "Batch and Incremental Dynamic Factor Machine Learning for Multivariate and Multi-Step-Ahead Forecasting." In: *International Journal of Data Science and Analytics*. ISSN: 2364-4168. DOI: 10.1007/s41060-018-0150-x. URL: https://doi.org/10.1007/s41060-018-0150-x (visited on 09/12/2018)
- Jacopo De Stefani, Olivier Caelen, Dalila Hattab, Yann-Aël Le Borgne, and Gianluca Bontempi (2019b). "A Multivariate and Multi-Step Ahead Machine Learning Approach to Traditional and Cryptocurrencies Volatility Forecasting." In: *ECML PKDD 2018 Workshops - MIning DAta for financial applicationS (MIDAS 2018)*. Ed. by Carlos Alzate, Anna Monreale, Livio Bioglio, Valerio Bitetta, Ilaria Bordino, Guido Caldarelli, Andrea Ferretti, Riccardo Guidotti, Francesco Gullo, Stefano Pascolutti, Ruggero G. Pensa, Celine Robardet, and Tiziano Squartini. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 7–22. ISBN: 978-3-030-13463-1. DOI: 10.1007/978-3-030-13463-1_1
- Jacopo De Stefani and Gianluca Bontempi (2021b). "Factor-Based Framework for Multivariate and Multi-Step-Ahead Forecasting of Large Scale Time Series." In: *Frontiers in Big Data* 4, p. 75. ISSN: 2624-909X. DOI: 10.3389/fdata.2021.690267. URL: https://www.frontiersin.org/article/10.3389/fdata.2021.690267 (visited on 09/10/2021)

4.1 INTRODUCTION

In this chapter, we focus on the assessment of the Dynamic Factor Machine Learner strategy.

The key idea of this strategy is that the multivariate problem can be decomposed into a reduced set of univariate problems by means of a dimensionality reduction procedure. Once the decomposition has been performed, the large scale forecasting problem is transformed into smaller scale problems that can be tackled in a computationally efficient way. With the solution to the smaller scale problems, a dimensionality increase procedure is performed to obtain the solution to the original large-scale problem.

With this framework in mind, we focus on an incremental assessment of the forecasting strategy. We start by presenting the different datasets employed for the assessment, in Section 4.2.

Then, in Section 4.3 we perform an extensive comparison of different dimensionality reduction techniques (both linear and non-linear) and different forecasting techniques, from both the model-driven and data-driven families, with the aim to determine the best pairing, for several real-life forecasting task.

Moreover, in Section 4.4, we focus on the specific problem of volatility forecasting, highlighting the computational limitations of some of the state-of-the-art forecasting methods, and showing how the DFML approach can overcome these limitations.

Finally, Section 4.5 assesses the impact of two extensions on the performance of the basic DFML framework. The first one focuses on making the process of dimensionality reduction incremental (or online), in order to extend the applicability of the proposed strategy to streaming settings. The second one introduces an automatic hyperparameter selection, jointly providing the optimal parameters for both the dimensionality reduction and the forecasting component, with the aim of simplify the usage of the DFML strategy as forecasting solution.

Section 4.6 concludes the chapter summarizing the main findings from the assessment and outlining some perspectives for future research studies.

4.2 DATASETS

For our studies, we consider a set of publicly available datasets related to multivariate forecasting problems with high dimensionality ($> 10^2$ variables and $> 10^3$ samples) coming from different domains ranging from finance, to mobility and climate science. In addition, we employ a synthetic dataset in order to control the number of generated samples and variables in order to be able to perform scalability tests. Table 4.1 contains an overview of the different datasets and the corresponding empirical assessments, while the details of the different datasets and associated data sources are presented in the remainder of the section.

4.2.1 *Electricity consumption*

This dataset [1] contains 26304 samples of 321 variables. Each variable represents the hourly electricity consumption in KWh of 321 clients between 2012 and 2014 (Lai et al., 2018). This dataset has been obtained by preprocessing the original dataset (NREL, 2021) in order to remove null time series and to resample the original data (with a sampling of 15 minutes) to have an hourly frequency.

4.2.2 *Traffic usage*

This dataset [2] contains 17544 samples of 862 variables, representing 48 months of hourly data from the California Department of Transportation (Lai et al., 2018). Each variable measures the road occupancy rates (between 0 and 1) returned by sensors monitoring the San Francisco Bay area freeways during 2015-2016 (California Departement of Transport, 2021).

4.2.3 *OBU Mobility data*

This dataset [3] contains 1416 samples of 389 variables. Each variable represents the average hourly occupancy (measured by the number of trucks) of a street in the Brussels region. The original dataset (Bruxelles Mobilité and Machine Learning Group - ULB, 2021) has been preprocessed in order to remove the variables with variance smaller than 0.2, thus reducing

1 The dataset is available at https://github.com/laiguokun/ multivariate-time-series-data.
2 The dataset is available at https://github.com/laiguokun/ multivariate-time-series-data.
3 The dataset is available at https://www.kaggle.com/giobbu/belgium-obu.

Assessment	Dataset	N	n
Factor estimation and forecasting	Electricity	26304	321
	Traffic	17544	862
	OBU Mobility Data	1416	389
Volatility forecasting	CAC40	1645	40
	Cryptocurrencies	495	291
Automatic hyperparameter search and iterative factor estimation	Synthetic	1500	20, 50, 100, 200, 400, 1000
	Earth Surface Temperature	1625	100, 200
	Volatility	1489	40

Table 4.1: Summary of the different datasets employed for the assessment of the DFML strategy.

the number of variables from 4529 to 388.

4.2.4 *CAC40*

The available data consists of 1645 data points representing the stock market valuation of the 40 companies composing the French stock market index CAC40 from 02/01/2012 to 08/06/2018 (approximately 6 years and 5 months) in Opening High Low Closing (Price Data) (OHLC) format. The OHLC data is processed in order to compute the following volatility proxies: σ^0, σ^4, σ^6, $\sigma^{SD,5}$, $\sigma^{SD,10}$, $\sigma^{SD,21}$ (see Appendix A for the proxy definitions). In addition to the proxies, we include also the continuously compounded return and the volume variable (representing the number of trades in given trading day).

4.2.5 *Cryptocurrencies*

The available data comes from the Kaggle dataset "Every Cryptocurrency Daily Market Price" [4] constitued of 785,024 observation of 1644 different cryptotokens from 28/04/2013 to 06/06/2018. However the number of available datapoints per cryptotoken is inversely proportional to the lifespan of the token itself. In other words, the further we go into the past, the fewer values we have for our analysis, as depicted in Figure 4.1. For these reasons, we restricted our analysis to the period from 28/01/2017 to 06/06/2018 (495 data points) for which we have complete OHLC data for 291 tokens. The OHLC data is processed in order to compute the following volatility proxies: σ_t^0, σ_t^4, σ_t^6, $\sigma_t^{SD,5}$, $\sigma_t^{SD,10}$, $\sigma_t^{SD,21}$ (see Appendix A - Section A.2 for the proxy definitions).

4.2.6 *Synthetic cross-sectional and temporal time series*

We simulated 14 multivariate stochastic processes with cross-sectional and temporal dependencies [5] . In order to define such processes, we adapted 12 NARX processes available in literature (Bontempi, 2014) by adding cross-sectional dependencies (Table 4.2). This was

4 The dataset is available at https://www.kaggle.com/ jessevent/all-crypto-currencies
5 The dataset is available at https://github.com/gbonte/panel/tree/master/data

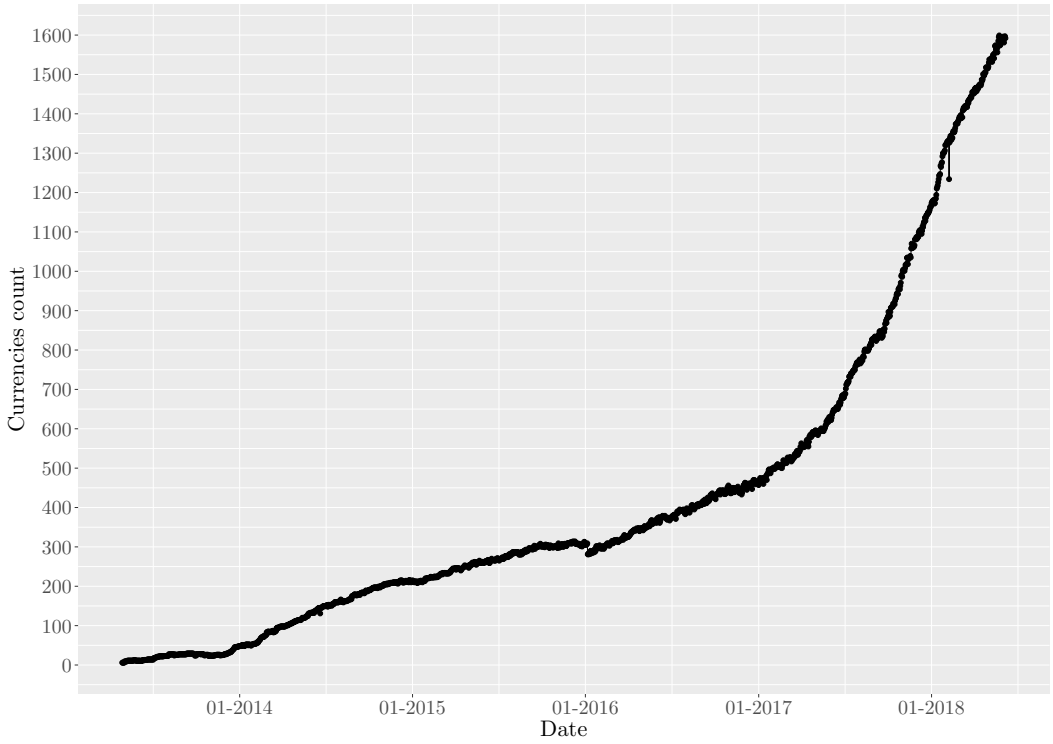


Figure 4.1: Number of available datapoints for the Cryptocurrencies dataset as a function of time

made by defining for each univariate component $Y[j], j = 1, \ldots, n$ a set $\mathcal{N}_j$ of neighboring series on which it depends. The dependance is obtained by making $Y_{t+1}[j]$ dependent on the average of the lagged values of the neighbouring series

$$\bar{Y}_t[\mathcal{N}_j] = \frac{\sum_{k \in \mathcal{N}_j} Y_t[k]}{|\mathcal{N}_j|}, i = 1, \ldots, n$$

where $|\mathcal{N}_j|$ is the size of $\mathcal{N}_j$. We considered a number of forecasting horizons $H \in \{5, 10, 20\}$ and we took $|\mathcal{N}_j| = 4$. From each stochastic process we generated a number of multivariate time series of size $N = 1500$. Each series differs in terms of the number of components ($n \in \{20, 50, 100, 200, 400, 1000\}$) and standard deviations $\sigma_w \in \{0.1, 0.2, 0.3\}$ of the Gaussian noise term. Some series are obtained by n replicates of the same stochastic process, with different initial conditions and independent noise terms. Some are obtained by mixing different stochastic processes.

4.2.7 *Earth Surface Temperature series*

We considered the Earth Surface Temperature series [6] made available by Berkeley Earth in a Kaggle dataset. We focused on the monthly `GlobalLandTemperatures ByCountry` dataset from which we derived two multivariate spatio-temporal series of size $N = 1625$. The two series refer to the temperature evolution in the n countries having the largest number of contiguous measures ($n = 100$ and $n = 200$, respectively). In the experiments we considered the forecasting horizons $H \in \{2, 5, 10, 20, 50\}$.

6 The dataset is available at http://berkeleyearth.org/data/ and http://www.kaggle.com/berkeleyearth/climate-change-earth-surface-temperature-data.

$$
\begin{aligned}
Y_{t+1}[j] =& -0.4\frac{(3-\bar{Y}_t[\mathcal{N}_j]^2)}{(1+\bar{Y}_t[\mathcal{N}_j]^2)} + 0.6\frac{3-(\bar{Y}_{t-1}[\mathcal{N}_j]-0.5)^3}{1+(\bar{Y}_{t-1}[\mathcal{N}_j]-0.5)^4} + W_{t+1}[j] \\
Y_{t+1}[j] =& (0.4 - 2\exp(-50\bar{Y}_{t-5}[\mathcal{N}_j]^2))\bar{Y}_{t-5}[\mathcal{N}_j] + \\
& (0.5 - 0.5\exp(-50\bar{Y}_{t-9}[\mathcal{N}_j]^2))\bar{Y}_{t-9}[\mathcal{N}_j] + W_{t+1}[j] \\
Y_{t+1}[j] =& (0.4 - 2\cos(40\bar{Y}_{t-5}[\mathcal{N}_j])\exp(-30\bar{Y}_{t-5}[\mathcal{N}_j]^2))\bar{Y}_{t-5}[\mathcal{N}_j] + \\
& (0.5 - 0.5\exp(-50\bar{Y}_{t-9}[\mathcal{N}_j]^2))\bar{Y}_{t-9}[\mathcal{N}_j] + W_{t+1}[j] \\
Y_{t+1}[j] =& 2\exp(-0.1\bar{Y}_t[\mathcal{N}_j]^2)\bar{Y}_t[\mathcal{N}_j] - \exp(-0.1\bar{Y}_{t-1}[\mathcal{N}_j]^2)\bar{Y}_{t-1}[\mathcal{N}_j] + W_{t+1}[j] \\
Y_{t+1}[j] =& -2\bar{Y}_t[\mathcal{N}_j]I(\bar{Y}_t[\mathcal{N}_j] < 0) + 0.4\bar{Y}_t[\mathcal{N}_j]I(\bar{Y}_t[\mathcal{N}_j] < 0) + W_{t+1}[j] \\
Y_{t+1}[j] =& 0.8\log(1+3\bar{Y}_t[\mathcal{N}_j]^2) - 0.6\log(1+3\bar{Y}_{t-2}[\mathcal{N}_j]^2) + W_{t+1}[j] \\
Y_{t+1}[j] =& 1.5\sin(\pi/2\bar{Y}_{t-1}[\mathcal{N}_j]) - \sin(\pi/2\bar{Y}_{t-2}[\mathcal{N}_j]) + W_{t+1}[j] \\
Y_{t+1}[j] =& (0.5 - 1.1\exp(-50\bar{Y}_t[\mathcal{N}_j]^2))\bar{Y}_t[\mathcal{N}_j] + \\
& (0.3 - 0.5\exp(-50\bar{Y}_{t-2}[\mathcal{N}_j]^2))\bar{Y}_{t-2}[\mathcal{N}_j] + W_{t+1}[j] \\
Y_{t+1}[j] =& 0.3\bar{Y}_t[\mathcal{N}_j] + 0.6\bar{Y}_{t-1}[\mathcal{N}_j] + \frac{(0.1 - 0.9\bar{Y}_t[\mathcal{N}_j] + 0.8\bar{Y}_{t-1}[\mathcal{N}_j])}{(1+\exp(-10\bar{Y}_t[\mathcal{N}_j]))} + W_{t+1}[j] \\
Y_{t+1}[j] =& \text{sign}(\bar{Y}_t[\mathcal{N}_j]) + W_{t+1}[j] \\
Y_{t+1}[j] =& 0.8\bar{Y}_t[\mathcal{N}_j] - \frac{0.8\bar{Y}_t[\mathcal{N}_j]}{(1+\exp(-10\bar{Y}_t[\mathcal{N}_j]))} + W_{t+1}[j] \\
Y_{t+1}[j] =& 0.3\bar{Y}_t[\mathcal{N}_j] + 0.6\bar{Y}_{t-1}[\mathcal{N}_j] + \frac{(0.1 - 0.9\bar{Y}_t[\mathcal{N}_j] + 0.8\bar{Y}_{t-1}[\mathcal{N}_j])}{(1+\exp(-10\bar{Y}_t[\mathcal{N}_j]))} + W_{t+1}[j]
\end{aligned}
$$

Table 4.2: Cross-sectional and temporal series: $\mathcal{N}_j$ denotes the indices of the set of time series which are neighbors of the jth component. $\bar{Y}_t[\mathcal{N}_j]$ stands for the average of the value of the neighboring series at time t. The covariance matrix of the Gaussian noise vector W is diagonal.

4.2.8 *Volatility series*

This dataset [7] consists of 7 multivariate volatility proxies derived from the 40 series of the French stock market index CAC40 in the period ranging from 05-01-2009 to 22-10-2014 (almost 6 years). Original data contains $N = 1489$ OHLC and Volume samples for each time series. Note that, although they are derived from the same raw data, this dataset differs from the one discussed in Section 4.2.4 by the reduced number of chosen samples, as well as a different number of considered volatility proxies. Here, we considered the following daily proxies available in the literature: σ_t^0 and the six proxies denoted by $\sigma_t^i, i = 1, \ldots, 6$. For a more detailed analysis and discussion concerning the volatility proxies and their definitions we refer the interested reader to Appendix A, more precisely Section A.1.

4.3 FACTOR ESTIMATION AND FORECASTING ASSESSMENT

The experimental study assesses and compares several implementations of the DFML, composing the different factor estimation techniques and factor forecasting techniques discussed in Section 3.2. Note, that for the sake of a robust assessment, we set the lag to $m = 3$ and the number of latent factors to $q = 3$ for all the considered methods. In order

7 The dataset is available at https://github.com/gbonte/panel/tree/master/data

to improve the readability of the results, we employ the following naming convention: the prefix *DF* is used to indicate a dynamic factor based model, whereas the *UNI* prefix is used to indicate the benchmark, univariate methods used for comparison. In both cases, the prefix is followed by the name of the employed forecasting technique.

4.3.1 *Benchmarks*

The majority of benchmark techniques used in this assessment is based on a univariate decomposition of the original n-dimensional MIMO task into n SISO forecasting tasks. The motivation of this choice is twofold. On one hand, several forecasting competitions based on real data clearly showed the competitiveness of these approaches, despite their simplicity (Hyndman, 2020). On the other hand, for several state-of-the-art multivariate techniques a MIMO implementation is either unavailable or computationally intractable due to a large number of variables (e.g. VAR) or the computational cost (e.g. deep learning based methods). Besides the Exponential Smoothing (*UNI-ES*), the Theta (*UNI-Theta*) and the Combined (*UNI-Comb*) we consider the Naive model (Section 2.3.4.1). Despite its trivial nature, in real-world tasks the Naive method is known to outperform more complex learning strategies, especially in presence of continuous sequences of constant values: for that reason it is considered as a baseline to normalize all our accuracy results in Section 4.3. The methods above are implemented with the code provided for the M4 competition (Center, 2020).

The other multivariate benchmarks are the original Dynamic Factor Model (Forni et al., 2005) (DFM, here DF-PCA-VAR) and the original DFML (Bontempi, Le Borgne, and De Stefani, 2017; De Stefani et al., 2018) (DFML$_{PC}$, here DF-PCA-Lazy-DIR and DFML$_{A}$, here DF-Base-Lazy-DIR).

4.3.1.1 *Considered DFML variants*

We test 5 different factor estimations techniques and 9 different factor forecasting techniques, for a total of 45 different models. The factor estimation techniques are listed below together with the software used for the experiments.

- PCA: the mathematical formulation is described in Section 2.4.1, while the implementation uses the basic R functions `cov` and `eigen`.
- Base: the base autoencoder (Section 2.4.2), is implemented by the `rstudio/keras` library. The architecture is symmetric with a single hidden layer of size q and a ReLU and sigmoid activation functions are used for the hidden and output layer, respectively.
- Deep: the deep autoencoder (Section 2.4.2) is implemented by the `rstudio/keras` library. The architecture is symmetric with three hidden layers (with sizes $(10, 5, q)$), a ReLU activation function for the hidden layer and a sigmoid for the output layer.
- LSTM: The LSTM autoencoder (Section 2.4.3) is implemented by the `rstudio/keras` library. The architecture is symmetric with a single hidden layer (q LSTM cells) and a ReLU and a sigmoid activation functions for the hidden and output layer, respectively.
- GRU: The GRU autoencoder (Section 2.4.3) is implemented by the `rstudio/keras` library. The architecture is symmetric with a single hidden layer (q GRU cells) and a ReLU and a sigmoid activation function for the hidden and the output layer, respectively.

For all the neural-based techniques, the maximum number of epochs used for the training is set to 50. The factor forecasting techniques are listed below together with the software used for the experiments.

- Naive, ES, Theta, Comb: their model formulation is discussed, respectively in Sections 2.3.2.1, 2.3.2.3, 2.3.2.4, 2.3.2.4. We employ the implementations made available by the M4 competition (Center, 2020). The multi-step-ahead forecast is obtained with a Recursive strategy (Section 2.2.3.2).
- VAR: the model formulation is discussed in Section 2.3.4.3, while the implementation is provided by the `vars` R library and a Recursive strategy (Section 2.2.3.2) returns the multi-step-ahead forecast.
- Lazy-DIR, Lazy-REC, MIMO: these methods denote the lazy learning (Section 2.3.3.2) with a Direct (Section 2.2.3.2), Recursive (Section 2.2.3.2) and Joint multi-step-ahead forecasting strategy, respectively. The implementation is made available in the Github library `gbonte/gbcode` by the `multisteapAhead` function with methods `lazydir,lazyiter,mimo` respectively.
- LightGBM-DIR and LightGBM-REC: these methods denote a gradient boosting technique (Section 2.3.3.4), whose implementation is provided by the `lightgbm` R library. The Direct and Recursive strategies for multi-step-ahead forecasting have been implemented by the authors.

Unless specified otherwise, we employed the default values for the forecasting techniques hyperparameters in the experiments. The entire code used to run the experiments is available in (De Stefani and Bontempi, 2021a).

4.3.2 *Experimental setup and results presentation*

We consider a rolling window approach (Tashman, 2000) using a window size of w_{tr} multivariate samples for training, and $H \in \{4, 6, 12, 24\}$ multivariate samples for validation. The window size w_{tr} is set to 2000 samples for the Electricity (Section 4.2.1) and Traffic (Section 4.2.2) datasets, while for the Mobility dataset (Section 4.2.3), the window size is 400 in order to ensure the feasibility of the rolling approach. It should be noted that this evaluation technique, proposed in (Tashman, 2000) consists of an extension of the well-known cross-validation principle for time-dependent data. All the time series are preprocessed via a z-score normalization (using the `scale` R function) and first-order differentiation to de-trend the data. For each window, a multivariate error measure, the Naive Normalized Mean Squared Error (NMMSE - Eq. 2.28) is computed using the `MEMTS` package. The statistical significance of the results is assessed via a Friedman statistical test (with post-hoc Nemenyi test - Section 2.2.3.4). In the results visualization, the methods are ordered according to their performance from left to right (the leftmost the best), while the black bar connects methods that are not significantly different (at $p = 0.05$).

4.3.3 *Mobility*

Figure 4.2 shows the overall ranking and the corresponding critical distance according to a Friedman-Nemenyi test (Demšar, 2006). Tables 4.3 and 4.4 report the average NNMSE for different horizons and groups of methods.

From the analysis of the results we can derive the following considerations:

- Statistical techniques for factor forecasting (*DF-Stat* methods) appear among the top 10 methods (Figure 4.2).
- Across all the horizons, the non-linear autoencoders (Base, Deep, LSTM, GRU) consistently outperform linear factor estimation techniques (PCA). Also, apart from the top two methods, the differences are rarely statistically significant (Figure 4.2).

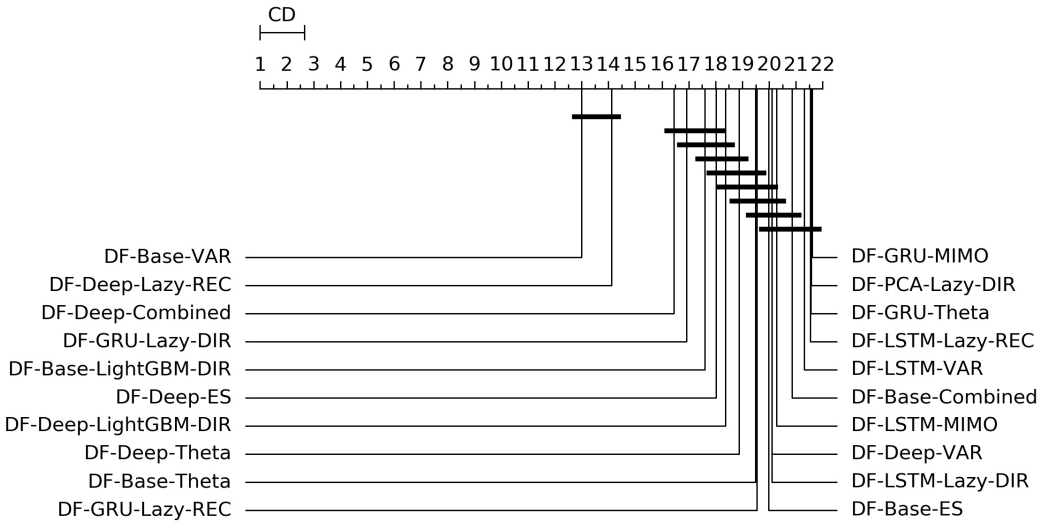


Figure 4.2: **Mobility** - Graphical representation according to (Demšar, 2006) of the results of Friedman statistical test (with post-hoc Nemenyi test) comparing the NNMSE of the best 20 methods against each other, aggregated across all horizons h. The methods are ordered according to their performance from left to right (the leftmost the best), while the black bar connects methods that are not significantly different (at $p = 0.05$).

- DFML strategies consistently outperform the Naive baseline for different horizons (Tables 4.3 and 4.4).
- The superiority of DFML over UNI-STAT methods (UNI-ES, UNI-Theta, UNI-Comb) is less clear-cut. Taking into considerations all horizons DFML is significantly better than UNI-STAT: nevertheless for large horizons, the accuracy of UNI-STAT and DFML techniques (both DF-Stat and DF-ML) tend to converge.

4.3.4 *Electricity*

Tables B.1 and B.2 report the NNMSE, averaged over all the tests sets. Figure B.4 shows the ranking and the corresponding critical distance according to a Friedman-Nemenyi test (Demšar, 2006).

On the basis of the results we can make the following considerations:

- Across all the horizons, DFML with non-linear autoencoders (Base, Deep, LSTM, GRU) is generally among the best methods (cf. Figure B.4). However, some specific factor estimation/forecasting pairs are consistently among the top performers (DF-PCA-{LAZY-DIR,LAZY-REC,MIMO} Bontempi, Le Borgne, and De Stefani, 2017; De Stefani et al., 2018).
- Apart from two combinations based on direct gradient boosting, the top 20s only includes lazy techniques (in the top 10s) and VAR (in the bottom 10s) (Figure B.4).

		H=4					H=6				
		PCA	LSTM	GRU	Base	Deep	PCA	LSTM	GRU	Base	Deep
DF-Stat	DF-ES	0.554	0.5477	0.5445	0.5583	0.5475	0.5835	0.5786	0.5767	0.596	0.5764
	DF-Theta	0.5541	0.5486	0.5445	0.5582	0.5475	0.5835	0.5789	0.5767	0.596	0.5764
	DF-Combined	0.5542	0.5481	0.5448	0.5555	0.5472	0.5837	0.5789	0.5766	0.5961	0.576
	DF-VAR	0.5514	0.5448	0.5515	0.5467	0.548	0.5789	0.5777	0.5777	0.5795	0.5768
DF-ML	DF-Lazy-DIR	0.5435	0.5471	0.5491	0.6037	0.5454	0.5773	0.5783	0.5765	0.6605	0.6161
	DF-Lazy-REC	0.5481	0.5479	0.549	0.609	0.5431	0.5793	0.5785	0.5765	0.6868	**0.5748**
	DF-MIMO	0.5546	0.547	0.5514	0.6197	**0.5383**	0.6001	0.5784	0.5764	0.6422	0.633
	DF-LightGBM-DIR	0.5705	0.5558	0.5489	0.543	0.5506	0.6025	0.5813	0.5816	0.5872	0.577
	DF-LightGBM-REC	0.5848	0.5465	0.551	0.5855	0.5491	0.6233	0.5834	0.6039	0.617	0.5796
UNI-Stat	UNI-Naive	1	1	1	1	1	1	1	1	1	1
	UNI-ES	0.5494	0.5494	0.5494	0.5494	0.5494	0.5776	0.5776	0.5776	0.5776	0.5776
	UNI-Theta	0.5494	0.5494	0.5494	0.5494	0.5494	0.5776	0.5776	0.5776	0.5776	0.5776
	UNI-Comb	0.5519	0.5519	0.5519	0.5519	0.5519	0.5794	0.5794	0.5794	0.5794	0.5794

Table 4.3: **Mobility** Naive Normalized MSE for $H \in \{4, 6\}$. The content of the cell c_{ij} represents the model using the jth dimensionality reduction technique with the ith forecasting method. The NNMSE for the Naive method is equal to 1. An NNMSE < 1 indicates that the proposed method outperforms the Naive method. Bold text denotes the best configuration for the given horizon H.

- DFML techniques generally outperform the Naive technique and the other univariate benchmarks (UNI-ES, UNI-Theta, UNI-Comb), also for longer horizons. The only exception is represented by the integration of recurrent autoencoders (LSTM, GRU) with statistical techniques (DF-Stat) which performs worse than the UNI-Stat benchmarks (Tables B.1 and B.2).

4.3.5 *Traffic*

Tables B.3 and B.4 report the NNMSE, averaged over all the tests sets. Figure B.5 shows the ranking and the corresponding critical distance according to a Friedman-Nemenyi test (Demšar, 2006).

From the analysis of the results we can make the following considerations:

- The main characteristic among the techniques outperforming the univariate benchmarks is the use of a lazy learning technique (either with the Direct (LAZY-DIR) or the Joint (MIMO) strategy) (Figure B.5).
- Another recurring forecasting technique in the top 20s is the VAR, in combination with both linear and nonlinear dimensionality reduction techniques (Figure B.5).
- DFML strategies consistently outperform the Naive baseline (Tables B.3 and B.4).
- The combination of lazy learning with a recursive technique and recurrent autoencoder tends to produce abnormal values, probably due to error propagation or vanishing/-exploding gradients problems (Table B.4).

		H=12					H=24				
		PCA	LSTM	GRU	Base	Deep	PCA	LSTM	GRU	Base	Deep
DF-Stat	DF-ES	0.6487	0.6427	0.6336	0.6666	0.6349	0.8048	0.8005	0.8017	0.8007	0.7986
	DF-Theta	0.6487	0.6426	0.6336	0.6666	0.635	0.8048	0.8005	0.8017	0.8007	0.7987
	DF-Combined	0.649	0.6428	0.6336	0.6685	0.635	0.8048	0.8005	0.8017	0.803	0.7986
	DF-VAR	0.6343	0.6359	0.6336	0.6392	0.6334	0.7996	0.7999	0.7992	0.7992	0.799
DF-ML	DF-Lazy-DIR	0.6282	0.6334	0.6335	0.7084	0.6345	0.7983	0.7998	0.7987	0.8078	0.7994
	DF-Lazy-REC	0.6822	0.6335	**0.6328**	0.7237	0.634	0.8417	0.7999	0.7997	0.8291	0.7983
	DF-MIMO	0.6558	0.6339	0.6369	0.6928	0.6431	0.8079	0.7995	0.7997	0.8086	0.8014
	DF-LightGBM-DIR	0.6794	0.6399	0.6352	0.6741	0.6348	0.796	0.7992	0.7979	**0.7955**	0.7957
	DF-LightGBM-REC	0.6798	0.6336	0.6341	0.7254	0.6375	0.8335	0.7993	0.7996	0.8296	0.7967
UNI-Stat	UNI-Naive	1	1	1	1	1	1	1	1	1	1
	UNI-ES	0.6334	0.6334	0.6334	0.6334	0.6334	0.7994	0.7994	0.7994	0.7994	0.7994
	UNI-Theta	0.6335	0.6335	0.6335	0.6335	0.6335	0.7993	0.7993	0.7993	0.7993	0.7993
	UNI-Comb	0.6346	0.6346	0.6346	0.6346	0.6346	0.7997	0.7997	0.7997	0.7997	0.7997

Table 4.4: **Mobility** Naive Normalized MSE for $H \in \{12, 24\}$. The content of the cell c_{ij} represents the model using the jth dimensionality reduction technique with the ith forecasting method. The NNMSE for the Naive method is equal to 1. An NNMSE < 1 indicates that the proposed method outperforms the Naive method. Bold text denotes the best configuration for the given horizon H.

4.3.6 *Computational time*

The total computational time (in seconds) of the different DFML techniques is represented in Figures 4.3 and 4.4, representing respectively, the computational time of the shortest and longest horizon for the dataset having the largest scale (i.e. Traffic). The total computational time includes the time required to train the factor estimation technique and the factor forecasting technique, as well as the time required to generate the forecasts. It should be noted that, while the time required to estimate the factors varies according to the selected techniques, the time allocated to factor forecasting (for a given method) is constant across the different factor estimation techniques, as they employ the same number of components, and therefore the same amount of data. As we can observe in the figure, the majority of the computational time is allocated to the factor estimation technique, with the differences between factor forecasting techniques being negligible (smaller than 1s). The only exception is represented by the LightGBM-DIR technique, where the increase in computational time is justified by the number of models to be trained which is proportional to the forecasting horizon H. The fastest technique in terms of computational time is the PCA, while the recurrent based-autoencoder (GRU and LSTM) are the slowest ones. It should be noted that, with the selected number of epochs, the upper bound for all the variants of the DFML is around 75s (Figure 4.4).

4.3.7 *Discussion*

The idea of employing neural components in the framework of a dynamic factor model has already been tested by (Nakagawa et al., 2019) for a MISO one-step-ahead prediction of the returns in the Japanese stock market and by (Kim et al., 2019) in the framework of generative modeling for image reconstruction. However, at the time of our first DFML publication, to the best of our knowledge, we are not aware of any study implementing neural components in the framework of dynamic factor model for multivariate and multistep ahead forecasting. An additional contribution is constituted by the extensive study, on multiple real datasets,

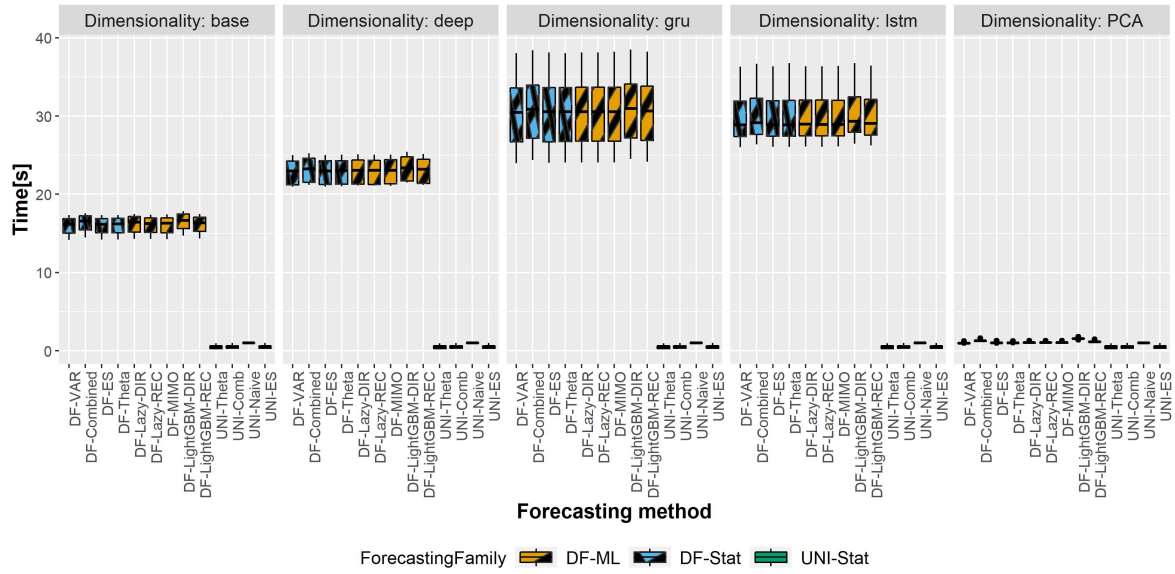


Figure 4.3: **Traffic** - Boxplots representing the distribution of the computational time (in s) across the different rolling windows for the shortest horizon ($H = 4$). Each column in the grid represents a different factor estimation technique.

of the different compositions of linear and non-linear factor estimation techniques as well as model-driven and data-driven factor forecasting techniques. We can summarize the findings from our experiments with the following considerations:

- About the *choice of a factor estimation technique* in DFML: linear techniques seem to be the most promising ones, both in terms of forecasting accuracy and computational cost (Figures B.4, B.5, 4.3, 4.4). Non-linear techniques both with and without recurrent components are comparable in terms of accuracy: nevertheless, the trade-off between the increase in accuracy and the overhead in terms of computational cost (and the consequent energetic overhead) needs to be carefully taken into account.

- About the *choice of a factor forecasting technique* in DFML: there is no clear winner between model-driven and data-driven techniques. However, two forecasting techniques: VAR (model-driven) and lazy learning (data-driven) appear to be consistently in the top performers across different datasets (Figures B.4 and B.5).

- About the *choice of a multi-step-ahead forecasting strategy* in DFML: the Direct (DIR) and Joint (MIMO) strategies consistently outperform the recurrent strategies, confirming the findings of (Bontempi and Ben Taieb, 2011) and (Taieb, 2014) (Figures B.4 and B.5).

- In the majority of the experiments, the DFML is significantly more accurate than the classical DFM (DF-PCA-VAR)), the Naive baseline and the univariate benchmarks (Figures 4.2, B.4). Note that outperforming a Naive baseline is not absolutely obvious in multivariate multi-step forecasting as discussed in publications like (Paldino et al., 2021).

- Last but not least, depending on the type of forecasting problem (cf. Section 4.3.5), univariate factor forecasting techniques still represent a competitive alternative to more complex models (Tables 4.3, 4.4, B.3, B.4).

Further experiments are foreseen to understand the impact of hyperparameters like the number of components q or the embedding order of the machine learning models m. In

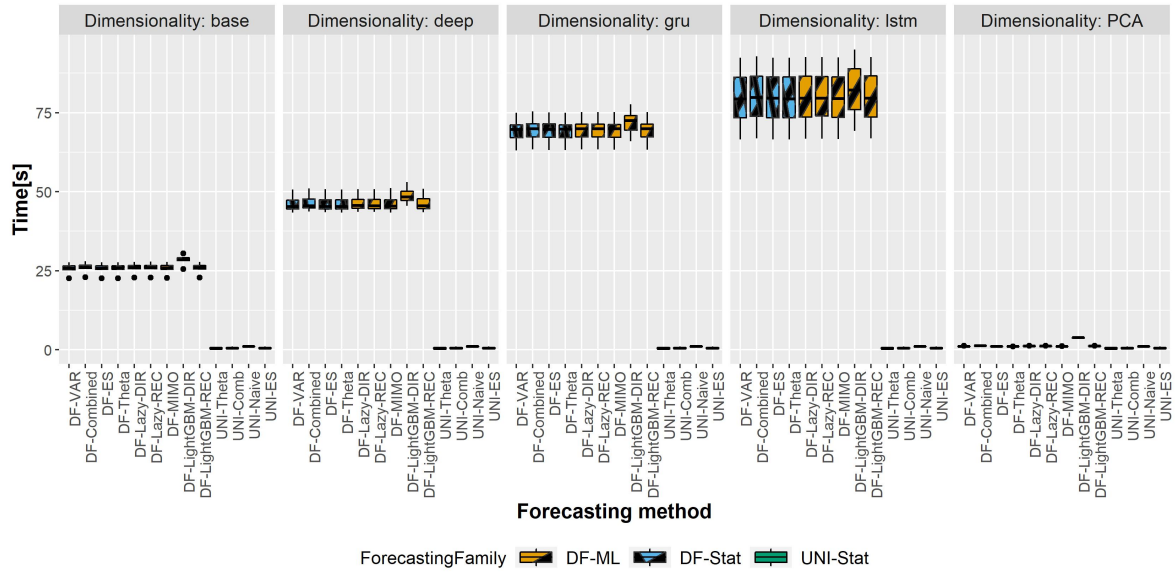


Figure 4.4: **Traffic** - Boxplots representing the distribution of the computational time (in s) across the different rolling window for the longest horizon ($H = 24$). Each column in the grid represents a different factor estimation technique.

addition, the choice of a problem specific neural network architecture, fine-tuning of the parameters, as well as longer training times could further improve the performances of the neural-based techniques, if the problem setting allows it.

4.4 VOLATILITY FORECASTING

This section focus on the comparison of the performances of the DFML framework on two settings: a well-conditioned ($n << N$) and an ill-conditioned ($n \sim N$) forecasting setting.

The well-conditioned forecasting setting is represented by the CAC40 dataset, presented in Section 4.2.4, where the 40 stocks composing the CAC40 market index have to be forecast, given 1645 available daily data samples, while the ill-posed forecasting problem is represented by the Cryptocurrency dataset, presented in Section 4.2.5, where after pre-processing, the problem is presented with high dimensionality ($n > 10^2$) and a reduced number of samples ($N = 495$). Even though this combination of high dimensionality and reduced data availability is particularly challenging for some of the considered multivariate benchmark techniques (i.e. VAR, DSE, SSA), relying on matrix decomposition techniques for parameters estimation, sometimes even preventing a successful parameter estimation, the modular architecture of the DFML, in combination with different linear/non-linear factor estimation techniques still allows the problem to be computationally tractable.

4.4.1 *Benchmarks*

The experimental study assessed and compared several multivariate forecasting methods, both local and global. The methods are listed below together with the software used for the experiments. Note that, for the sake of assessment, we set the lag $m = 2$ and the maximum number of latent factors to $q = 3$ for all methods, unless specified otherwise.

1. Naive: univariate baseline method (Section 2.3.2.1) using the last observed value for each time series as prediction for the following H steps.

2. UNI: univariate multi-step-ahead Direct forecasting Bontempi and Ben Taieb, 2011 of each individual series with a feature selection process based on correlation.

3. PLS: partial-least-squares forecasting (Section 2.3.4.4) implemented by the function `mvr` of the R package `pls`. The optimal values for the size of the input space and the number of principal components q is determined through an out-of-sample criterion.

4. RNN: recurrent neural network (Section 2.3.5.1) implemented by the `keras_predict` function of `kerasR`, the R keras interface to the `keras` Deep Learning library for Theano. The network is a fully-connected RNN with 10 hidden units. Since an automated setting of the number of units would not have been feasible due to an excessive computational time, this number has been set on the basis of trial and error over a small number of synthetic series.

5. LSTM: As RNN, the model is a fully connected RNN, with 10 hidden units implemented using `kerasR`. It differs from RNN as it employs LSTM cells (Hochreiter and Schmidhuber, 1997) in the hidden layer, instead of regular neurons.

6. DFM: linear DFM where PCA is used for factor estimation, the number of factors is set to q and the forecasting of the factors is carried out with a VAR method implemented by the `estBlackBox` function of the R package `dse`. The batch PCA is computed using the base R `eigen` function.

7. DFML$_{PCA}$: DFML framework where PCA is used for factor estimation, the number of factors is set to q and the forecasting of each factors is carried out independently in a univariate fashion using a local learning predictor (lazy learning (Bontempi and Ben Taieb, 2011)) and a multi-step-ahead Direct strategy.

8. DFML$_A$: Similar configuration as DFML$_{PCA}$ with the use of a feed-forward autoencoder instead of PCA in the process of factor estimation.

9. DFML'$_{PCA}$: Similar configuration as DFML$_{PCA}$ with the addition of the automatic selection strategy (described in (Bontempi, Le Borgne, and De Stefani, 2017)): the number of factors (in the range $[1, q]$) and the multi-step-ahead strategy (among Direct, Iterated and MIMO) and the lag m are selected by an out-of-sample strategy carried out on the training set.

4.4.2 *Experimental setup and results presentation*

For each multivariate dataset we performed time series cross-validation following a rolling origin strategy (Tashman, 2000). The size of the training set is $2N/3$ and a sequence of 50 different test sets of length H is considered. For each test set, all methods are assessed in terms of the average Normalized Mean Squared Error (Eq. 2.28)

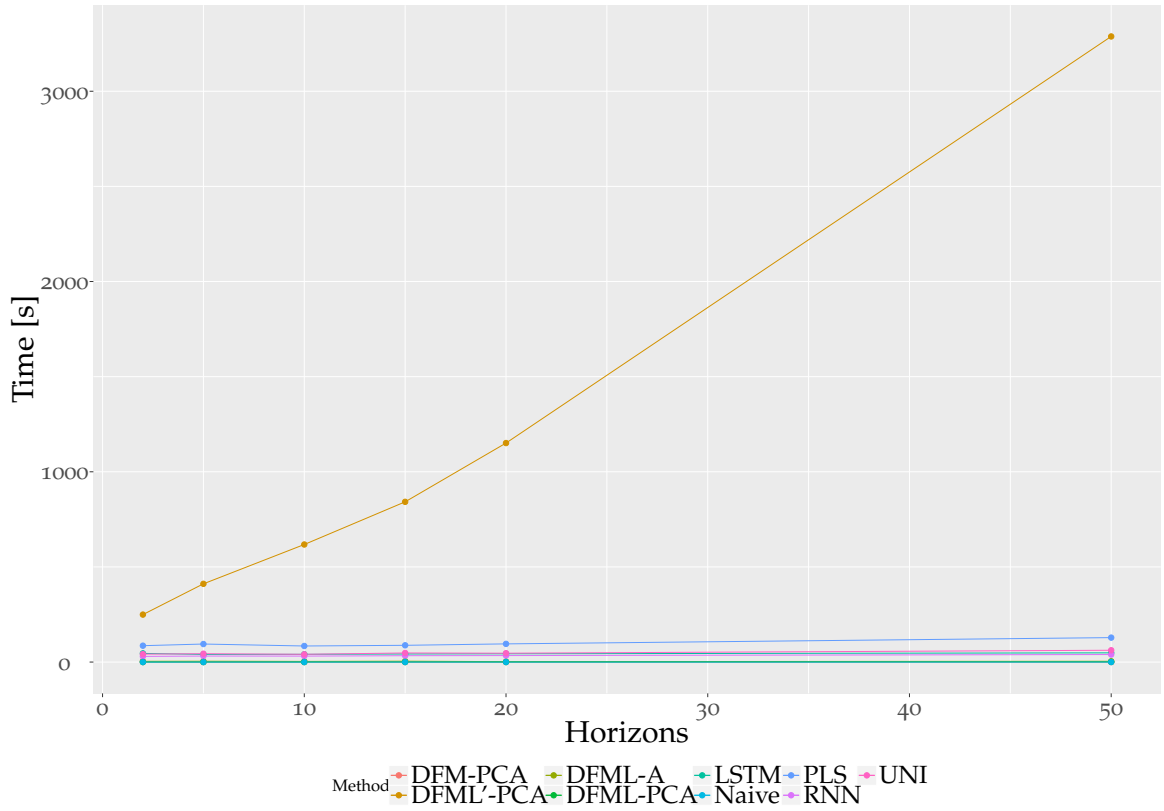
We focus on two measures to benchmark our methods: forecasting accuracy and computational time. The accuracy is presented in a tabular form (Table 4.6, 4.5), comparing the DFML framework with statistical benchmark techniques (Naive, UNI, PLS, first three columns) and representation learning based benchmark techniques (LSTM and RNN, last two columns), across different forecasting horizons H and different datasets corresponding to the different volatility proxies (σ^0, σ^4, σ^6, $\sigma^{SD,5}$, $\sigma^{SD,10}$, $\sigma^{SD,21}$, cf. Section A.1), in terms of NMSE. The computational time is presented graphically (Figure 4.5), plotting the computational time of model training and forecast, as a function of the chosen forecasting horizon, across the two considered case studies: CAC40 (a) and Cryptocurrencies (b).

4.4.3 *Cryptocurrencies*

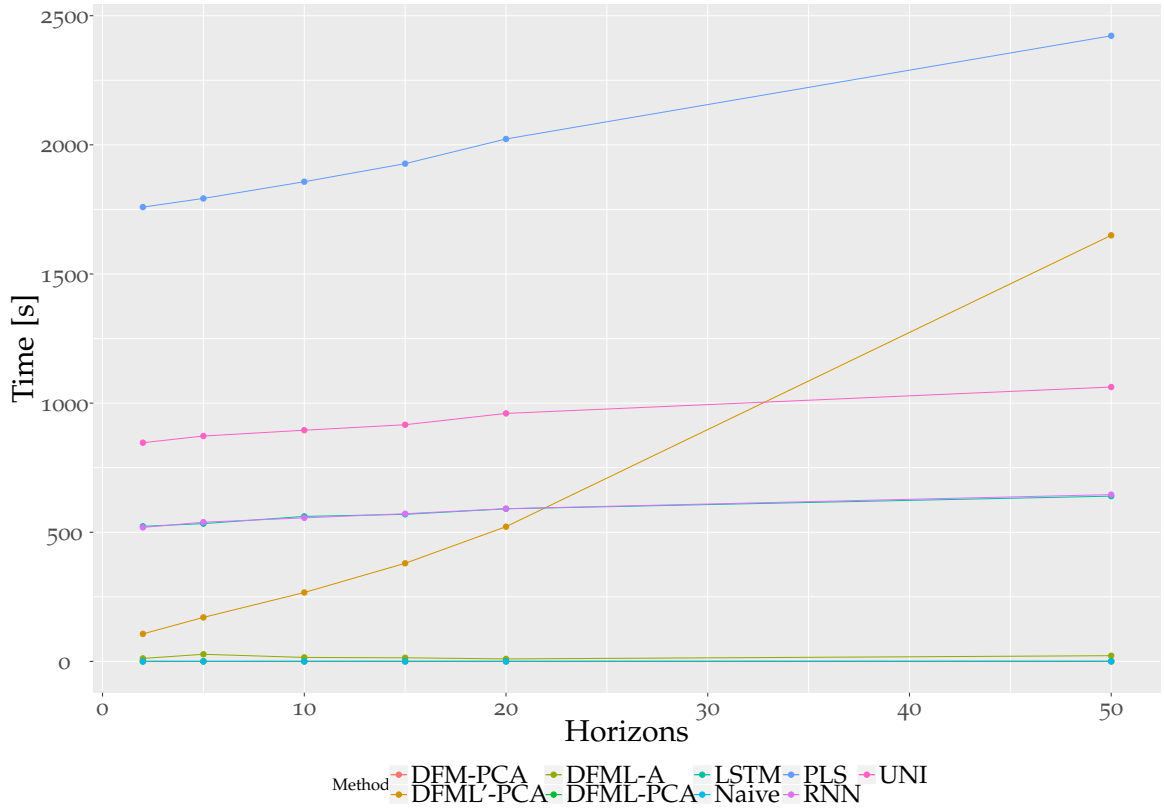
While dealing with high dimensionality ($n = 291$) coupled with a relatively low number of observations ($N = 495$), as in the case of the Cryptocurrency dataset (Table 4.5), using the σ_t^i family of proxies, the DFML, even without hyperparameter optimisation, clearly outperforms all the concurrent methods. It should also be noted that some methods tested in the original DFML paper ((Bontempi, Le Borgne, and De Stefani, 2017)) (i.e. VAR, DSE, SSA) could not be tested due to numerical problems related to the limited number of available observations. The performances of DFML are mitigated while using proxies coming from the $\sigma_t^{SD,w}$ family, where the performance of the Naive method improves, even for forecasting horizons up to 20 steps ahead, as the smoothing provided by the window size parameter w increases. In both the cases, a linear dimensionality reduction technique with no optimization (DFM, DFML$_{PCA}$) is shown to improve the performances of the forecaster, compared to nonlinear (DFML$_A$) and optimized (DFML'$_{PCA}$) ones.

4.4.4 *CAC40*

A similar ranking among the methods is observed in the case of the CAC40 dataset (Table 4.6), characterized by a lower dimensionality ($n = 40$) but an higher number of points ($N = 1641$). Here we can observe a generally higher average normalized NMSE, indicating a higher complexity of the forecasting problem. For the σ_t^i family, PLS and DFM appear as competitive alternatives of the DFML, especially for longer horizons ($H > 15$). As in the previous case, for the $\sigma_t^{SD,w}$ family of proxies, the performances of the DFML family are affected by the value of the smoothing factor w, where, the higher the smoothing factor is, the less effective the DFML becomes for shorter horizons, with the Naive method becoming the best one, while still maintaining good forecasting accuracy for longer horizons.



(a)



(b)

Figure 4.5: Total computational time (model training + forecast) of the tested methods on the CAC40 - σ^4 (a) ($n = 40$) and Cryptocurrencies - σ^4 (b) ($n = 291$) dataset-proxy combination.

Dataset	H	Naive	UNI	PLS	DFM	$DFML_A$	$DFML_{PCA}$	$DFML'_{PCA}$	LSTM	RNN
σ_t^0	2	0.988	0.660	0.630	0.595	0.631	**0.594**	0.596	0.630	0.670
	5	0.982	0.646	0.613	0.579	0.613	**0.576**	0.588	0.605	0.656
	10	1.042	0.608	0.581	0.543	0.575	**0.539**	0.538	0.570	0.615
	15	1.172	0.602	0.584	0.540	0.569	**0.537**	0.547	0.563	0.599
	20	1.247	0.579	0.555	0.515	0.544	**0.512**	0.514	0.540	0.593
	50	1.024	0.517	0.503	**0.451**	0.483	**0.451**	0.466	0.479	0.521
σ_t^4	2	0.831	0.607	0.602	0.540	0.611	**0.528**	0.543	0.585	0.647
	5	0.816	0.598	0.585	0.521	0.580	**0.510**	0.522	0.559	0.638
	10	0.945	0.582	0.579	0.505	0.564	**0.491**	0.494	0.542	0.590
	15	0.924	0.582	0.592	0.508	0.565	**0.495**	0.498	0.551	0.580
	20	1.061	0.578	0.584	0.501	0.554	**0.489**	10.969	0.539	0.575
	50	0.950	0.553	0.563	0.474	0.524	**0.472**	0.476	0.510	0.543
σ_t^6	2	0.946	0.587	0.588	0.528	0.580	**0.512**	0.527	0.547	0.613
	5	1.103	0.561	0.578	0.507	0.551	**0.479**	0.480	0.531	0.587
	10	1.101	0.583	0.590	0.516	0.579	**0.499**	0.500	0.553	0.591
	15	1.041	0.592	0.616	0.525	0.574	**0.505**	0.509	0.554	0.591
	20	1.000	0.589	0.592	0.522	0.568	**0.507**	0.509	0.551	0.586
	50	1.185	0.557	0.582	0.481	0.530	0.481	**0.474**	0.514	0.560
$\sigma_t^{SD,5}$	2	**0.269**	0.351	0.648	0.499	0.662	0.500	0.524	0.647	0.739
	5	**0.511**	0.533	0.674	0.519	0.668	0.514	0.534	0.647	0.717
	10	0.719	0.612	0.669	0.523	0.647	**0.516**	0.534	0.635	0.790
	15	0.818	0.627	0.662	0.527	0.638	**0.520**	0.523	0.616	0.794
	20	0.852	0.636	0.653	0.517	0.629	**0.514**	0.526	0.614	0.771
	50	0.974	0.611	0.636	**0.484**	0.577	0.497	0.481	0.558	0.748
$\sigma_t^{SD,10}$	2	**0.113**	0.258	0.754	0.491	0.756	0.494	0.534	0.722	0.796
	5	**0.238**	0.415	0.769	0.501	0.751	0.504	0.541	0.728	0.838
	10	**0.466**	0.598	0.781	0.513	0.746	0.507	0.543	0.719	1.027
	15	0.606	0.668	0.780	0.526	0.737	**0.514**	0.558	0.713	0.898
	20	0.668	0.706	0.777	0.523	0.741	**0.522**	0.574	0.701	0.911
	50	0.891	0.726	0.778	**0.514**	0.683	0.547	0.533	0.657	0.907
$\sigma_t^{SD,21}$	2	**0.052**	0.203	0.989	0.493	0.992	0.493	0.570	0.899	1.034
	5	**0.108**	0.316	0.986	0.501	0.977	0.504	0.571	0.864	1.144
	10	**0.199**	0.522	0.987	0.513	0.958	0.514	0.573	0.891	1.065
	15	**0.295**	0.658	0.988	0.523	0.963	0.528	0.572	0.848	1.340
	20	**0.397**	0.748	0.990	0.535	0.874	0.544	0.571	0.863	1.117
	50	0.775	0.833	1.014	**0.586**	0.882	0.649	0.612	0.808	1.252

Table 4.5: Cryptocurrencies - Volatility time series: NMSE (averaged over all the continuation sets) of the different forecasting methods. The bold notation is used to highlight all techniques which are not significantly worse (p-value=0.05) than the one with the lowest NMSE score.

Dataset	H	Naive	UNI	PLS	DFM	$DFML_A$	$DFML_{PCA}$	$DFML'_{PCA}$	LSTM	RNN
σ_t^0	2	1.332	1.047	0.972	0.969	0.987	**0.962**	1.010	1.018	1.006
	5	2.177	1.916	1.826	1.857	1.872	1.838	**1.822**	1.849	1.865
	10	1.438	1.246	**1.157**	1.173	1.184	1.164	1.155	1.164	1.184
	15	2.499	1.304	1.220	1.220	1.227	**1.209**	1.222	1.219	1.242
	20	1.566	1.227	1.155	1.153	1.163	1.174	**1.146**	1.160	1.160
	50	2.026	1.221	1.136	1.135	1.144	1.134	**1.120**	1.160	1.164
σ_t^4	2	0.585	0.504	0.463	**0.433**	0.521	0.434	0.450	0.564	0.496
	5	2.295	1.347	1.318	1.292	1.356	**1.268**	1.275	1.328	1.346
	10	1.047	1.003	0.948	0.936	0.991	**0.911**	0.946	1.014	1.018
	15	1.372	1.132	1.078	1.067	1.118	**1.048**	1.071	1.126	1.120
	20	1.272	1.023	0.948	0.926	0.977	**0.908**	0.933	1.010	1.007
	50	1.111	1.036	0.936	0.942	0.987	**0.919**	0.981	1.052	1.042
σ_t^6	2	1.780	0.854	0.805	0.776	0.859	0.767	**0.758**	0.852	0.822
	5	1.859	1.800	1.750	**1.741**	1.809	1.747	1.715	1.781	1.770
	10	1.264	1.171	1.106	1.102	1.154	**1.083**	1.118	1.149	1.139
	15	1.222	1.074	**1.001**	**0.999**	1.049	**1.001**	1.011	1.093	1.046
	20	1.332	1.185	**1.103**	1.107	1.156	1.108	1.116	1.172	1.170
	50	1.280	1.188	1.112	1.098	1.139	**1.089**	1.126	1.206	1.177
$\sigma_t^{SD,5}$	2	**0.276**	0.649	0.834	0.783	0.877	0.787	0.769	0.823	0.864
	5	**1.122**	1.275	1.304	1.289	1.355	1.242	1.215	1.329	1.352
	10	1.329	1.199	1.163	1.139	1.167	**1.095**	1.162	1.131	1.201
	15	1.408	1.149	1.095	1.068	1.113	**1.064**	1.066	1.111	1.134
	20	1.576	1.215	1.154	1.133	1.166	**1.141**	1.150	1.203	1.182
	50	2.584	1.292	1.316	1.444	**1.184**	1.243	1.192	1.229	1.273
$\sigma_t^{SD,10}$	2	**0.453**	0.667	0.901	0.805	0.964	0.805	0.788	0.827	0.881
	5	**0.698**	0.886	1.018	0.932	1.073	0.934	0.927	0.910	1.009
	10	1.133	1.010	1.044	**0.970**	1.073	1.005	1.005	1.000	1.104
	15	1.495	1.065	1.140	1.292	1.271	1.092	**1.013**	1.066	1.032
	20	1.642	1.141	1.181	1.340	1.223	1.145	**1.078**	1.108	1.178
	50	1.916	1.258	1.233	1.256	1.158	**1.144**	1.171	1.310	1.338
$\sigma_t^{SD,21}$	2	**0.033**	0.306	0.747	0.509	0.772	0.510	0.561	0.776	0.725
	5	**0.123**	0.372	0.732	0.530	0.736	0.595	0.566	0.867	0.716
	10	**0.346**	0.520	0.808	0.660	0.853	0.682	0.673	0.992	0.932
	15	**0.608**	0.680	0.862	0.771	0.893	0.795	12.315	0.970	0.868
	20	**0.827**	**0.827**	0.923	0.905	0.890	0.840	0.777	1.010	1.256
	50	1.603	1.259	1.210	1.357	1.109	**1.076**	1.311	1.282	1.585

Table 4.6: CAC40 - Volatility time series: NMSE (averaged over all the continuation sets) of the different forecasting methods. The bold notation is used to highlight all techniques which are not significantly worse (p-value=0.05) than the one with the lowest NMSE score.

4.4.5 *Computational time*

In addition to forecasting accuracy, we also analyzed the total computational time required to produce a forecast. The total computational time is obtained by summing up the time required to train the considered model and the time needed to generate a forecast. Figure 4.5 shows that, for low dimensionalities ($n = 40$) the total computational time of the different techniques is comparable, and independent of the forecasting horizon, except for the optimized DFML'$_{PCA}$, where the comparison of different forecasting strategies require a computational time proportional to the length of the forecasting horizon. On the other hand, for higher dimensionalities ($n > 40$), the computational time required to train multiple univariate models (UNI), neural based models (RNN and LSTM) and PLS increases considerably due to the increase of both dimensionality and forecasting horizons, while DFML models, thanks to the dimensionality reduction component, maintain a reduced computational time regardless of the forecasting horizon.

4.4.6 *Discussion*

The empirical analysis shows that DFML is able to produce accurate volatility forecasts, especially in the case of high-dimensional noisy series (i.e. Cryptocurrencies dataset) with non-smoothed volatility proxies σ^i, by summarizing well the intrisic market correlations in a reduced number of factors. However, the presence of a smoothing factor (as in the $\sigma^{SD,w}$ proxies family) is shown to worsen the performances of the DFML methods. Moreover, we have shown that, thanks to the dimensionality reduction component, DFML methods can produce multi-step ahead forecasts with the same accuracy as concurrent methods with a great reduction in terms of computational cost.

4.5 ITERATIVE FACTOR ESTIMATION AND AUTOMATIC HYPERPARAMETER SELECTION

This section focuses on analyzing the impact of two extensions to the basic DFML framework: iterative factor estimation and automatic hyperparameter selection (Section 3.2.4). The assessment is performed over a set of synthetic (Section 4.2.6) and real datasets (Sections 4.2.7 and 4.2.8) datasets, through ablation studies, comparing the forecasting performance of the strategy both with and without the considered extension.

4.5.1 *Benchmarks*

The experimental study assessed and compared several multivariate forecasting benchmarks, both data-driven and model driven approaches. [8] The methods are listed below together with the software used for the experiments. Note that, for the sake of assessment, we set the lag to $m = 2$ and the maximum number of latent factors to $q = 3$ for all methods.

1. DFM: linear Dynamic Factor Model where PCA is used for factor estimation, the number of factors is set to q and the forecasting of the factors is carried out with a VAR method implemented by the `estBlackBox` function of the R package `dse`. The batch PCA is computed using the base R `eigen` function whereas the tested incremental PCA method is implemented by the function `incRPCA` of the R package `onlinePCA`. The initialisation of the components for incremental PCA is performed with $\frac{2}{3}$ of the available data.

2. DFML$_{PC}$: machine learning Dynamic Factor Model where PCA is used for factor estimation, the number of factors is set to q and the forecasting of each factors is carried out in a univariate manner using a local learning predictor (lazy learning (Bontempi, Birattari, and Bersini, 1999a; Bontempi and Ben Taieb, 2011)) and a multi-step-ahead Direct strategy. The PCA estimation is performed in the same manner as DFM.

3. DFML'$_{PC}$: Similar configuration as DFML$_{PCA}$ with the addition of the automatic selection strategy (described in Fig. 3.5): both the number of factors (in the range $[1, q]$) and the multi-step-ahead strategy (among Direct, Iterated and MIMO) are selected by an out-of-sample strategy carried out on the training set.

4. DFML$_A$: Similar configuration as DFML$_{PC}$ with the use of a feed-forward autoencoder instead of PCA in the process of factor estimation.

5. DFML'$_A$: Similar configuration as DFML'$_{PC}$ with the use of a feed-forward autoencoder instead of PCA in the process of factor estimation.

6. RNN: recurrent neural network implemented by the `keras_predict` function of `kerasR`, the R keras interface to the `keras` Deep Learning library for Theano. The network is a fully-connected RNN with 10 hidden units. Since an automated setting of the number of units would not have been feasible for computation time reasons, this number has been set on the basis of trial and error over a small number of synthetic series.

7. DSE: VAR method implemented by the `estBlackBox` function of the R package `dse`. This function implements an automatic model selection procedure based on information criteria (Gilbert., 1993).

8. PLS: partial-least-squares forecasting implemented by the function `mvr` of the R package `pls`. An out-of-sample criterion is used to select the size of the input space and the number of principal components.

8 The experimental sessions can be reproduced by means of the R code available in the github repository `https://github.com/gbonte/panel`.

9. UNI: univariate multi-step-ahead Direct forecasting of each individual series with a preliminary selection of the three most correlated features.

10. VAR: VAR method implemented by the function `VARpred` of the R package `MTS` for Multivariate Time Series (Tsay, 2014).

11. SSA: SSA method implemented by the function `rforecast` of the R package `Rssa` for Singular Spectrum Analysis (Golyandina et al., 2015).

12. NAIVE: baseline method using the last observed vector as prediction for the following H steps.

4.5.2 *Experimental setup and results presentation*

For each multivariate time series we performed a number of training and test tasks by following a rolling origin strategy (Tashman, 2000). The size of the training set is $N/3$ and a number of different continuation sets of length H are considered. For each continuation set, all methods are assessed in terms of the average Normalized Mean Squared Error (Eq. 2.28)

Tables B.8, B.9, B.10 report the NMSE (averaged over all continuation sets) of all the compared methods for the case studies in Sections 4.2.6, 4.2.7 and 4.2.8, respectively. The bold notation is used to highlight all techniques which are not significantly worse (p-value=0.05) than the one with the lowest NMSE score. Missing scores (or missing columns) in the Tables of results are due to the excessive computational time required by the corresponding technique for the given task or to numerical problems (e.g. in VAR). In order to suggest how the computational time of some techniques scales with the dimensionality we show in Fig. 4.6 the computational training time normalized with respect to the one of DFML'$_{PC}$ for different values of n.

4.5.3 *Batch versus iterative PCA*

This section discusses the interest of adopting an incremental PCA approach for the Volatility, Synthetic and Temperature datasets, using a 100-fold rolling origin (Tashman, 2000) cross-validation procedure, as opposed to the 10-fold one of the other experiments. As shown in Table 4.7, iterative PCA reduces the time required to compute the principal components ($Time_{PCA}$) for all the considered datasets. This is particularly evident for large dimensionality (e.g. $n \geq 400$) where the computation of PCA accounts for most of the total computational time.

On the other hand, in terms of forecasting accuracy, we can conclude that:

- For the *Temperature* datasets (Table B.6), while dealing with 100 variables, DFML outperforms DFM for each $H > 2$, while with 200 variables this only occurs for $h = 50$. Nevertheless, the usage of iterative (online) PCA brings improvements to DFML for $h > 2$ in both cases.

- For the *Volatility* datasets (Table B.7), DFML methods perform better than the corresponding DFM method for $H > 10$. Even though, for the same forecasting horizons, the *Direct-batch* combination outperforms all the other methods in the majority of the cases, for the other combinations of both DFML and DFM, iterative PCA generally improves with respect the batch one for $\sigma_4, \sigma_5, \sigma_6$. For DFM methods, the use of iterative PCA improves the global accuracy also on shorter horizons.

- For the *Synthetic* datasets (Table B.5), DFML outperforms DFM for each H, except the cases for 1000 variables and $H = 50$. Here, while dealing with more than 50 variables, the usage of iterative PCA brings improvements to both DFML and DFM for every forecasting horizon.

4.5.4 *Manual versus automatic hyperparameter search strategy*

This section discusses the interest of implementing an automatic hyperparameter search strategy for the Volatility, Synthetic and Temperature datasets. In the experimental session we considered the forecasting horizons $H \in \{2, 5, 10, 20, 50\}$, performing a 10-fold rolling origin (Tashman, 2000) cross-validation procedure for each considered horizon H.

4.5.4.1 *DFML versus state-of-art*

From the analysis of the results we can derive the following considerations:

- Methods based on the DFM principle appear consistently among the most accurate ones. In particular the comparison with UNI shows how the creation of a small number of latent variables is extremely advantageous in synthetic as well as real tasks, even for moderate dimensions (e.g. $n = 20$).

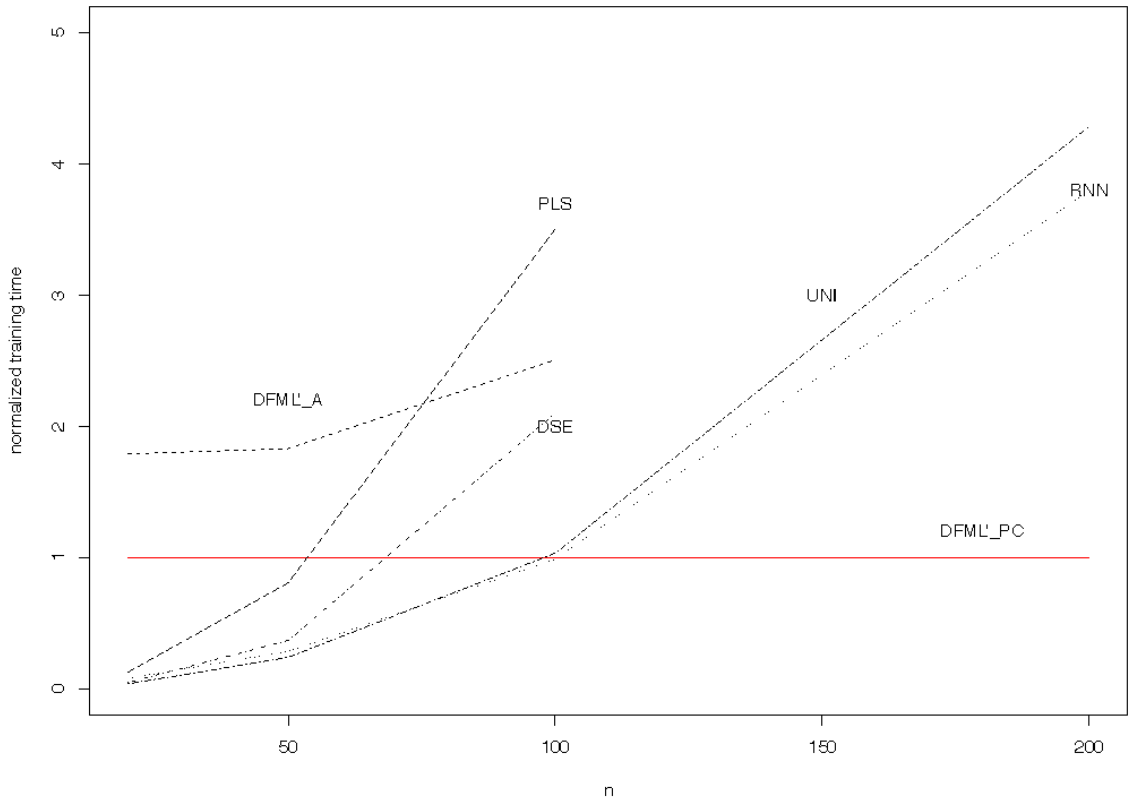


Figure 4.6: Computational training times normalized with respect to DFML'$_{PC}$. Note that the training time of SSA is not reported since the current R code demands the retraining of the model for each forecast. VAR training time is one order of magnitude smaller than the ones of DFML'$_{PC}$ and DSE (since VAR makes no order selection) but the method encounters numerical problems for $n \geq 200$.

		$Time_{PCA}$		$Time_{Total}$							
				$DFML_{PC}$						DFM	
				Direct		Iterated		MIMO			
Dataset	n	Batch	Online	Batch	Online	Batch	Online	Batch	Online	Batch	Online
σ_0	40	0.0018	0.0012	0.1532	0.1513	0.1535	0.1537	0.3232	0.3246	0.0994	0.0971
σ_1	40	0.0018	0.0011	0.1519	0.1525	0.1532	0.1514	0.3225	0.3249	0.0995	0.0948
σ_2	40	0.0018	0.0011	0.1502	0.1529	0.1548	0.1541	0.3290	0.3321	0.1130	0.1118
σ_3	40	0.0020	0.0011	0.1536	0.1566	0.1566	0.1563	0.3290	0.3279	0.1118	0.1044
σ_4	40	0.0018	0.0011	0.1507	0.1551	0.1558	0.1513	0.3312	0.3321	0.1128	0.1064
σ_5	40	0.0018	0.0012	0.1533	0.1562	0.1573	0.1528	0.3294	0.3318	0.1109	0.1090
σ_6	40	0.0018	0.0012	0.1544	0.1564	0.1580	0.1547	0.3329	0.3353	0.1107	0.1114
Synthetic	20	0.0007	0.0011	0.1560	0.1582	0.1567	0.1569	0.3405	0.3400	0.1069	0.1088
Synthetic	50	0.0025	0.0014	0.1591	0.1593	0.1550	0.1575	0.3335	0.3372	0.1114	0.1091
Synthetic	100	0.0087	0.0013	0.1617	0.1524	0.1613	0.1515	0.3373	0.3337	0.1186	0.1076
Synthetic	200	0.0331	0.0016	0.1829	0.1514	0.1832	0.1505	0.3594	0.3266	0.1381	0.1043
Synthetic	400	0.1329	0.0030	0.2793	0.1485	0.2779	0.1477	0.4537	0.3236	0.2344	0.1004
Synthetic	1000	0.8784	0.0087	1.0155	0.1481	1.0158	0.1457	1.1833	0.3158	0.9787	0.0990
Temperature	50	0.0047	0.0020	0.2946	0.3011	0.3058	0.2969	0.6767	0.6723	0.2032	0.1983
Temperature	100	0.0094	0.0016	0.1722	0.1615	0.1692	0.1640	0.3686	0.3595	0.1225	0.1147
Temperature	200	0.0350	0.0017	0.1947	0.1598	0.1944	0.1582	0.3799	0.3493	0.1413	0.1065

Table 4.7: Computational time (in seconds) of the analyzed forecasting methods (averaged across all the forecasting horizons). $Time_{PCA}$ denotes the computational time required to estimate the principal components, whereas $Time_{Total}$ includes both the PCA calculation and the forecasting time.

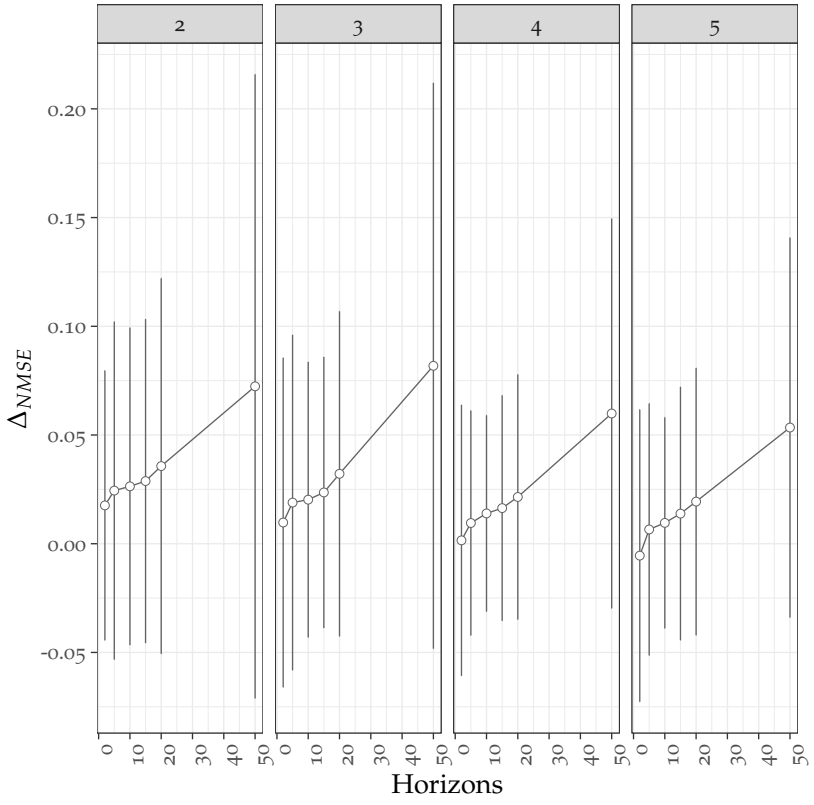
- If we restrict to consider linear approaches, DFM techniques consistently outperform their competitors (e.g. DSE, VAR, SSA or PLS).
- The automatic selection strategy improves significantly the perfomance of the DFM strategy For the two real case studies and for the synthetic benchmark (up to $n = 400$). While the simple usage of nonlinear prediction models in DFML_{PC} does not lead to major improvements, the introduction of automatic model selection of the number of factors and of the multi-step strategy makes often DFML'_{PC} the most accurate technique.
- In spite of the nonlinear nature of most series (notably the synthetic ones) the use of the autoencoder instead of PCA does not have significant impact. This is probably due to the excessive variance of the method.
- Recurrent neural network techniques are often outperformed by the DFM strategy. Note that no model selection technique is used for this method but that their implementation would have been difficult for lack of guidelines and very expensive in terms of computation requirements.
- When the dimensionality of the series becomes very high ($n = 1000$), linear DFM methods outperform all the other methods. This can be interpreted in terms of bias/variance tradeoff: though DFML'_{PC} has smaller bias than DFM, its variance tends to increase with the dimensionality of the problem. For very large n, biased yet low variant techniques like DFM are preferable.
- It is worth remarking that some of the techniques we presented are able to perform very accurate forecasting (and significantly better than random) also for tasks reputed

to be very complex (e.g. volatility forecasting), very long horizons (e.g. $H = 50$ for the Earth temperature forecasting) and very large dimensionality.

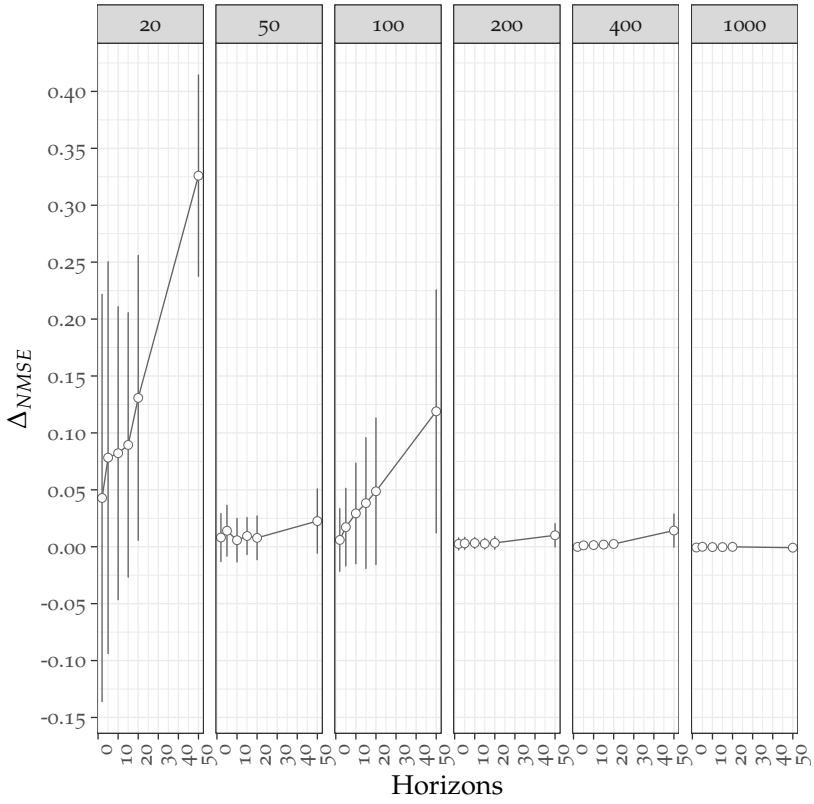
4.5.4.2 *DFML vs DFM*

The aim of this section is to focus on Dynamic Factor models, and to analyze in more details the differences between a traditional dynamic factor model (DFM) and the proposed machine learning one (DFML). Experimental results show that:

- For the Synthetic and Temperature datasets (cf. Fig. 4.7a, 4.8a) the DFML performs consistently better than the DFM, with the performance gains increasing as the forecasting horizon becomes larger, while for the Volatility series (cf. Fig. 4.9a) the DFML only outperforms DFM starting from medium sized horizons. This can be explained by the fact that the non-linear nature of DFML is able to better capture the non-linear long-term dependencies among the time series that can be found in the Synthetic and Temperature datasets. It is worth noting that, for all datasets, the addition of dynamic factors reduces the benefits of DFML over DFM.
- By fixing the number of dynamic factors to 3 (as in the previous experiments), and analyzing for different dataset dimensionalities, we can observe that, the accuracy gains provided by DFML tends to decrease when the number of series increases (Fig. 4.7b and 4.8b). DFML however still outperforms the traditional DFM, especially for larger horizons.

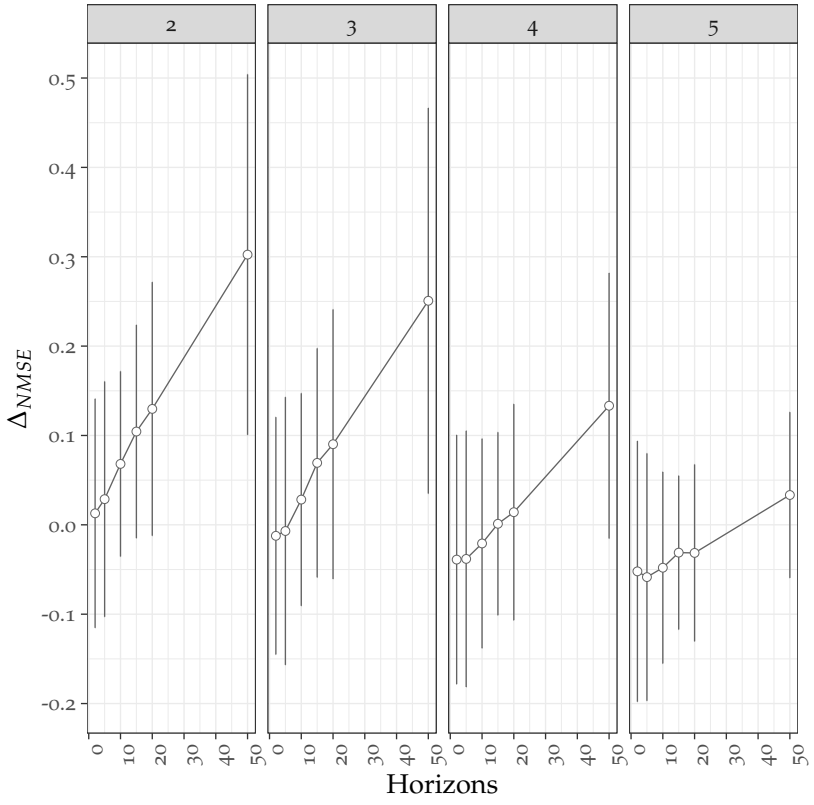


(a)

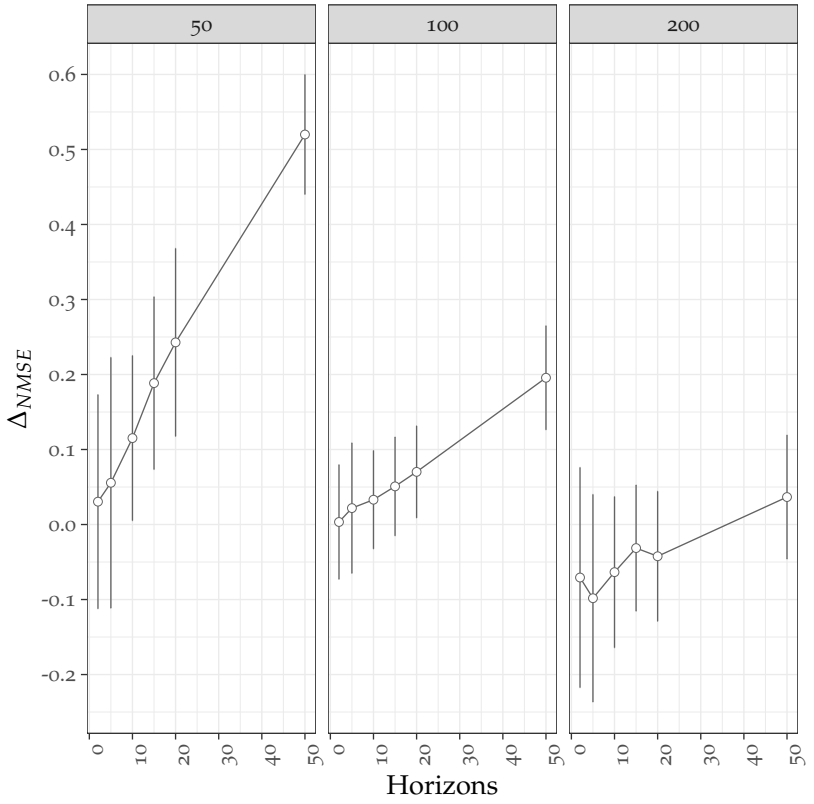


(b)

Figure 4.7: Synthetic multivariate time series: $\Delta_{NMSE} = NMSE_{DFM} - NMSE_{DFML}$. Difference between the NMSE of dynamic factor models with (DFML) and without (DFM) nonlinear forecasting component as a function of the forecasting horizon (x-axis) and (a) the number of dynamic factors or (b) the number of variables. The vertical black bars represent the confidence interval $\mu_{\Delta_{NMSE}} \pm \sigma_{\Delta_{NMSE}}$, $\mu_{\Delta_{NMSE}}$ and $\sigma_{\Delta_{NMSE}}$ being respectively the mean and standard deviation of Δ_{NMSE} across trials.

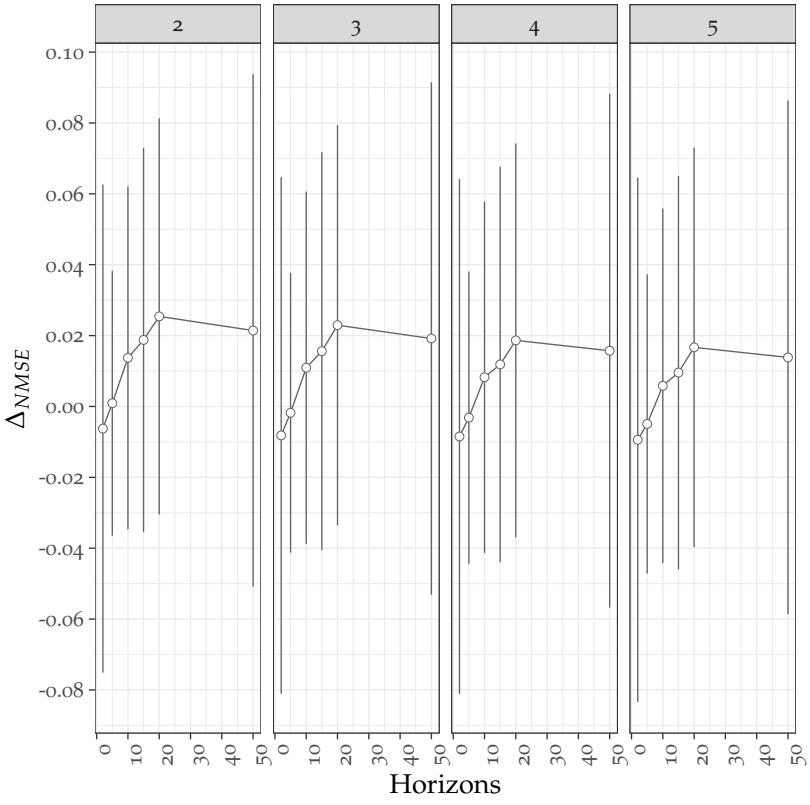


(a)

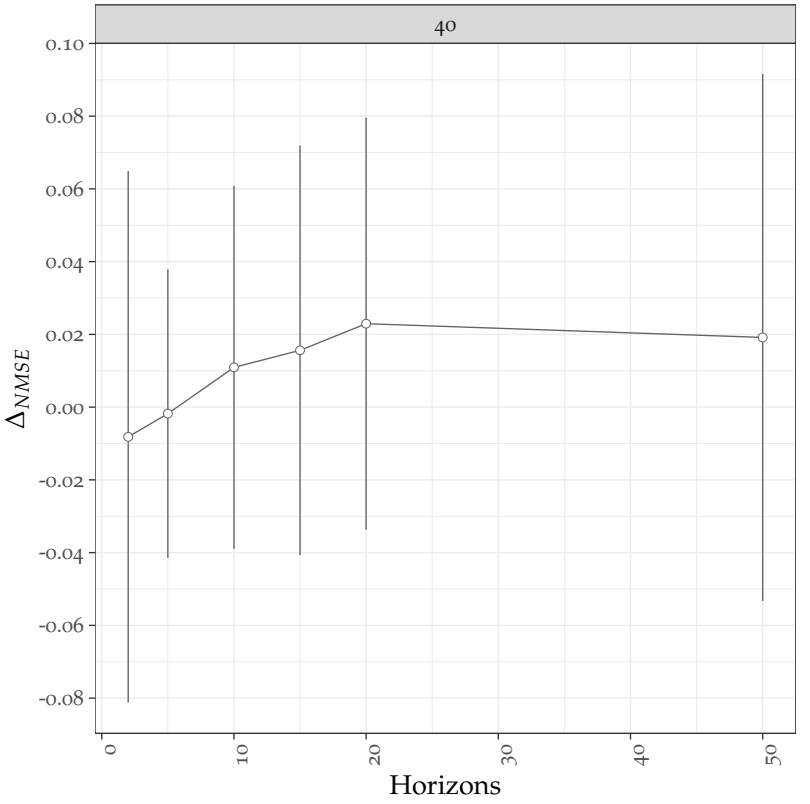


(b)

Figure 4.8: Earth Surface Temperature series: $\Delta_{NMSE} = NMSE_{DFM} - NMSE_{DFML}$. Difference between the NMSE of dynamic factor models with (DFML) and without (DFM) nonlinear forecasting component as a function of the forecasting horizon (x-axis) and (a) the number of dynamic factors or (b) the number of variables. The vertical black bars represent the confidence interval $\mu_{\Delta_{NMSE}} \pm \sigma_{\Delta_{NMSE}}$, $\mu_{\Delta_{NMSE}}$ and $\sigma_{\Delta_{NMSE}}$ being respectively the mean and standard deviation of Δ_{NMSE} across trials.



(a)



(b)

Figure 4.9: Volatility time series: $\Delta_{NMSE} = NMSE_{DFM} - NMSE_{DFML}$. Difference between the NMSE of dynamic factor models with (DFML) and without (DFM) nonlinear forecasting component as a function of the forecasting horizon (x-axis) and (a) the number of dynamic factors or (b) the number of variables. The vertical black bars represent the confidence interval $\mu_{\Delta_{NMSE}} \pm \sigma_{\Delta_{NMSE}}$, $\mu_{\Delta_{NMSE}}$ and $\sigma_{\Delta_{NMSE}}$ being respectively the mean and standard deviation of Δ_{NMSE} across trials.

4.6 CONCLUDING REMARKS

In this chapter, we conducted an extensive study of the different components of the DFML, across several real dataset.

First, we studied the performances of different factor forecasting and factor estimation techniques within the DFML. Then, we proceeded to test the performances of the DFML, on specific corner cases (e. g., ill-conditioned problems), where the benchmark techniques have shown several limitations. Finally, we assesed the integration in the DFML of two extensions: an incremental factor estimation technique and an automatic hyperparameter search strategy for the factor forecasting compnent.

Overall, the results have shown that linear factor estimation techniques in combination with either lazy learning or VAR for forecasting (corresponding to DF-PCA-Lazy/DFML and DF-PCA-VAR/DFM, according to the experiments) are consistently among the top performers across all the considered datasets, with the DFML significantly outperforming DFM. Non-linear factor estimation techniques, and additional model-driven and data-driven forecasting techniques have comparable performances within the DFML but the trade-off between the increase in forecasting accuracy and the computational overhead tends to favor simpler solutions (e. g., linear factor estimation and lazy forecasting techniques). In addition, our experiments show that, on the considered forecasting problems, univariate techniques are tough competitors for dynamic factor model techniques in multivariate, multi-step-ahead forecasting. Although often neglected in the scientific literature, where representation-based models (e. g., neural networks) tends to be predominant as benchmarks, employing well-known univariate techniques such as Exponential Smoothing (ES) or Holt-Winters, in a parallel fashion, instead of a complex global model, provided comparable performances to some configurations of DFML.

In the case of volatility forecasting, we can observe a similar ranking of the methods, with the DFML approach being significantly better than the benchmark methods, with univariate techniques being close in performances in several situations. Here, an additional advantage of a factor based approach is that the high dimensionality of the problem can be effectively reduced of at least one order of magnitude, thus preventing the computational issues that some of the global models (e. g., VAR, SSA) encounter.

Lastly, the empirical assessment showed that the extensions to the original DFML can be beneficial for a broader employment of this framework. On one hand, we show the effectiveness of an incremental approach to factor estimation, supporting the employment of a factor based framework on a large scale streaming setting, where high dimensional time series could be processed online, at a reduced computational cost. On the other hand, we showed that an automatic selection of the optimal value of the parameters can further improve the performance of the DFML, and simplify the model tuning.

In the next chapter, we develop an alternative multivariate multi-step-ahead forecasting strategy, still based on a multivariate to univariate problem reduction, by employing a combination of conventional machine learning approaches (i. e., feature selection and model ensembling) and feature engineering relying both on expert-based and data-driven approaches.

5

SELECTIVE MULTIVARIATE TO UNIVARIATE REDUCTION THROUGH FEATURE ENGINEERING AND SELECTION - EXPERIMENTAL ASSESSMENT

Results presented in this chapter have been published in the following papers:

- Fabrizio De Caro, Jacopo De Stefani, Gianluca Bontempi, Alfredo Vaccaro, and Domenico Villacci (Oct. 18, 2020). "Robust Assessment of Short-Term Wind Power Forecasting Models on Multiple Time Horizons." In: *Technology and Economics of Smart Grids and Sustainable Energy* 5.1, p. 19. ISSN: 2199-4706. DOI: 10.1007/s40866-020-00090-8. URL: https://doi.org/10.1007/s40866-020-00090-8 (visited on 07/27/2021)
- Fabrizio De Caro, Jacopo De Stefani, Alfredo Vaccaro, and Gianluca Bontempi (2021). "DAFT-E : Feature-based Multivariate and Multi-step-ahead Wind Power Forecasting." In: *IEEE Transactions on Sustainable Energy*, pp. 1–1. DOI: 10.1109/TSTE.2021.3130949

5.1 INTRODUCTION

In this chapter we consider two implementations of the SMURF-ES strategy: the DAF-E and the DAFT-E. We start by presenting the different datasets employed for the assessment (Section 5.2.2) before proceeding to the assessment of the two implementations: DAF-E and DAFT-E.

The first implementation (DAF-E - Section 3.3.5.1) employs a feature engineering process based on rolling statistics on the historical available data. A feature selection technique based on an information-theoretical filter reduces the number of input features to the forecasting component. The forecasting component considers two alternative rules for forecast combination: a static and a dynamic one. The static combination gives equal weight to all the underlying forecasting models, and is employed for benchmarking purposes whereas in the dynamic one the combination weights are adapted to be inversely proportional to the current forecasting error.

The second implementation (DAFT-E - Section 3.3.5.2) provides a two-fold improvement over the first one. First, the feature engineering component is augmented by the introduction of expert-based features tailored to wind power forecasting problems in order to capture domain-specific temporal dynamics. Secondly, the forecasting combination is improved by considering both an adaptive forecasting combination rule based on the current forecasting error (similarly to the first implementation) and a set of forgetting factors. Forgetting factors with decreasing weights also allow to include information about the past predictive performances in the forecasting combination, favoring more recent information over older one.

The two implementations are assessed in Sections 5.3 and 5.4, respectively. Section 5.5 concludes the chapter by summarizing the main findings from the assessment and outlining some perspectives for future research studies.

5.2 DATASETS

For our studies, we consider a mixture of publicly available and private datasets related to multivariate problems in the domain of renewable energies forecasting ($15 - 30$ variables and $> 10^4$ samples). Table 5.1 contains an overview of the different datasets and

the corresponding empirical assessments, while the details of the different datasets and preprocessing techniques are presented in the remainder of the section.

Assessment	Dataset	N	n
DAF-E	Apennines dataset	425000	15
DAFT-E	Australian dataset	35040	22
	Italian dataset	37920	28

Table 5.1: Summary of the different datasets employed for the assessment of the SMURF-ES strategy

5.2.1 *Apennines dataset*

For the DAF-E assessment, we consider a private dataset, representative of a wind farm located in southern Italy on the ridge of Apennines chain, composed of 15 wind generators and with an installed power of 30 MW.

The raw signals collected in a wind farm are typically the output of a Supervisory Control And Data Acquisition (SCADA) system that returns series with a time resolution of 10 minutes. This system collects information from all wind generators/anemometers of the wind farm and supplies both environmental and mechanical data. All data are radio broadcast to a central server where a database is continuously updated. The first data processing consists in extracting and grouping data according to the source (generator/anemometer).

As shown in Figure 5.1, the terrain is characterized by ridge steeps and complex orography, which causes chaotic behavior (notably fast changes in wind direction, strong shears, turbulence, sudden gusts).

The impact of site morphology is confirmed by the spatial wind distribution shown by the wind roses of the two anemometers, where dominant winds come from south-west and north for anemometer 1 and 2, respectively. This setting increases the difficulties in relating anemometers measures with the wind farm power generation, requiring then the adoption of complex models to return an accurate forecasting.

The data considered for the experiment includes wind speeds/directions at different heights, supplied by two anemometer spots, and the generated power, with a time resolution of 10 minutes over a period of 2 years.

5.2.2 *Australian and Italian datasets*

For the DAFT-E assessment, we consider two real wind power forecasting case studies, related respectively to an Australian and Italian wind farm. The *Australian* dataset includes 22 time series, representative of Eastern Australian wind farm power generation, with 5 minutes time resolution for 1 year. The data is publicly available via the R package of (Messner and Pinson, 2019). The positioning of the wind farms is depicted in Figure 5.2

The *Italian* dataset includes 28 time series of a private domain farms located in southern Italian wind farms, with a 15 minutes time resolution for 1 year. For confidentiality reasons, the input data could not be made available.

In order to perform a fair comparison with the second dataset, the time scale of the first datasets is raised to 15 minutes per sample by averaging the 5 minutes samples.

Figure 5.1: Apennine case study - Satellite view with overlaid wind rose visualization of direction and wind intensity for the considered wind farms.

Figure 5.2: Australian case study - Positioning of wind farms

5.3 DAF-E ASSESSMENT

This experimental study assesses a first implementation of the SMURF-ES strategy (DAF-E - Section 3.3.5.1), to provide multiple-step-ahead forecasts of a real wind farm situated on the ridge of Apennines in southern Italy (Section 5.2.1). The assessment compares the proposed method against different Statistical and Machine Learning-based forecasting techniques on several prediction horizons ranging from 1 to 6 hours ahead (short-term forecasting).

In addition, we propose an in-depth statistical analysis of the performances of the different forecasting models, inspired to the assessment procedure effected by the M3 competition (Makridakis, Spiliotis, and Assimakopoulos, 2018).

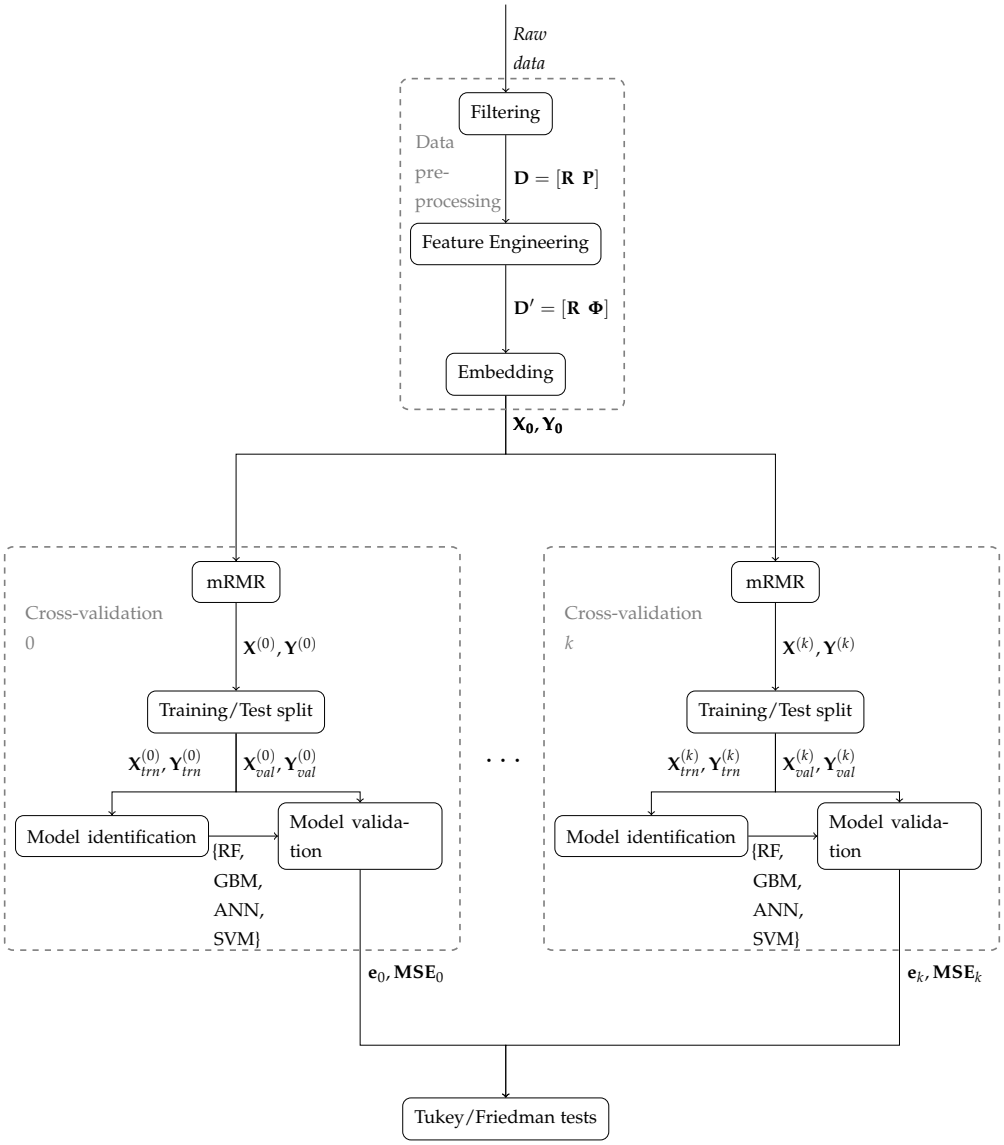


Figure 5.3: Flow chart of the proposed experiment assesment pipeline

On one hand, this aims to fill a gap in literature on wind power forecasting, where there is an increasing need for a quantitative and statistically founded comparison of existing approaches. Among the few examples we could find, we cite (Foley et al., 2012), which tested several Wind Power Forecasting (WPF) models on different horizons in terms of Mean Absolute Percentage Error (MAPE); (Demolli et al., 2019), discussing a spatial comparison on daily wind power forecasting and the work of (Korprasertsak and Leephakpreeda, 2019) which took into consideration only the mean value for an ensemble of accuracy metrics. (Korprasertsak and Leephakpreeda, 2019) presents an assessment in terms of prediction

interval of several Artificial Neural Networks models while (Ozkan and Karagoz, 2015) introduced the t-student test between the assessed models for short term WPF models. Unfortunately, the t-test is subject to α (probability error of I type) inflation in case of multiple comparisons (Lee and Lee, 2018), reducing its reliability.

On the other hand, we hope that such statistically sound methodology could support wind power producers and system operators (Albadi and El-Saadany, 2010) in choosing the most suitable model for their needs, in light of the opening of the capacity market to the not programmable renewable energies (Soares et al., 2016).

5.3.1 *Statistical assessment*

The choice of the most suitable metric (Ren, Suganthan, and Srikanth, 2015) to assess wind power forecasting is still an open issue (Würth et al., 2019). Here we adopt the following metrics: Mean Square Error (MSE), Mean Absolute Error (MAE) and R squared (R^2) (Cameron and Windmeijer, 1997). The details of their mathematical definition are included in Table 5.2.

Metric	**Equation**	**Domain**
MSE	$\frac{1}{N_{ts}} \sum_{i=1}^{N_{ts}} (y_i - \hat{y}_i)^2$	$[0, \infty]$
MAE	$\frac{1}{N_{ts}} \sum_{i=1}^{N_{ts}} \lvert y_i - \hat{y}_i \rvert$	$[0, \infty]$
R^2	$1 - \frac{\text{RSS}}{\text{TSS}}$ $\text{RSS} = \sum_{i=1}^{N_{ts}} (y_i - \hat{y}_i)^2$ $\text{TSS} = \sum_{i=1}^{N_{ts}} (y_i - \bar{y})^2$	$[0, 1]$
nMSE	$\frac{\text{MSE}}{\sigma_{ts}^2}$	$[0, \infty]$

Table 5.2: Metrics employed for the DAF-E assessement. N_{ts} is the number of samples in validation dataset. σ_{ts}^2 is the variance of validation dataset.

In order to robustly assess the alternative WPF models, a cross-validation approach has been considered. Each case is based on different pairs of training and validation sets (Hyndman and Koehler, 2006), generated by applying the rolling window technique (Tashman, 2000).

The aggregation of the resulting accuracy measures enables a thorough assessment [1] of each WPF model by means of the Friedman's test (Friedman, 1937), a non-parametric randomized block analysis of variance whose null hypothesis H_0 is that the error distributions are the same across repeated measures. If the test rejects the H_0 hypothesis, a post-hoc analysis is run to find which pairs of methods are significantly different (Pereira, Afonso, and Medeiros, 2015). The analysis is based on Tukey's test and supplies an upper diagonal square matrix with the column element sorted by their rank.

Eventually, once the Friedman's test with post-hoc analysis has returned a rank of the different competing models, we visualize the performance of all the models by means of box-plot/heat-map graphs.

1 Note that such test differs from the conventional Analysis of variance (ANOVA) since it does not rely on any assumption of normal distribution and equal variances of residual.

5.3.2 *Benchmarks*

The experimental study assessed and compared the proposed ensemble techniques against standard univariate forecasting methods, both from the statistical and machine learning literature. The methods are listed below together with the software used for the experiments.

Model-driven	**Data-driven**	**Proposed Ensembles**
Exponential Smoothing	Simple ANN	Av-GBM-RF
Average	GRUdrop (ANN)	Av-Average-RF
	GRU (ANN)	Av-GBM-Average
	SVM	Av-SVM-GBM
	GBM	Av-SVM-Average
	RF	Ad-ANNs-Average
		Ad-SVM-Average
		Ad-GBM-Average
		Ad-RF-Average
		Ad-Average-RF-SVM-GBM

Table 5.3: Overview of the assessed models for the *DAF-E* assessement. *Av.* denotes an ensemble of the proposed models based on the averaging of their forecasts, whereas *Ad.* denotes a weighted combination of the forecasts, inversely propotional to their forecasting error (Section 3.3.5.1)

- **Average**, **ES:** their model formulation is discussed, respectively in Sections 2.3.2.2 and 2.3.2.3. We employ the implementations made available by the M4 competition (Center, 2020). The multi-step-ahead forecast is obtained with a Recursive strategy (Section 2.2.3.2).
- **SimpleANN:** the model formulation is discussed in Section 2.3.3.1 and implemented via `rstudio/keras` library. The architecture consists of an input layer, an hidden layer of 32 nodes and ReLU activation function and a single output layer with H neurons. The weights are optimized using `RMSProp`, with the MAE metric, over 20 epochs of training.
- **GRU**, **GRUDrop:** The model formulation is discussed in Section 2.3.5.1 and implemented via `rstudio/keras` library. The architecture consists of a fully connected RNN with 32 GRU cells a single output layer with H neurons. The weights are optimized using `RMSProp`, with the MAE metric, over 20 epochs of training. The *GRUDrop* variant employs dropout and recurrent dropout with a probability of 0.2.
- **SVM:** the model formulation is discussed in Section 2.3.3.3, while the implementation is provided by the `e1071` library
- **GBM:** the model formulation is discussed in Section 2.3.3.4, while the implementation is provided by the `gbm` library.
- **RF:** the model formulation is discussed in Section 2.3.3.5, while the implementation is provided by the `randomForest` library.

For the neural based models, since an automated setting of the number of units would not have been feasible for computation time reasons, this number has been set based on trial and error over a small number of training sets.

5.3.3 *Experimental setup and results presentation*

The considered dataset (Apennines - Section 5.2.1) does not contain any missing values, and is already available in matrix form $\mathbf{Y}$ having N (number of observations) rows and n (number of variables/time series) columns. Each time series in the dataset is re-scaled in order to have null mean and unit standard deviation via Z-score standardization.

The data is split to generate $V = 17$ experimental cases, with a rolling window procedure (Tashman, 2000). Each experimental case is made of 25000 samples partitioned according a $5:1$ ratio into training and validation datasets.

The assessment results in terms of all the considered metrics and horizons are illustrated by a number of heat-maps (Figures 5.7 and 5.8) and box-plots (Figures 5.4, 5.5 and 5.6). The heat-maps illustrate the ranking of wind power forecasting models according to the Friedman's test where the most accurate models are situated at the bottom-left side. Each cell of the heat-map takes two possible colors: grey if the WPF model in the row is significantly worse than the one in the column, orange otherwise. By looking at the colors, this representation allows to visualize clusters of equivalent predictors.

READING GUIDE FOR FRIEDMAN'S TEST The procedure to interpret the results of the Friedman's test is as follows: given a forecasting horizon H, the reader should read in the figure from the bottom by selecting a model and moving up until the corresponding diagonal cell, then move perpendicularly from left to right over the row. All the orange elements considered during this trip correspond to models whose performances are not significantly different. For example, in Figure 5.7, for the forcasting horizon $H = 2$ hours, *GBM* lies in 4^{th} position according to the ranking based on Friedman's test. Thus, counting from the bottom to the diagonal element, there are two orange cells above the selected column, which means that the previous two models in the ranking (*Av-GBM-RF* and *Ad-RF-Naive*) are not significantly different to *GBM* (the diagonal element is always excluded from the count because it corresponds to the considered model itself). Then, starting from the diagonal element, there are three orange cells by moving from left to right, which means that the successive three models in the rank (*Av-GBM-Naive*, *Av-SVM-GBM* and *RF*) are not significantly worse than *GBM*.

5.3.4 *Results*

The information provided by the heat-maps, based on the accuracy ranks, is coupled with that provided by the box-plots, which shows the metric distribution for each considered WPF model and forecasting horizon. In this way, the decision maker has a complete summary about the ranks and performance dispersion for each method leading to many interesting considerations made based on the experimental results:

- The method which appears consistently on the top ranking positions is the adaptive ensemble model combining RF, SVM, GBM and Naïve.
- Exponential Smoothing has the highest accuracy only for $H = 1$ forecasting horizon whereas for increasing values of H its performances deteriorate dramatically in terms of MSE, which seems to penalize larger prediction error more than MAE.
- The ensemble forecasting of ANNs plus Naive tends to have higher ranking as the forecasting horizon H increases.
- The influence of the system physics becomes dominant in $H = 5$ and $H = 6$, causing a general accuracy reduction in all the WPF models. Indeed, the performance spreads become much wider and there is no model clearly outperforming the rest.

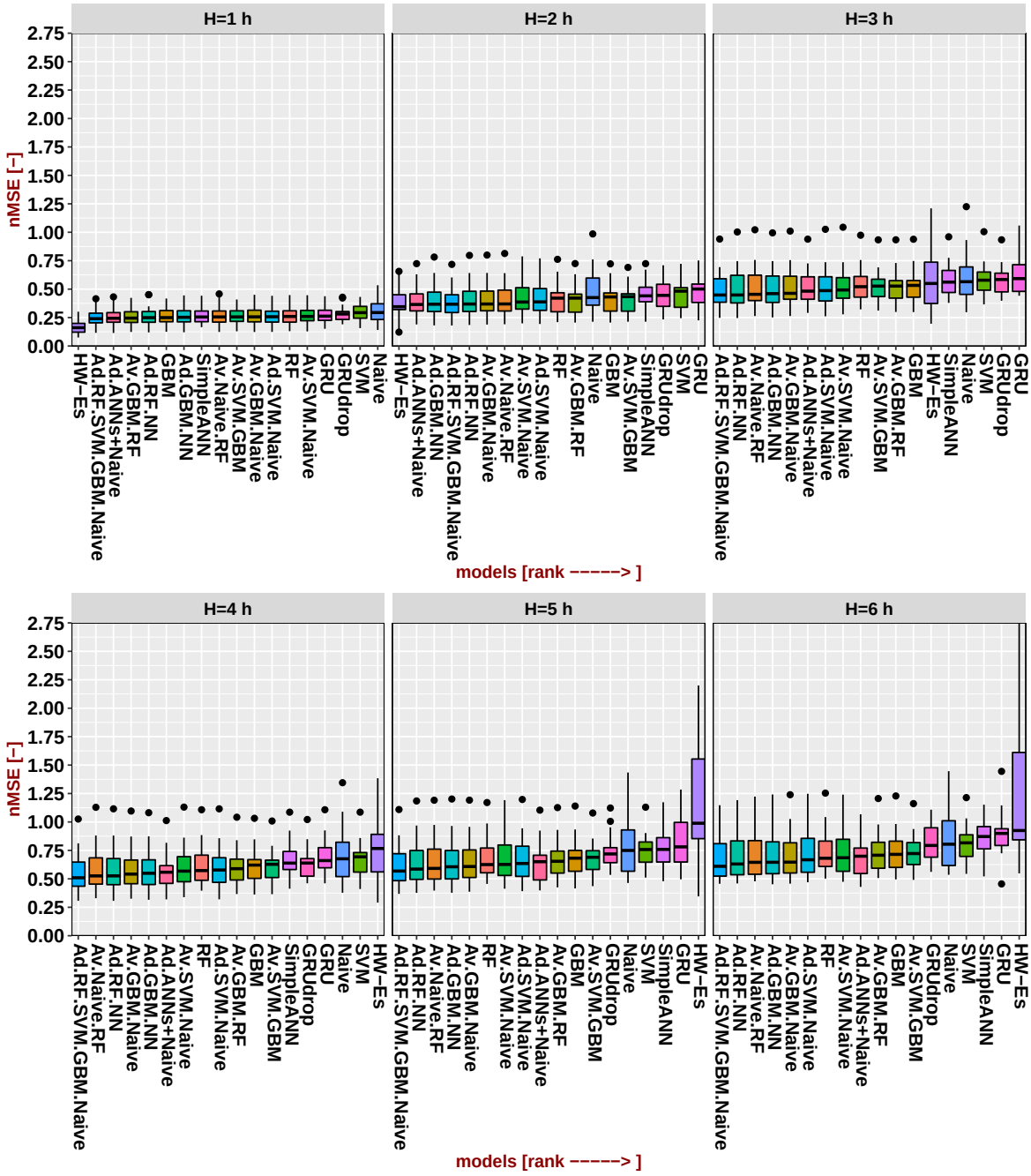


Figure 5.4: Visualization of model performance distribution in terms of nMSE across $V = 17$ cases. The models are ordered by considering the rank supplied by the Friedman test.

On the basis of the previous considerations, our recommendation would be the adoption of ensemble forecasting, which is able to guarantee both accuracy and stability over different forecasting settings.

5.3.5 *Conclusions*

The growing wind power production is introducing uncertainty in power grid operations with negative consequences for both system operators and wind producers because of the stochastic and intermittent nature of the wind. This issue motivates the design of effective wind power forecasting models to mitigate the power generation uncertainty. Nevertheless a thorough procedure for comparing and assessing existing approaches is still lacking in the literature on wind power forecasting.

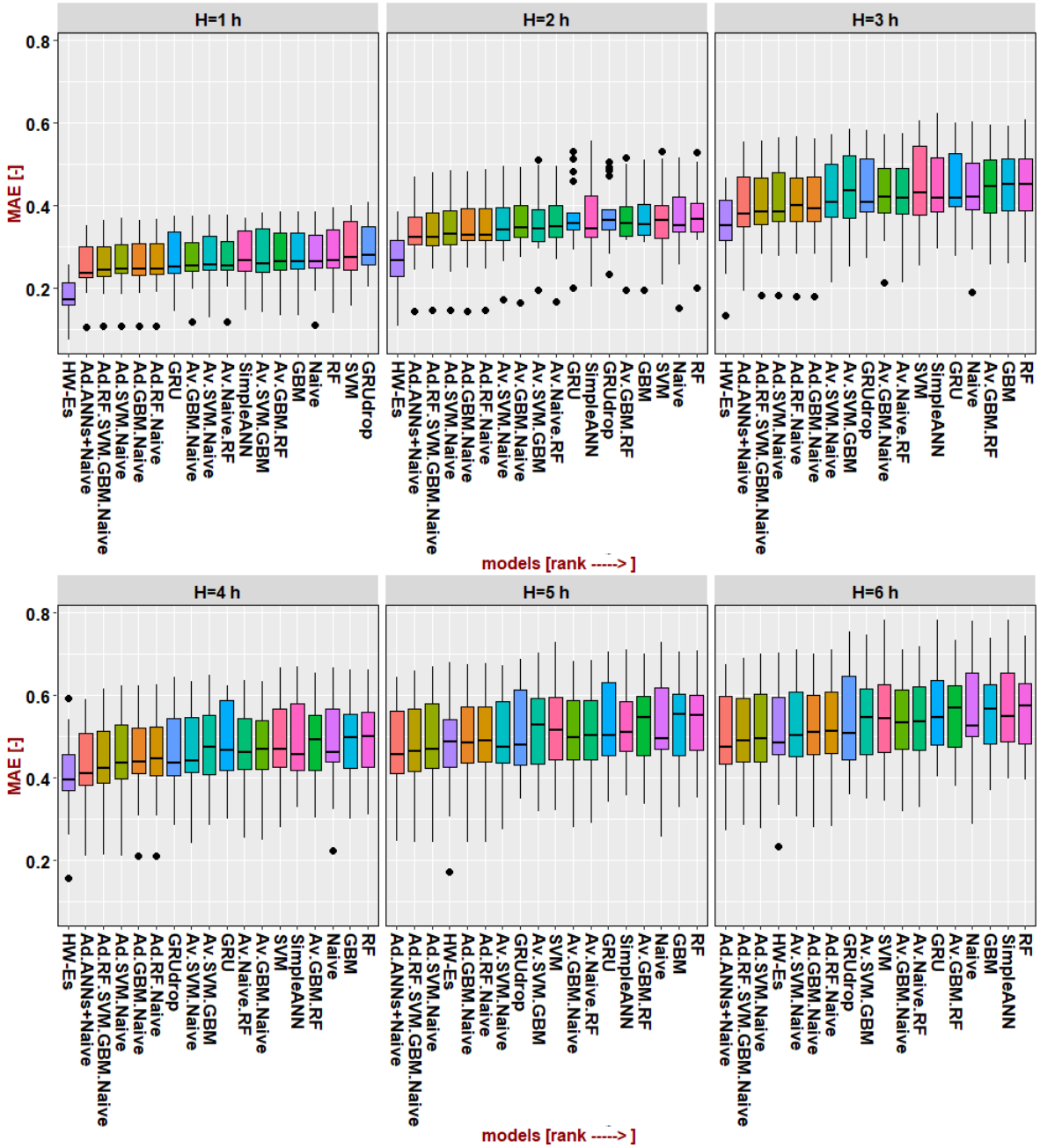


Figure 5.5: Visualization of model performance distribution in terms of MAE across $V = 17$ cases. The models are ordered by considering the rank supplied by the Friedman test.

This section proposed a methodology for the robust assessment of different types of short-term WPF models over multiple horizons, which is based on data resampling and statistical tests.

The study highlighted that ensemble forecasting of statistical and machine learning models dominates in terms of accuracy ranks, by supplying additional robustness with respect to single approaches.

In particular, for horizons longer than two hours, the proposed technique is able to outperform exponential smoothing, a well-known state-of-the-art statistical approach.

As far as the generalization of these results is concerned, it is important to remark that the pipeline and the features selection technique have been designed to process heterogeneous data sets, characterized by different size and complexity. Moreover, additional time series, i.e. temperature and pressure profiles, can be integrated in the input data-set and processed

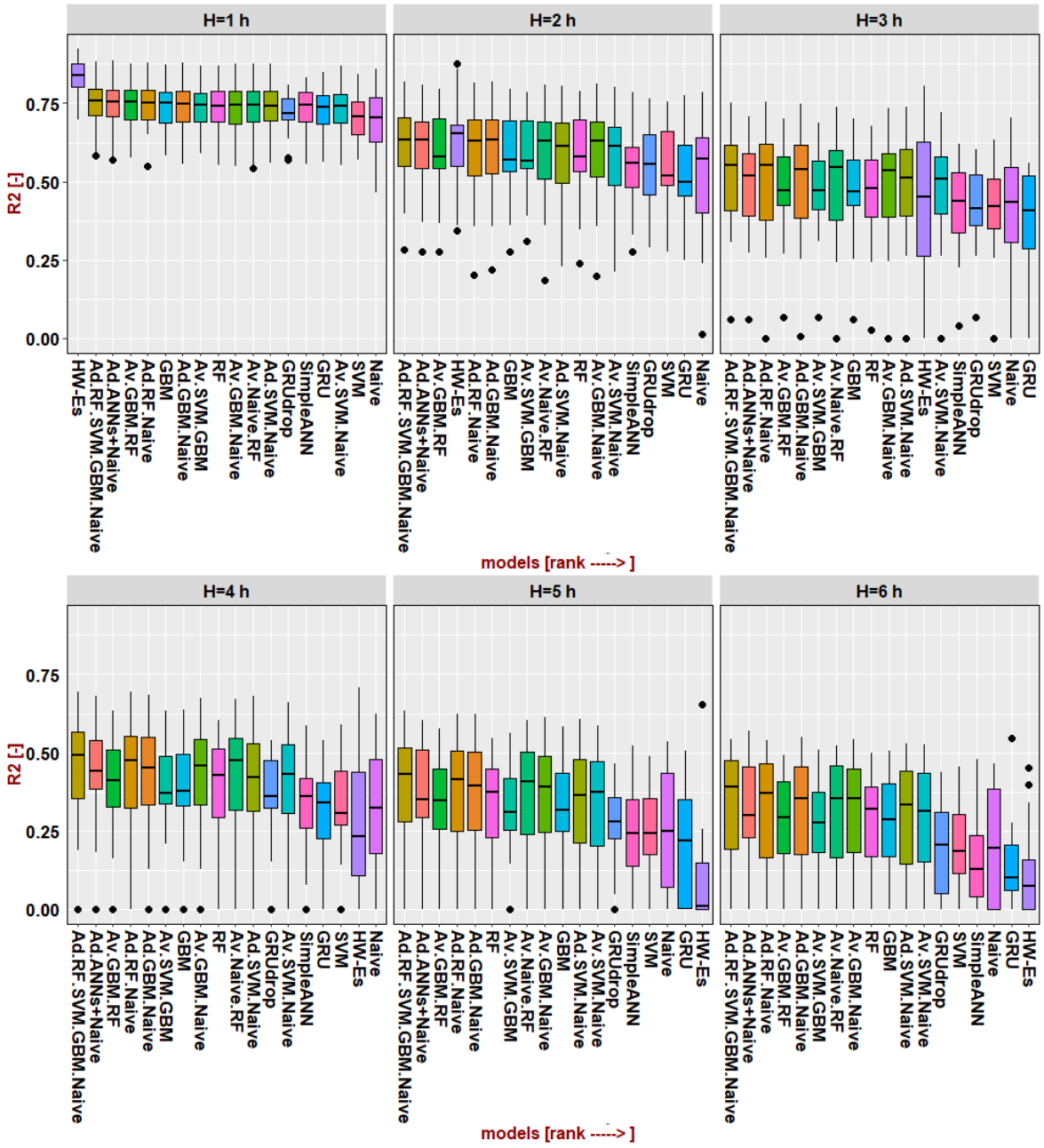


Figure 5.6: Visualization of model performance distribution in terms of R^2 across $V = 17$ cases. The models are ordered by considering the rank supplied by the Friedman test.

by the forecasting algorithms, if their contribution is considered relevant by the feature selection technique.

Future research will focus on the improvement of the performance of ensemble forecasting models. This kind of models will be adapted to supply wind power forecast on larger areas, trying to detect correlation between the several wind power plants by supplying multivariate and hierarchical forecasts over the time.

5.3.6 *Future work*

The proposed approach has been developed to predict the overall wind farm power generation. During the preliminary design phase, a wind power forecasting on the single wind generator has also been tested, showing worse results. This phase had the merit of highlighting the need for enhancing condition monitoring of the generator asset, improving

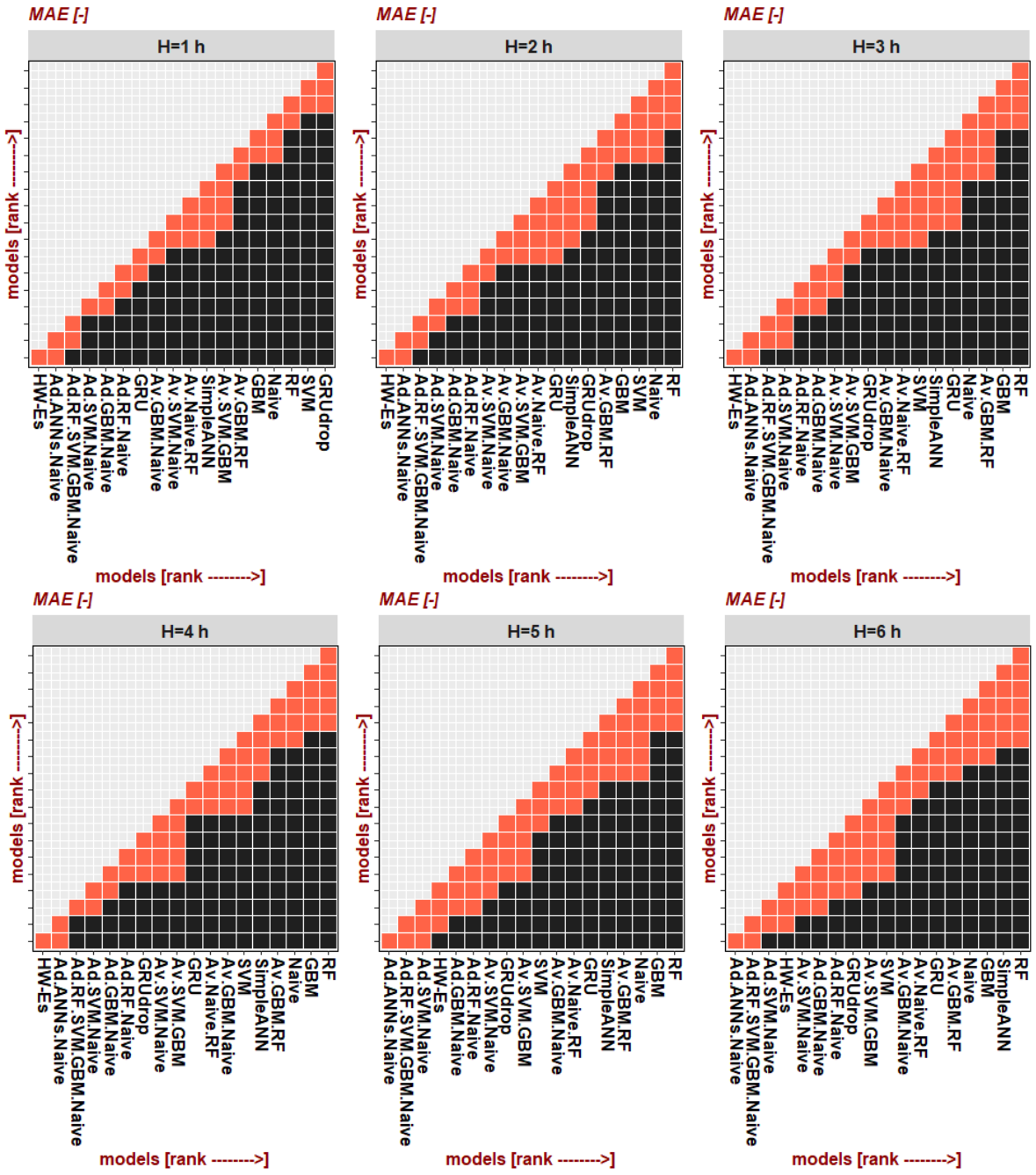


Figure 5.7: Visualization of Friedman's test with Post-hoc Analysis in terms of MAE. The best models are on the leftmost/lower side. The elements placed on the left side are placed in a specular manner to those on the downside for each insert. An orange case means that the models on the respective row/column are not significantly different.

the level of confidence in the analysis of the operation data. In particular, the comparison between the output of the wind turbine generator models and the measured data allowed detecting several critical operation states, especially in analyzing the wind - power output with a 10-min time resolution. Indeed, as shown by Fig. 5.9, the active power spread for each wind speed bin is significant, especially for low values. This is caused by the effect of generator inertia, which smooths the fast transients in wind gusts as shown in the lower side of the figure. Actually, the overall combined effect of inertia, blade/wind peripheral rotor speed and yaw controls is the main factor inducing this power spreading (Yan et al., 2015). Further, critical patterns include the deflections from the maximum power operation point for the wind speeds greater than the nominal value (rightmost side), where several

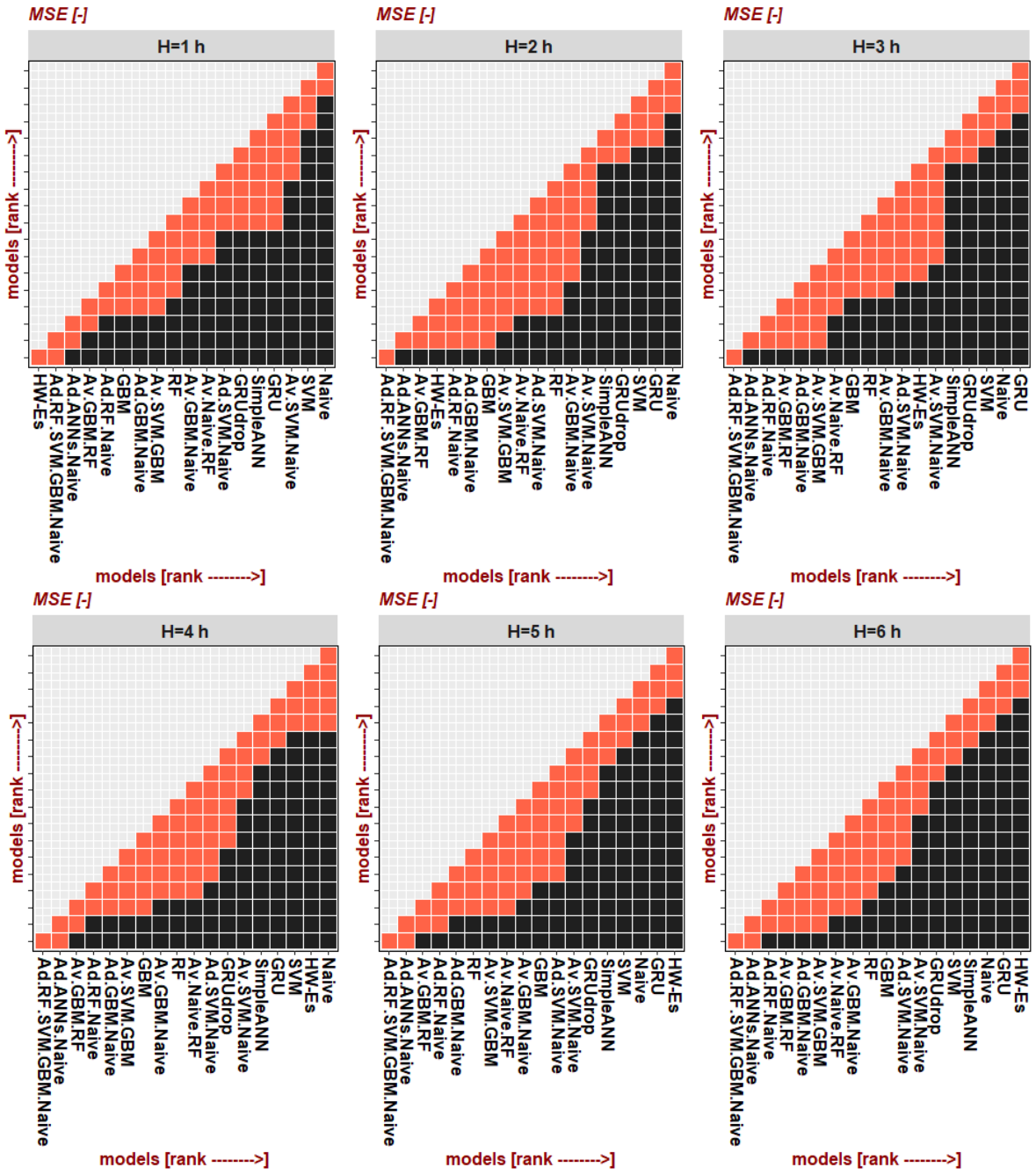


Figure 5.8: Visualization of Friedman's test with Post-hoc Analysis in terms of MSE. The best models are on the leftmost/lower side. The elements placed on the left side are placed in a specular manner to those on the downside for each insert. An orange case means that the models on the respective row/column are not significantly different.

data clusters can be clearly identified. These derated states are activated in order to satisfy the power curtailment orders imposed by the transmission system operator in the presence of power system congestion. Any other deviation between the predicted and the measured data is a clear indication of a system anomaly, which could be induced by wrong control settings, performance deterioration, incipient faults etc.

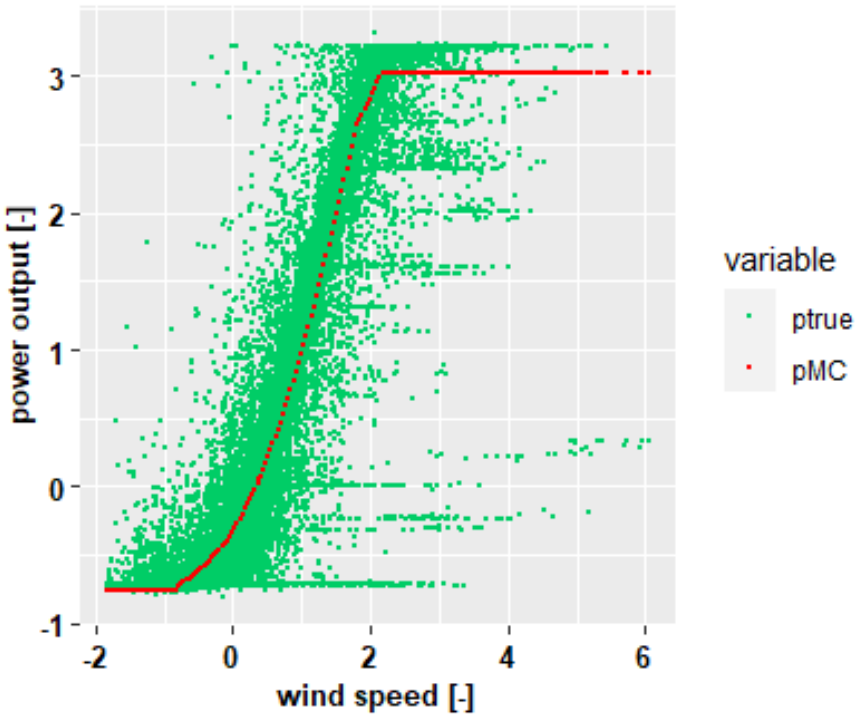


Figure 5.9: Comparison between wind turbine generator experimental measures ("ptrue") and manufacturer curve ("pMC")

5.4 DAFT-E ASSESSMENT

This experimental study assesses a second implementation of the SMURF-ES strategy (Section 3.3) on two real wind power forecasting case studies: one based on a public dataset and the other based on a proprietary dataset (Section 5.2.2).

This second strategy extends the first implementation by weighting the forecast combination based not only on the current performance of the underlying forecasting techniques, but also on their historical performances, via a set of forgetting factors. This dynamic model adaptation is introduced in order to better model the intrinsic time-varying behavior characterizing the wind dynamics (Hanifi et al., 2020).

The assessment compares the proposed method (DAFT-E) against different Statistical and Machine Learning-based forecasting techniques on several prediction horizons ranging from 1 to 3 hours ahead ($H \in \{1, \ldots, 12\}$ steps ahead, considering a sampling frequency of 15 minutes).

In addition, we propose two novel model assessment procedures inspired from the financial domain. The first one borrows from modern portfolio theory (Markowitz, 1952): the bivariate risk-return analysis (i. e., bias-variance principle), which allows to study the spatial distribution of multivariate model performance through a visualization based on bivariate box-plots (Rousseeuw, Ruts, and Tukey, 1999); The second one focuses on the comparison between different families of multivariate forecasting strategies over different horizons, time resolutions and taking into account, besides accuracy on the average case, measures focusing on extreme case performances such as Value at Risk (VaR) and conditional Value at Risk (cVaR) (Rockafellar and Uryasev, 2002).

These procedures have been developed to support the Transmission System Operator (TSO) in assessing forecasting of load and renewable energy sources, with a focus on multiple aspects: spatial, temporal and financial, over a time window ranging from 5 min to 1 week ahead. In short-term system operation (pre-dispatch phase), which extends from 1 week to 1 day ahead of actual operation, TSO needs predictions to identify and allocate the system

reserve. In real-time operation, which extends from 5 min to 30 min ahead of actual operation, TSO needs prediction to bear RES/load increments/decrements following deviations from the predicted profiles; to evaluate the power system status; to take preventive actions for assuring a secure and reliable grid operation (Ning and You, 2019).

5.4.1 *Statistical assessment*

Validation is a crucial step in the assessment of a forecasting model but in the literature, the focus is typically on the average prediction accuracy (e.g. Mean-Squared-Error) disregarding other relevant aspects for the decision maker. Here we propose a more general approach to assess the performance accounting for other performance measures like the spatial spread analysis and the tail error distribution analysis.

5.4.1.1 *Spatial performance analysis*

The spatial analysis of forecasting performance is important to assess possible distortions in studies that employ the obtained predictions, since an unbalanced forecasting performance may compromise the quality of the decision making process.

A reliable multivariate wind power forecasting methodology should assure a consistent prediction performance across measurement points. Inconsistent performance may compromise the quality of the decision-making process based on the obtained predictions.

Traditionally, the prediction accuracy of a multivariate forecasting methodology is assessed by computing a single error metric over all the target variables.

Unfortunately, this approach neglects the distribution of the forecasting model errors across the target variables. Therefore, a novel validation procedure is proposed to fulfill this gap inspired by the modern portfolio theory (Markowitz, 1952).

In particular, the following quantities are computed:

- $\mu_{\text{ERR}}^{(\omega,k,h)}$: the average value of the considered error metric ERR across the target variables for the forecasting model ω, forecasting horizon h and trial k.
- $\sigma_{\text{ERR}}^{(\omega,k,h)}$: the standard deviation of the considered error metric across the target variables for the forecasting model ω, forecasting horizon h and trial k.

Finally, a bivariate distribution is built by collecting $\mu_{\text{ERR}}^{(\omega,k,h)}$ and $\sigma_{\text{ERR}}^{(\omega,k,h)}$ for each trial in the experiment.

In particular, Fig. 5.10 shows some examples of the possible occurrences in the built bi-variate distribution. (1) low bias and low spread; (2) low spread and high bias; (3) high spread, low bias; (4) high bias, high spread. The closer is the point to the lower-left corner, the better is the performance of the forecasting model, given the k-th trial.

The results are visualised by means of a bivariate extension (Rousseeuw, Ruts, and Tukey, 1999) of the univariate box-plot where the conventional Interquatile Range and whiskers are replaced by two convex hull polytopes. In particular, the 50% of population is included in dark-colored area (*bag*), the 99.7% in the the light-colored one (*fence*), and the points outside the fence are considered outliers (Fig. 5.13,5.14).

5.4.1.2 *Tail Analysis of Wind Power Forecasting Error*

Although the spatial analysis of the performance returns a clear picture of the forecasting model behavior, it does not supply enough information about the worst-case configuration. To address such aspect, we adopt the tail analysis of the forecasting error distribution by using some well-known metrics in the financial domain: the value at risk (VaR), and the

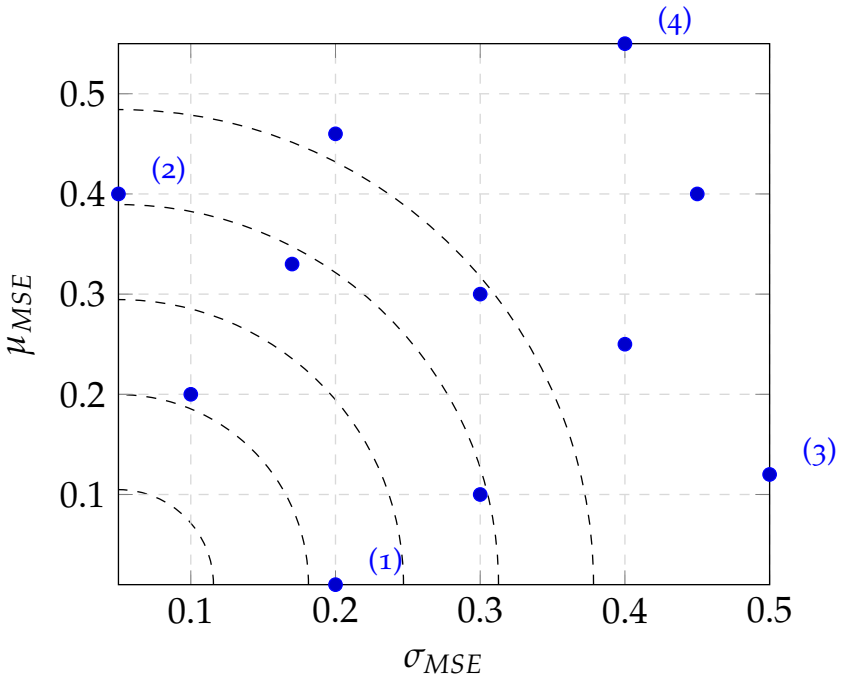


Figure 5.10: Example of bivariate plot $\sigma_{\text{ERR}} - \mu_{\text{ERR}}$, for with the selected error metric being MSE. The highlighted points represent configuration with low bias and low variance (1), large bias and low variance (2), large variance and low bias (3) and high bias and high variance (4).

conditional value at risk (cVaR). Traditionally, these metrics summarize the probability distribution of financial returns to determine the worst-case financial losses given a risk threshold (Rockafellar and Uryasev, 2002) and a strategy. In this manuscript, we apply them to the probability distribution of the forecasting errors, where each forecasting model corresponds to a different strategy. In other words, we are assessing, given an identical risk threshold for all forecasting models, which one returns the least absolute error.

In order to maintain a coherent interpretation with respect to the financial counterparts, the analysis considers absolute errors. The rationale is twofold: first, any gap between predicted and actual value is detrimental independently of the sign; second, we do not convert the forecasting error into TSO economic losses since this would require a deeper analysis (and more complex simulations) which are out of the scope of the manuscript.

Although it relies on some simplifying hypotheses, this approach is crucial to assess the reliability of the forecasting model. Indeed, it is reasonable to assume that the lower is the risk to commit a large prediction error, the lower is the possible associated economic losses.

Algorithm 2 Algorithm for MAE,VaR, and cVaR estimation for the ω-th model and forecasting horizon H, across K case studies

for $h \in \{1, \ldots, H\}$ **do**
  for $k \in \{1, \ldots, K\}$ **do**
    Compute the absolute error matrix
    Compute $\text{MAE}^{(\omega,k,h)}$,$\text{VaR}^{(\omega,k,h)}$,and $\text{cVaR}^{(\omega,k,h)}$
  end for
  ▷ *A K MAE, VaR, and cVaR value collection is obtained, hence the statistical quantities are computed again on the obtained distributions*
  $\overline{\text{MAE}}^{(\omega,h)} \leftarrow \text{mean}(\{\text{MAE}^{(\omega,1,h)}, \ldots, \text{MAE}^{(\omega,K,h)}\})$
  $\overline{\text{VaR}}^{(\omega,h)} \leftarrow \text{VaR}_\alpha(\{\text{VaR}^{(\omega,1,h)}, \ldots, \text{VaR}^{(\omega,K,h)}\})$
  $\overline{\text{cVaR}}^{(\omega,h)} \leftarrow \text{cVaR}_\alpha(\{\text{cVaR}^{(\omega,1,h)}, \ldots, \text{cVaR}^{(\omega,K,h)}\})$
end for

Algorithm 2 summarizes the procedure to estimate VaR and cVaR from the experimental results, given the ω-th forecasting model, and the h-th forecasting horizon, where:

$$\text{VaR}_\alpha(X) := \min\{z | \widehat{F}_X(z) \geq \alpha\} \tag{5.1}$$

$$\text{cVaR}_\alpha(X) := \mathbb{E}[X | X \geq \text{VaR}_\alpha(X)] \tag{5.2}$$

where X is the absolute error distribution, which is obtained collecting forecasting error across all the wind farms, given the ω-th model and h-th forecasting horizon, $\widehat{F}_X$ is the empirical cumulative distribution function, and α is the confidence level. In the results, VaR and cVaR are compared to the average value of X that is the Mean Absolute Error (MAE) (Fig. 5.11).

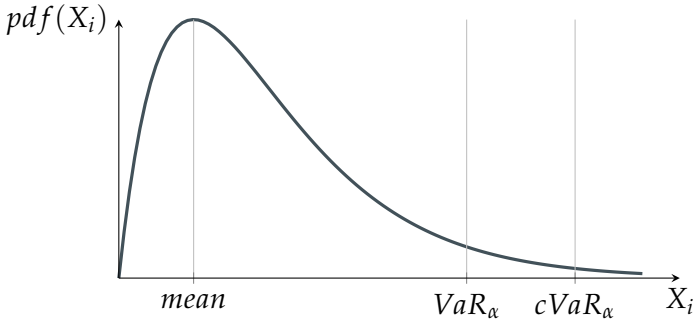


Figure 5.11: Qualitative visualization of the expected value (mean), VaR_α, and $cVaR_\alpha$ for a generic non-negative loss distribution.

It should be noted that VaR is the maximum value the decision-maker accepts to lose in a percentage of the cases equal to α, as shown in (5.1). Mathematically, the VaR is equivalent to the α-th percentile of a empirical distribution. The smaller is this value, the smaller is the maximum absolute forecasting error we accept to commit by using a forecasting model in the α % of cases. In other words, the lower the VaR, the more reliable the model is. Unfortunately, VaR is not such a good metric of risk since it neglects the loss values greater than VaR. Differently, cVaR is a coherent and robust measure of risk. Hence, it supplies information about the expected losses greater than VaR as shown in (5.2). Particularly, cVaR is computed through the Convex Combination Formula (Rockafellar and Uryasev, 2002). In other terms, the cVaR represents the upper bound for the worst expected forecasting error.

5.4.2 *Benchmarks*

The experimental study assessed and compared the proposed DAFT-E approach against different standard classes of models from the statistical and machine learning literature, representative of both the state-of-the-art in wind power forecasting (Deng et al., 2020; Messner and Pinson, 2019) and time series forecasting (Makridakis, Spiliotis, and Assimakopoulos, 2020b). Note that the considered multivariate forecasting task can be either approached in a univariate fashion by decomposing it in n SISO tasks or n MISO or in a multivariate fashion (MIMO) by capturing in a single model both the temporal and cross-series (e.g. spatial) dependencies between time series. The main features of the considered forecasting methods are summarized in Table 5.4, while Table 5.5 contains an overview of the associated forecasting pipelines. Finally, the list below resumes the software used for the experiments.

Model	Model class	Approach	Parameters
Dynamic Adaptive Feature Temporal Ensemble – DAFTE	Hybrid Ensemble	n MISO	V, Λ
Naive it – Persistence	Naive	n SISO	$\emptyset$
Holt-Winters Exponential Smoothing – Es it	Statistical	n SISO	α_{HW}
Lazy Learning FS all	Machine Learning	n MISO	k
Lazy Learning FS raw	Machine Learning	n MISO	k
Lazy PCA	Machine Learning	n MISO	k
Random Forest FS all	Machine Learning	n MISO	N_{RF}
Random Forest FS raw	Machine Learning	n MISO	N_{RF}
Random Forest PCA	Machine Learning	n MISO	N_{RF}
Long Short Term Memory RNN - LSTM	Deep Learning	n MISO	N_{LSTM}, δ_{LSTM}
VAR it	Statistical	MIMO	L
Online VAR	Statistical	MIMO	L, λ_{VAR}

Table 5.4: Characteristics of the considered models for the *DAFT-E* assessment. The models composing the *DAFT-E* ensemble are highlighted in bold.

Model	Feature Engineering	Embedding	Feature Selection	Dimensionality Reduction	Strategy
Dynamic Adaptive Feature Temporal Ensemble – DAFTE	(Yes)	(Yes)	(Yes) - MRMR	(No)	Direct
Naive it – Persistence	(No)	(No)	(No)	(No)	Recursive
Holt-Winters Exponential Smoothing – Es it	(No)	(No)	(No)	(No)	Recursive
Lazy Learning FS all	(Yes)	(Yes)	(Yes) - MRMR	(No)	Direct
Lazy Learning FS raw	(No)	(Yes)	(Yes) - MRMR	(No)	Direct
Lazy PCA	(Yes)	(Yes)	(No)	(Yes) - PCA	Direct
Random Forest FS all	(Yes)	(Yes)	(Yes) - MRMR	(No)	Direct
Random Forest FS raw	(No)	(Yes)	(Yes) - MRMR	(No)	Direct
Random Forest PCA	(Yes)	(Yes)	(No)	(Yes) - PCA	Direct
Long Short Term Memory RNN - LSTM	(No)	(No)	(No)	(Yes) - PCA	Direct
VAR it	(No)	(No)	(No)	(No)	Direct
Online VAR	(No)	(No)	(No)	(No)	Direct

Table 5.5: Characteristics of the forecasting pipelines of considered models for the *DAFT-E* assessment. The models composing the *DAFT-E* ensemble are highlighted in bold.

- **Naive, ES it:** their model formulation is discussed, respectively in Sections 2.3.2.1 and 2.3.2.3. We employ the implementations made available by the M4 competition

(Center, 2020). The multi-step-ahead forecast is obtained with a Recursive strategy (Section 2.2.3.2). The parameter α_{HW} controlling the exponential smoothing forecasting, is automatically fitted from the supplied data.

- **Random Forest:** the model formulation is discussed in Section 2.3.3.5, while the implementation is provided by the `randomForest` library, employing N_{RF} decision trees. The number N_{RF} of decision trees is optimized during the learning process.
- **Lazy Learning:** the model formulation is discussed in Section 2.3.3.2, while the implementation is provided by `gbonte/gbcode` library. The optimal value of k employed for the predictions is automatically optimized according to the input data supplied to the method.
- VAR **it**: the model formulation is discussed in Section 2.3.4.3, while the implementation is provided by `vars` library.
- **Online** VAR: In addition to the basic VAR(L) model, we included in our experiments an optimized version, where the number of model parameters is reduced by means of a LASSO regularization (controlled by the parameter λ_{VAR}, optimized on the input data). Moreover, an online procedure with a reduced computational cost (Messner and Pinson, 2019) is employed to update the values of the model parameters.
- **LSTM**: The model formulation is discussed in Section 2.3.5.1 and implemented via `rstudio/keras` library. In our experiments we fixed the hidden layer number to one, and we performed a grid search, on the test set, over different values of cells per layer, dropout rate in the input and recurrent elements, and regularization technique (no regularization, L_1, L_2, and a combination of both). The resulting architecture employs $N_{LSTM} = 100$ cells, no regularization and a dropout rate $\delta_{LSTM} = 0.2$ for both the input and recurrent elements. It should be noted that using the test set for the model selection might yield over-optimistic performances. Similar architectures have been considered as state-of-the art techniques in a recent survey (Deng et al., 2020).

Note that for both *Lazy* and *RF* the same model architecture is considered three times (Tables 5.4, 5.5), corresponding to the *raw*, *all* and *PCA* variants. The *raw* variant considers only the original features (historical data), without including the variables generated through Feature Engineering, whereas the *all* includes also the latter. In both cases, features selection is employed to reduce the number of features as input to the models. The *PCA* variant, on the other hand, replaces the process of feature augmentation and reduction by a dimensionality reduction approach based on PCA.

5.4.3 *Experimental setup and results presentation*

The considered datasets (Italian and Australian - Section 5.2.2) do not contain any missing values, and are already available in matrix form $\mathbf{Y}$ having N (number of observations) rows and n (number of variables/time series) columns. Each time series in the dataset is already normalized in the range $[0, 1]$.

Both datasets are split to generate $K = 45$ experimental cases, with a rolling window procedure (Tashman, 2000). Each experimental case is split into training and validation sets of 300 (12 days) and 100 (4 days) samples respectively. 1 statistical analysis is performed on the collected accuracy metrics to analyze the behavior of the models across time.

Figures 5.13 and 5.14 show the bag-plots of the bivariate distribution μ_{MSE}-σ_{MSE} obtained for the Australian and Italian benchmark, respectively. The closer is the cloud point to the lower-left corner (low mean - low variance of the forecasting error), the better is the

performance of the ω-th forecasting model over the K-th trials. Note that in this visualisation the variability over the vertical (horizontal) axis is related to the variability across (within) trials.

5.4.4 *Results*

Figures from 5.12 to 5.16 visualize the experimental results of the two benchmarks. The full results for both the Italian and Australian dataset are made available in Appendix C.

The MSE reduction ratio (nMSE) between the ω-th model and the Naive baseline for the kth trial and the hth forecasting horizon is computed as:

$$\text{nMSE}^{(\omega,k,h)} = (\text{MSE}^{(\omega,k,h)} / \text{MSE}^{(Naive,k,h)}) - 1 \tag{5.3}$$

Figure 5.12 shows the distribution of the nMSE according to (5.3) across $K = 45$ trials and for different forecasting horizons. Note that only negative values of nMSE correspond to an improvement in accuracy with respect to the Naive baseline.

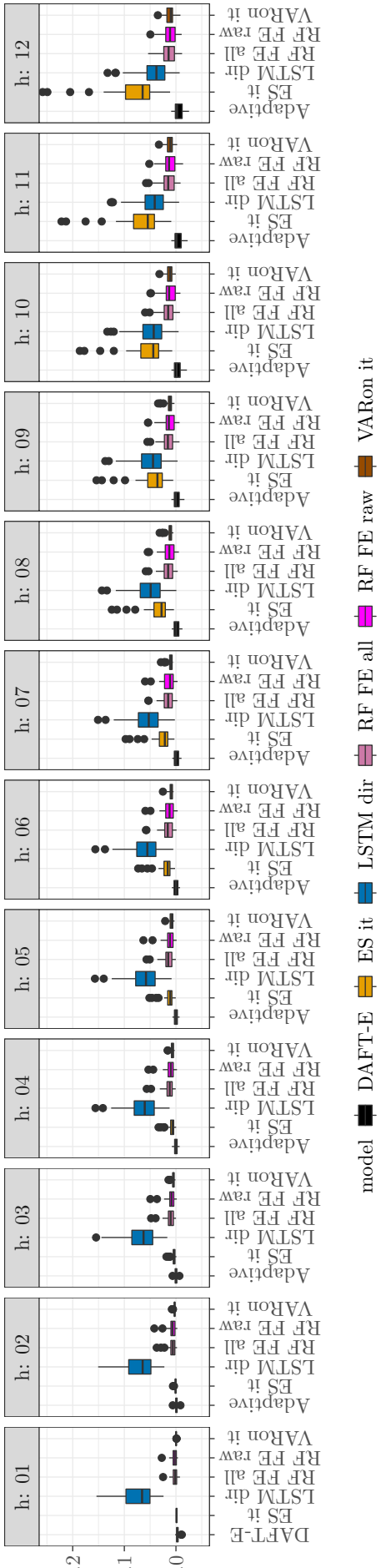


Figure 5.12: Visualization of the nMSE across the forecasting horizon for the Australian case study. The lower the nMSE is, the better the corresponding model is performing. An nMSE < 0 indicates that the corresponding model is outperforming the Naive model. The plot includes the Proposed Model (DAFT-E), the best performing algorithms among those combined in DAFT-E (RF FS all, RF FS raw), and the benchmark models showing the best performance.

Figure 5.13 shows that the DAFT-E combination strategy outperforms the single components (Lazy FS all, RF FS all, RF FS raw) taken individually, as well as state-of-the-art approaches. We get a similar result for the Italian benchmark (Fig. 5.14) but, for the sake of conciseness, we report only a smaller set of approaches.

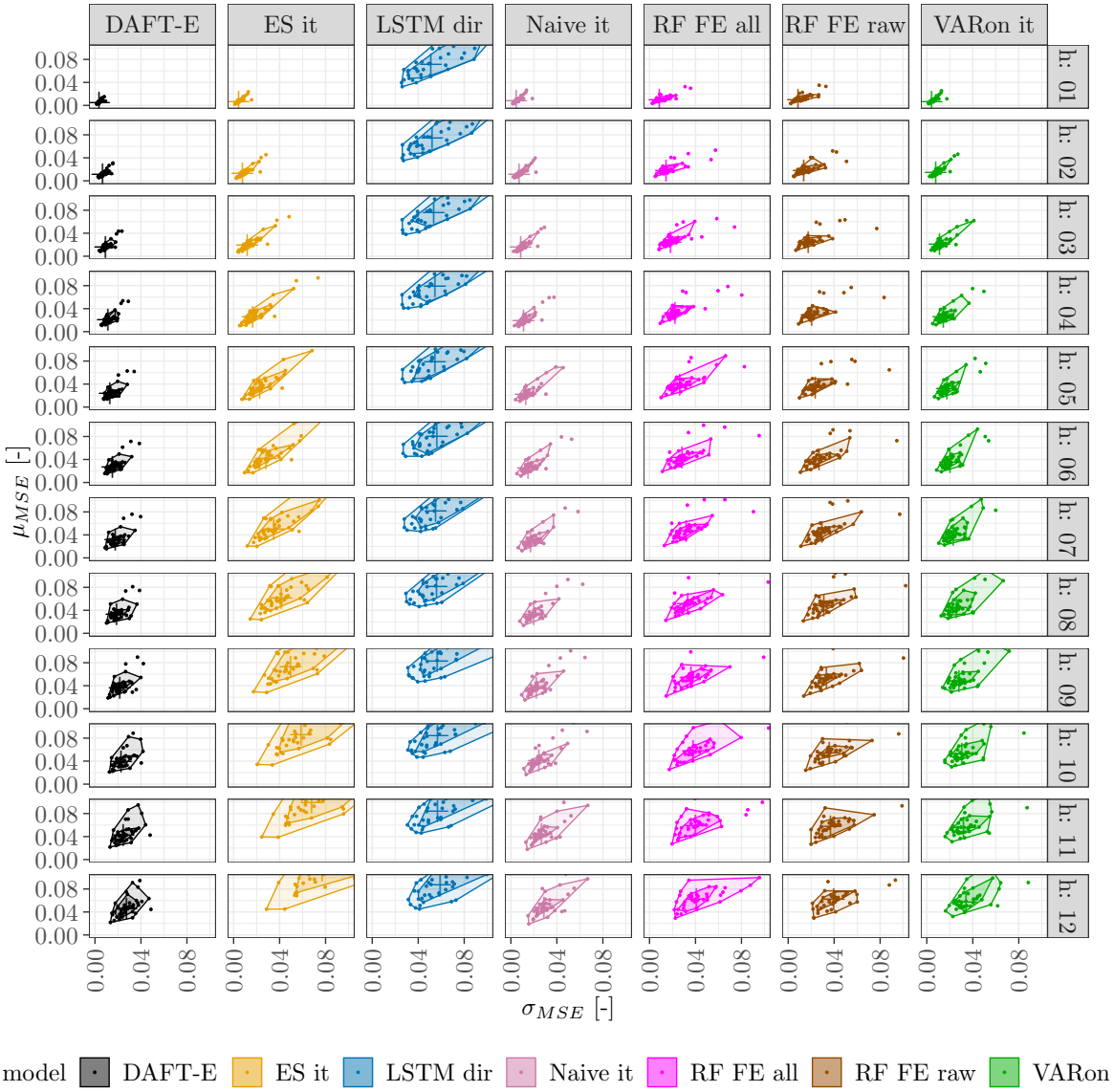


Figure 5.13: Visualization of the bagplot of the bivariate distribution μ_{MSE}-σ_{MSE} across the forecasting horizon h for the proposed model (DAFT-E), the internal algorithms of DAFT-E (Lazy FS all, RF FS all, RF FS raw, Naive it), and the the benchmark models showing the best performance - Australian case study. A smaller bagplot area indicates a reduced variability in the the corresponding method's predictions.

Fig. 5.15 shows the risk measures (MAE,VaR Equation (5.1) and cVaR Equation (5.2)) across different forecasting horizons. This representation allows to compare the average performance of different forecasting techniques (MAE) versus the worst-case ($\alpha = 95\%$) configurations. In particular, we consider that the lower the increment of those quantities for increasing horizons, the higher is the robustness.

Finally, Fig. 5.16 shows the computational times of all models. The total computation time accounts for feature engineering, embedding, feature selection, and model training steps. If a model does not include some of these steps, the corresponding computational time is null.

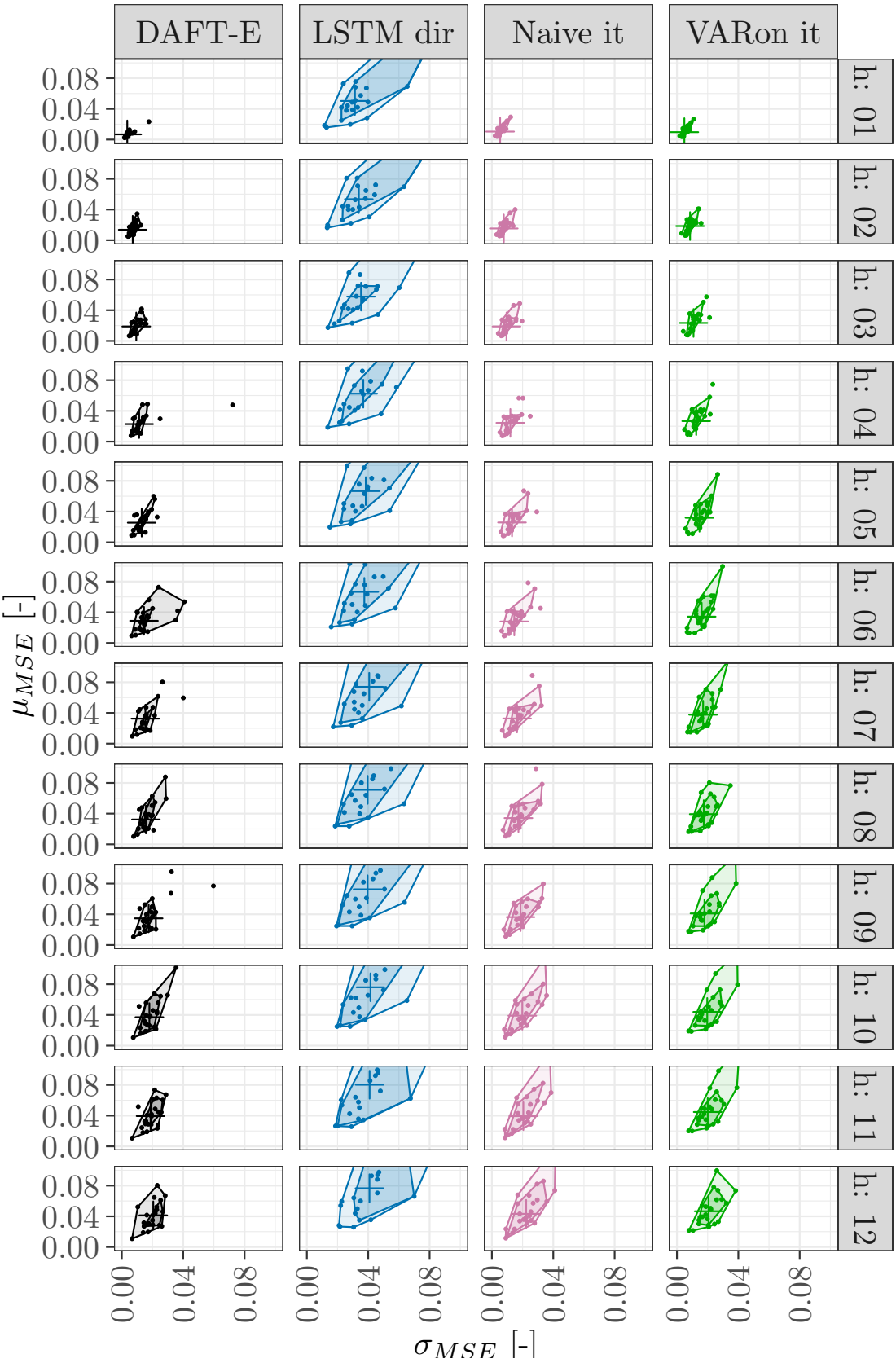


Figure 5.14: Visualization of the bagplot of the bivariate distribution μ_{MSE}-σ_{MSE} across the forecasting horizon h for the proposed model (DAFT-E), and the the benchmark models showing the best performance - Italian case study. A smaller bagplot area indicates a reduced variability in the the corresponding method's predictions.

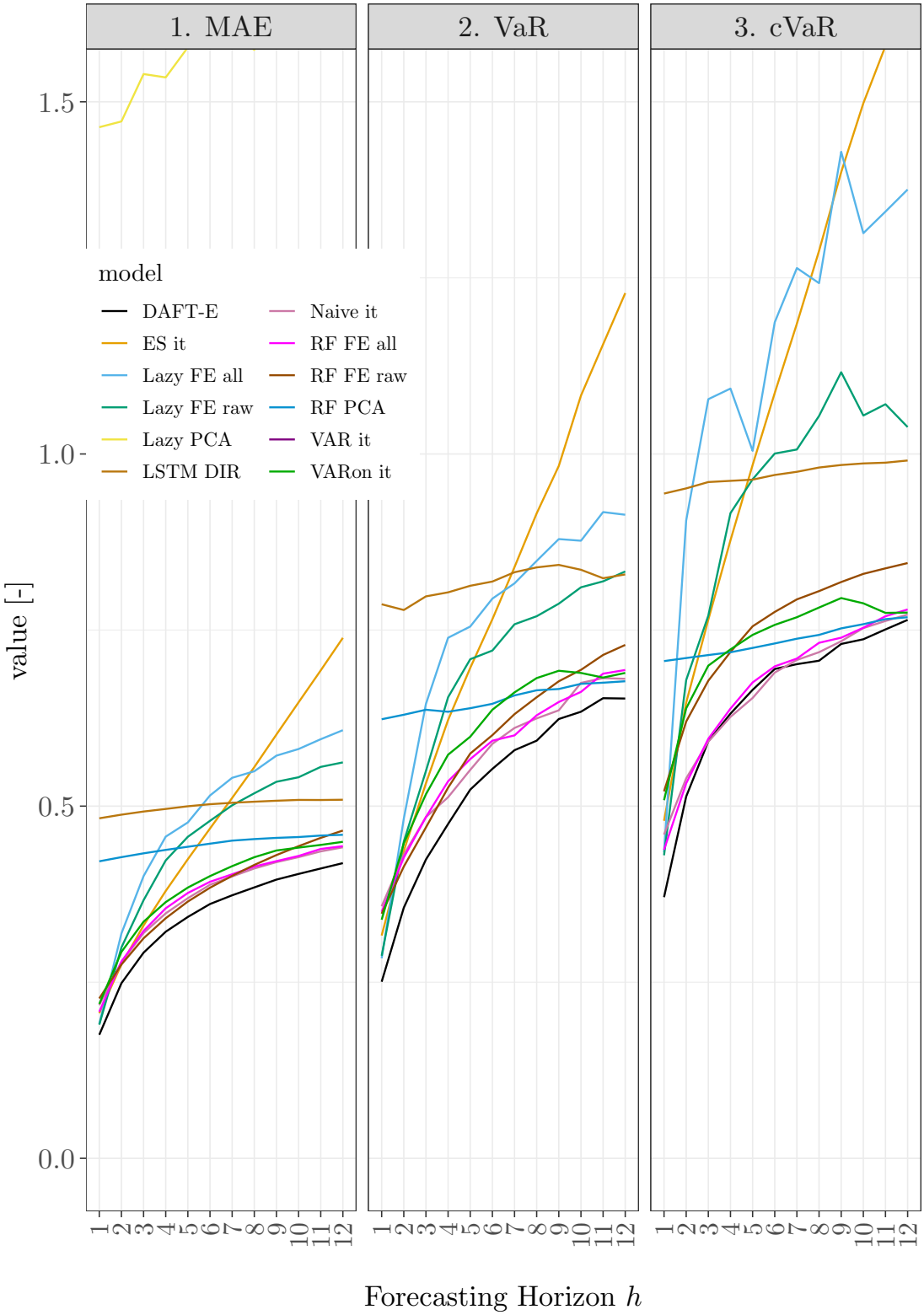


Figure 5.15: Visualization of the expected value (mean), VaR_α, and $cVaR_\alpha$ ($\alpha = 95\%$) across the forecasting horizon for all models - Australian case study. For all the three metrics, the lower the metric value is, the smaller is the risk (in terms of absolute forecasting error) that we are going to expect using the corresponding method.

5.4.5 *Conclusions*

Overall, some general considerations may be made on the basis of the experimental results:

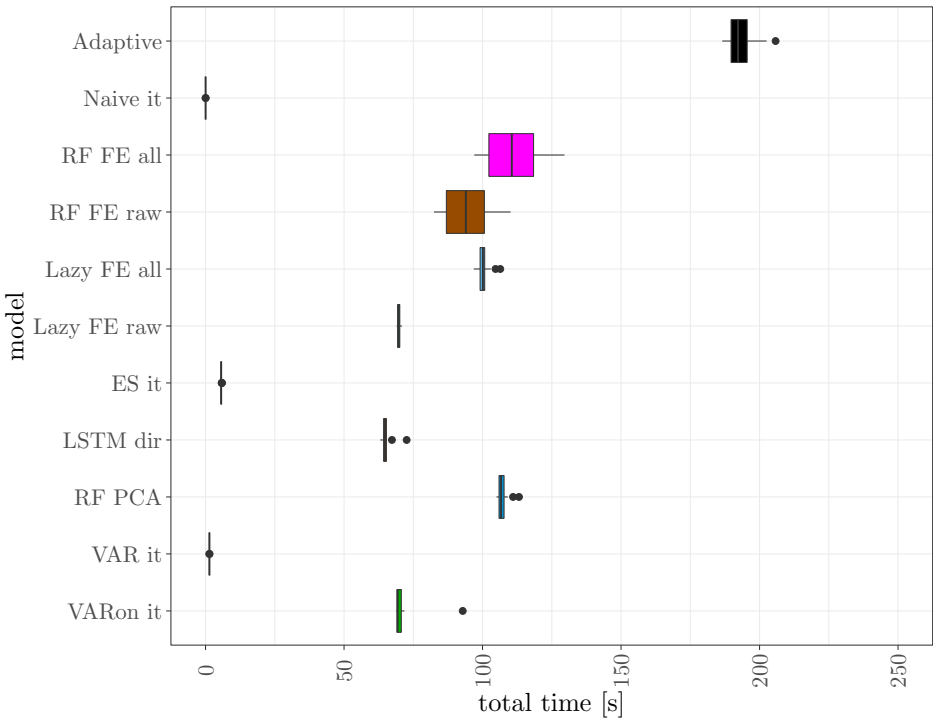


Figure 5.16: Spread of the computational time across $K = 45$ trials, Australian Case Study. Smaller computational times indicate an higher computational efficiency of the methods.

1. Multi-step-ahead wide area wind power forecasting relying solely on historical power data is a challenging task, as shown by the difficulty in improving over simpler baselines (Naive, Holt-Winters).

2. The proposed approach *DAFT-E* is a promising alternative to the state-of-the-art forecasting strategies in this context. In terms of nMSE, DAFT-E is the best model, followed by *VARon it* (Fig. 5.12). The *LSTM dir* accuracy is definitely the worst one. The *ES* performance dramatically decreases as h increases.

3. The *DAFT-E* has a balanced performance on the plane μ_{MSE}-σ_{MSE} over h as shown by Fig. 5.13. This is shown by the fact that its bagplot area is the smallest and is closer to the origin with respect to those of multivariate approaches like *VARon it*.

4. The *DAFT-E* has the least absolute error in terms of VaR and CVaR, for a $\alpha = 95\%$ confidence level. In particular, the best *DAFT-E* is the one using *RF FS all* and *RF FS raw* (Fig. 5.15).

5. The addition of smooth features to the information set improves the accuracy compared to the adoption of raw features only, particularly for large h (Fig. 5.13).

6. *DAFT-E* demands a much larger computational time than the fastest method (*VARon it*), yet such overhead is compensated by a better performance accuracy (Fig. 5.16). The same reasoning does not apply to *LSTM dir* whose heavier computational time is not compensated by an increase in accuracy.

7. The embedding procedure covers about 50% of the entire computational time. Future work will focus on speeding up this step (e.g. by making it parallel).

5.5 CONCLUDING REMARKS

In this chapter, we analyzed two implementations of the SMURF-ES strategy, each of them having different approaches on feature engineering and forecasting combinations, both devised to address the problem of wind power forecasting, a particularly challenging problem given the stochastic and intermittent nature of the wind.

The first strategy (*Error-based dynamic combination*) employs a feature engineering process aimed at exploiting the historical information embedded in the series, by computing a set of rolling statistics over the available values of the series, while the forecasting combination is made through a dynamic rule, giving more weight in the forecast combination to the models having a smaller forecasting error.

The second strategy (*DAFT-E*) extends the feature engineering process proposed by the first strategy by including a set of expert-based features, devised to capture specific dynamics wind power generation, which can dramatically impact the forecasting performance. The forecasting combination is also improved on two aspects: the nature of the underlying models and the combination rule. Instead of considering fully heterogeneous models, in this ensemble we consider the models of the same family, but trained on different input features, in order to still preserve their diversity. As for the combination rule, the introduction of a set of forgetting factors allows to combine models based not only on their performance on the current set of values, but also to account for past forecasting performance.

Both strategies have been assessed on data coming from real wind farms, employing an extensive cross-validation and post-hoc statistical analysis. The results confirmed the efficiency of heterogeneous ensembles with dynamic combinations rules, a well-known result from the scientific literature (Cerqueira et al., 2017a). Moreover, the statistical analysis allowed to assess the robustness of the proposed technique across different forecasting horizons (Figures 5.8 and 5.7) as well as the bias-variance tradeoff associated to the different models (Figures 5.13 and 5.14), confirming the forecasting error variance reduction associated to forecasting combination techniques.

Part III

CONCLUSIONS AND FUTURE WORK

"What we know is limited, and what we don't know is infinite."

– P. Laplace

6

CONCLUSION

6.1 CONCLUSIONS

The technological revolution of the last two decades has brought an exponential increase in the quantity of produced data across the world (Figure 1.2). More precisely, the development and miniaturization of interconnected computing devices (i. e., the IoT revolution) has led to a pervasive diffusion of these devices in several industrial domains. These devices, when equipped with various sensors, could be employed for different tasks, ranging from production monitoring (in the energy and manufacturing domain), to traffic analysis (both on streets and computer networks), to continuous processing of publicly available data (e. g., stock market analysis). The common trait among these seemingly different domains is the need for analyzing a large quantity of data coming from heterogeneous sensors, potentially displaying a highly dynamical behavior and spatio-temporal dependencies among them. In addition, in some of these domains (e. g., energy production and stock market), it might also be beneficial to produce accurate predictions of the analyzed quantities for one or multiple steps in the future.

In more formal terms, the majority of the aforementioned problems can be addressed as multivariate (to account for interdependencies among the series) and multiple-step-ahead forecasting problems (to produce accurate forecasts for several steps in the future). Multivariate and multi-step-ahead forecasting is a particularly difficult problem, combining both the problems related to the long-term estimation in the future (error accumulation, increasing uncertainty) with the computational burden required to estimate multivariate models with a large number of variables. In order to simplify the problem, the multivariate and multiple-step-ahead are often treated separately in the scientific literature. In other words, either the problem is approached as multivariate, but considering a one-step-ahead forecasting horizon, or a multiple-step-ahead approach is employed, but the task is decomposed in a set of univariate forecasting problems. This thesis focused on tackling the two aspects jointly, with a particular focus on techniques inspired by the machine learning domain.

In Chapter 2, we analyzed the scientific literature concerning multivariate time series forecasting for both statistical (model-driven) and machine learning (data-driven approaches).

Among the statistical approaches, VAR-based and generalized DFM approaches are the most employed in practice, especially in the field of econometrics. The popularity of VAR-based approaches is closely related to the extensive study of the model and its properties in the scientific literature (De Gooijer and Hyndman, 2006), as well as the interpretability of the model, due to its simple structure. The ease of interpretability comes at the price of an increased number of parameters to be estimated, which limits the applicability of the technique when the dimensionality of the data increases (Escribano, Peña, and Ruiz, 2021). Generalized DFM (Forni et al., 2000) models address the dimensionality issue by including a dimensionality reduction component to compress the original high dimensional space into a smaller one, while still retaining the majority of the informative content. Although this choice represents an improvement over the VAR model, both the approaches still rely on strong assumptions about the distribution of the original data (normality and stationarity) and are limited to a one-step-ahead forecast.

In recent years, after their remarkable performance in several different applications such as image and signal analysis (Dargan et al., 2020) with limited knowledge and assumptions on the underlying data, Deep Learning models have also made their way in the domain

of time series forecasting (Torres et al., 2021). The recent success of these Deep Learning models can be explained by their ability to overcome the limitations of the aforementioned statistical approaches, by offering intrinsic support to multiple-step-ahead forecasting, no assumptions on the nature of the input data, in addition to the possibility of easily scaling up the architecture of the model as the input dimensionality increase. The drawback of these approaches is that the resulting models have an extremely large amount of parameters to be estimated, which consequently require computationally intensive training procedure and fine-tuning of the parameters to obtain a consistent forecasting performance.

In this context, we observed that traditional machine learning approaches have received limited attention in the multivariate and multiple-step-ahead literature and that the proposed approaches focus on a particular ML technique (i. e., SVM, k-NN and fuzzy approaches), specifically tailored to the considered problem (Bitencourt and Guimarães, 2021; Talavera-Llames et al., 2019; Wang et al., 2020). In order to fill this gap, we proposed two forecasting strategies, discussed in Chapter 3, employing traditional machine learning components such as dimensionality reduction, feature engineering and feature selection approaches, and combining concepts coming from both the statistical and the data-driven literature.

The first strategy, DFML, is inspired by the Generalized DFM approach and combines the ease of interpretability of the DFM framework with the flexibility offered by the choice between non-linear non-parametric and parametric models for both the factor estimation and the factor forecasting components.

In Chapter 4 we assessed the DFML strategy on several heterogeneous real-life problems. In these contexts DFML showed its competitive performance on very large scale problems ($n > 10^2$), in particular when employing a linear dimensionality reduction technique (PCA - Section 2.4.1), in combination with multiple-step-ahead lazy techniques (Section 2.3.3.2), both with and without automatic parameter selection. Although this assessment contributed to a better understanding of which family of techniques should be employed within the DFML for a given problem, the choice between linear/non-linear dimensionality reduction and model-driven/data-driven techniques still remains a challenging problem in practical applications. Moreover, some experiments on smaller-scale problems (i. e., wind power forecasting problems with $n \sim 30$) highlighted that performance improvements related to the DFML approach (i. e., dimensionality reduction followed by forecasting) were greatly reduced, bringing the model accuracy closer, when not inferior, to that of the benchmarks.

This decrease in performance led us to rethink the problem approach to multivariate forecasting in order to overcome the encountered limitations, yielding to the development of the SMURF-ES strategy through three structural modifications to the DFML strategy.

The first modification consisted in replacing the dimensionality reduction module with a feature engineering component followed by feature selection. The rationale of this choice is that, given the reduced problem dimensionality, the increase in the number of features could still be computationally affordable if the added informative content by the feature engineering process can improve the forecasting performances. Moreover, the feature selection process allows to greatly reduce the number of input features to the problem while still preserving the informative content.

The second modification is related to the problem decomposition: instead of generating a reduced set of transformed (latent) variables, the SMURF-ES approach works in the original feature space, tackling each time series in the multivariate data as an independent problem. Once again, such approach is computationally manageable only when the problem size is small $n < 30$.

The last modification is related to the forecasting component. Instead of relying on a single forecasting technique at a time, as in the DFML, the SMURF-ES strategy employs a forecasting combination technique, to further improve the predictive accuracy on the different individual problems. The forecast combination includes a dynamic combination of

heterogeneous (i. e., coming from both the statistical and machine learning domains) and computationally inexpensive models, in order to improve the diversity of the ensemble, while still limiting the computational burden.

Overall, as the DFML strategy, SMURF-ES combines traditional statistical approaches with data-driven ones, into different components of the traditional machine learning pipeline of feature engineering, feature selection, and forecasting.

In Chapter 5 we assessed the SMURF-ES strategy, through two specific implementations, the DAF-E and the DAFT-E (Sections 3.3.5.1 and 3.3.5.2) on the problem of wind power forecasting. This assessment showed the complementarity of the SMURF-ES strategy with respect to the DFML on smaller scale datasets. Although the combined process of feature engineering and selection could cause a dramatic increase in the number of available features, in the case of smaller datasets ($n \sim 30$) this approach remains an effective technique to improve the informative content available for forecasting. Analogously, an ensemble approach (requiring the training of multiple models) with a multiple-step-ahead direct strategy for each univariate series remains computationally feasible only when the dimensionality of the input data is reduced. The results on real datasets show the outperformance of the proposed strategies with respect to state-of-the-art techniques in both statistical (i. e., online VAR) and DL domain. Moreover, the robustness of the proposed strategies has been assessed through conventional statistical testing and by applying a novel statistical approach inspired by the risk management theory.

The main message of the thesis can be summarized as follows: algorithms and methodologies from the traditional machine learning literature have been neglected in the multivariate and multistep-ahead forecasting literature, where the dominant approaches are either from the econometric or the deep learning domains.

Nevertheless, we have shown that traditional machine learning approaches can produce competitive results when employed in tailored strategies for multivariate and multiple-step ahead forecasting. Moreover, we have assessed that, for a similar predictive accuracy, traditional machine learning approaches have a reduced computational time compared to deep learning techniques. This reduction also implies reduced energy consumption which could be beneficial for practical applications (e. g., embedded systems).

We showed that the forecasting performance can be further improved by employing conventional approaches, such as automated model selection and feature engineering, without the need for overly complex model structures (e. g., those proposed in DL). Still, in our approach to multivariate and multi-step-ahead forecasting several methodological choices need to be performed for both of the strategies, for instance, the dimensionality reduction/feature selection techniques and the forecasting models. Our assessments helped to shed some light on the most relevant components in the proposed strategies, but further research still needs to be performed to determine the best methodological choices for the problem at hand.

Yet, our proposed strategies confirmed that a combination of statistical and machine learning components, often seen as competitors, is beneficial to improve forecasting accuracy for multivariate multiple-step-ahead problems, as the peer-reviewed work (i. e., journal papers and patent) in both the academic and industrial domain showed. Learned lessons and perspectives for future research are discussed in the following sections.

6.2 GUIDELINES FOR STRATEGY CHOICE

This section aims to briefly summarize the main findings of this thesis in an quick guide to the practitioner, as summarized in Figure 6.1. First of all, we would like to direct the attention of the reader to an aspect often neglected in the framework of multivariate and multi-step-ahead forecasting. Even though it might seem counterintuitive, the first test a

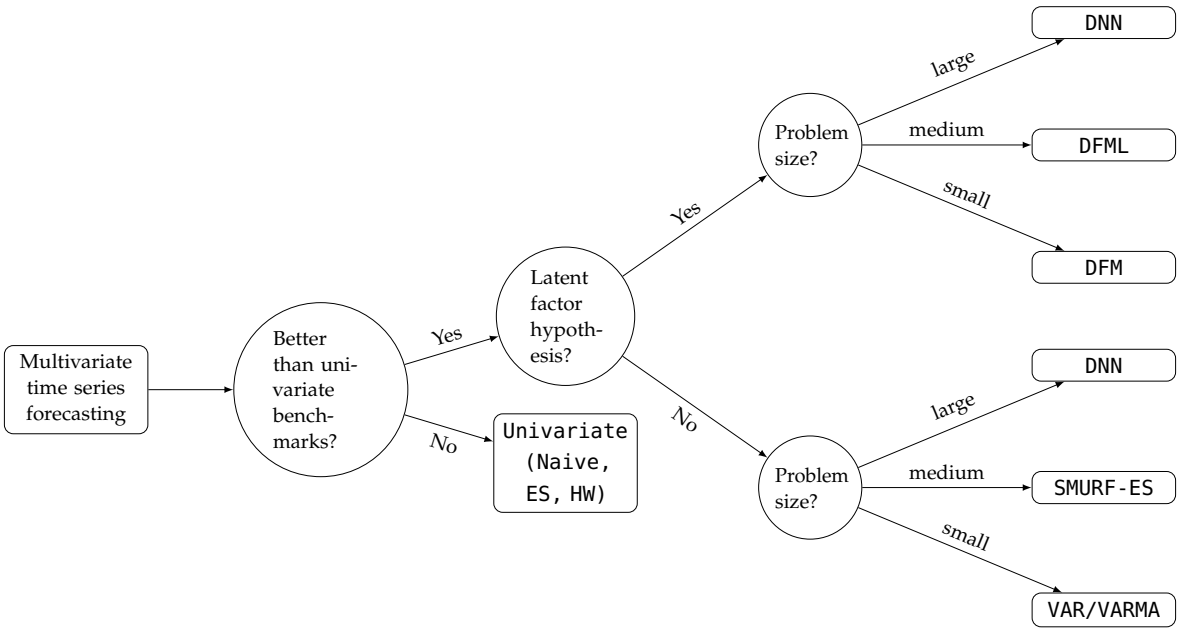
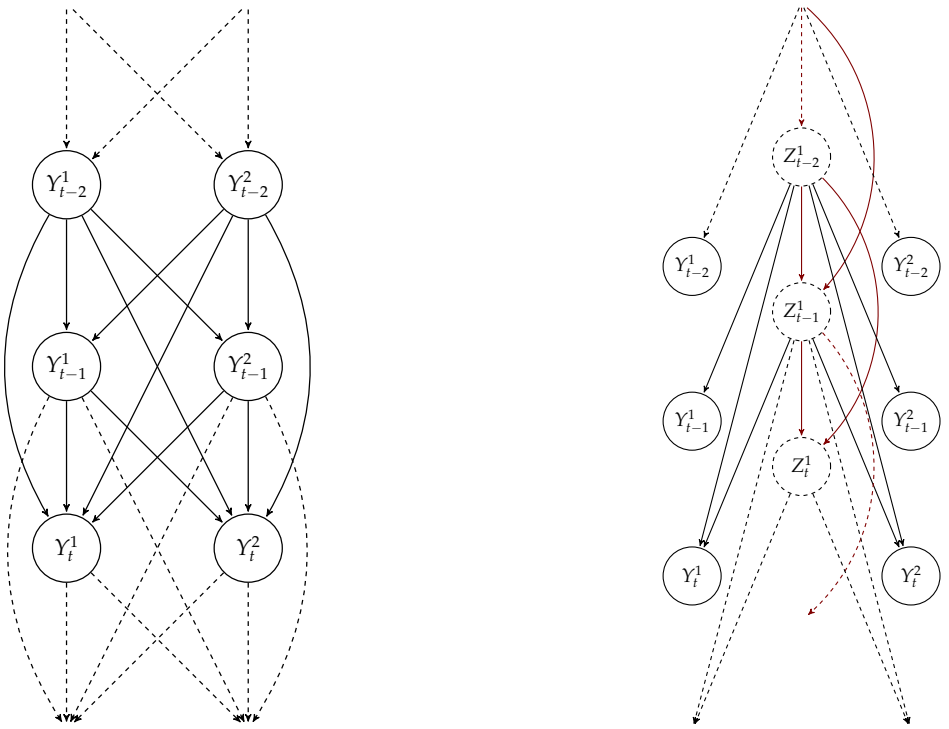


Figure 6.1: Indicative decision tree summarizing the criteria for the choice of the forecasting approach for multivariate problems, based on the results presented in this thesis.

practitioner should do is to employ univariate forecasting techniques (such as those employed as benchmarks in forecasting competitions: Naive - Section 2.3.2.1, ES - Section 2.3.2.3, for instance) as baseline references for their techniques, and test for a statistically significant outperformance. As recent works showed (Paldino et al., 2021; Shah and Shroff, 2021), univariate techniques are still tough competitors to beat, even with high dynamical series, especially for shorter horizons. Once the out-performance of the baselines has been established, the choice of a forecasting approach is mostly driven by two factors: the presence of latent factors and the dimensionality of the input data (i. e., the number of time series n). First, the practitioner needs to consider whether the existence of latent temporal factors (Figure 6.2) can be hypothesized. For instance, in the case of renewable energy forecasting, we might assume that a latent dynamic shared by all the series might be related to the atmospheric weather, while in the case of financial time series, the market's behavior could be the underlying latent dynamic.

If this latent dynamic exists, DFM models are a promising choice for smaller multivariate series (indicatively $n < 10$, based on our experiments). As the dimensionality increases, our proposed strategy DFML should be preferred thanks to its efficient way to reduce dimensionality while still preserving informative content, especially with strongly correlated datasets. For very large scale datasets ($n > 800$, beyond the scale of our tests), based on the current literature, we would advise to employ deep learning models. If the tests with dynamic factor models, both traditional and machine-learning based, did not give satisfying results, one should resort to statistical techniques. Especially when the problem size is small, techniques such as basic VAR or its regularized version are tough competitors to beat. For small to medium-sized multivariate series (indicatively $10 < n < 30$, according to our assessments), SMURF-ES still offers a good balance between forecasting accuracy and computational complexity, and should be preferred to the DFML strategy. On the other hand, if the number of time series becomes larger, deep learning models could be a more flexible choice.



(a) Without latent dependencies

(b) With latent dependencies

Figure 6.2: Graphical models illustrating two different setups for multivariate ($n = 2$) time series forecasting: a case without the presence of a latent dynamics (6.2a) and a case displaying a latent dynamics with a single latent factor (6.2b). Dashed lines indicate unobserved elements, while latent dependencies are plotted in red.

6.3 LIMITATIONS OF OUR APPROACH AND FUTURE WORK

This thesis represents the first step towards a machine learning based solution of the multivariate and multiple-step-ahead forecasting problem, considered as the most complex forecasting problem.

In Section 3.1 we presented the formalization of the multivariate forecasting problem and we discussed the different approaches to its solution (i. e., global, local, and hybrid).

The domains of global (especially with neural approaches) (Lara-Benítez, Carranza-García, and Riquelme, 2021) and hybrid forecasting (with a focus on hierarchical forecasting) have seen an extremely fast evolution in recent years. Due to a combination of lack of time, expertise, and reproducible implementations, we have not been able to compare our approaches to state-of-the-art techniques in these domains (i. e., deep recurrent network, such as Transformers (Lara-Benítez et al., 2021)), although our experiments showed promising results in terms of trade-off between forecasting accuracy and computational burden required to produce the results. Moreover, our assessements focused on problems characterized by highly dynamical settings. This setting favors models capturing more short-term dependencies in the data (such as the proposed DFML and SMURF-ES strategies, with small model orders) but at the same time being a least favorable setting for models based on LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014a) cells, specifically built to capture longer term dependencies.

On a short-term horizon, from this literature analysis, two promising research directions come naturally.

The first one would aim to deepen the study of neural approaches (such as recurrent autoencoders) and their integration in the forecasting strategy. The non-linear nature of such approaches, combined with their capabilities to model long-term temporal dynamics, could

be beneficial to improve the dimensionality reduction components in the DFML as well as provide meaningful additional features in SMURF-ES.

The second one would aim to develop hierarchical extensions to the DFML and SMURF-ES strategy, in order to exploit the intrinsic information available in problems presenting a hierarchical structure (such as mobility forecasting on a network of roads).

In addition, although we focused on large-scale problems ($n > 10^2$), the Big Data revolution is shifting the focus towards even bigger problems, with the additional complication of having continuous streams of data.

To address the dimensionality problem one could rely on unsupervised machine learning techniques (such as clustering (Pathak et al., 2021)) to split the original problem into smaller sub-problems that could be tackled independently of each other with state-of-the-art techniques such as the proposed strategies.

To address the streaming issue, further studies on incremental/iterative implementations of the proposed strategies should be analyzed. In Chapter 4, we empirically validated an iterative extension of the DFML framework for linear factor estimation techniques and data-driven factor forecasting techniques, but the modular structure of the framework could easily support the extension for non-linear factor estimation techniques and model-driven factor forecasting techniques.

Forecast combination is another relevant approach that has been proven effective by practitioners (see the top-performing techniques in M4 (Makridakis, Spiliotis, and Assimakopoulos, 2020b) and M5 (Makridakis, Spiliotis, and Assimakopoulos, 2020a) competitions) and that deserves further assessment. Although we did assess its effectiveness (as forecasting component in the SMURF-ES strategy), further experiments need to be performed to generalize the results we obtained and to provide implementation guidelines for multivariate problems. In addition, forecast combination could also be directly be employed as factor forecasting technique within the DFML, potentially improving the forecasting capabilities of the strategy at a reduced computational cost (as it works in the reduced dimensionality factor space).

Last but not least, the work presented in this thesis focus solely on point forecasting (i. e., predicting a single point for each horizon h), instead of probabilistic forecasting, which focuses on trying to predict the conditional distribution of the forecasts at each time step. Several variants of probabilistic forecasting exists (Guen and Thome, 2021) based respectively on: Monte-Carlo estimation (Laptev et al., 2017; Taieb, Taylor, and Hyndman, 2017a), quantile forecasting (Gasthaus et al., 2019) or predictive distribution approximation via parametrized distributions (Salinas et al., 2020) or generative models (Koochali, Dengel, and Ahmed, 2020). Even though both the DFML and SMURF-ES strategies have not been natively developed for probabilistic forecasting, their modular architecture could easily be adapted to support probabilistic forecasting, by plugging probabilistic models in their forecasting components.

6.4 LONG TERM PERSPECTIVES OF MULTIVARIATE MULTI-STEP FORECASTING

On the other hand, on a longer-term horizon, several theoretical problems need to be addressed.

For instance, a question often neglected in the domain of forecasting is the trade-off between the computational cost required to train a model and the performance gains offered by this model with respect to a baseline technique. In the case of Deep Learning, as presented by the authors of (Thompson et al., 2020), the increase in computational complexity required to obtain a minimal gain in terms of forecasting error is of several orders of magnitude (Figure 1.4), making such approaches clearly unsustainable on the long term.

Further work is required to characterize this trade-off (for instance, in a meta-learning fashion). In addition, the application of techniques such as racing (Birattari et al., 2002)

or early-stopping/regularization could be beneficial in decreasing the computational cost while still retaining predictive capabilities.

Another related issue involves dealing with the change of computational paradigm that is progressively occurring in recent years, often referred as Internet of Things. A consequence of this shift is that connected devices are becoming more and more pervasive in people's everyday life (e. g., smart meters for household consumption, voice-activated digital assistants, wearable devices). These devices have the capabilities of collecting data through an array of different sensors and of processing said data directly on-board. This paradigm change will require novel learning approaches, such as federated learning (Konečný et al., 2017; McMahan et al., 2017), in order to perform accurate forecasting while still complying with ethical and privacy-related constraints.

Last but not least, in terms of practical applications, multivariate and multi-step-ahead approaches to forecasting problems could provide useful insights in contrasting the effect of anthropogenic climate change. More precisely, better forecasts in terms of renewable energy production and people's mobility trends could provide economic benefits as well as reductions in terms of emissions of several types of pollutants. Moreover, improved multivariate monitoring and forecasting of environmental conditions through distributed sensors could help prevent catastrophic events (e. g., floods and forest fires).

Part IV

APPENDIX

APPENDIX A - VOLATILITY

A.1 VOLATILTY DEFINITION

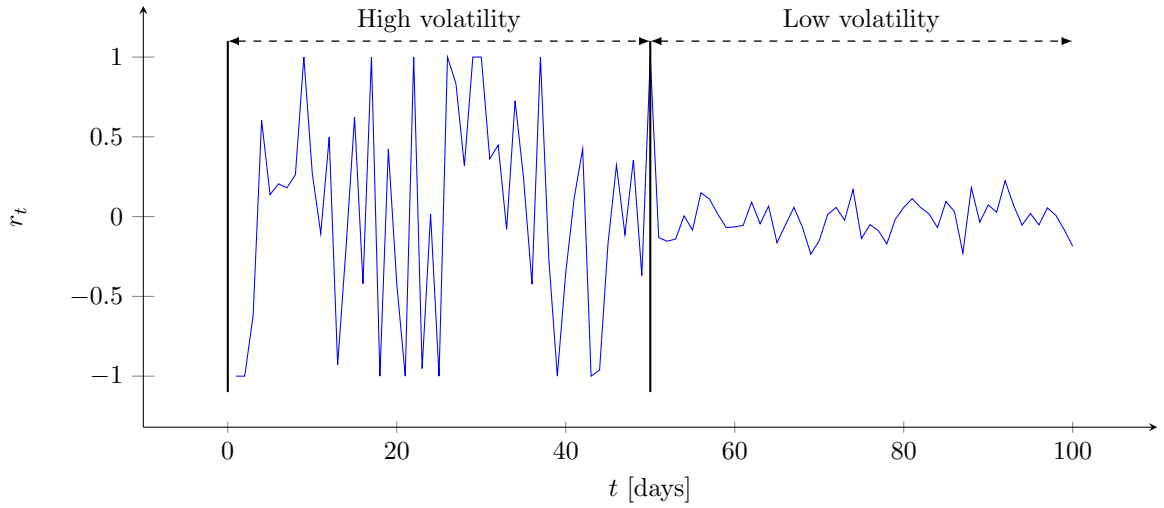


Figure A.1: Graphical representation of OHLC price data on a dummy time series. Time is expressed as a percentage of calendar days. f represent the fraction of the day in which the trading is closed. In the case of CAC40, $f = 0.6458333$.

Before discussing the details of the definition of volatility, we will establish the notation of all the relevant quantities involved. In our study, we are considering temporal data of financial instruments traded on a stock market. From a temporal standpoint, a stock market is characterized by two fundamental quantities: trading days and trading hours. A stock market's trading days is the subset of the days of the calendar year in which trading take place. In every trading day, the market does not allow trading (i.e. is closed) for a fraction of the day f. Conversely, for the remaining part of the day $1 - f$, the market is said to be open, and trading could take place.

For instance, the stocks composing the CAC 40 index are traded on the Euronext NV market, whose trading hours are from 9 AM to 5:30 PM CET, Monday to Friday. [1] For instance, after accounting for bank holidays, the year 2016 will have 256 trading days.

For every trading day t, we define the opening price $P_t^{(o)}$ as the price of the considered instrument at the opening of the trading day. Analogously, we can define the closing price $P_t^{(c)}$ as the instrument value at the end of the trading day. The high price $P_t^{(h)}$ and low price

1 See Euronext NV calendar for details on the trading times.

$P_t^{(l)}$ represent the extreme price values of the trading day t, respectively the maximum and minimum. To summarize:

$$P_t^{(o)} = P_{t_{09:00}} \tag{A.1}$$

$$P_t^{(h)} = \max_{s \in \{t_{09:00}, t_{17:30}\})} P_s \tag{A.2}$$

$$P_t^{(l)} = \min_{s \in \{t_{09:00}, t_{17:30}\})} P_s \tag{A.3}$$

$$P_t^{(c)} = P_{t_{17:30}} \tag{A.4}$$

$$\tag{A.5}$$

An example of the available data, annotated with the different quantities, can be seen in figure A.1.

In the context of trading, such information is generally summarized in a single box and whiskers cart, called candlestick charts, where the height of the box represent the difference between opening and closing prices, while the whiskers represent respectively the extreme prices of the days (cf. Figure A.2).

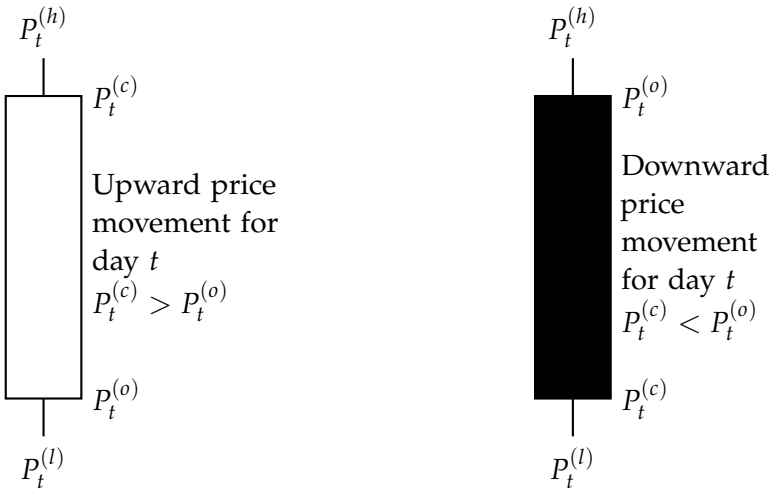


Figure A.2: Candlestick representation of OHLC price data for days displaying upward/downward price movements.

In the continuation of the thesis, we will refer to the closing price $P_t^{(c)}$ as price (A.6) at time t.

$$P_t = P_t^{(c)} \tag{A.6}$$

This will allow us to define, in turn, the notions of continuously compounded return (A.7) and return (A.8).

$$r_t = \ln\left(\frac{P_t}{P_{t-1}}\right) = \ln(P_t) - \ln(P_{t-1}) \tag{A.7}$$

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}} = \frac{P_t}{P_{t-1}} - 1 \tag{A.8}$$

In addition to the price, we define the volume V_t as the number of shares that have been traded (i.e. they changed owner), during a given trading day t.

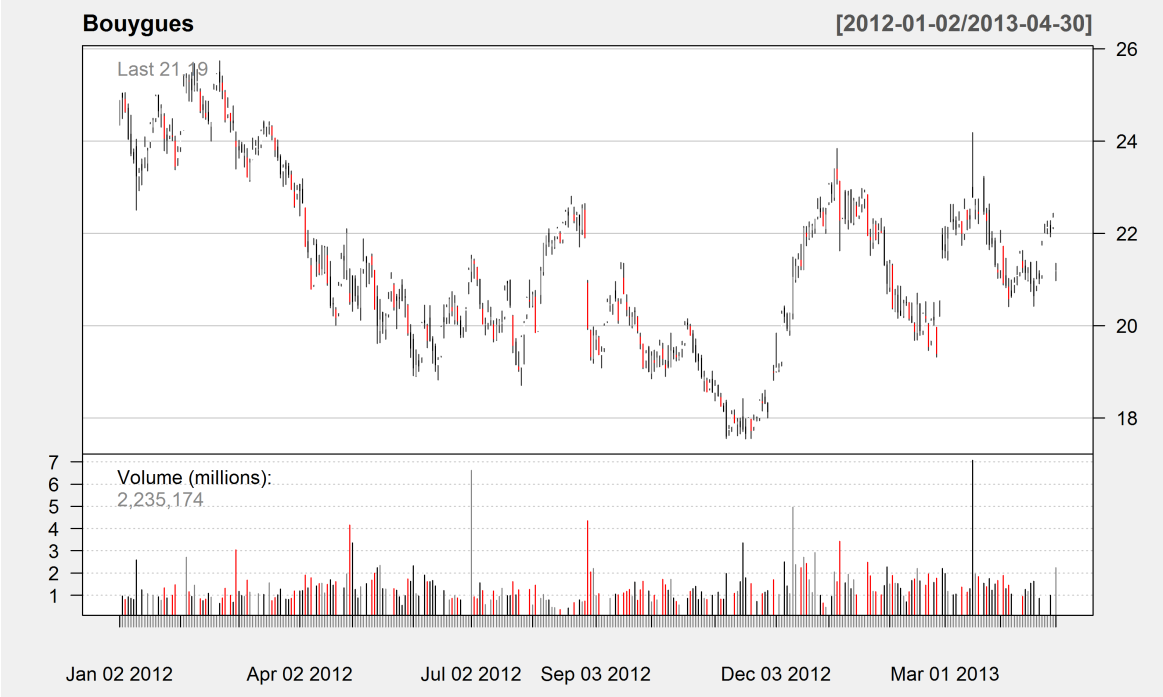


Figure A.3: Representation of the evolution of the stock market valuation of Bouygues Telecom (one of the composing series of CAC40) as a candlebar graph.

A.2 LITERATURE REVIEW

In finance, volatility is a measure of the degree of variation of a trading price series over time. It can be also seen as a quantification of risk associated to the investment, capturing how strongly the value of the stock is going to oscillate at a certain moment in time. It should be noted that, by being a measure of dispersion, volatility does not give precise information on the direction of such oscillations, and that specific measures have been developed to focus primarily on the dispersion of values associated to economic losses. (cf. (Poon and Granger, 2003)). Unlike prices or returns, volatility is not directly observable (i.e. latent variable); thus a proxy needs to be employed (cf. (Santamaría-Bonfil, Frausto-Solís, and Vázquez-Rodarte, 2015)). The choice of such proxies is highly dependent on the nature and the granularity of the available data.

A.2.0.1 *Volatility as variance*

The problem of measuring the degree of variation of a given time series over time can also be seen as the characterization of the dispersion of the data over time. With this in mind, the most natural proxy for the volatility is standard deviation of the original price series or a derived measure. As a matter of fact, (Poon and Granger, 2003) proposes to define volatility as the empirical standard deviation of the continuously compounded return series, over a time window of N observations.

$$\hat{\sigma}_t^{(SD)} = \sqrt{\frac{1}{N-1}\sum_{i=0}^{N-1}(r_{t-i} - \bar{r})^2} \tag{A.9}$$

where $\bar{r}$ corresponds to the average continuously compounded return for the N time frame. The sample size N, however, is an important parameter to correctly capture the variability of the underlying data. Common choices of N are corresponding to the number

of trading days in a week (cf. (Santamaría-Bonfil, Frausto-Solís, and Vázquez-Rodarte, 2015)) on in a month (cf. (Mittnik, Robinzonov, and Spindler, 2015)), in order compute weekly/monthly volatility.

Such definition has several advantages. Firstly, the empirical standard deviation $\hat{\sigma}^{(SD)}$ is a distribution free statistic. As such, it could be computed regardless of the assumptions made on the nature of the underlying distribution of the data. Secondly, only the knowledge of daily closing prices or returns is needed for its computation.

It should be noted that when $\bar{r} = 0$, this proxy reduces to the scaled sum of the squared continuously compounded returns.

$$\hat{\sigma}_t^{(SD)} = \sqrt{\frac{1}{n-1}\sum_{i=0}^{n-1} r_{t-i}^2} \tag{A.10}$$

Such formula could then be applied on de-trended time series (that by construction have $\bar{r} = 0$). Conversely, a null mean does not necessarily imply the absence of a trend.

A.2.0.2 *Volatility as a proxy of the coarse grained intraday information*

A common practice, in the financial world, is to augment the available price information with additional discrete intraday measures, namely opening, maximum (high), minimum (low), closing prices and the volume of transactions. As previously mentioned, this data format is usually referred as OHLC. The study by (Garman and Klass, 1980) propose to include such supplementary available information to improve the statistical properties of the proxy, yielding to an estimator with reduced bias and variance (cf. Table A.1). In the following, we will introduce the different proposed estimator, organizing them according to information required to compute them.

Estimator $\hat{\mathbf{y}}$	**Efficiency = Eff$(\hat{\mathbf{y}}) = \frac{var(\hat{\sigma}_0(t))}{var(\hat{y})}$**
$\hat{\sigma}_t^0$	1
$\hat{\sigma}_t^1$	2
$\hat{\sigma}_t^2$	≈ 5.2
$\hat{\sigma}_t^3$	≈ 6.2
$\hat{\sigma}_t^4$	≈ 7.4
$\hat{\sigma}_t^5$	≈ 7.4
$\hat{\sigma}_t^6$	≈ 8.4

Table A.1: Efficiency of the proposed volatility estimators in the literature ((Garman and Klass, 1980)). The baseline efficiency value is 1, while higher values represent better efficiency.

CLOSE PRICES The first estimator $\hat{\sigma}_0$, which they propose as benchmark value, simply consist of the squared value of the returns (i.e. the ratio of the logarithms of the closing price time series):

$$\hat{\sigma}_t^0 = \left[\ln\left(\frac{P_{t+1}^{(c)}}{P_t^{(c)}}\right)\right]^2 = r_t^2 \tag{A.11}$$

OPEN/CLOSE PRICES The second proposition $\hat{\sigma}_t^1$ is able to reduce the variance of the estimator, by including the opening price, and computing a weighted average between two components, representing respectively the nightly and daily volatility:

$$\hat{\sigma}_t^1 = \underbrace{\frac{1}{2f} \cdot \left[\ln\left(\frac{P_{t+1}^{(o)}}{P_t^{(c)}} \right) \right]^2}_{\text{Nightly volatility}} + \underbrace{\frac{1}{2(1-f)} \cdot \left[\ln\left(\frac{P_t^{(c)}}{P_t^{(o)}} \right) \right]^2}_{\text{Intraday volatility}} \tag{A.12}$$

The value of f is by definition bounded in the interval $[0, 1]$, with 0 representing the case when the market never close and 1 indicating that the market is always closed In the case of CAC40, we have that $f > 1 - f$, since trading is only performed of roughly one third of the day. In this case, the weighting scheme proposed in A.12 will give higher weight to the intraday volatility, with respect to the nightly one.

HIGH/LOW PRICES The third estimator, derived by (Parkinson, 1980) through the modeling of the price evolution as a stochastic diffusion process with unknown variance, bases its estimation on a function of the variation range (i.e. the difference between maximum and minimum value for the current trading day):

$$\hat{\sigma}_t^2 = \frac{1}{2\ln 4} \cdot \left[\ln\left(\frac{P_t^{(h)}}{P_t^{(l)}} \right) \right]^2 \tag{A.13}$$

where the value $\frac{1}{2\ln 4}$ corresponds to the variance of the distribution of the high-low displacement difference, under the assumption of a stochastic Wiener process (i.e. with normally distributed increments).

OHLC PRICES (Garman and Klass, 1980) further improves the efficiency of the estimator in Equation A.13 by including the information concerning nightly volatility.

$$\hat{\sigma}_t^3 = \underbrace{\frac{a}{f} \cdot \left[\ln\left(\frac{P_{t+1}^{(o)}}{P_t^{(c)}} \right) \right]^2}_{\text{Nightly volatility}} + \underbrace{\frac{1-a}{1-f} \cdot \hat{\sigma}_2(t)}_{\text{Intraday volatility}} \tag{A.14}$$

where a is a weighting parameter, whose optimal value, according to the authors is shown to be 0.17, regardless of the value of f.

Furthermore, the same study introduces a family of estimators based on the normalization of the maximum, minimum and closing values by the opening price of the considered day. We can then define:

$$u = \ln\left(\frac{P_t^{(h)}}{P_t^{(o)}} \right) \qquad d = \ln\left(\frac{P_t^{(l)}}{P_t^{(o)}} \right) \qquad c = \ln\left(\frac{P_t^{(c)}}{P_t^{(o)}} \right) \tag{A.15}$$

where u is the normalized high price, d is the normalized low price and c is the normalized closing price.

Given this derived information, we can derive Equation A.16, by starting from a general, analytic form for the estimator, and then deriving the optimal values of the coefficient by minimizing the estimation variance.

$$\hat{\sigma}_t^4 = 0.511(u-d)^2 - 0.019[c(u+d) - 2ud] - 0.383c^2 \tag{A.16}$$

The values of the coefficients are then derived by assuming that the price dynamics follows a Brownian motion and enforcing scale invariance properties and price and time symmetry conditions. For all the details concerning the proof, we refer the interested reader to (Garman and Klass, 1980).

Such estimator is also employed in (Xiong, Nichols, and Shen, 2015) using daily data, and given as input to a deep neural network, along with the prices and the volume of search engine research concerning financial market related keywords in order to provide a forecast of future volatility.

Equation A.17 is derived from Equation A.16 by eliminating the cross product terms. Even though this formulation is simplified, and easily computable, it is shown to have a comparable efficiency to the analogous more complex estimator A.1.

$$\hat{\sigma}_t^5 = 0.511(u-d)^2 - (2\ln 2 - 1)c^2 \tag{A.17}$$

Last but not least, the best estimator in terms of estimation variance efficiency is obtained by combining the overnight volatility measure with the optimal estimator described in Equation A.16.

$$\hat{\sigma}_t^6 = \underbrace{\frac{a}{f} \cdot \log\left(\frac{P_{t+1}^{(o)}}{P_t^{(c)}}\right)^2}_{\text{Nightly volatility}} + \underbrace{\frac{1-a}{1-f} \cdot \hat{\sigma}_4(t)}_{\text{Intraday volatility}} \tag{A.18}$$

A.2.0.3 *Volatility as a proxy of the fine grained intraday information*

In conclusion, if a finer data granularity is available, for instance intraday data, the realized variance of the series can be used as a proxy, as proposed by Hansen and Lunde (Hansen and Lunde, 2005).

The return over a time interval with length $\frac{1}{m}$ on day t $R_{t,i,m}$, given intraday observations $i \in \{1, \cdots, m\}$, and trading days $t \in \{1, \cdots, n\}$ is computed as:

$$R_{t,i,m} = P_{t-\frac{i-1}{m}} - P_{t-\frac{i}{m}} \tag{A.19}$$

With this available data the daily return r_t can be easily computed by aggregating the intraday observations:

$$R_t = \sum_{i=1}^{m} R_{t,i,m} \tag{A.20}$$

The realized variance for trading day t is then computed as the squared sum of the m available intraday returns:

$$RV^{(m)}(t) = \sum_{i=1}^{m} r_{t,i,m}^2 \tag{A.21}$$

The proxy is then obtained by computing the realized variance and accounting for the fact that financial markets are opened for trading during a subset of the whole trading days,

which in turn only allows to observe s values out of the m that compose the trading day. The realized variance is then scaled by a coefficient representing the daily price variance normalized by the aggregated realized variance over all the n observed trading days.

$$(\hat{\sigma}_t^{RV})^2 = \frac{\frac{1}{n}\sum_{k=1}^{n}(r_k - \hat{\mu_k})^2}{\frac{1}{n}\sum_{k=1}^{n} RV^{(s)}(k)} RV^{(s)}(t) \tag{A.22}$$

For additional details concerning the derivation of the coefficient, we refer the interested reader to (Hansen and Lunde, 2005).

Given the available OHLC data, we will opt for the proxy proposed in (Garman and Klass, 1980).

B

APPENDIX B - DFML SUPPLEMENTARY MATERIAL

B.1 FACTOR ESTIMATION AND FORECASTING ASSESSMENT

B.1.1 *Datasets - Correlation Analysis*

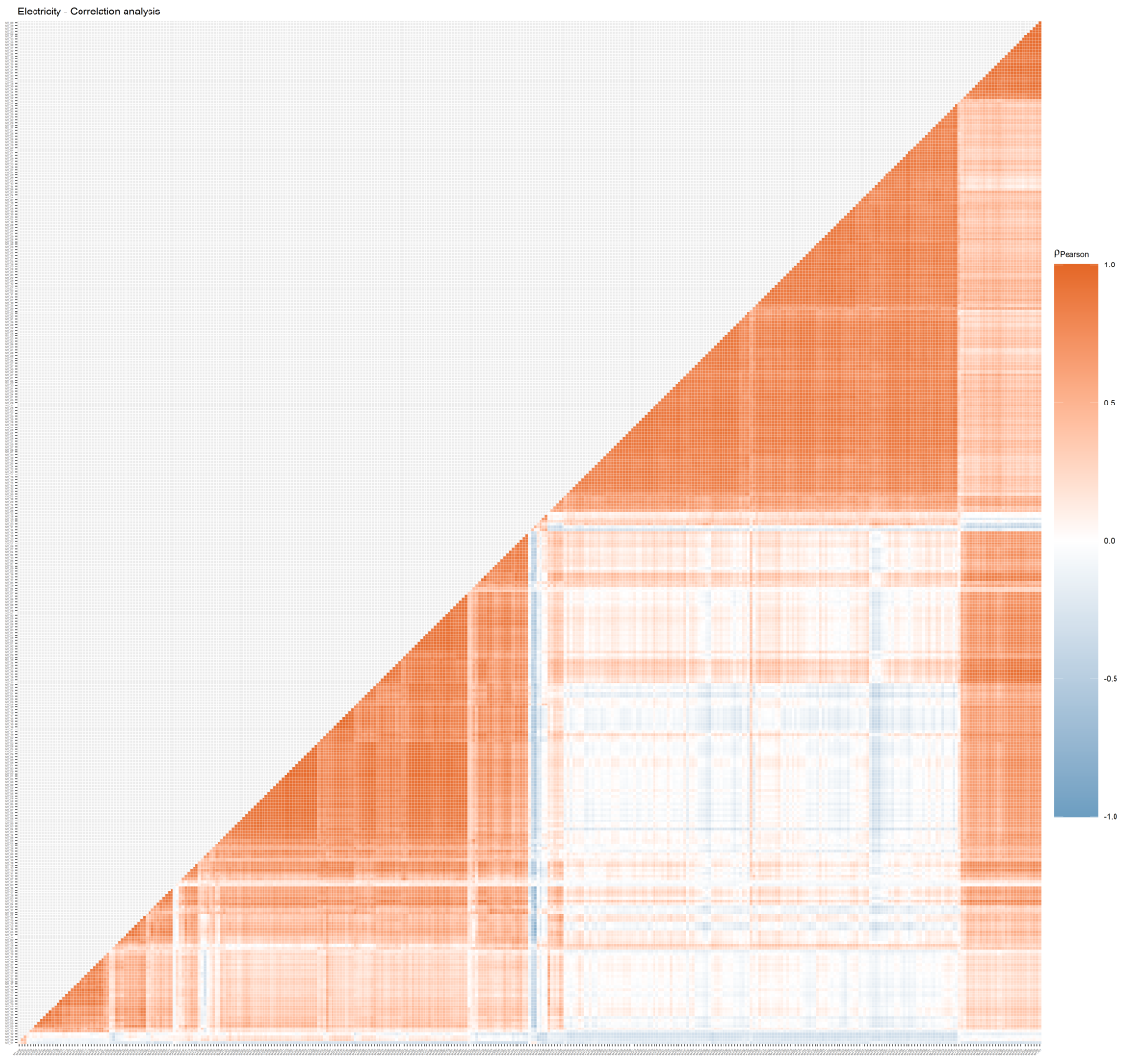


Figure B.1: **Electricity** - Correlation matrix of the underlying time series composing the dataset, using Pearson's correlation coefficient ρ

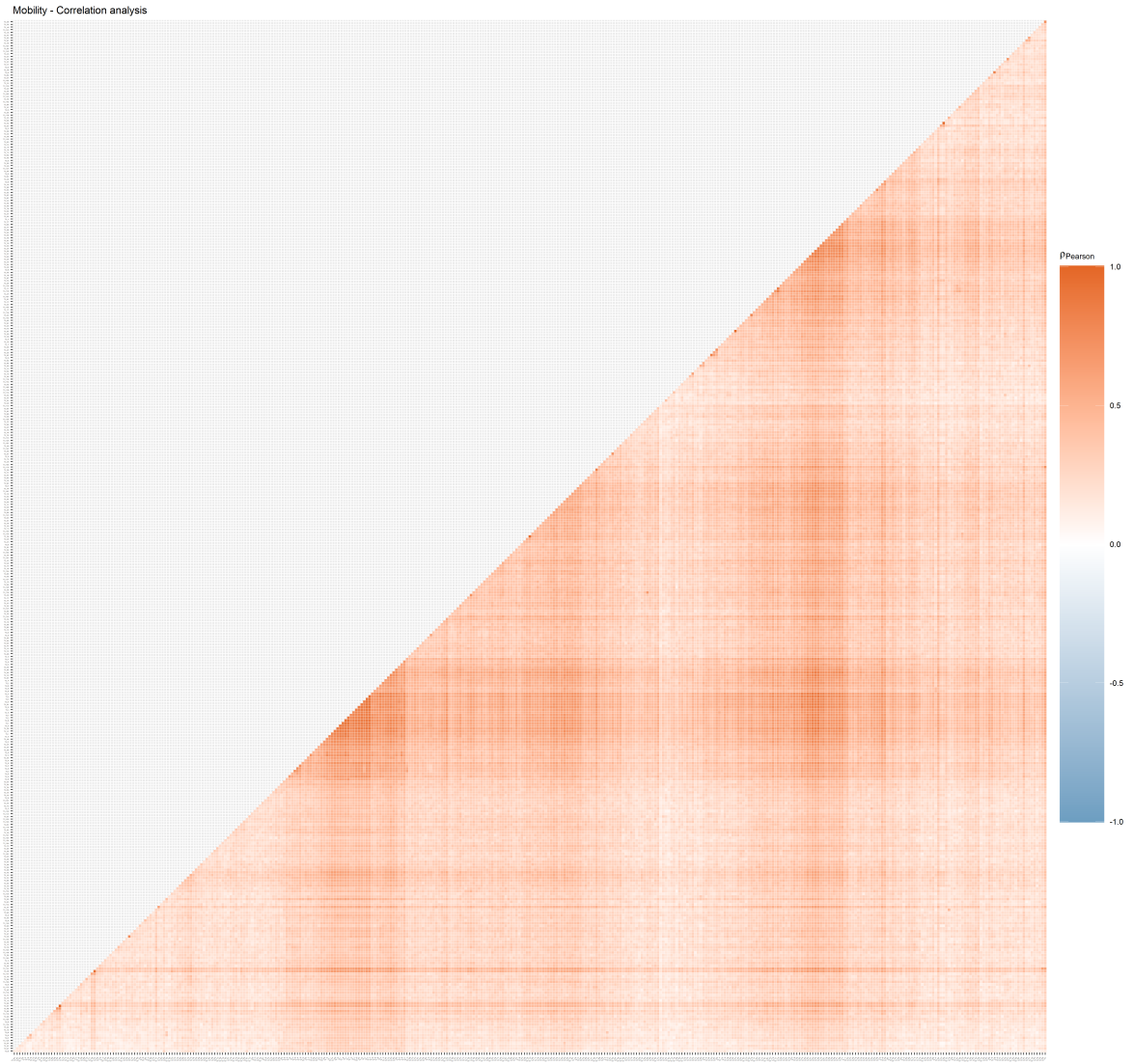


Figure B.2: **Mobility** - Correlation matrix of the underlying time series composing the dataset, using Pearson's correlation coefficient ρ

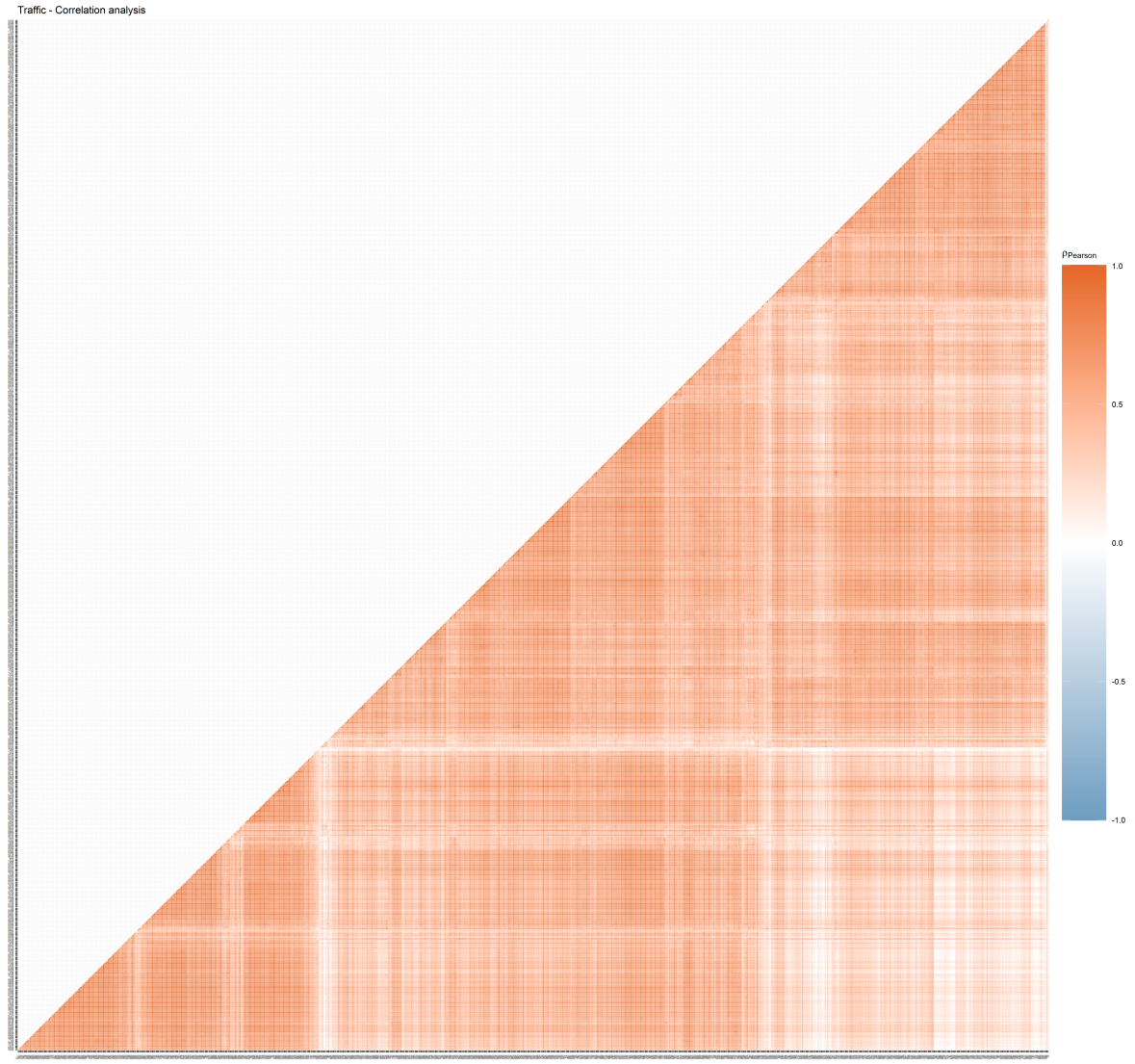


Figure B.3: **Traffic** - Correlation matrix of the underlying time series composing the dataset, using Pearson's correlation coefficient ρ

B.1.2 *Supplementary results*

B.1.2.1 *Electricity*

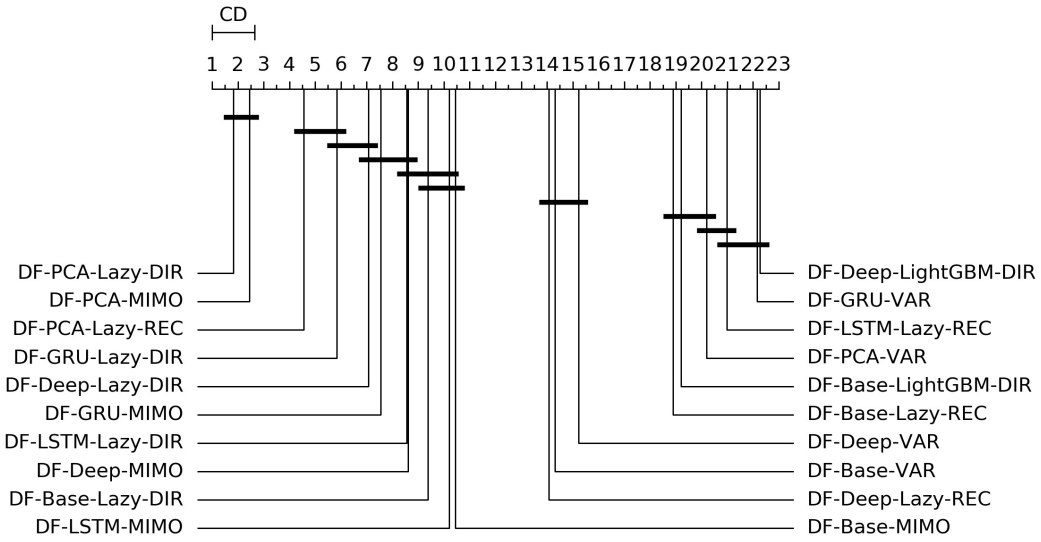


Figure B.4: **Electricity** - Graphical representation according to (Demšar, 2006) of the results of Friedman statistical test (with post-hoc Nemenyi test) comparing the NNMSE of the best 20 methods against each other, aggregated across all horizons h. The methods are ordered according to their performance from left to right (the leftmost the best), while the black bar connects methods that are not significantly different (at $p = 0.05$).

Table B.1: **Electricity** Naive Normalized MSE for $H \in \{4, 6\}$. The content of the cell c_{ij} represents the model using the jth dimensionality reduction technique with the ith forecasting method. The NNMSE for the Naive method is equal to 1. An NNMSE < 1 indicates that the proposed method outperforms the Naive method. Bold text denotes the best configuration for the given horizon H.

		H=4					**H=6**				
		PCA	LSTM	GRU	Base	Deep	PCA	LSTM	GRU	Base	Deep
DF-Stat	DF-ES	0.5418	0.7396	0.6106	0.4415	0.4424	0.6026	0.6314	0.6594	0.4772	0.4928
	DF-Theta	0.5436	0.7482	0.6094	0.4416	0.442	0.5681	0.6306	0.6593	0.4772	0.4884
	DF-Combined	0.5402	0.7443	0.6116	0.4391	0.4421	0.6024	0.6325	0.6227	0.4773	0.5236
	DF-VAR	0.4684	0.7245	0.4463	0.4149	0.4208	0.4692	0.5191	0.5378	0.4369	0.4326
DF-ML	DF-Lazy-DIR	0.3202	0.4561	0.3955	0.4318	0.3773	0.3195	0.3746	0.379	0.4018	0.3751
	DF-Lazy-REC	0.3297	0.4904	0.4526	0.439	0.3891	**0.336**	0.4418	0.432	0.4448	0.4684
	DF-MIMO	0.3498	0.4331	0.3959	0.4481	**0.3716**	0.341	0.3869	0.3823	0.4015	0.388
	DF-LightGBM-DIR	0.6249	0.5435	0.5902	0.4426	0.4776	0.6985	0.6227	0.6636	0.4936	0.5117
	DF-LightGBM-REC	0.56	0.5395	0.5575	0.4895	0.5166	0.6882	0.5866	0.6103	0.5097	0.5567
UNI-Stat	UNI-Naive	1	1	1	1	1	1	1	1	1	1
	UNI-ES	0.5881	0.5881	0.5881	0.5881	0.5881	0.623	0.623	0.623	0.623	0.623
	UNI-Theta	0.4722	0.4722	0.4722	0.4722	0.4722	0.499	0.499	0.499	0.499	0.499
	UNI-Comb	0.59	0.59	0.59	0.59	0.59	0.6266	0.6266	0.6266	0.6266	0.6266

Table B.2: **Electricity** Naive Normalized MSE for $H \in \{12, 24\}$. The content of the cell c_{ij} represents the model using the jth dimensionality reduction technique with the ith forecasting method. The NNMSE for the Naive method is equal to 1. An NNMSE < 1 indicates that the proposed method outperforms the Naive method. Bold text denotes the best configuration for the given horizon H.

		H=12					**H=24**				
		PCA	LSTM	GRU	Base	Deep	PCA	LSTM	GRU	Base	Deep
DF-Stat	DF-ES	0.6357	0.7244	0.6648	0.5075	0.5188	0.6793	2.797	0.6316	0.5959	0.5856
	DF-Theta	0.5928	0.7261	0.6648	0.5079	0.5183	0.6388	2.7969	0.6313	0.5957	0.5856
	DF-Combined	0.6371	0.747	0.6649	0.5115	0.5575	0.6825	3.3199	0.6334	0.5985	0.6162
	DF-VAR	0.4744	0.5188	0.4464	0.4496	0.4649	0.5352	2.2798	0.5358	0.5226	0.5227
DF-ML	DF-Lazy-DIR	0.2878	0.4179	0.3615	0.4339	0.4621	**0.3307**	0.4616	0.451	0.485	0.4691
	DF-Lazy-REC	0.3747	0.4957	0.569	0.4682	0.4894	0.4429	0.5979	0.5798	0.569	0.5513
	DF-MIMO	**0.304**	0.4445	0.3656	0.4384	0.4878	0.341	0.457	0.4633	0.491	0.4766
	DF-LightGBM-DIR	0.7044	0.641	0.5773	0.5217	0.5494	0.3284	0.4988	0.4403	0.4824	0.4683
	DF-LightGBM-REC	0.7427	0.5387	0.5251	0.5267	0.5503	0.842	0.5783	0.8388	0.5647	0.5836
UNI-Stat	UNI-Naive	1	1	1	1	1	1	1	1	1	1
	UNI-ES	0.6504	0.6504	0.6504	0.6504	0.6504	0.6746	0.6746	0.6746	0.6746	0.6746
	UNI-Theta	0.5295	0.5295	0.5295	0.5295	0.5295	0.5802	0.5802	0.5802	0.5802	0.5802
	UNI-Comb	0.6537	0.6537	0.6537	0.6537	0.6537	0.6783	0.6783	0.6783	0.6783	0.6783

B.1.2.2 *Traffic*

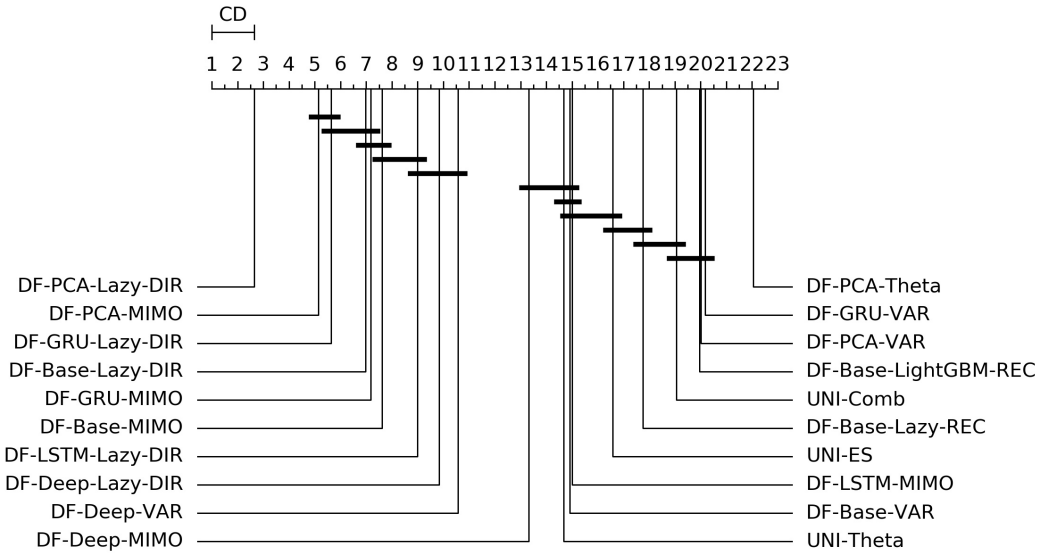


Figure B.5: **Traffic** - Graphical representation according to (Demšar, 2006) of the results of Friedman statistical test (with post-hoc Nemenyi test) comparing the NNMSE of the best 20 methods against each other, aggregated across all horizons h. The methods are ordered according to their performance from left to right (the leftmost the best), while the black bar connects methods that are not significantly different (at $p = 0.05$).

Table B.3: **Traffic** Naive Normalized MSE for $H \in \{4, 6\}$. The content of the cell c_{ij} represents the model using the jth dimensionality reduction technique with the ith forecasting method. The NNMSE for the Naive method is equal to 1. An NNMSE < 1 indicates that the proposed method outperforms the Naive method. Bold text denotes the best configuration for the given horizon H.

		H=4					**H=6**				
		PCA	LSTM	GRU	Base	Deep	PCA	LSTM	GRU	Base	Deep
DF-Stat	DF-ES	0.5637	0.5819	0.6036	0.4823	0.4788	0.5726	0.4589	0.5386	0.4858	0.4731
	DF-Theta	0.5236	0.576	0.6037	0.4823	0.4788	0.5267	0.4583	0.542	0.4857	0.4731
	DF-Combined	0.5639	0.5961	0.5731	0.4826	0.5026	0.5729	0.4594	0.5404	0.4835	0.4754
	DF-VAR	0.5127	0.6259	0.5115	0.4667	0.4497	0.4718	0.4411	0.4451	0.4538	0.4433
DF-ML	DF-Lazy-DIR	**0.3502**	0.437	0.4234	0.4303	0.4659	0.3599	**0.3455**	0.3719	0.4631	0.3711
	DF-Lazy-REC	0.4746	0.399	0.4594	0.445	0.4693	0.4532	0.4775	0.5467	0.6137	0.3735
	DF-MIMO	0.3793	0.4264	0.4421	0.436	0.5122	0.3949	0.3378	0.3823	0.4588	0.37
	DF-LightGBM-DIR	0.7315	0.6496	0.5325	0.4996	0.6158	0.5644	0.4804	0.4823	0.4648	0.559
	DF-LightGBM-REC	0.6973	0.7253	0.6136	0.4998	0.6084	0.6133	0.5077	0.5237	0.5113	0.5183
UNI-Stat	UNI-Naive	1	1	1	1	1	1	1	1	1	1
	UNI-ES	0.5157	0.5157	0.5157	0.5157	0.5157	0.5062	0.5062	0.5062	0.5062	0.5062
	UNI-Theta	0.4731	0.4731	0.4731	0.4731	0.4731	0.4645	0.4645	0.4645	0.4645	0.4645
	UNI-Comb	0.5226	0.5226	0.5226	0.5226	0.5226	0.5129	0.5129	0.5129	0.5129	0.5129

Table B.4: **Traffic** Naive Normalized MSE for $H \in \{12, 24\}$. The content of the cell c_{ij} represents the model using the jth dimensionality reduction technique with the ith forecasting method. The NNMSE for the Naive method is equal to 1. An NNMSE < 1 indicates that the proposed method outperforms the Naive method. Bold text denotes the best configuration for the given horizon H.

		H=12					**H=24**				
		PCA	LSTM	GRU	Base	Deep	PCA	LSTM	GRU	Base	Deep
DF-Stat	DF-ES	0.5786	0.6448	0.583	0.5128	0.5033	0.6083	0.5209	0.5728	0.568	0.5385
	DF-Theta	0.5105	0.6089	0.5829	0.5128	0.5037	0.5432	0.5243	0.5728	0.5679	0.539
	DF-Combined	0.5792	0.6678	0.583	0.5187	0.5196	0.6097	0.5228	0.5692	0.5881	0.5578
	DF-VAR	0.4583	0.4797	0.4958	0.454	0.4453	0.4966	0.5102	0.4993	0.4903	0.4885
DF-ML	DF-Lazy-DIR	**0.3823**	0.4649	0.4177	0.4307	0.4325	**0.4123**	0.4799	0.489	0.4568	0.4557
	DF-Lazy-REC	0.4723	0.716	0.5079	0.5039	0.5264	0.6572	41.4895	2.3971	0.5305	0.6124
	DF-MIMO	0.4211	0.5064	0.4268	0.4469	0.4401	0.4769	0.5116	0.4933	0.467	0.4529
	DF-LightGBM-DIR	0.6798	0.6337	0.5732	0.5711	0.5125	0.4182	0.4606	0.4538	0.4577	0.4698
	DF-LightGBM-REC	0.8134	0.5839	0.5478	0.4921	0.4648	0.8062	0.7161	0.7522	0.5216	0.5378
UNI-Stat	UNI-Naive	1	1	1	1	1	1	1	1	1	1
	UNI-ES	0.5154	0.5154	0.5154	0.5154	0.5154	0.5423	0.5423	0.5423	0.5423	0.5423
	UNI-Theta	0.4801	0.4801	0.4801	0.4801	0.4801	0.5167	0.5167	0.5167	0.5167	0.5167
	UNI-Comb	0.5206	0.5206	0.5206	0.5206	0.5206	0.5446	0.5446	0.5446	0.5446	0.5446

B.2 ITERATIVE FACTOR ESTIMATION AND AUTOMATIC HYPERPARAMETER SELECTION

B.2.1 *Batch versus iterative PCA*

Table B.5: Synthetic multivariate time series: NMSE (averaged over all the continuation sets) of the different forecasting methods.Comparison between batch and iterative PCA. The superscript $^+$ denotes a significant improvement (pv=0.05) while using the iterative PCA calculation instead of the batch one. The bold notation is used to identify the best method for each horizon.

Rolling		$DFML_{PC}$						DFM	
		Direct		Iterated		MIMO			
n	H	Batch	Online	Batch	Online	Batch	Online	Batch	Online
20	2	**0.245**	0.238	0.258	0.259	0.263	0.249	0.288	0.272
20	5	**0.265**	0.268	0.325	0.325	0.287	0.298	0.343	0.314^+
20	10	0.280	**0.279**	0.356	0.340	0.303	0.302	0.362	0.341
20	15	0.302	**0.301**	0.381	0.376	0.335	0.325	0.391	0.360^+
20	20	0.306	**0.302**	0.385	0.380	0.339	0.326	0.436	0.397^+
20	50	**0.340**	0.344	0.414	0.408	0.376	0.378	0.666	0.567^+
50	2	0.225	$\mathbf{0.133}^+$	0.226	0.137^+	0.228	0.137^+	0.233	0.140^+
50	5	0.221	$\mathbf{0.129}^+$	0.228	0.141^+	0.224	0.133^+	0.235	0.147^+
50	10	0.227	$\mathbf{0.134}^+$	0.232	0.150^+	0.231	0.138^+	0.233	0.151^+
50	15	0.232	$\mathbf{0.140}^+$	0.240	0.156^+	0.235	0.144^+	0.241	0.164^+
50	20	0.237	$\mathbf{0.148}^+$	0.245	0.163^+	0.241	0.152^+	0.244	0.177^+
50	50	0.264	$\mathbf{0.179}^+$	0.270	0.185^+	0.270	0.182^+	0.286	0.228^+
100	2	0.314	$\mathbf{0.180}^+$	0.315	0.183^+	0.315	$\mathbf{0.180}^+$	0.320	0.200^+
100	5	0.320	$\mathbf{0.193}^+$	0.329	0.207^+	0.324	$\mathbf{0.193}^+$	0.337	0.243^+
100	10	0.322	0.213^+	0.332	0.356^+	0.324	$\mathbf{0.210}^+$	0.352	0.289^+
100	15	0.329	0.225^+	0.341	99.729	0.332	$\mathbf{0.224}^+$	0.367	0.321^+
100	20	0.331	0.240^+	0.343	134.651	0.334	$\mathbf{0.239}^+$	0.380	0.341^+
100	50	0.359	0.332^+	0.369	862.325	0.363	$\mathbf{0.307}^+$	0.477	0.443^+
200	2	0.254	$\mathbf{0.121}^+$	0.255	0.122^+	0.255	0.122^+	0.257	0.124^+
200	5	0.252	0.120^+	0.256	0.127^+	0.253	$\mathbf{0.118}^+$	0.255	0.125^+
200	10	0.255	0.128^+	0.259	1.819	0.256	$\mathbf{0.123}^+$	0.258	0.138^+
200	15	0.258	0.134^+	0.263	5.787	0.260	$\mathbf{0.130}^+$	0.261	0.154^+
200	20	0.261	0.151^+	0.266	12.806	0.263	$\mathbf{0.138}^+$	0.265	0.170^+
200	50	0.279	0.180^+	0.284	65.992	0.280	$\mathbf{0.176}^+$	0.289	0.246^+
400	2	0.278	0.119^+	0.278	0.119^+	0.278	$\mathbf{0.117}^+$	0.278	0.119^+
400	5	0.277	0.115^+	0.280	0.117^+	0.277	$\mathbf{0.113}^+$	0.278	0.120^+
400	10	0.281	0.129^+	0.286	0.145^+	0.282	$\mathbf{0.121}^+$	0.283	0.138^+
400	15	0.285	0.146^+	0.290	0.149^+	0.285	$\mathbf{0.132}^+$	0.286	0.155^+
400	20	0.289	0.170^+	0.295	7.144	0.289	$\mathbf{0.143}^+$	0.291	0.170^+
400	50	0.311	0.234^+	0.319	283.169	0.311	$\mathbf{0.204}^+$	0.325	0.243^+
1000	2	0.287	0.111^+	0.287	0.111^+	0.287	$\mathbf{0.109}^+$	0.287	0.111^+
1000	5	0.287	0.107^+	0.288	0.107^+	0.287	$\mathbf{0.105}^+$	0.287	0.106^+
1000	10	0.291	0.119^+	0.293	0.121^+	0.291	$\mathbf{0.113}^+$	0.290	0.114^+
1000	15	0.294	0.138^+	0.297	0.158^+	0.294	$\mathbf{0.121}^+$	0.294	0.122^+
1000	20	0.298	0.173^+	0.300	0.294	0.298	$\mathbf{0.131}^+$	0.297	0.132^+
1000	50	0.319	0.440^+	0.322	50.946	0.319	0.188^+	0.318	$\mathbf{0.180}^+$

Table B.6: Earth Surface Temperature series: NMSE (averaged over all the continuation sets) of the different forecasting methods.Comparison between batch and iterative PCA. The superscript $^+$ denotes a significant improvement (pv=0.05) while using the iterative PCA calculation instead of the batch one. The bold notation is used to identify the best method for each horizon.

Rolling		$DFML_{PC}$						DFM	
		Direct		Iter		MIMO			
n	H	Batch	Online	Batch	Online	Batch	Online	Batch	Online
100	2	**0.106**	**0.106**	0.110	0.107	0.110	0.109	0.109	**0.106**
100	5	**0.118**	**0.118**	0.144	0.206	0.125	0.128	0.140	0.137
100	10	**0.118**	**0.118**	0.178	0.173	0.127	0.123	0.151	0.148
100	15	0.114	**0.112**	0.175	0.172	0.120	0.120	0.165	0.153^+
100	20	0.112	**0.111**	0.170	0.168	0.118	0.120	0.182	0.166^+
100	50	**0.111**	**0.111**	0.173	0.173	0.117	0.112^+	0.307	0.282^+
200	2	0.195	0.195	0.199	0.197	0.198	0.195	**0.124**	0.135^+
200	5	0.274	0.266	0.345	0.337	0.281	0.269	**0.176**	0.184
200	10	0.259	0.253	0.388	0.369^+	0.278	0.268	0.195	**0.191**
200	15	0.242	0.237	0.399	0.388^+	0.254	0.244	0.211	**0.206**
200	20	0.269	0.262	0.412	99.062	0.270	0.269	0.226	**0.225**
200	50	0.253	**0.245**	0.434	43.889	0.266	0.256	0.290	0.285^+

Table B.7: Volatility time series: NMSE (averaged over all the continuation sets) of the different forecasting methods. Comparison between batch and iterative PCA. The superscript $^+$ denotes a significant improvement (pv=0.05) while using the iterative PCA calculation instead of the batch one. The bold notation is used to identify the best method for each horizon.

Rolling		$DFML_{PC}$						DFM	
		Direct		Iter		MIMO			
Ind	H	Batch	Online	Batch	Online	Batch	Online	Batch	Online
σ_0	2	**0.405**	0.414	0.408	0.413	0.423	0.437^+	0.406	0.410
σ_0	5	**0.421**	0.430	0.432	0.429	0.454	0.452	0.424	0.428
σ_0	10	**0.418**	0.422	0.427	0.427	0.453	0.456	0.420	0.422
σ_0	15	**0.418**	0.422	0.437	0.426	0.454	0.456	0.420	0.422
σ_0	20	**0.417**	0.418	0.456	0.447	0.458	0.452	0.419	0.419
σ_0	50	**0.422**	0.426	0.510	0.460^+	0.482	0.490	0.424	0.424
σ_1	2	0.270	0.272	0.270	0.273	0.299	0.296	**0.263**	0.264
σ_1	5	0.385	0.392	0.402	0.397	0.426	0.421	**0.380**	0.382
σ_1	10	**0.364**	0.368	0.401	0.406	0.400	0.401	0.370	0.372
σ_1	15	**0.368**	0.371	0.420	0.414	0.419	0.418	0.374	0.374
σ_1	20	**0.348**	0.354^+	0.420	0.436	0.401	0.387	0.367	0.367
σ_1	50	**0.367**	0.369	0.466	0.481	0.425	0.423	0.368	0.368
σ_2	2	0.315	0.312	0.311	0.308	0.326	0.322	**0.305**	**0.305**
σ_2	5	0.314	0.318	0.331	0.335	0.336	0.336	0.315	**0.311**
σ_2	10	**0.315**	0.317	0.350	0.353	0.335	0.334	0.330	0.325^+
σ_2	15	**0.308**	0.314	0.378	0.382	0.340	0.335	0.339	0.333^+
σ_2	20	**0.320**	0.318	0.390	0.405	0.352	0.347	0.346	0.339^+
σ_2	50	**0.329**	0.332	0.513	0.493	0.404	0.407	0.365	0.361^+
σ_3	2	0.279	0.265	0.280	0.265	0.284	0.279	0.273	**0.261**
σ_3	5	0.338	0.350	0.343	0.345	0.358	0.348^+	0.332	**0.330**
σ_3	10	**0.329**	0.335	0.342	0.344	0.362	0.367	0.341	0.333^+
σ_3	15	**0.325**	0.332	0.354	0.348	0.362	0.360	0.350	0.347
σ_3	20	**0.324**	0.332	0.368	0.376	0.381	0.377	0.355	0.349^+
σ_3	50	**0.343**	0.344	0.365	0.399	0.407	0.398	0.372	0.370^+
σ_4	2	0.304	0.308	0.295	0.310	0.348	0.341	0.286	**0.284**
σ_4	5	0.317	0.322	0.320	0.322	0.338	0.339	**0.315**	0.317
σ_4	10	**0.308**	**0.308**	0.318	0.320	0.334	0.324^+	0.322	0.321
σ_4	15	0.318	**0.317**	0.339	0.340	0.380	0.364	0.331	0.332
σ_4	20	**0.307**	**0.307**	0.329	0.330	0.327	0.325	0.338	0.335
σ_4	50	0.339	0.337	**0.336**	0.339	0.444	0.444	0.359	0.357^+
σ_5	2	0.301	0.306	0.296	0.312	0.328	0.324	0.286	**0.284**
σ_5	5	0.317	0.321	0.323	0.325	0.336	0.340	**0.315**	0.316
σ_5	10	**0.308**	0.309	0.320	0.326	0.344	0.330	0.322	0.320
σ_5	15	0.318	**0.314**	0.327	0.329	0.361	0.352	0.332	0.332
σ_5	20	0.311	**0.308**	0.331	0.335	0.333	0.328	0.339	0.335^+
σ_5	50	**0.336**	0.339	0.350	0.342^+	0.446	0.440	0.360	0.357^+
σ_6	2	0.299	0.308	**0.298**	0.305	0.317	0.312	**0.298**	0.300
σ_6	5	0.314	0.317	0.318	0.315	0.327	0.330	**0.311**	0.317
σ_6	10	**0.305**	**0.305**	0.317	0.319	0.333	0.337	0.317	0.317
σ_6	15	**0.304**	**0.304**	0.326	0.319	0.339	0.344	0.323	0.323
σ_6	20	**0.306**	0.307	0.335	0.324	0.339	0.324	0.330	0.330
σ_6	50	0.328	0.331	0.332	**0.325**	0.430	0.419	0.351	0.350

B.2.2 *Manual versus automatic hyperparameter search strategy*

Table B.8: Synthetic multivariate time series: NMSE (averaged over all the continuation sets) of the different forecasting methods

n	H	DFM	$DFML_{PC}$	$DFML'_{PC}$	$DFML_{A}$	$DFML'_{A}$	RNN	DSE	PLS	UNI	VAR	SSA	NAIVE
20	5	0.815	0.813	**0.783**	0.834	0.815	0.793	0.872	0.891	1.012	0.819	0.913	1.913
20	10	0.863	0.851	**0.829**	0.872	0.854	**0.824**	0.925	0.915	1.058	0.62	0.925	1.925
20	20	0.914	0.895	0.875	0.911	0.898	**0.862**	0.957	0.929	1.078	0.909	0.95	1.977
50	5	0.818	0.819	**0.782**	0.842	0.809	0.833	0.890	0.909	1.004	0.821	0.921	1.909
50	10	0.851	0.846	**0.816**	0.868	0.839	0.863	0.906	0.924	1.043	0.850	0.922	1.929
50	20	0.885	0.875	**0.854**	0.895	0.875	0.893	0.923	0.930	1.069	0.881	0.929	1.961
100	5	0.852	0.857	**0.824**	0.909	0.846	0.916	0.922	0.957	1.026	0.913	0.911	1.901
100	10	0.872	0.876	**0.853**	0.924	0.873	0.934	0.958	0.963	1.062	0.908	0.914	1.919
100	20	0.852	0.840	**0.809**	0.883	0.827	0.901	1.028	0.944	1.032	0.861	0.919	1.972
200	5	0.882	0.881	**0.854**	0.942	-	0.956	-	-	1.022	-	-	1.907
200	10	0.895	0.893	**0.872**	0.952	-	0.971	-	-	1.060	-	-	1.928
200	20	0.909	0.908	**0.892**	0.958	-	0.967	-	-	1.086	-	-	1.972
400	5	0.898	0.902	**0.892**	0.984	-	0.985	-	-	-	-	-	1.888
400	10	0.906	0.908	**0.903**	0.991	-	0.994	-	-	-	-	-	1.907
400	20	**0.915**	**0.916**	**0.915**	0.993	-	0.998	-	-	-	-	-	1.949
1000	5	**0.915**	0.919	0.925	1.062	-	-	-	-	-	-	-	1.893
1000	10	**0.919**	0.922	0.930	1.061	-	-	-	-	-	-	-	1.915
1000	20	**0.924**	0.926	0.934	1.058	-	-	-	-	-	-	-	1.958

Table B.9: Earth Surface Temperature series: NMSE (averaged over all the continuation sets) of the different forecasting methods

n	H	DFM	$DFML_{PC}$	$DFML'_{PC}$	$DFML_{A}$	$DFML'_{A}$	RNN	PLS	UNI	NAIVE
100	2	**0.099**	**0.111**	**0.099**	0.566	0.594	**0.099**	0.227	0.265	0.692
100	5	0.13	0.151	**0.092**	1.144	0.394	0.102	0.664	0.271	1.981
100	10	0.142	0.164	**0.093**	1.709	0.6	0.113	0.673	0.295	2.247
100	20	0.165	0.173	**0.089**	1.721	0.873	0.11	0.653	0.255	2.165
100	50	0.288	0.187	**0.091**	1.621	0.838	0.111	0.612	0.259	1.894
200	2	**0.124**	0.198	**0.14**	0.7	0.483	0.188	0.49	0.33	0.703
200	5	0.155	0.292	**0.135**	1.131	0.596	0.183	0.834	0.328	1.852
200	10	0.179	0.352	**0.135**.	1.329	0.613	0.202	0.854	0.327	2.125
200	20	0.206	0.381	**0.157**	1.472	0.645	0.229	0.837	0.34	2.038
200	50	0.266	0.405	**0.169**	1.721	0.764	0.242	0.807	0.344	1.801

Table B.10: Volatility time series: NMSE (averaged over all the continuation sets) of the different forecasting methods

Ind	H	DFM	$DFML_{PC}$	$DFML'_{PC}$	$DFML_{A}$	$DFML'_{A}$	RNN	PLS	UNI	NAIVE
σ_0	2	0.417	0.439	**0.409**	0.462	0.465	0.463	0.416	0.564	0.774
σ_0	5	0.424	0.439	**0.413**	0.463	0.468	0.442	0.421	0.563	0.838
σ_0	10	0.426	0.435	0.454	0.461	0.462	0.439	**0.419**	0.561	0.871
σ_0	20	0.434	0.445	**0.425**	0.465	0.474	0.446	**0.423**	0.573	0.753
σ_0	50	**0.433**	0.449	**0.431**	0.465	0.472	0.443	0.438	0.573	0.759
σ_1	2	**0.362**	0.391	**0.358**	0.422	0.444	0.382	0.369	0.484	0.617
σ_1	5	0.363	0.373	**0.354**	0.416	0.425	0.382	0.364	0.492	0.694
σ_1	10	0.37	0.381	**0.354**	0.414	0.425	0.379	0.361	0.494	0.685
σ_1	20	**0.384**	0.397	**0.383**	0.423	0.433	**0.385**	0.39	0.509	0.718
σ_1	50	0.389	0.411	**0.383**	0.430	0.454	0.390	0.387	0.518	0.647
σ_2	2	**0.305**	0.318	**0.309**	0.384	0.406	0.333	0.321	0.41	0.518
σ_2	5	0.31	0.317	**0.304**	0.380	0.394	0.349	0.316	0.404	0.553
σ_2	10	0.324	0.323	**0.3**	0.376	0.389	0.347	0.316	0.407	0.522
σ_2	20	0.35	**0.343**	**0.354**	0.385	0.416	0.360	**0.338**	0.438	0.534
σ_2	50	0.375	0.383	**0.328**	0.402	0.421	0.399	**0.326**	0.493	0.543
σ_3	2	**0.322**	0.335	0.332	0.426	0.441	0.356	0.341	0.407	0.507
σ_3	5	**0.325**	0.334	**0.323**	0.419	0.433	0.363	0.336	0.416	0.587
σ_3	10	0.338	0.345	**0.328**	0.420	0.431	0.364	0.337	0.422	0.587
σ_3	20	0.364	**0.367**	**0.354**	0.433	0.445	0.379	0.36	0.453	0.59
σ_3	50	0.386	0.388	**0.344**	0.436	0.460	0.403	**0.345**	0.506	0.561
σ_4	2	**0.3**	0.312	0.314	0.389	0.412	0.332	0.318	0.392	0.523
σ_4	5	**0.304**	**0.319**	**0.309**	0.388	0.397	0.326	0.316	0.396	0.567
σ_4	10	0.319	0.331	**0.302**	0.385	0.406	0.340	0.315	0.4	0.511
σ_4	20	0.344	**0.34**	**0.328**	0.388	0.426	0.359	**0.329**	0.427	0.494
σ_4	50	0.373	0.386	0.329	0.405	0.402	0.388	**0.322**	0.504	0.536
σ_5	2	**0.299**	0.311	0.312	0.389	0.412	0.329	0.317	0.391	0.521
σ_5	5	**0.304**	**0.317**	**0.304**	0.387	0.400	0.341	0.316	0.394	0.564
σ_5	10	0.319	0.327	**0.301**	0.387	0.402	0.336	0.315	0.398	0.51
σ_5	20	0.345	0.341	**0.309**	0.389	0.430	0.352	0.329	0.426	0.495
σ_5	50	0.373	0.383	0.328	0.395	0.410	0.405	**0.322**	0.504	0.535
σ_6	2	**0.297**	0.312	0.312	0.385	0.428	0.322	0.325	0.385	0.506
σ_6	5	**0.299**	0.309	**0.302**	0.380	0.415	0.334	0.314	0.39	0.554
σ_6	10	0.312	0.32	**0.296**	0.381	0.423	0.356	0.313	0.4	0.507
σ_6	20	0.336	0.34	**0.319**	0.386	0.428	0.351	0.327	0.424	0.491
σ_6	50	0.365	0.382	0.324	0.401	0.433	0.401	**0.319**	0.494	0.528

C

APPENDIX C - SMURF-ES SUPPLEMENTARY MATERIAL

This document provides the experimental results for two different case studies concerning Australian and Italian wind farms, respectively. Particularly, the figures show all analyzed wind power forecasting models. The figures are grouped by the corresponding type of analysis:

- Normalized MSE boxplot
  - Australian Case Study
  - Italian Case Study
- Error Mean-Variance Analysis
  - Australian Case Study
  - Italian Case Study
- Error Tail Analysis
  - Australian Case Study
  - Italian Case Study
- Computational Time Analysis
  - Australian Case Study
  - Italian Case Study

All details are described in the manuscript.

C.1 NORMALIZED MSE BOXPLOT

C.1.1 *Australian Case Study*

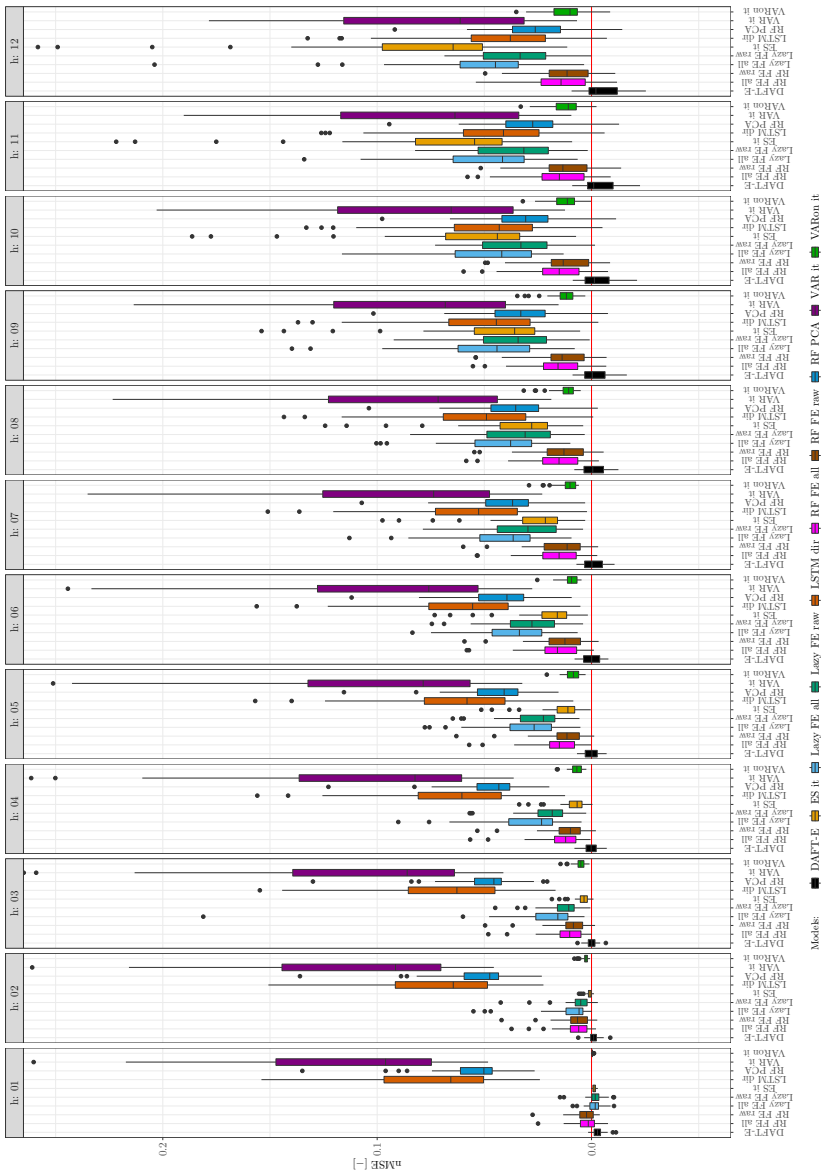


Figure C.1: Visualization of the nMSE across the forecasting horizon for the Australian case study. An nMSE < 0 indicates that the corresponding model is outperforming the Naive model.

C.1.2 *Italian Case Study*

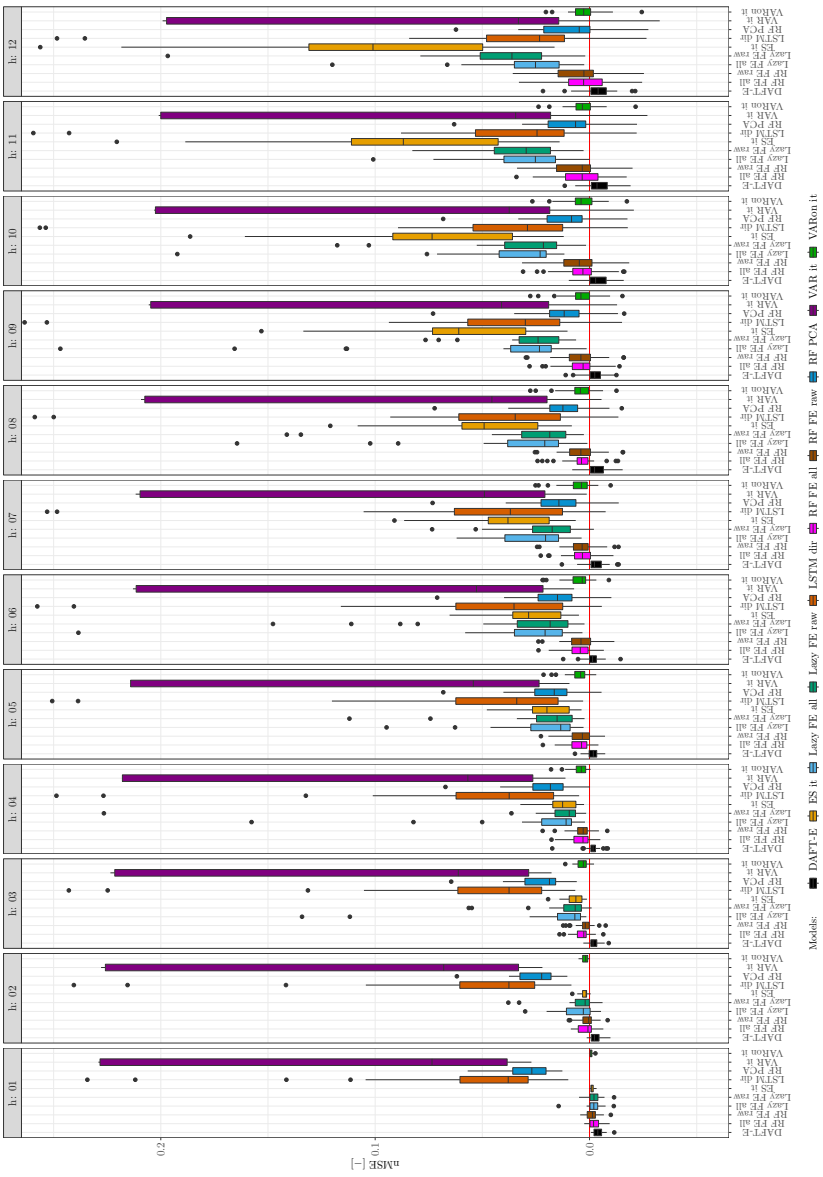


Figure C.2: Visualization of the nMSE across the forecasting horizon for the Australian case study. An nMSE < 0 indicates that the corresponding model is outperforming the Naive model.

C.2 BI-VARIATE PLOT

C.2.1 *Australian Case Study*

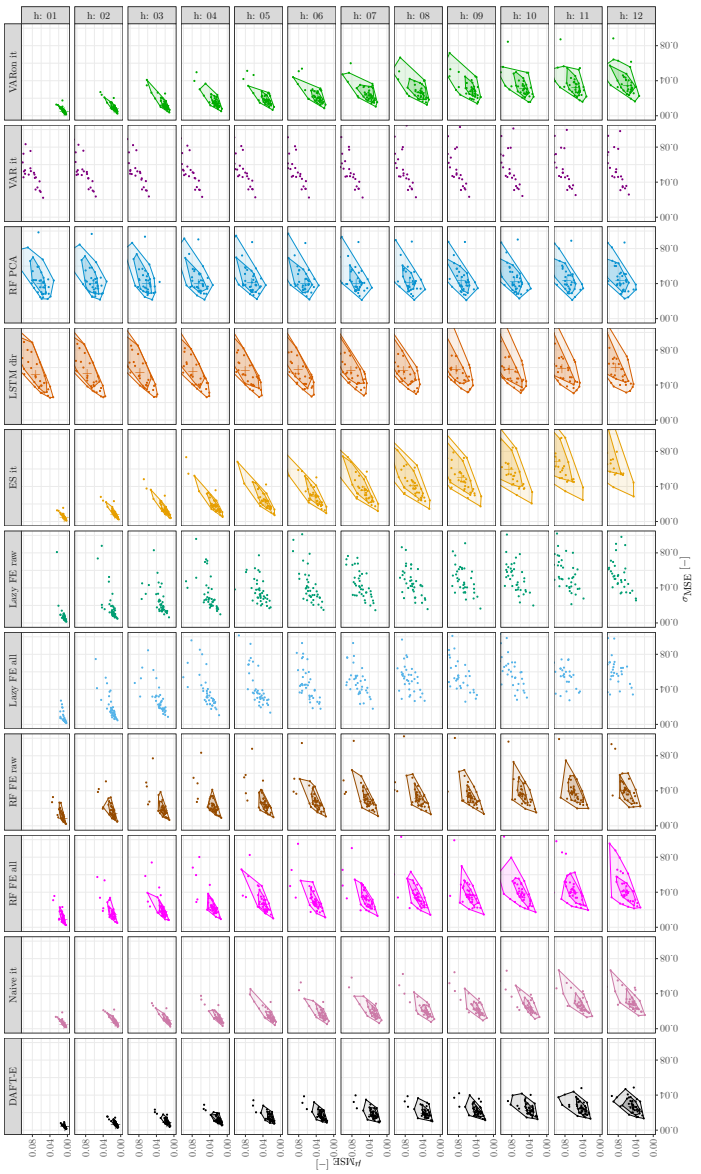


Figure C.3: Visualization of the bagplot across the forecasting horizon h for the proposed model (DAFT-E), the internal algorithms of DAFT-E (Lazy FS all, RF FS all, RF FS raw, Naive it), and the the benchmark models showing the best performance - Australian case study. A smaller bagplot area indicates a reduced variability in the the corresponding method's predictions.

C.2.2 *Italian Case Study*

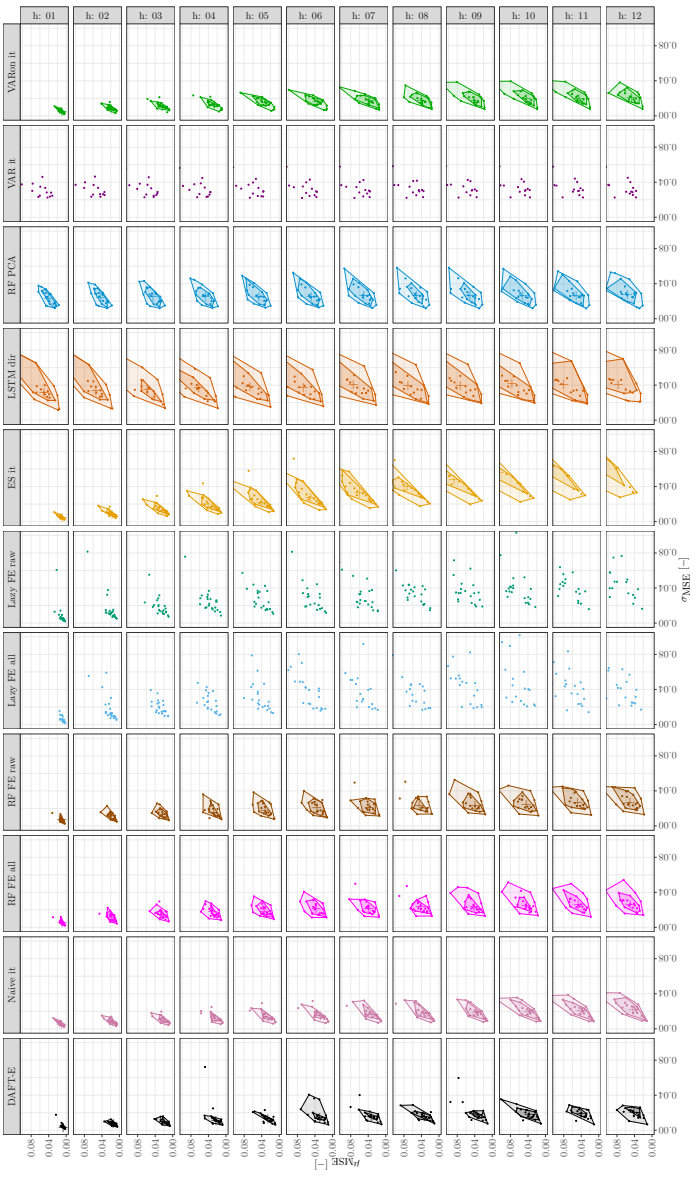


Figure C.4: Visualization of the bagplot across the forecasting horizon h for the proposed model (DAFT-E), and the the benchmark models showing the best performance - Italian case study. A smaller bagplot area indicates a reduced variability in the the corresponding method's predictions.

C.3 ERROR TAIL ANALYSIS

C.3.1 *Australian Case Study*

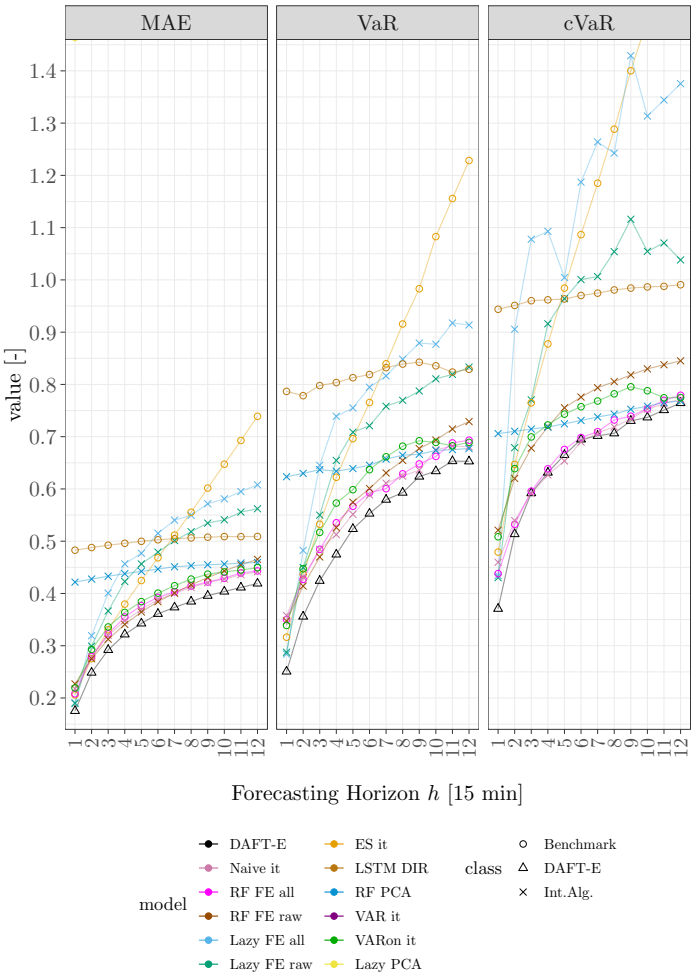


Figure C.5: Visualization of the expected value (mean), VaR, and cVaR across the forecasting horizon for all models - Australian case study. The lower the metric value is, the smaller is the risk (in terms of absolute forecasting error) that we are going to expect using the corresponding method.

C.3.2 *Italian Case Study*

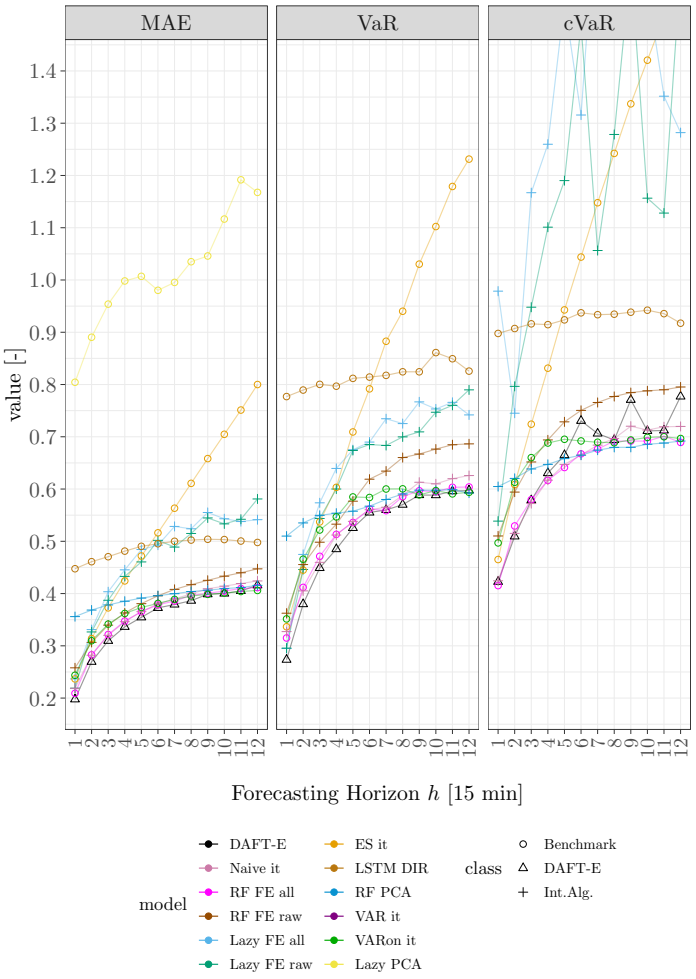


Figure C.6: Visualization of the expected value (mean), VaR, and cVaR across the forecasting horizon for all models - Italian case study. The lower the metric value is, the smaller is the risk (in terms of absolute forecasting error) that we are going to expect using the corresponding method.

C.4 COMPUTATIONAL ANALYSIS

C.4.1 *Australian Case Study*

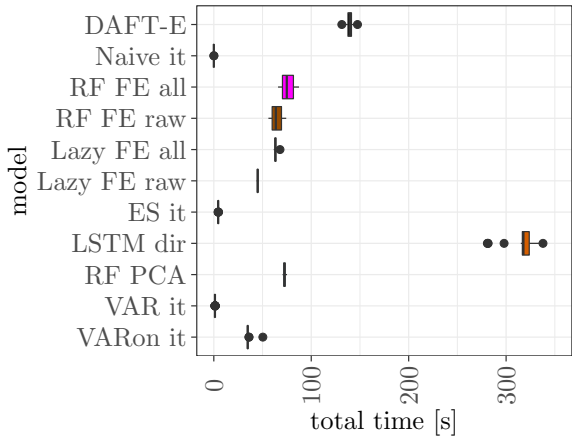


Figure C.7: Spread of the computational time across K trials, Australian Case Study

C.4.2 *Italian Case Study*

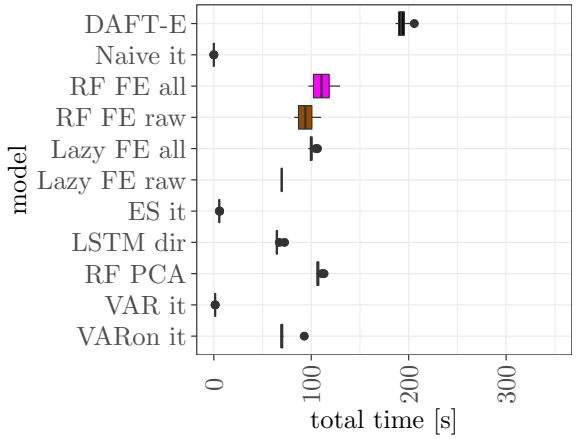


Figure C.8: Spread of the computational time across K trials, Italian Case Study

BIBLIOGRAPHY

Aggarwal, Charu C et al. (2018). "Neural networks and deep learning." In: *Springer* 10, pp. 978–3 (Cited on page 13).

Aha, DW (1997). "Special issue on lazy learning." In: *Artificial Intelligence Review* 11, pp. 7–10 (Cited on page 44).

Albadi, M.H. and E.F. El-Saadany (2010). "Overview of wind power intermittency impacts on power systems." In: *Electric Power Systems Research* 80.6, pp. 627 –632. ISSN: 0378-7796. DOI: https://doi.org/10.1016/j.epsr.2009.10.035. URL: http://www.sciencedirect.com/science/article/pii/S0378779609002764 (Cited on page 119).

Allen, David M (1974). "The relationship between variable selection and data agumentation and a method for prediction." In: *technometrics* 16.1, pp. 125–127 (Cited on page 45).

Altman, Naomi S (1992). "An introduction to kernel and nearest-neighbor nonparametric regression." In: *The American Statistician* 46.3, pp. 175–185 (Cited on page 44).

Andrecut, Mircea (2009). "Parallel GPU implementation of iterative PCA algorithms." In: *Journal of Computational Biology* 16.11, pp. 1593–1599 (Cited on page 57).

Arora, Raman, Andrew Cotter, Karen Livescu, and Nathan Srebro (2012). "Stochastic optimization for PCA and PLS." In: *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*. IEEE, pp. 861–868 (Cited on pages 56, 57).

Assimakopoulos, Vassilis and Konstantinos Nikolopoulos (2000). "The theta model: a decomposition approach to forecasting." In: *International journal of forecasting* 16.4, pp. 521–530 (Cited on pages 40, 70).

Athanasopoulos, George, Rob J Hyndman, Nikolaos Kourentzes, and Fotios Petropoulos (2017). "Forecasting with temporal hierarchies." In: *European Journal of Operational Research* 262.1, pp. 60–74 (Cited on page 68).

Atkeson, Christopher G, Andrew W Moore, and Stefan Schaal (1997). "Locally weighted learning for control." In: *Lazy learning*. Springer, pp. 75–113 (Cited on page 44).

Bai, Shaojie, J. Zico Kolter, and Vladlen Koltun (Apr. 2018). "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling." In: *arXiv:1803.01271 [cs]*. arXiv: 1803.01271 [cs]. URL: http://arxiv.org/abs/1803.01271 (visited on 11/25/2021) (Cited on page 52).

Bates, J. M. and C. W. J. Granger (Dec. 1969). "The Combination of Forecasts." en. In: *Journal of the Operational Research Society*. DOI: 10.1057/jors.1969.103. URL: https://www.tandfonline.com/doi/abs/10.1057/jors.1969.103 (visited on 09/21/2021) (Cited on page 59).

Ben Taieb, S., G. Bontempi, A. Sorjamaa, and A. Lendasse (2009). "Long-Term Prediction of Time Series by combining Direct and MIMO Strategies." In: *Proceedings of the 2009 IEEE International Joint Conference on Neural Networks*. Atlanta, U.S.A., pp. 3054–3061 (Cited on page 25).

Ben Taieb, S., A. Sorjamaa, and G. Bontempi (2010). "Multiple-Output Modelling for Multi-Step-Ahead Forecasting." In: *Neurocomputing* 73, pp. 1950–1957 (Cited on page 25).

Ben Taieb, Souhaib, Gianluca Bontempi, Amir F. Atiya, and Antti Sorjamaa (June 2012). "A Review and Comparison of Strategies for Multi-Step Ahead Time Series Forecasting Based on the NN5 Forecasting Competition." In: *Expert Systems with Applications* 39.8, pp. 7067–7083. ISSN: 09574174. DOI: 10.1016/j.eswa.2012.01.039. URL: http://linkinghub.elsevier.com/retrieve/pii/S0957417412000528 (visited on 09/11/2018) (Cited on pages 7, 25, 42, 67, 72).

Ben Taieb, Souhaib, Rob J Hyndman, and Gianluca Bontempi (2014). "Machine learning strategies for multi-step-ahead time series forecasting." In: (Cited on page 32).

Bengio, Yoshua (Jan. 2009). "Learning Deep Architectures for AI." In: *Found. Trends Mach. Learn.* 2.1, pp. 1–127. ISSN: 1935-8237. DOI: 10.1561/2200000006. URL: http://dx.doi.org/10.1561/2200000006 (Cited on page 57).

Bergmeir, Christoph, Rob J. Hyndman, and José M. Benítez (Apr. 2016). "Bagging Exponential Smoothing Methods Using STL Decomposition and Box–Cox Transformation." en. In: *International Journal of Forecasting* 32.2, pp. 303–312. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2015.07.002. URL: https://www.sciencedirect.com/science/article/pii/S0169207015001120 (visited on 09/23/2021) (Cited on page 61).

Bermingham, Mairead L, Ricardo Pong-Wong, Athina Spiliopoulou, Caroline Hayward, Igor Rudan, Harry Campbell, Alan F Wright, James F Wilson, Felix Agakov, Pau Navarro, et al. (2015). "Application of high-dimensional feature selection: evaluation for genomic prediction in man." In: *Scientific reports* 5.1, pp. 1–12 (Cited on page 76).

Bianchi, Filippo Maria, Enrico Maiorino, Michael C. Kampffmeyer, Antonello Rizzi, and Robert Jenssen (2017). "An Overview and Comparative Analysis of Recurrent Neural Networks for Short Term Load Forecasting." In: *arXiv:1705.04378 [cs]*. DOI: 10.1007/978-3-319-70338-1. arXiv: 1705.04378 [cs]. URL: http://arxiv.org/abs/1705.04378 (visited on 03/28/2021) (Cited on pages 54, 58).

Birattari, Mauro, Thomas Stützle, Luis Paquete, Klaus Varrentrapp, et al. (2002). "A Racing Algorithm for Configuring Metaheuristics." In: *Gecco*. Vol. 2. 2002 (Cited on page 148).

Bitencourt, Hugo Vinicius and Frederico Gadelha Guimarães (July 2021). "High-Dimensional Multivariate Time Series Forecasting in IoT Applications Using Embedding Non-Stationary Fuzzy Time Series." In: *arXiv:2107.09785 [cs, eess]*. arXiv: 2107.09785 [cs, eess]. URL: http://arxiv.org/abs/2107.09785 (visited on 08/19/2021) (Cited on page 144).

Bolón-Canedo, Verónica, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos (Mar. 2013). "A Review of Feature Selection Methods on Synthetic Data." In: *Knowledge and Information Systems* 34.3, pp. 483–519. ISSN: 0219-3116. DOI: 10.1007/s10115-012-0487-8. URL: https://doi.org/10.1007/s10115-012-0487-8 (visited on 11/09/2021) (Cited on page 16).

Bontempi, G. (2008). "Long term time series prediction with multi-input multi-output local learning." In: *Proceedings of the 2nd European Symposium on Time Series Prediction (TSP), ESTSP08*, pp. 145–154 (Cited on page 25).

Bontempi, G (2013). "Handbook Statistical foundations of machine learning." In: *Universite Libre de Bruxelles* (Cited on pages 13, 14, 16, 21, 59).

Bontempi, G. (July 2014). "A Monte Carlo Strategy for Structured Multiple-Step-Ahead Time Series Prediction." In: *2014 International Joint Conference on Neural Networks (IJCNN)*, pp. 853–858. DOI: 10.1109/IJCNN.2014.6889666 (Cited on page 89).

Bontempi, G., M. Birattari, and H. Bersini (1999a). "Lazy Learning for Modeling and Control Design." In: *International Journal of Control* 72.7/8, pp. 643–658 (Cited on page 105).

– (1999b). "Local learning for iterated time-series prediction." In: *Machine Learning: Proceedings of the Sixteenth International Conference*. Ed. by I. Bratko and S. Dzeroski. San Francisco, CA: Morgan Kaufmann Publishers, pp. 32–38 (Cited on page 25).

Bontempi, Gianluca (1999). "Local learning techniques for modeling, prediction and control." In: (Cited on page 45).

Bontempi, Gianluca and Souhaib Ben Taieb (July 2011). "Conditionally Dependent Strategies for Multiple-Step-Ahead Prediction in Local Learning." In: *International Journal of Forecasting* 27.3, pp. 689–699. ISSN: 01692070. DOI: 10.1016/j.ijforecast.2010.09.004. URL: http://linkinghub.elsevier.com/retrieve/pii/S0169207010001433 (visited on 09/06/2018) (Cited on pages 7, 25, 46, 70, 72, 97, 99, 105).

Bontempi, Gianluca, Souhaib Ben Taieb, and Yann-Aël Le Borgne (2013). "Machine Learning Strategies for Time Series Forecasting." In: *Business Intelligence*. Ed. by Marie-Aude Aufaure and Esteban Zimányi. Red. by Wil van der Aalst, John Mylopoulos, Michael Rosemann, Michael J. Shaw, and Clemens Szyperski. Vol. 138. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 62–77. ISBN: 978-3-642-36317-7 978-3-642-36318-4. DOI: 10.1007/978-3-642-36318-4_3. URL: http://link.springer.com/10.1007/978-3-642-36318-4_3 (visited on 09/25/2018) (Cited on pages 24, 25).

Bontempi, Gianluca, Mauro Birattari, and Hugues Bersini (1999c). "Lazy learning for local modelling and control design." In: *International Journal of Control* 72.7-8, pp. 643–658 (Cited on page 84).

Bontempi, Gianluca, Yann-Aël Le Borgne, and Jacopo De Stefani (2017). "A Dynamic Factor Machine Learning Method for Multi-Variate and Multi-Step-Ahead Forecasting." In: *Proceedings of DSAA 2017, the 4th IEEE International Conference on Data Science and Advanced Analytics 2017* (Cited on pages 9, 10, 87).

Bontempi, Gianluca, Yann-Aël Le Borgne, and Jacopo De Stefani (2017). "A dynamic factor machine learning method for multi-variate and multi-step-ahead forecasting." In: *Data Science and Advanced Analytics (DSAA), 2017 IEEE International Conference on*. IEEE, pp. 222–231. DOI: 10.1109/DSAA.2017.1 (Cited on pages 92, 94, 100).

Bontempi, Gianluca, Souhaib Ben Taieb, and Yann-Aël Le Borgne (2012). "Machine learning strategies for time series forecasting." In: *European business intelligence summer school*. Springer, Berlin, Heidelberg, pp. 62–77 (Cited on page 24).

Borovykh, Anastasia, Sander Bohte, and Cornelis W. Oosterlee (Mar. 14, 2017). "Conditional Time Series Forecasting with Convolutional Neural Networks." In: arXiv: 1703.04691 [stat]. URL: http://arxiv.org/abs/1703.04691 (visited on 07/18/2018) (Cited on page 52).

Boulegane, Dihia, Albert Bifet, and Giyyarpuram Madhusudan (Dec. 2019). "Arbitrated Dynamic Ensemble with Abstaining for Time-Series Forecasting on Data Streams." In: *2019 IEEE International Conference on Big Data (Big Data)*, pp. 1040–1045. DOI: 10.1109/BigData47090.2019.9005541 (Cited on page 60).

Bourlard, Hervé and Yves Kamp (1988). "Auto-association by multilayer perceptrons and singular value decomposition." In: *Biological cybernetics* 59.4, pp. 291–294 (Cited on page 57).

Box, G. E. P. and D. R. Cox (1964). "An Analysis of Transformations." In: *Journal of the Royal Statistical Society. Series B (Methodological)* 26.2, pp. 211–252. ISSN: 0035-9246. URL: https://www.jstor.org/stable/2984418 (visited on 11/19/2021) (Cited on page 26).

Box, G.E..P. and G.C. Tiao (1977). "A canonical analysis of multiple time series." In: *Biometrika* 64.2, pp. 355–365 (Cited on page 50).

Box, George EP, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung (2015). *Time series analysis: forecasting and control*. John Wiley & Sons (Cited on page 3).

Breiman, Leo (2001a). "Random forests." In: *Machine learning* 45.1, pp. 5–32 (Cited on page 47).

– (Aug. 2001b). "Statistical Modeling: The Two Cultures (with Comments and a Rejoinder by the Author)." In: *Statistical Science* 16.3, pp. 199–231. ISSN: 0883-4237, 2168-8745. DOI: 10.1214/ss/1009213726. URL: https://projecteuclid.org/journals/statistical-science/volume-16/issue-3/Statistical-Modeling--The-Two-Cultures-with-comments-and-a/10.1214/ss/1009213726.full (visited on 07/30/2021) (Cited on page 14).

Breiman, Leo, Jerome H Friedman, Richard A Olshen, and Charles J Stone (2017). *Classification and regression trees*. Routledge (Cited on page 47).

Brockwell, Peter J, Peter J Brockwell, Richard A Davis, and Richard A Davis (2016). *Introduction to time series and forecasting*. Springer (Cited on page 13).

Bruxelles Mobilité and Machine Learning Group - ULB (2021). *Mobility Dataset*. URL: https://www.kaggle.com/giobbu/belgium-obu?select=Bxl_60.csv (Cited on page 88).

California Departement of Transport (2021). *Traffic Dataset*. URL: http://pems.dot.ca.gov/ (Cited on page 88).

Cameron, A Colin and Frank AG Windmeijer (1997). "An R-squared measure of goodness of fit for some common nonlinear regression models." In: *Journal of econometrics* 77.2, pp. 329–342 (Cited on page 119).

Caruana, Rich (July 1, 1997). "Multitask Learning." In: *Machine Learning* 28.1, pp. 41–75. ISSN: 1573-0565. DOI: 10.1023/A:1007379606734. URL: https://doi.org/10.1023/A:1007379606734 (visited on 09/04/2018) (Cited on page 54).

Cawley, Gavin C. (2012). "Over-Fitting in Model Selection and Its Avoidance." In: *Advances in Intelligent Data Analysis XI*. Ed. by Jaakko Hollmén, Frank Klawonn, and Allan Tucker. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 1–1. ISBN: 978-3-642-34156-4. DOI: 10.1007/978-3-642-34156-4_1 (Cited on page 22).

Center, M Open Forecasting (2020). *M4-methods*. URL: \url{https://github.com/Mcompetitions/M4-methods} (Cited on pages 92, 93, 120, 132).

Cerqueira, Vitor, Luis Torgo, and Igor Mozetic (May 2019). "Evaluating Time Series Forecasting Models: An Empirical Study on Performance Estimation Methods." In: *arXiv:1905.11744 [cs, stat]*. arXiv: 1905.11744 [cs, stat]. URL: http://arxiv.org/abs/1905.11744 (visited on 08/19/2020) (Cited on pages 34–36, 78).

Cerqueira, Vitor, Luis Torgo, Mariana Oliveira, and Bernhard Pfahringer (2017a). "Dynamic and heterogeneous ensembles for time series forecasting." In: *2017 IEEE international conference on data science and advanced analytics (DSAA)*. IEEE, pp. 242–251 (Cited on pages 60, 75, 139).

Cerqueira, Vítor, Luís Torgo, Fábio Pinto, and Carlos Soares (2017b). "Arbitrated ensemble for time series forecasting." In: *Joint European conference on machine learning and knowledge discovery in databases*. Springer, pp. 478–494 (Cited on pages 60, 61, 75).

Cerqueira, Vitor, Luis Torgo, and Carlos Soares (2019). "Machine learning vs statistical methods for time series forecasting: Size matters." In: *arXiv preprint arXiv:1909.13316* (Cited on page 75).

Chakraborty, Kanad, Kishan Mehrotra, Chilukuri K. Mohan, and Sanjay Ranka (Nov. 1992a). "Forecasting the Behavior of Multivariate Time Series Using Neural Networks." In: *Neural Networks* 5.6, pp. 961–970. ISSN: 08936080. DOI: 10.1016/S0893-6080(05)80092-9. URL: http://linkinghub.elsevier.com/retrieve/pii/S0893608005800929 (visited on 09/11/2018) (Cited on page 43).

– (Nov. 1992b). "Forecasting the Behavior of Multivariate Time Series Using Neural Networks." en. In: *Neural Networks* 5.6, pp. 961–970. ISSN: 0893-6080. DOI: 10.1016/S0893-6080(05)80092-9. URL: http://www.sciencedirect.com/science/article/pii/S0893608005800929 (visited on 06/16/2020) (Cited on page 85).

Chang, Yen-Yu, Fan-Yun Sun, Yueh-Hua Wu, and Shou-De Lin (Sept. 6, 2018). "A Memory-Network Based Solution for Multivariate Time-Series Forecasting." In: arXiv: 1809.02105 [cs, stat]. URL: http://arxiv.org/abs/1809.02105 (visited on 09/08/2018) (Cited on page 54).

Cheng, Haibin, Pang-Ning Tan, Jing Gao, and Jerry Scripps (2006). "Multistep-Ahead Time Series Prediction." In: *PAKDD*, pp. 765–774 (Cited on page 25).

Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (2014a). "Learning phrase representations using RNN encoder-decoder for statistical machine translation." In: *arXiv preprint arXiv:1406.1078* (Cited on pages 58, 70, 147).

Cho, Kyunghyun, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (Oct. 2014b). "Learning Phrase Representations

Using RNN Encoder–Decoder for Statistical Machine Translation." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 1724–1734. URL: http://www.aclweb.org/anthology/D14-1179 (visited on 09/11/2018) (Cited on page 53).

Chollet, François et al. (2015). *Keras*. https://github.com/fchollet/keras (Cited on page 4).

Clements, Michael and David Hendry (1998). *Forecasting Economic Time Series*. Cambridge University Press. URL: https://EconPapers.repec.org/RePEc:cup:cbooks:9780521634809 (Cited on page 60).

Cortes, Corinna and Vladimir Vapnik (1995). "Support vector machine." In: *Machine learning* 20.3, pp. 273–297 (Cited on page 46).

Cunningham, John P. and Zoubin Ghahramani (2015). "Linear Dimensionality Reduction: Survey, Insights, and Generalizations." In: *Journal of Machine Learning Research* 16.89, pp. 2859–2900. ISSN: 1533-7928. URL: http://jmlr.org/papers/v16/cunningham15a.html (visited on 06/09/2021) (Cited on page 55).

Dargan, Shaveta, Munish Kumar, Maruthi Rohit Ayyagari, and Gulshan Kumar (Sept. 2020). "A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning." In: *Archives of Computational Methods in Engineering* 27.4, pp. 1071–1092. ISSN: 1886-1784. DOI: 10.1007/s11831-019-09344-w. URL: https://doi.org/10.1007/s11831-019-09344-w (visited on 11/02/2021) (Cited on page 143).

De Caro, Fabrizio, Jacopo De Stefani, Gianluca Bontempi, Alfredo Vaccaro, and Domenico Villacci (Oct. 18, 2020). "Robust Assessment of Short-Term Wind Power Forecasting Models on Multiple Time Horizons." In: *Technology and Economics of Smart Grids and Sustainable Energy* 5.1, p. 19. ISSN: 2199-4706. DOI: 10.1007/s40866-020-00090-8. URL: https://doi.org/10.1007/s40866-020-00090-8 (visited on 07/27/2021) (Cited on pages 9, 115).

De Caro, Fabrizio, Jacopo De Stefani, Alfredo Vaccaro, and Gianluca Bontempi (2021). "DAFT-E : Feature-based Multivariate and Multi-step-ahead Wind Power Forecasting." In: *IEEE Transactions on Sustainable Energy*, pp. 1–1. DOI: 10.1109/TSTE.2021.3130949 (Cited on pages 9, 115).

De Gooijer, Jan G. and Rob J. Hyndman (Jan. 2006). "25 Years of Time Series Forecasting." In: *International Journal of Forecasting*. Twenty Five Years of Forecasting 22.3, pp. 443–473. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2006.01.001. URL: https://www.sciencedirect.com/science/article/pii/S0169207006000021 (visited on 10/10/2021) (Cited on pages 49, 143).

De Jay, Nicolas, Simon Papillon-Cavanagh, Catharina Olsen, Nehme El-Hachem, Gianluca Bontempi, and Benjamin Haibe-Kains (2013). "mRMRe: an R package for parallelized mRMR ensemble feature selection." In: *Bioinformatics* 29.18, pp. 2365–2368 (Cited on pages 17, 84).

De Stefani, Jacopo and Gianluca Bontempi (2021a). *E-DFML-Experiment*. URL: \url{https://github.com/jdestefani/E-DFML-Experiments} (Cited on page 93).

– (2021b). "Factor-Based Framework for Multivariate and Multi-Step-Ahead Forecasting of Large Scale Time Series." In: *Frontiers in Big Data* 4, p. 75. ISSN: 2624-909X. DOI: 10.3389/fdata.2021.690267. URL: https://www.frontiersin.org/article/10.3389/fdata.2021.690267 (visited on 09/10/2021) (Cited on pages 9, 87).

De Stefani, Jacopo, Gianluca Bontempi, Olivier Caelen, and Dalila Hattab (Jan. 3, 2019a). "System and Method for Managing Risks in a Process." Pat. WO2019002582 (A1). Worldline. **Classifications: IPC:** G06Q10/04; G06Q50/04, **CPC:** G06Q10/04 (EP); G06Q50/04 (EP); Y02P90/30 (EP). URL: https://worldwide.espacenet.com/publicationDetails/biblio?FT=D&date=20190103&DB=&locale=en_EP&CC=WO&NR=2019002582A1&KC=A1&ND=5 (visited on 07/27/2021) (Cited on page 10).

De Stefani, Jacopo, Olivier Caelen, Dalila Hattab, and Gianluca Bontempi (n.d.). "Machine Learning for Multi-Step Ahead Forecasting of Volatility Proxies." In: *ECML PKDD 2017 Workshops - MIning DAta for financial applicationS (MIDAS 2017)* (Cited on page 9).

De Stefani, Jacopo, Olivier Caelen, Dalila Hattab, Yann-Aël Le Borgne, and Gianluca Bontempi (2019b). "A Multivariate and Multi-Step Ahead Machine Learning Approach to Traditional and Cryptocurrencies Volatility Forecasting." In: *ECML PKDD 2018 Workshops - MIning DAta for financial applicationS (MIDAS 2018)*. Ed. by Carlos Alzate et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 7–22. ISBN: 978-3-030-13463-1. DOI: 10.1007/978-3-030-13463-1_1 (Cited on pages 9, 87).

De Stefani, Jacopo, Yann-Aël Le Borgne, Olivier Caelen, Dalila Hattab, and Gianluca Bontempi (Aug. 31, 2018). "Batch and Incremental Dynamic Factor Machine Learning for Multivariate and Multi-Step-Ahead Forecasting." In: *International Journal of Data Science and Analytics*. ISSN: 2364-4168. DOI: 10.1007/s41060-018-0150-x. URL: https://doi.org/10.1007/s41060-018-0150-x (visited on 09/12/2018) (Cited on pages 9, 87, 92, 94).

DeMers, David and Garrison W Cottrell (1993). "Non-linear dimensionality reduction." In: *Advances in neural information processing systems*. Citeseer, pp. 580–587 (Cited on page 71).

Deep Learning's Diminishing Returns (Sept. 2021). URL: https://spectrum.ieee.org/deep-learning-computational-cost (visited on 10/06/2021) (Cited on page 6).

Demolli, Halil, Ahmet Sakir Dokuz, Alper Ecemis, and Murat Gokcek (2019). "Wind power forecasting based on daily wind speed data using machine learning algorithms." In: *Energy Conversion and Management* 198, p. 111823 (Cited on page 118).

Demšar, Janez (2006). "Statistical comparisons of classifiers over multiple data sets." In: *The Journal of Machine Learning Research* 7, pp. 1–30 (Cited on pages 36, 37, 93–95, 164, 166).

Deng, Xing, Haijian Shao, Chunlong Hu, Dengbiao Jiang, and Yingtao Jiang (2020). "Wind Power Forecasting Methods Based on Deep Learning: A Survey." In: *Computer Modeling in Engineering & Sciences* 122.1, pp. 273–302 (Cited on pages 131, 132).

Domingos, Pedro (2012). "A few useful things to know about machine learning." In: *Communications of the ACM* 55.10, pp. 78–87 (Cited on page 84).

Downs, G. W. and D. M. Rocke (1983). "Municipal Budget Forecasting with Multivariate ARMA Models." In: *Journal of Forecasting* 2.4, pp. 377–387. ISSN: 1099-131X. DOI: 10.1002/for.3980020407. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/for.3980020407 (visited on 10/10/2021) (Cited on page 49).

Du, Shengdong, Tianrui Li, Yan Yang, and Shi-Jinn Horng (2020). "Multivariate time series forecasting via attention-based encoder–decoder framework." In: *Neurocomputing* 388, pp. 269–279 (Cited on page 58).

Edlund, Per-Olov and Sune Karlsson (Apr. 1993). "Forecasting the Swedish Unemployment Rate VAR vs. Transfer Function Modelling." In: *International Journal of Forecasting* 9.1, pp. 61–76. ISSN: 0169-2070. DOI: 10.1016/0169-2070(93)90054-Q. URL: https://www.sciencedirect.com/science/article/pii/016920709390054Q (visited on 10/10/2021) (Cited on page 49).

Eisenach, Carson, Yagna Patel, and Dhruv Madeka (Sept. 2020). "MQTransformer: Multi-Horizon Forecasts with Context Dependent and Feedback-Aware Attention." In: *arXiv:2009.14799 [cs, stat]*. arXiv: 2009.14799 [cs, stat]. URL: http://arxiv.org/abs/2009.14799 (visited on 10/06/2020) (Cited on page 54).

Engle, Robert F. and C. W. J. Granger (1987). "Co-Integration and Error Correction: Representation, Estimation, and Testing." In: *Econometrica* 55.2, pp. 251–276. ISSN: 0012-9682. DOI: 10.2307/1913236. URL: https://www.jstor.org/stable/1913236 (visited on 10/28/2021) (Cited on page 3).

Escribano, Alvaro, Daniel Peña, and Esther Ruiz (Oct. 2021). "30 Years of Cointegration and Dynamic Factor Models Forecasting and Its Future with Big Data: Editorial." In:

International Journal of Forecasting 37.4, pp. 1333–1337. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2021.06.004. URL: https://www.sciencedirect.com/science/article/pii/S0169207021001047 (visited on 10/12/2021) (Cited on pages 3, 7, 143).

Fan, Cheng, Meiling Chen, Xinghua Wang, Jiayuan Wang, and Bufu Huang (2021). "A Review on Data Preprocessing Techniques Toward Efficient and Reliable Knowledge Discovery From Building Operational Data." In: *Frontiers in Energy Research* 9, p. 77. ISSN: 2296-598X. DOI: 10.3389/fenrg.2021.652801. URL: https://www.frontiersin.org/article/10.3389/fenrg.2021.652801 (visited on 11/07/2021) (Cited on page 16).

Fedorov, Valerii Vadimovich (2013). *Theory of optimal experiments*. Elsevier (Cited on page 15).

Foley, Aoife M, Paul G Leahy, Antonino Marvuglia, and Eamon J McKeogh (2012). "Current methods and advances in forecasting of wind power generation." In: *Renewable Energy* 37.1, pp. 1–8 (Cited on page 118).

Forni, Mario, Marc Hallin, Marco Lippi, and Lucrezia Reichlin (2000). "The Generalized Dynamic-Factor Model: Identification and Estimation." In: *The Review of Economics and Statistics* 82.4, pp. 540–554. DOI: 10.1162/003465300559037. eprint: https://doi.org/10.1162/003465300559037. URL: https://doi.org/10.1162/003465300559037 (Cited on page 143).

– (2005). "The Generalized Dynamic Factor Model." In: *Journal of the American Statistical Association* 100.471, pp. 830–840. DOI: 10.1198/016214504000002050 (Cited on pages 69, 92).

Franses, P.H. and R. Legerstee (2010). "A Unifying View on Multi-Step Forecasting Using an Autoregression." In: *Journal of Economic Surveys* 24.3, pp. 389–401 (Cited on page 50).

Friedman, Jerome, Trevor Hastie, Robert Tibshirani, et al. (2001). *The elements of statistical learning*. Vol. 1. 10. Springer series in statistics New York (Cited on pages 4, 76).

Friedman, Milton (1937). "The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance." In: *Journal of the American Statistical Association* 32.200, pp. 675–701. DOI: 10.1080/01621459.1937.10503522. eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1937.10503522. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1937.10503522 (Cited on pages 36, 119).

Galicia, Antonio, José F. Torres, Francisco Martínez-Álvarez, and Alicia Troncoso (2017). "Scalable Forecasting Techniques Applied to Big Electricity Time Series." In: *Advances in Computational Intelligence: 14th International Work-Conference on Artificial Neural Networks, IWANN 2017, Cadiz, Spain, June 14-16, 2017, Proceedings, Part II*. Ed. by Ignacio Rojas, Gonzalo Joya, and Andreu Catala. Cham: Springer International Publishing, pp. 165–175. ISBN: 978-3-319-59147-6. DOI: 10.1007/978-3-319-59147-6_15. URL: https://doi.org/10.1007/978-3-319-59147-6_15 (Cited on page 4).

Gardner Jr, Everette S (1985). "Exponential smoothing: The state of the art." In: *Journal of forecasting* 4.1, pp. 1–28 (Cited on page 40).

– (2006). "Exponential smoothing: The state of the art—Part II." In: *International journal of forecasting* 22.4, pp. 637–666 (Cited on page 40).

Garman, Mark B and Michael J Klass (1980). "On the Estimation of Security Price Volatilities from Historical Data." In: *Journal of business*, pp. 67–78 (Cited on pages 156–159).

Gasthaus, Jan, Konstantinos Benidis, Yuyang Wang, Syama S Rangapuram, David Salinas, Valentin Flunkert, and Tim Januschowski (2019). "Probabilistic Forecasting with Spline Quantile Function RNNs." In: p. 10 (Cited on page 148).

Gilbert., P.D. (1993). "State space and ARMA models: an overview of the equivalence." In: (Cited on pages 48, 105).

Goel, Hardik, Igor Melnyk, and Arindam Banerjee (Sept. 10, 2017). "R2N2: Residual Recurrent Neural Networks for Multivariate Time Series Forecasting." In: arXiv: 1709.03159 [cs, stat]. URL: http://arxiv.org/abs/1709.03159 (visited on 08/17/2018) (Cited on page 53).

Golyandina, N., A. Korobeynikov, A. Shlemov, and K. Usevich (2015). "Multivariate and 2D Extensions of Singular Spectrum Analysis with the Rssa Package." In: *Journal of Statistical Software* 67 (Cited on page 106).

Golyandina, Nina, Vladimir Nekrutkin, and Anatoly Zhigljavsky (2001). *Analysis of Time Series Structure: SSA and related techniques*. CRC Press (Cited on page 56).

Graves, Alex (2012). *Supervised sequence labelling with Recurrent Neural Networks*. Springer (Cited on page 52).

Graybill, Franklin A. and R. B. Deal (1959). "Combining Unbiased Estimators." In: *Biometrics* 15.4, pp. 543–550. ISSN: 0006-341X. DOI: 10.2307/2527652. URL: https://www.jstor.org/stable/2527652 (visited on 11/12/2021) (Cited on page 22).

Grigsby, Jake, Zhe Wang, and Yanjun Qi (Sept. 2021). "Long-Range Transformers for Dynamic Spatiotemporal Forecasting." In: *arXiv:2109.12218 [cs, stat]*. arXiv: 2109.12218 [cs, stat]. URL: http://arxiv.org/abs/2109.12218 (visited on 10/03/2021) (Cited on page 54).

Grömping, Ulrike (2009). "Variable importance assessment in regression: linear regression versus random forest." In: *The American Statistician* 63.4, pp. 308–319 (Cited on page 47).

Guen, Vincent Le and Nicolas Thome (Apr. 2021). "Probabilistic Time Series Forecasting with Structured Shape and Temporal Diversity." In: *arXiv:2010.07349 [cs, stat]*. arXiv: 2010.07349 [cs, stat] (Cited on page 148).

Guo, M., Z. Bai, and H.Z. An (1999). "Multi-step prediction for nonlinear autoregressive models based on empirical distributions." In: *Statistica Sinica*, pp. 559–570 (Cited on page 25).

Guo, Tian, Zhao Xu, Xin Yao, Haifeng Chen, Karl Aberer, and Koichi Funaya (2016). "Robust online time series prediction with recurrent neural networks." In: *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. Ieee, pp. 816–825 (Cited on page 50).

Hanifi, Shahram, Xiaolei Liu, Zi Lin, and Saeid Lotfian (2020). "A Critical Review of Wind Power Forecasting Methods—Past, Present and Future." In: *Energies* 13.15, p. 3764 (Cited on page 127).

Hansen, Peter R and Asger Lunde (2005). "A forecast comparison of volatility models: does anything beat a GARCH (1, 1)?" In: *Journal of applied econometrics* 20.7, pp. 873–889 (Cited on pages 158, 159).

Hastie, T., R. Tibshirani, and J.H. Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer. ISBN: 978-0-387-84884-6. URL: https://books.google.be/books?id=eBSgoAEACAAJ (Cited on pages 13, 47).

Hegde, Anant, Jose C Principe, Deniz Erdogmus, Umut Ozertem, Yadunandana N Rao, and Hemanth Peddaneni (2006). "Perturbation-based eigenvector updates for on-line principal components analysis and canonical correlation analysis." In: *The Journal of VLSI Signal Processing* 45.1, pp. 85–95 (Cited on pages 56, 57).

Heuts, R.M.J. and J.H.J.M. Bronckers (1988). "Forecasting the Dutch Heavy Truck Market. A Multivariate Approach." In: *International Journal of Forecasting* 4.1, pp. 57–79. ISSN: 0169-2070. DOI: 10.1016/0169-2070(88)90010-6 (Cited on page 49).

Hewamalage, Hansika, Christoph Bergmeir, and Kasun Bandara (2021). "Recurrent neural networks for time series forecasting: Current status and future directions." In: *International Journal of Forecasting* 37.1, pp. 388–427 (Cited on pages 5, 54, 58, 85).

Hillmer, S. C., D. F. Larcker, and D. A. Schroeder (1983). "Forecasting Accounting Data: A Multiple Time-Series Analysis." In: *Journal of Forecasting* 2.4, pp. 389–404. ISSN: 1099-131X. DOI: 10.1002/for.3980020408. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/for.3980020408 (visited on 10/10/2021) (Cited on page 49).

Hochreiter, Sepp, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. (2011). "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies." In: Kolen,

John F. and Stefan C. Kremer. *A Field Guide to Dynamical Recurrent Networks*. IEEE. ISBN: 978-0-470-54403-7. DOI: 10.1109/9780470544037.ch14. URL: http://ieeexplore.ieee.org/search/srchabstract.jsp?arnumber=5264952 (visited on 09/11/2018) (Cited on page 52).

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory." In: *Neural computation* 9.8, pp. 1735–1780 (Cited on pages 53, 58, 70, 99, 147).

Hoerl, Arthur E and Robert W Kennard (1970). "Ridge regression: Biased estimation for nonorthogonal problems." In: *Technometrics* 12.1, pp. 55–67 (Cited on page 21).

Holt, Charles C (2004). "Forecasting seasonals and trends by exponentially weighted moving averages." In: *International journal of forecasting* 20.1, pp. 5–10 (Cited on pages 39, 70).

Hospedales, Timothy, Antreas Antoniou, Paul Micaelli, and Amos Storkey (Nov. 2020). "Meta-Learning in Neural Networks: A Survey." In: *arXiv:2004.05439 [cs, stat]*. arXiv: 2004.05439 [cs, stat]. URL: http://arxiv.org/abs/2004.05439 (visited on 11/11/2021) (Cited on page 19).

Hotelling, Harold (1933). "Analysis of a complex of statistical variables into principal components." In: *Journal of educational psychology* 24.6, p. 417 (Cited on page 71).

Hutter, Frank, Lars Kotthoff, and Joaquin Vanschoren, eds. (2019). *Automated Machine Learning: Methods, Systems, Challenges*. The Springer Series on Challenges in Machine Learning. Cham: Springer International Publishing. ISBN: 978-3-030-05317-8 978-3-030-05318-5. DOI: 10.1007/978-3-030-05318-5. URL: http://link.springer.com/10.1007/978-3-030-05318-5 (visited on 11/11/2021) (Cited on page 19).

Hwang, Yunseong, Anh Tong, and Jaesik Choi (2016). "Automatic construction of nonparametric relational regression models for multiple time series." In: *International Conference on Machine Learning*. PMLR, pp. 3030–3039 (Cited on page 69).

Hyndman, Rob J (2020). "A brief history of forecasting competitions." In: *International Journal of Forecasting* 36.1, pp. 7–14 (Cited on pages 40, 72, 92).

Hyndman, Rob J and George Athanasopoulos (2018). *Forecasting: principles and practice*. OTexts (Cited on pages 13, 27, 28, 39).

Hyndman, Rob J and Baki Billah (2003). "Unmasking the Theta method." In: *International Journal of Forecasting* 19.2, pp. 287–290 (Cited on page 40).

Hyndman, Rob J. and Anne B. Koehler (2006). "Another look at measures of forecast accuracy." In: *International Journal of Forecasting* 22.4, pp. 679 –688. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2006.03.001. URL: http://www.sciencedirect.com/science/article/pii/S0169207006000239 (Cited on pages 11, 35–37, 119).

Hyndman, Rob, Yanfei Kang, Pablo Montero-Manso, Thiyanga Talagala, Earo Wang, Yangzhuoran Yang, and Mitchell O'Hara-Wild (2020). *tsfeatures: Time Series Feature Extraction*. R package version 1.0.2. URL: https://CRAN.R-project.org/package=tsfeatures (Cited on page 29).

Inoue, Atsushi and Lutz Kilian (Mar. 2004). *Bagging Time Series Models*. en. SSRN Scholarly Paper ID 540262. Rochester, NY: Social Science Research Network. URL: https://papers.ssrn.com/abstract=540262 (visited on 09/23/2021) (Cited on page 61).

Jain, A. K., R. P. W. Duin, and Jianchang Mao (2000). "Statistical pattern recognition: a review." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.1, pp. 4–37. DOI: 10.1109/34.824819 (Cited on page 16).

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani (2013). *An introduction to statistical learning*. Vol. 112. Springer (Cited on page 13).

Januschowski, Tim, Jan Gasthaus, Yuyang Wang, David Salinas, Valentin Flunkert, Michael Bohlke-Schneider, and Laurent Callot (2020). "Criteria for classifying forecasting methods." In: *International Journal of Forecasting* 36.1, pp. 167–177 (Cited on pages 4, 38, 65, 72).

Jolliffe, I.T. (2002). *Principal Component Analysis*. Springer (Cited on page 56).

Ke, Guolin, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu (2017). "Lightgbm: A highly efficient gradient boosting decision tree." In: *Advances in neural information processing systems* 30, pp. 3146–3154 (Cited on pages 47, 71, 72).

Kim, Minyoung, Yuting Wang, Pritish Sahu, and Vladimir Pavlovic (2019). "Bayes-Factor-VAE: Hierarchical Bayesian Deep Auto-Encoder Models for Factor Disentanglement." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2979–2987. URL: https://openaccess.thecvf.com/content_ICCV_2019/html/Kim_Bayes-Factor-VAE_Hierarchical_Bayesian_Deep_Auto-Encoder_Models_for_Factor_Disentanglement_ICCV_2019_paper.html (visited on 10/18/2020) (Cited on page 96).

Kirchgassner, G. and J. Wolters (2007). *Introduction to Modern Time Series Analysis*. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, Online–Ressource. ISBN: 978-3-540-73291-4 (Cited on page 7).

Kline, D. M. (2004). "Methods for Multi-Step Time Series Forecasting Neural Networks." In: *Neural Networks in Business Forecasting*. IGI Global (Cited on page 25).

Konečný, Jakub, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon (Oct. 2017). "Federated Learning: Strategies for Improving Communication Efficiency." In: *arXiv:1610.05492 [cs]*. arXiv: 1610.05492 [cs]. URL: http://arxiv.org/abs/1610.05492 (visited on 11/06/2021) (Cited on page 149).

Koochali, Alireza, Andreas Dengel, and Sheraz Ahmed (May 2020). "If You Like It, GAN It. Probabilistic Multivariate Times Series Forecast With GAN." In: *arXiv:2005.01181 [cs, eess]*. arXiv: 2005.01181 [cs, eess] (Cited on page 148).

Koprinska, Irena, Mashud Rana, and Ashfaqur Rahman (2019). "Dynamic Ensemble Using Previous and Predicted Future Performance for Multi-Step-Ahead Solar Power Forecasting." en. In: *Artificial Neural Networks and Machine Learning – ICANN 2019: Text and Time Series*. Ed. by Igor V. Tetko, Věra Kůrková, Pavel Karpov, and Fabian Theis. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 436–449. ISBN: 978-3-030-30490-4. DOI: 10.1007/978-3-030-30490-4_35 (Cited on page 59).

Korprasertsak, Natapol and Thananchai Leephakpreeda (2019). "Robust short-term prediction of wind power generation under uncertainty via statistical interpretation of multiple forecasting models." In: *Energy* 180, pp. 387–397 (Cited on page 118).

Kuhn, Max (2008). "Building Predictive Models in R Using the caret Package." In: *Journal of Statistical Software, Articles* 28.5, pp. 1–26. ISSN: 1548-7660. DOI: 10.18637/jss.v028.i05. URL: https://www.jstatsoft.org/v028/i05 (Cited on page 4).

Kuncheva, Ludmila I. (2004). "Classifier Ensembles for Changing Environments." en. In: *Multiple Classifier Systems*. Ed. by Takeo Kanade et al. Vol. 3077. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–15. ISBN: 978-3-540-22144-9 978-3-540-25966-4. DOI: 10.1007/978-3-540-25966-4_1. URL: http://link.springer.com/10.1007/978-3-540-25966-4_1 (visited on 09/22/2021) (Cited on pages 60, 61).

Kuznetsov, Vitaly and Zelda Mariet (2018). "Foundations of sequence-to-sequence modeling for time series." In: *arXiv preprint arXiv:1805.03714* (Cited on page 58).

Lai, Guokun, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu (Mar. 20, 2017). "Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks." In: arXiv: 1703.07015 [cs]. URL: http://arxiv.org/abs/1703.07015 (visited on 09/08/2018) (Cited on page 53).

– (2018). "Modeling long-and short-term temporal patterns with deep neural networks." In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 95–104 (Cited on page 88).

Laptev, Nikolay, Jason Yosinski, Li Erran Li, and Slawek Smyl (2017). "Time-Series Extreme Event Forecasting with Neural Networks at Uber." In: p. 5 (Cited on page 148).

Laptev, Nikolay, Jiafan Yu, and Ram Rajagopal (May 10, 2018). "DEEPCAST : UNIVERSAL TIME-SERIES FORECASTER." In: URL: https://openreview.net/forum?id=HJ1ZxXbAM (visited on 08/17/2018) (Cited on page 55).

Lara-Benítez, Pedro, Manuel Carranza-García, and José C. Riquelme (Mar. 2021). "An Experimental Review on Deep Learning Architectures for Time Series Forecasting." In: *International Journal of Neural Systems* 31.03, p. 2130001. ISSN: 0129-0657, 1793-6462. DOI: 10.1142/S0129065721300011. arXiv: 2103.12057. URL: http://arxiv.org/abs/2103.12057 (visited on 07/28/2021) (Cited on pages 51, 54, 147).

Lara-Benítez, Pedro, Luis Gallego-Ledesma, Manuel Carranza-García, and José M. Luna-Romera (2021). "Evaluation of the Transformer Architecture for Univariate Time Series Forecasting." In: *Advances in Artificial Intelligence*. Ed. by Enrique Alba, Gabriel Luque, Francisco Chicano, Carlos Cotta, David Camacho, Manuel Ojeda-Aciego, Susana Montes, Alicia Troncoso, José Riquelme, and Rodrigo Gil-Merino. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 106–115. ISBN: 978-3-030-85713-4. DOI: 10.1007/978-3-030-85713-4_11 (Cited on pages 54, 147).

Lee, Sangseok and Dong Kyu Lee (2018). "What is the proper way to apply the multiple comparison test?" In: *Korean journal of anesthesiology* 71.5, p. 353 (Cited on page 119).

Lian, X. and L. Chen (2009). "General Cost Models for Evaluating Dimensionality Reduction in High-Dimensional Spaces." In: *IEEE Transactions on Knowledge and Data Engineering* 21.10, pp. 1447–1460. DOI: 10.1109/TKDE.2008.170 (Cited on page 78).

Liaw, Andy, Matthew Wiener, et al. (2002). "Classification and regression by randomForest." In: *R news* 2.3, pp. 18–22 (Cited on page 84).

Lin, Winston T. (Jan. 1989). "Modeling and Forecasting Hospital Patient Movements: Univariate and Multiple Time Series Approaches." In: *International Journal of Forecasting* 5.2, pp. 195–208. ISSN: 0169-2070. DOI: 10.1016/0169-2070(89)90087-3. URL: https://www.sciencedirect.com/science/article/pii/0169207089900873 (visited on 10/10/2021) (Cited on page 49).

Lipton, Zachary C, John Berkowitz, and Charles Elkan (2015). "A critical review of recurrent neural networks for sequence learning." In: *arXiv preprint arXiv:1506.00019* (Cited on page 52).

Liu, Chen, Weiyan Hou, and Deyin Liu (Dec. 1, 2017). "Foreign Exchange Rates Forecasting with Convolutional Neural Network." In: *Neural Processing Letters* 46.3, pp. 1095–1119. ISSN: 1370-4621, 1573-773X. DOI: 10.1007/s11063-017-9629-z. URL: https://link.springer.com/article/10.1007/s11063-017-9629-z (visited on 08/17/2018) (Cited on page 52).

Liu, Fagui, Muqing Cai, Liangming Wang, and Yunsheng Lu (2019). "An Ensemble Model Based on Adaptive Noise Reducer and Over-Fitting Prevention LSTM for Multivariate Time Series Forecasting." In: *IEEE Access* 7, pp. 26102–26115. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2900371 (Cited on page 60).

Liu, Minhao, Ailing Zeng, Qiuxia Lai, and Qiang Xu (June 2021). "Time Series Is a Special Sequence: Forecasting with Sample Convolution and Interaction." In: *arXiv:2106.09305 [cs]*. arXiv: 2106.09305 [cs]. URL: http://arxiv.org/abs/2106.09305 (visited on 07/28/2021) (Cited on page 52).

Lütkepohl, Helmut (2005). *New Introduction to Multiple Time Series Analysis*. OCLC: ocm61028971. Berlin: New York : Springer. 764 pp. ISBN: 978-3-540-40172-8 (Cited on pages 3, 13, 49, 70).

MacKay, David JC (2003). *Information theory, inference and learning algorithms*. Cambridge university press (Cited on page 41).

Makridakis, S, E Spiliotis, and V Assimakopoulos (2020a). "The M5 accuracy competition: Results, findings and conclusions." In: *Int J Forecast* (Cited on pages 5, 71, 79, 148).

Makridakis, Spyros, Evangelos Spiliotis, and Vassilios Assimakopoulos (2018). "Statistical and Machine Learning forecasting methods: Concerns and ways forward." In: *PloS one* 13.3, e0194889 (Cited on page 118).

– (2020b). "The M4 Competition: 100,000 Time Series and 61 Forecasting Methods." In: *International Journal of Forecasting* 36.1, pp. 54–74. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2019.04.014. URL: https://www.sciencedirect.com/science/article/pii/S0169207019301128 (Cited on pages 4, 40, 66, 70, 79, 131, 148).

Markowitz, Harry (1952). "PORTFOLIO SELECTION*." In: *The Journal of Finance* 7.1, pp. 77–91. DOI: https://doi.org/10.1111/j.1540-6261.1952.tb01525.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.1952.tb01525.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1952.tb01525.x (Cited on pages 127, 128).

Maron, Oden and Andrew W. Moore (Feb. 1997). "The Racing Algorithm: Model Selection for Lazy Learners." In: *Artificial Intelligence Review* 11.1, pp. 193–225. ISSN: 1573-7462. DOI: 10.1023/A:1006556606079. URL: https://doi.org/10.1023/A:1006556606079 (visited on 09/09/2021) (Cited on page 20).

Matías, José M. (2005). "Multi-output Nonparametric Regression." In: *EPIA*, pp. 288–292 (Cited on page 25).

McMahan, H. Brendan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas (Feb. 2017). "Communication-Efficient Learning of Deep Networks from Decentralized Data." In: *arXiv:1602.05629 [cs]*. arXiv: 1602.05629 [cs]. URL: http://arxiv.org/abs/1602.05629 (visited on 11/06/2021) (Cited on page 149).

McNames, J. (1998). "A nearest trajectory strategy for time series prediction." In: *Proceedings of the InternationalWorkshop on Advanced Black-Box Techniques for Nonlinear Modeling*. K.U. Leuven. Belgium, pp. 112–128 (Cited on page 25).

Mendes-Moreira, Joao, Carlos Soares, Alípio Mário Jorge, and Jorge Freire De Sousa (2012). "Ensemble approaches for regression: A survey." In: *Acm computing surveys (csur)* 45.1, p. 10 (Cited on page 79).

Messner, Jakob W and Pierre Pinson (2019). "Online adaptive lasso estimation in vector autoregressive models for high dimensional wind power forecasting." In: *International Journal of Forecasting* 35.4, pp. 1485–1498 (Cited on pages 50, 116, 131, 132).

Meyer, Patrick E, Frederic Lafitte, and Gianluca Bontempi (2008). "minet: AR/Bioconductor package for inferring large transcriptional networks using mutual information." In: *BMC bioinformatics* 9.1, pp. 1–10 (Cited on page 17).

Micchelli, Charles A. and Massimiliano A. Pontil (2005). "On Learning Vector-Valued Functions." In: *Neural Comput.* 17.1, pp. 177–204. ISSN: 0899-7667. DOI: http://dx.doi.org/10.1162/0899766052530802 (Cited on page 25).

Mierswa, Ingo (2005). "Automatic Feature Extraction from Large Time Series." In: *Classification — the Ubiquitous Challenge*. Ed. by Claus Weihs and Wolfgang Gaul. Studies in Classification, Data Analysis, and Knowledge Organization. Berlin, Heidelberg: Springer, pp. 600–607. ISBN: 978-3-540-28084-2. DOI: 10.1007/3-540-28084-7_71 (Cited on page 28).

Mitchell, T. M. (1997). *Machine Learning*. McGraw Hill (Cited on page 13).

Mitliagkas, Ioannis, Constantine Caramanis, and Prateek Jain (2013). "Memory limited, streaming PCA." In: *Advances in Neural Information Processing Systems*, pp. 2886–2894 (Cited on page 57).

Mittnik, Stefan, Nikolay Robinzonov, and Martin Spindler (2015). "Stock market volatility: Identifying major drivers and the nature of their impact." In: *Journal of Banking & Finance* 58, pp. 1–14 (Cited on page 156).

Molnar, Christoph (n.d.). *Interpretable Machine Learning*. URL: https://christophm.github.io/interpretable-ml-book/ (visited on 10/27/2021) (Cited on page 21).

Moritz, Steffen and Thomas Bartz-Beielstein (2017). "imputeTS: Time Series Missing Value Imputation in R." In: *The R Journal* 9.1, p. 207. ISSN: 2073-4859. DOI: 10.32614/RJ-2017-009. URL: https://journal.r-project.org/archive/2017/RJ-2017-009/index.html (visited on 11/14/2021) (Cited on page 26).

NREL (2021). *Electricity Dataset*. URL: https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014 (Cited on page 88).

Nakagawa, Kei, Tomoki Ito, Masaya Abe, and Kiyoshi Izumi (2019). "Deep recurrent factor model: interpretable non-linear and time-varying multi-factor Model." In: *arXiv preprint arXiv:1901.11493* (Cited on page 96).

Newbold, P. and C. W. J. Granger (1974). "Experience with Forecasting Univariate Time Series and the Combination of Forecasts." en. In: *Journal of the Royal Statistical Society: Series A (General)* 137.2, pp. 131–146. ISSN: 2397-2327. DOI: 10.2307/2344546. URL: https://onlinelibrary.wiley.com/doi/abs/10.2307/2344546 (visited on 09/21/2021) (Cited on page 59).

Ning, C. and F. You (2019). "Data-Driven Adaptive Robust Unit Commitment Under Wind Power Uncertainty: A Bayesian Nonparametric Approach." In: *IEEE Transactions on Power Systems* 34.3, pp. 2409–2418. DOI: 10.1109/TPWRS.2019.2891057 (Cited on page 128).

Oja, Erkki (1992). "Principal components, minor components, and linear neural networks." In: *Neural networks* 5.6, pp. 927–935 (Cited on pages 56, 57).

Oliveira, Mariana and Luis Torgo (2015). "Ensembles for Time Series Forecasting." In: *Proceedings of the Sixth Asian Conference on Machine Learning*. Ed. by Dinh Phung and Hang Li. Vol. 39. Proceedings of Machine Learning Research. Nha Trang City, Vietnam: PMLR, pp. 360–370. URL: https://proceedings.mlr.press/v39/oliveira14.html (Cited on page 75).

Ozkan, Mehmet Baris and Pinar Karagoz (2015). "A novel wind power forecast model: Statistical hybrid wind power forecast technique (SHWIP)." In: *IEEE Transactions on Industrial Informatics* 11.2, pp. 375–387 (Cited on page 119).

Paldino, Gian Marco, Jacopo De Stefani, Fabrizio De Caro, and Gianluca Bontempi (2021). "Does AutoML Outperform Naive Forecasting?" In: *Engineering Proceedings* 5.1 (1), p. 36. DOI: 10.3390/engproc2021005036. URL: https://www.mdpi.com/2673-4591/5/1/36 (visited on 07/27/2021) (Cited on pages 10, 39, 97, 146).

Papadimitriou, Spiros, Jimeng Sun, and Christos Faloutsos (2005). "Streaming pattern discovery in multiple time-series." In: *Proceedings of the 31st international conference on Very large data bases*, pp. 697–708 (Cited on page 56).

Parkinson, Michael (1980). "The extreme value method for estimating the variance of the rate of return." In: *Journal of Business*, pp. 61–65 (Cited on page 157).

Parmezan, Antonio Rafael Sabino, Vinicius MA Souza, and Gustavo EAPA Batista (2019). "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." In: *Information sciences* 484, pp. 302–337 (Cited on page 84).

Paszke, Adam et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library." In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf (Cited on page 4).

Pathak, Reese, Rajat Sen, Nikhil Rao, N. Benjamin Erichson, Michael I. Jordan, and Inderjit S. Dhillon (Oct. 2021). "Cluster-and-Conquer: A Framework For Time-Series Forecasting." In: *arXiv:2110.14011 [cs, stat]*. arXiv: 2110.14011 [cs, stat]. URL: http://arxiv.org/abs/2110.14011 (visited on 11/02/2021) (Cited on page 148).

Pavlyshenko, Bohdan M. (Aug. 2020). "Using Bayesian Regression for Stacking Time Series Predictive Models." In: *2020 IEEE Third International Conference on Data Stream Mining*

Processing (DSMP), pp. 305–309. DOI: 10.1109/DSMP47368.2020.9204312 (Cited on page 61).

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12, pp. 2825–2830 (Cited on page 4).

Pereira, Dulce G, Anabela Afonso, and Fátima Melo Medeiros (2015). "Overview of Friedman's test and post-hoc analysis." In: *Communications in Statistics-Simulation and Computation* 44.10, pp. 2636–2653 (Cited on pages 37, 119).

Perez-Chacon, R., R. L. Talavera-Llames, F. Martinez-Alvarez, and A. Troncoso (2016). "Finding Electric Energy Consumption Patterns in Big Time Series Data." In: *Distributed Computing and Artificial Intelligence, 13th International Conference*. Ed. by Sigeru Omatu, Ali Semalat, Grzegorz Bocewicz, Paweł Sitek, Izabela E. Nielsen, Julián A. García García, and Javier Bajo. Cham: Springer International Publishing, pp. 231–238. ISBN: 978-3-319-40162-1. DOI: 10.1007/978-3-319-40162-1_25. URL: https://doi.org/10.1007/978-3-319-40162-1_25 (Cited on page 4).

Petropoulos, Fotios, Rob J. Hyndman, and Christoph Bergmeir (July 2018). "Exploring the Sources of Uncertainty: Why Does Bagging for Time Series Forecasting Work?" en. In: *European Journal of Operational Research* 268.2, pp. 545–554. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2018.01.045. URL: https://www.sciencedirect.com/science/article/pii/S037722171830081X (visited on 09/23/2021) (Cited on page 61).

Petropoulos, Fotios et al. (Oct. 2021). "Forecasting: Theory and Practice." In: *arXiv:2012.03854 [stat]*. arXiv: 2012.03854 [stat]. URL: http://arxiv.org/abs/2012.03854 (visited on 11/03/2021) (Cited on pages 13, 28).

Poon, Ser-Huang and Clive WJ Granger (2003). "Forecasting volatility in financial markets: A review." In: *Journal of economic literature* 41.2, pp. 478–539 (Cited on page 155).

Priebe, Florian (2019). "Dynamic Model Selection for Automated Machine Learning in Time Series." In: (Cited on page 60).

Rana, Mashud, Irena Koprinska, and Vassilios G. Agelidis (Aug. 2016). "Univariate and Multivariate Methods for Very Short-Term Solar Photovoltaic Power Forecasting." en. In: *Energy Conversion and Management* 121, pp. 380–390. ISSN: 0196-8904. DOI: 10.1016/j.enconman.2016.05.025. URL: https://www.sciencedirect.com/science/article/pii/S0196890416303934 (visited on 09/22/2021) (Cited on page 60).

Rego-Fernández, Diego, Verónica Bolón-Canedo, and Amparo Alonso-Betanzos (2014). "Scalability Analysis of mRMR for Microarray Data." In: *ICAART (1)*, pp. 380–386 (Cited on page 17).

Ren, Ye, P.N. Suganthan, and N. Srikanth (2015). "Ensemble methods for wind and solar power forecasting—A state-of-the-art review." In: *Renewable and Sustainable Energy Reviews* 50, pp. 82 –91. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2015.04.081. URL: http://www.sciencedirect.com/science/article/pii/S1364032115003512 (Cited on page 119).

Rockafellar, R Tyrrell and Stanislav Uryasev (2002). "Conditional value-at-risk for general loss distributions." In: *Journal of banking & finance* 26.7, pp. 1443–1471 (Cited on pages 127, 129, 130).

Rong, M., D. Gong, and X. Gao (2019). "Feature Selection and Its Use in Big Data: Challenges, Methods, and Trends." In: *IEEE Access* 7, pp. 19709–19725. DOI: 10.1109/ACCESS.2019.2894366 (Cited on page 78).

Rousseeuw, Peter J, Ida Ruts, and John W Tukey (1999). "The bagplot: a bivariate boxplot." In: *The American Statistician* 53.4, pp. 382–387 (Cited on pages 127, 128).

Ruder, Sebastian (June 15, 2017). "An Overview of Multi-Task Learning in Deep Neural Networks." In: arXiv: 1706.05098 [cs, stat]. URL: http://arxiv.org/abs/1706.05098 (visited on 08/28/2018) (Cited on page 55).

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1988). "Learning representations by back-propagating errors." In: *Cognitive modeling* 5.3, p. 1 (Cited on page 42).

Saad, E., D. Prokhorov, and D. Wunsch (1998). "Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks." In: *Neural Networks, IEEE Transactions on* 9.6, pp. 1456–1470. DOI: 10.1109/72.728395. URL: http://dx.doi.org/10.1109/72.728395 (Cited on page 25).

Saadallah, Amal, Florian Priebe, and Katharina Morik (Apr. 2020). "A Drift-Based Dynamic Ensemble Members Selection Using Clustering for Time Series Forecasting." In: pp. 678–694. ISBN: 978-3-030-46149-2. DOI: 10.1007/978-3-030-46150-8_40 (Cited on pages 60, 61).

Salinas, David, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski (July 2020). "DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks." In: *International Journal of Forecasting* 36.3, pp. 1181–1191. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2019.07.001 (Cited on page 148).

Sánchez, Ismael (Oct. 2008). "Adaptive Combination of Forecasts with Application to Wind Energy." In: *International Journal of Forecasting*. Energy Forecasting 24.4, pp. 679–693. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2008.08.008. URL: https://www.sciencedirect.com/science/article/pii/S0169207008001003 (visited on 09/22/2021) (Cited on page 61).

Sanger, Terence D (1989). "Optimal unsupervised learning in a single-layer linear feedforward neural network." In: *Neural networks* 2.6, pp. 459–473 (Cited on pages 56, 57).

Santamaría-Bonfil, Guillermo, Juan Frausto-Solís, and Ignacio Vázquez-Rodarte (2015). "Volatility forecasting using support vector regression and a hybrid genetic algorithm." In: *Computational Economics* 45.1, pp. 111–133 (Cited on pages 155, 156).

Sapankevych, Nicholas I and Ravi Sankar (2009). "Time series prediction using support vector machines: a survey." In: *IEEE Computational Intelligence Magazine* 4.2 (Cited on page 46).

Schapire, Robert E (1990). "The strength of weak learnability." In: *Machine learning* 5.2, pp. 197–227 (Cited on page 46).

Schölkopf, Bernhard, Alexander Smola, and Klaus-Robert Müller (1998). "Nonlinear component analysis as a kernel eigenvalue problem." In: *Neural computation* 10.5, pp. 1299–1319 (Cited on page 71).

Segal, Mark R (2004). "Machine learning benchmarks and random forest regression." In: (Cited on page 47).

Sen, Rajat, Hsiang-Fu Yu, and Inderjit Dhillon (2019). "Think Globally, Act Locally: A Deep Neural Network Approach to High-Dimensional Time Series Forecasting." In: *arXiv preprint arXiv:1905.03806*, p. 5 (Cited on page 69).

Sezer, Omer Berat and Ahmet Murat Ozbayoglu (Sept. 1, 2018). "Algorithmic Financial Trading with Deep Convolutional Neural Networks: Time Series to Image Conversion Approach." In: *Applied Soft Computing* 70, pp. 525–538. ISSN: 1568-4946. DOI: 10.1016/j.asoc.2018.04.024. URL: http://www.sciencedirect.com/science/article/pii/S1568494618302151 (visited on 08/17/2018) (Cited on page 52).

Shah, Vedant and Gautam Shroff (Oct. 2021). "Forecasting Market Prices Using DL with Data Augmentation and Meta-Learning: ARIMA Still Wins!" In: *arXiv:2110.10233 [cs]*. arXiv: 2110.10233 [cs]. URL: http://arxiv.org/abs/2110.10233 (visited on 10/26/2021) (Cited on page 146).

Smyl, Slawek (2020). "A Hybrid Method of Exponential Smoothing and Recurrent Neural Networks for Time Series Forecasting." In: *International Journal of Forecasting* 36.1, pp. 75–85. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2019.03.017. URL: https://www.sciencedirect.com/science/article/pii/S0169207019301153 (Cited on pages 5, 69).

Soares, Tiago, Pierre Pinson, Tue V Jensen, and Hugo Morais (2016). "Optimal offering strategies for wind power in energy and primary reserve markets." In: *IEEE Transactions on Sustainable Energy* 7.3, pp. 1036–1045 (Cited on page 119).

Sorjamaa, Antti, Jin Hao, Nima Reyhani, Yongnan Ji, and Amaury Lendasse (2007). "Methodology for long-term prediction of time series." In: *Neurocompuing* 70.16-18, pp. 2861–2869. ISSN: 0925-2312. DOI: http://dx.doi.org/10.1016/j.neucom.2006.06.015 (Cited on page 25).

Stock, J.H. and M.W. Watson (2002). "Forecasting using principal components from a large number of predictors." In: *Journal of the American Statistical Association* 97.460, pp. 1167–1179. ISSN: 0925-2312 (Cited on page 70).

– (2010). "Dynamic Factor Models." In: *Oxford Handbook of Economic Forecasting*. Ed. by M.P. Clements and D.F. Hendry. Oxford University Press (Cited on pages 69, 70).

Susik, Robert (2020). "Recurrent autoencoder with sequence-aware encoding." In: *arXiv preprint arXiv:2009.07349* (Cited on page 58).

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le (2014). "Sequence to sequence learning with neural networks." In: *arXiv preprint arXiv:1409.3215* (Cited on pages 58, 71).

Taieb, Souhaib Ben (2014). "Machine learning strategies for multi-step-ahead time series forecasting." PhD thesis. Ph. D. Thesis (Cited on pages 47, 97).

Taieb, Souhaib Ben, Gianluca Bontempi, Amir F Atiya, and Antti Sorjamaa (2012). "A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition." In: *Expert systems with applications* 39.8, pp. 7067–7083 (Cited on pages 75, 82, 85).

Taieb, Souhaib Ben, James W Taylor, and Rob J Hyndman (2017a). "Coherent Probabilistic Forecasts for Hierarchical Time Series." In: p. 10 (Cited on page 148).

– (2017b). "Coherent probabilistic forecasts for hierarchical time series." In: *International Conference on Machine Learning*. PMLR, pp. 3348–3357 (Cited on page 68).

Talavera-Llames, R, Rubén Pérez-Chacón, Alicia Troncoso, and Francisco Martínez-Álvarez (2019). "MV-kWNN: A novel multivariate and multi-output weighted nearest neighbours algorithm for big data time series forecasting." In: *Neurocomputing* 353, pp. 56–73 (Cited on page 144).

Tantithamthavorn, Chakkrit, Ahmed E. Hassan, and Kenichi Matsumoto (Jan. 2018). "The Impact of Class Rebalancing Techniques on the Performance and Interpretation of Defect Prediction Models." In: *arXiv:1801.10269 [cs]*. arXiv: 1801.10269 [cs]. URL: http://arxiv.org/abs/1801.10269 (visited on 11/07/2021) (Cited on page 15).

Tashman, Leonard J. (2000). "Out-of-Sample Tests of Forecasting Accuracy: An Analysis and Review." In: *International Journal of Forecasting* 16.4, pp. 437 –450. ISSN: 0169-2070. DOI: https://doi.org/10.1016/S0169-2070(00)00065-0. URL: http://www.sciencedirect.com/science/article/pii/S0169207000000650 (Cited on pages 29, 33, 34, 93, 100, 106, 107, 119, 121, 132).

Terasvirta, Timo, Dag Tjostheim, and Clive W. J. Granger (2010). *Modelling Nonlinear Economic Time Series*. Oxford University Press. URL: https://ideas.repec.org/b/oxp/obooks/9780199587155.html (visited on 11/04/2021) (Cited on page 13).

Thompson, Neil C., Kristjan Greenewald, Keeheon Lee, and Gabriel F. Manso (July 2020). "The Computational Limits of Deep Learning." In: *arXiv:2007.05558 [cs, stat]*. arXiv: 2007.05558 [cs, stat]. URL: http://arxiv.org/abs/2007.05558 (visited on 11/06/2021) (Cited on page 148).

Thrun, Sebastian and Lorien Pratt (1998). "Learning to Learn: Introduction and Overview." In: *Learning to Learn*. Ed. by Sebastian Thrun and Lorien Pratt. Boston, MA: Springer US, pp. 3–17. ISBN: 978-1-4615-5529-2. DOI: 10.1007/978-1-4615-5529-2_1. URL: https://doi.org/10.1007/978-1-4615-5529-2_1 (visited on 11/11/2021) (Cited on page 19).

Tibshirani, Robert (1996). "Regression shrinkage and selection via the lasso." In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1, pp. 267–288 (Cited on pages 21, 50).

Tong, H. (1983). *Threshold models in Nonlinear Time Series Analysis*. Berlin: Springer Verlag (Cited on page 25).

Torres, José F., Dalil Hadjout, Abderrazak Sebaa, Francisco Martínez-Álvarez, and Alicia Troncoso (Feb. 2021). "Deep Learning for Time Series Forecasting: A Survey." In: *Big Data* 9.1, pp. 3–21. ISSN: 2167-6461. DOI: 10.1089/big.2020.0159. URL: https://www.liebertpub.com/doi/10.1089/big.2020.0159 (visited on 10/26/2021) (Cited on pages 5, 144).

Tsay, R. S. (2014). *Multivariate Time Series Analysis with R and Financial Applications*. Wiley (Cited on pages 7, 13, 56, 106).

Tsay, Ruey S (2000). "Time series and forecasting: Brief history and future research." In: *Journal of the American Statistical Association* 95.450, pp. 638–643 (Cited on page 3).

Tuarob, Suppawong, Conrad S. Tucker, Soundar Kumara, C. Lee Giles, Aaron L. Pincus, David E. Conroy, and Nilam Ram (Apr. 1, 2017). "How Are You Feeling?: A Personalized Methodology for Predicting Mental States from Temporally Observable Physical and Behavioral Information." In: *Journal of Biomedical Informatics* 68, pp. 1–19. ISSN: 1532-0464. DOI: 10.1016/j.jbi.2017.02.010. URL: http://www.sciencedirect.com/science/article/pii/S1532046417300357 (visited on 09/03/2018) (Cited on page 48).

Turner Lee, Nicol (Jan. 2018). "Detecting Racial Bias in Algorithms and Machine Learning." In: *Journal of Information, Communication and Ethics in Society* 16.3, pp. 252–260. ISSN: 1477-996X. DOI: 10.1108/JICES-06-2018-0056. URL: https://doi.org/10.1108/JICES-06-2018-0056 (visited on 11/07/2021) (Cited on page 15).

Van Der Maaten, Laurens, Eric Postma, and Jaap Van den Herik (2009). "Dimensionality reduction: a comparative." In: *J Mach Learn Res* 10.66-71, p. 13 (Cited on pages 55, 71).

Vapnik, Vladimir (1999). *The nature of statistical learning theory*. Springer science & business media (Cited on page 19).

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin (Dec. 2017). "Attention Is All You Need." In: *arXiv:1706.03762 [cs]*. arXiv: 1706.03762 [cs]. URL: http://arxiv.org/abs/1706.03762 (visited on 11/26/2021) (Cited on page 54).

Vincent, Pascal, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol (2010). "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." In: *Journal of Machine Learning Research* 11.Dec, pp. 3371–3408 (Cited on page 57).

Wang, G., K.-S. Choi, J. Y.-C. Teoh, and J. Lu (2020). "Deep Cross-Output Knowledge Transfer Using Stacked-Structure Least-Squares Support Vector Machines." In: *IEEE Transactions on Cybernetics*, pp. 1–14. ISSN: 2168-2275. DOI: 10.1109/TCYB.2020.3008963 (Cited on page 144).

Wang, Miss Lei (Dec. 2018). "Advanced Multivariate Time Series Forecasting Models." en-US. In: *Journal of Mathematics and Statistics* 14.1, pp. 253–260. ISSN: 1558-6359. DOI: 10.3844/jmssp.2018.253.260. URL: https://thescipub.com/abstract/jmssp.2018.253.260 (visited on 09/29/2021) (Cited on page 85).

Wang, Zheng, Irena Koprinska, Alicia Troncoso, and Francisco Martínez-Álvarez (July 2018). "Static and Dynamic Ensembles of Neural Networks for Solar Power Forecasting." In: *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. DOI: 10.1109/IJCNN.2018.8489231 (Cited on pages 60, 61).

Weigend, A.S. and N.A. Gershenfeld (1994). *Time Series Prediction: forecasting the future and understanding the past*. Harlow, UK: Addison Wesley (Cited on page 25).

Weng, Juyang, Yilu Zhang, and Wey-Shiuan Hwang (2003). "Candid covariance-free incremental principal component analysis." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25.8, pp. 1034–1040 (Cited on pages 56, 57).

Wickramasuriya, Shanika L, George Athanasopoulos, Rob J Hyndman, et al. (2015). "Forecasting hierarchical and grouped time series through trace minimization." In: *Department of Econometrics and Business Statistics, Monash University* 105 (Cited on page 68).

Wolpert, David H. (2002). "The Supervised Learning No-Free-Lunch Theorems." In: *Soft Computing and Industry: Recent Applications*. Ed. by Rajkumar Roy, Mario Köppen, Seppo Ovaska, Takeshi Furuhashi, and Frank Hoffmann. London: Springer, pp. 25–42. ISBN: 978-1-4471-0123-9. DOI: 10.1007/978-1-4471-0123-9_3. URL: https://doi.org/10.1007/978-1-4471-0123-9_3 (visited on 11/11/2021) (Cited on page 19).

Würth, Ines, Laura Valldecabres, Elliot Simon, Corinna Möhrlen, Bahri Uzunoğlu, Ciaran Gilbert, Gregor Giebel, David Schlipf, and Anton Kaifel (2019). "Minute-scale forecasting of wind power—results from the collaborative workshop of IEA Wind task 32 and 36." In: *Energies* 12.4, p. 712 (Cited on page 119).

Xiong, Ruoxuan, Eric P Nichols, and Yuan Shen (2015). "Deep Learning Stock Volatility with Google Domestic Trends." In: *arXiv preprint arXiv:1512.04916* (Cited on page 158).

Yan, Jie, Yongqian Liu, Shuang Han, Yimei Wang, and Shuanglei Feng (2015). "Reviews on uncertainty analysis of wind power forecasting." In: *Renewable and Sustainable Energy Reviews* 52, pp. 1322–1330 (Cited on page 125).

Yang, Yang, Guillaume Sautiere, J Jon Ryu, and Taco S Cohen (2020). "Feedback recurrent autoencoder." In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 3347–3351 (Cited on page 58).

Yu, Fisher and Vladlen Koltun (Nov. 23, 2015). "Multi-Scale Context Aggregation by Dilated Convolutions." In: arXiv: 1511.07122 [cs]. URL: http://arxiv.org/abs/1511.07122 (visited on 09/11/2018) (Cited on page 52).

Zhang, Bing, Jhen-Long Wu, and Pei-Chann Chang (June 1, 2018). "A Multiple Time Series-Based Recurrent Neural Network for Short-Term Load Forecasting." In: *Soft Computing* 22.12, pp. 4099–4112. ISSN: 1432-7643, 1433-7479. DOI: 10.1007/s00500-017-2624-5. URL: https://link.springer.com/article/10.1007/s00500-017-2624-5 (visited on 08/17/2018) (Cited on page 54).

Zhang, Guoqiang, B. Eddy Patuwo, and Michael Y. Hu (Mar. 1998). "Forecasting with Artificial Neural Networks:" in: *International Journal of Forecasting* 14.1, pp. 35–62. ISSN: 01692070. DOI: 10.1016/S0169-2070(97)00044-7. URL: http://linkinghub.elsevier.com/retrieve/pii/S0169207097000447 (visited on 09/11/2018) (Cited on page 42).

Zheng, Alice and Amanda Casari (2018). *Feature engineering for machine learning: principles and techniques for data scientists*. " O'Reilly Media, Inc." (Cited on page 17).

COLOPHON

This thesis was typeset using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*".

Hermann Zapf's *Palatino* and *Euler* type faces (Type 1 PostScript fonts *URWPalladio L* and *FPL*) are used. The "typewriter" text is typeset in *Bera Mono*, originally developed by Bitstream, Inc. as "Bitstream Vera". (Type 1 PostScript fonts were made available by Malte Rosenau and Ulrich Dirr.)

This thesis has been typeset using LaTeX. On a magnetic HDD, with an approximate storage density of $750[\frac{Gb}{in^2}]$ and a magnetic layer height of $10[nm]$, the approximate weight of a GB is $0.612471[\frac{\mu g}{GB}]$, resulting in a total weight for this thesis of $\frac{20}{1024}[GB] \times 0.612471[\frac{\mu g}{GB}] = 0.0119623[\mu g]$.

Hoc opus est scriptum
Magister da mihi potum;
Dextera scriptoris careat grauitate doloris.

– VLF 5, f. 172v,