Machine learning strategies for multi-step-ahead time series forecasting

A thesis submitted for the degree of

Doctor of Philosophy

by

Souhaib Ben Taieb

Département d'Informatique
Université Libre de Bruxelles
Belgium

September 2014

Declaration

This thesis has been written under the supervision of Prof. Gianluca Bontempi (Université Libre de Bruxelles, Belgium) and Prof. Rob J. Hyndman (Monash University, Australia). The members of the Jury are:

- Prof. Gianluca Bontempi (Université Libre de Bruxelles, Belgium)
- Prof. Yves Desmet (Université Libre de Bruxelles, Belgium)
- Prof. Rob J. Hyndman (Monash University, Australia)
- Prof. Maarten Jansen (Université Libre de Bruxelles, Belgium)
- Prof. Tom Lenaerts (Université Libre de Bruxelles, Belgium)
- Prof. Timo Teräsvirta (Aarhus University, Denmark)
- Prof. Michel Verleysen (Université Catholique de Louvain, 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.

Souhaib Ben Taieb

To my son Harone and my future children

It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts. —Arthur Conan Doyle, *Sherlock Holmes*.

The scientist does not study nature because it is useful; he studies it because he delights in it, and he delights in it because it is beautiful. —Henri Poincaré.

I was like a boy playing on the sea-shore, and diverting myself now and then finding a smoother pebble or a prettier shell than ordinary, whilst the great ocean of truth lay all undiscovered before me. —Isaac Newton.

I see a pattern, but my imagination cannot picture the maker of that pattern. I see a clock, but I cannot envision the clockmaker. The human mind is unable to conceive of the four dimensions, so how can it conceive of a God, before whom a thousand years and a thousand dimensions are as one? —Albert Einstein.

Abstract

How much electricity is going to be consumed for the next 24 hours? What will be the temperature for the next three days? What will be the number of sales of a certain product for the next few months? Answering these questions often requires forecasting several future observations from a given sequence of historical observations, called a time series.

Historically, time series forecasting has been mainly studied in econometrics and statistics. In the last two decades, machine learning, a field that is concerned with the development of algorithms that can automatically learn from data, has become one of the most active areas of predictive modeling research. This success is largely due to the superior performance of machine learning prediction algorithms in many different applications as diverse as natural language processing, speech recognition and spam detection. However, there has been very little research at the intersection of time series forecasting and machine learning.

The goal of this dissertation is to narrow this gap by addressing the problem of multi-step-ahead time series forecasting from the perspective of machine learning. To that end, we propose a series of forecasting strategies based on machine learning algorithms.

Multi-step-ahead forecasts can be produced recursively by iterating a one-step-ahead model, or directly using a specific model for each horizon. As a first contribution, we conduct an in-depth study to compare recursive and direct forecasts generated with different learning algorithms for different data generating processes. More precisely, we decompose the multi-step mean squared forecast errors into the bias and variance components, and analyze their behavior over the forecast horizon for different time series lengths. The results and observations made in this study then guide us for the development of new forecasting strategies.

In particular, we find that choosing between recursive and direct forecasts is not an easy task since it involves a trade-off between bias and estimation variance that depends on many interacting factors, including the learning model, the underlying data generating process, the time series length and the forecast horizon. As a second contribution, we develop multi-stage forecasting strategies that do not treat the recursive and direct strategies as competitors, but seek to combine their best properties. More precisely, the multi-stage strategies generate recursive linear forecasts, and then adjust these forecasts by modeling the multi-step forecast residuals with direct nonlinear models at each horizon, called rectification models. We propose a first multi-stage strategy, that we called the rectify strategy, which estimates the rectification models using the nearest neighbors model. However, because recursive linear forecasts often need small adjustments with real-world time series, we also consider a second multi-stage strategy, called the boost strategy, that estimates the rectification models using gradient boosting algorithms that use so-called weak learners.

Generating multi-step forecasts using a different model at each horizon provides a large modeling flexibility. However, selecting these models independently can lead to irregularities in the forecasts that can contribute to increase the forecast variance. The problem is exacerbated with nonlinear machine learning models estimated from short time series. To address this issue, and as a third

contribution, we introduce and analyze multi-horizon forecasting strategies that exploit the information contained in other horizons when learning the model for each horizon. In particular, to select the lag order and the hyperparameters of each model, multi-horizon strategies minimize forecast errors over multiple horizons rather than just the horizon of interest.

We compare all the proposed strategies with both the recursive and direct strategies. We first apply a bias and variance study, then we evaluate the different strategies using real-world time series from two past forecasting competitions. For the rectify strategy, in addition to avoiding the choice between recursive and direct forecasts, the results demonstrate that it has better, or at least has close performance to, the best of the recursive and direct forecasts in different settings. For the multi-horizon strategies, the results emphasize the decrease in variance compared to single-horizon strategies, especially with linear or weakly nonlinear data generating processes. Overall, we found that the accuracy of multi-step-ahead forecasts based on machine learning algorithms can be significantly improved if an appropriate forecasting strategy is used to select the model parameters and to generate the forecasts.

Lastly, as a fourth contribution, we have participated in the Load Forecasting track of the Global Energy Forecasting Competition 2012. The competition involved a hierarchical load forecasting problem where we were required to backcast and forecast hourly loads for a US utility with twenty geographical zones. Our team, TinTin, ranked fifth out of 105 participating teams, and we have been awarded an IEEE Power & Energy Society award.

Acknowledgement

Les cinq années de ma thèse de doctorat ont été les plus productives tant au niveau professionnel qu'au niveau personnel, et ceci notamment grâce au soutien, à la patience et à la disponibilité d'un grand nombre de personnes de mon entourage que je tiens à remercier.

Tout d'abord, ma famille, et tout particulièrement ma mère et mon père, à qui je dois tout (et plus encore). Je tiens aussi à remercier mes frères et soeurs, qui chacun à leur manière m'ont aussi beaucoup donné.

Durant ces cinq dernières années, j'ai dû souvent m'absenter pour participer à plusieurs conférences à l'étranger. Je tiens particulièrement à remercier mon épouse pour son encouragement et sa patience lors de tous ces déplacements.

Il y a presque deux ans, notre petit Harone s'est joint à nous, et il a très bien choisi sa date de naissance, en la retardant d'une semaine, jusqu'au dernier jour de la compétition "GEFCOM2012" à laquelle je participais. Merci Harone de m'avoir permis de terminer la compétition, mais aussi pour toute la joie et la bonne humeur que tu m'as procurée ces deux dernières années, en particulier durant les moments les plus difficiles de ma thèse de doctorat.

J'ai une pensée particulière à tous les membres de ma famille que j'ai eu très peu l'occasion de rencontrer durant ces dernières années, et que j'aimerais aussi remercier pour leur encouragement.

Enfin, les moments de joie et de bonheur que j'ai passé avec mes amis m'ont souvent permis d'oublier les périodes difficiles. Je pense particulièrement à notre groupe, incluant Tarik, Jamel, Ayoube, Abdelrahman et Ibrahim ; mais aussi à Ahmed, Abdeslam, Gabriel et Hadrien. Le tour du Mont Blanc effectué avec Gabriel et Hadrien, ainsi que le voyage à Marrakech avec mon frère Ayoub et mon cousin Ahmed resteront des moments mémorables. Merci à tous !

After my family and friends, I would like to thank those who contributed more directly to my thesis.

My foremost gratitude goes to Gianluca Bontempi for his great advice and support throughout the entire PhD journey. I am particularly grateful to Gianluca for his willingness and patience to give me complete freedom to choose my research topics. His insightful understanding of machine learning has allowed me to progress a lot, and I thank him for all he taught me about that subject.

I am deeply grateful to Rob Hyndman who joined this thesis as co-supervisor, two years ago. Rob is one of the best forecasters in the world, and I have been very fortunate to work with him. I would like to express my deep gratitude to Rob for all the support and encouragement he provided to me, especially, his patience in answering all my questions during very late or very early Skype talks[1] that we had every week for the last two years. As a result, our participation to the GEFCOM2012 competition led to excellent results.

[1] Due to the time zone difference between Brussels and Melbourne.

I also thank Rob for the great time we had while traveling to Paris, Seoul and Rome, discussing about research and life for many hours. I also deeply appreciate Rob visiting me two times in Brussels even he was very tired of all his previous trips.

I would like also to thank Rob for welcoming me at Monash University in Melbourne for five months where I had the opportunity to work closely with him and to meet great people, including Georges, Farshid, Anastasios, Julia, Slava, Mehmet, Alexander, Behrooz, Gary, and many others. In particular, playing soccer with the red and blue teams was a very enjoyable experience.

More than a co-supervisor, I consider Rob as a close friend who is not only a brilliant researcher, but also a person with an extremely nice and sincere personality.

Research is a team effort and I was fortunate enough to work with many collaborators. In particular, I would like to thank Amir Atiya from Cairo University who accepted at a very early stage of the thesis to provide me guidance and support. I also thank Amir for taking the time to visit me in Brussels, last year. Giorgio Corani invited me at IDSIA in Lugano for one week, which was a very exciting experience and I am very grateful for that.

I thank all my colleagues of the Machine Learning Group and the Department of Computer Science that contributed to create an enjoyable daily atmosphere. Many thanks to all of you: Angélique, Ahmed, Alessia, Amine, Andrea, Benjamin, Bernard, Boris, Catharina, Claudio, Elisa, Emmanuel, Gérald, Gianluca, Gilles, Gwënael, Ioannis, Jean, Jean-François, Jean-Sébastien, Joël, Liran, Luciano, Martin, Maarten, Maryka, Matteo, Miguel, Nikita, Naim, Olivier, Pascaline, Patrick, Stephane, The Anh Han, Thierry, Tom, Vandy, Véronique, Yann-Aël, Yves, ...

I would like to thank all the people that have taken the time to read this thesis and that gave me feedback, including Gianluca, Rob, Catharina, Nikita, Liran and Slava. A special thanks has to be addressed to Catharina who accepted many times to give me feedback on my papers and applications. In particular, she accepted to read this thesis while she was on holidays. Thanks Catharina! I am also very grateful to Prof. Raymond Devillers who has proofread the preliminary version of this thesis in great details, even though he was on holidays.

Finally, my love for science and education was born in large part thanks to my excellent mathematics teacher in secondary school, Didier Pauwels. I would like to thank Didier for his enthusiasm and the time he takes to provide an excellent education for his students.

On this day, 19th of August 2014, my birthday, I am grateful for being able to do what I love to do.

Financial Support: the work presented in this thesis was supported by the Belgian National Funds for Scientific Research (FNRS) through a Research Fellow grant.

Contents

Chapter 1

Introduction

1.1 Time series forecasting and machine learning

Forecasts guide decisions in many areas of scientific, industrial and economic activities such as meteorology, telecommunication and finance. In many real-life scenarios, the forecaster encounters a multi-step-ahead forecasting problem where forecasts are required for short, medium or long horizons. In particular, multi-step-ahead forecasting of a univariate time series consists in predicting several future observations of a given sequence of historical observations. This thesis aims to address the problem of multi-step-ahead time series forecasting from the perspective of machine learning, a field that is concerned with the development of algorithms that can automatically learn from data (Abu-Mostafa, Magdon-Ismail, and Lin, 2012; Hastie, Tibshirani, and Friedman, 2009).

Time series forecasting has been widely studied in statistics (Hyndman and Athanasopoulos, 2014; Brockwell and Davis, 2002; Chatfield, 2000; Box and Jenkins, 1976) and econometrics (Greene, 2012; Wooldridge, 2012). It has also been influenced for a long time, by linear statistical models such as ARIMA models (Gooijer and Hyndman, 2006). However, since time series from real world phenomena typically behave nonlinearly (Kantz and Schreiber, 2004), nonlinear time series models were proposed such as the bilinear model (Poskitt and Tremayne, 1986) and the threshold autoregressive model (Tong and Lim, 1980; Tong, 1990). However, the study of nonlinear time series analysis and forecasting is still in its infancy compared to the development of linear time series (Gooijer and Hyndman, 2006; Fan and Yao, 2003; Teräsvirta, Tjostheim, and Granger, 2010).

In the last two decades, machine learning algorithms have drawn attention and have established themselves as serious contenders to classical statistical models in many different fields and applications including bioinformatics, natural language processing and speech recognition (Rudin and Wagstaff, 2014). Machine learning models, also called black-box or data-driven models (Mitchell, 1997), are examples of nonlinear and nonparametric models that make few assumptions about the underlying data generating process, and only use historical data to learn the stochastic dependency between a set of input and output variables. The superior performance of machine learning algorithms has been confirmed in many predictive modeling competitions where they are often ranked as top entries.

Although algorithms and methodologies from machine learning have proven to be very effective in many different fields, there has been very little research at the intersection of time series forecasting and machine learning. A notable exception is the use of the neural network algorithm for multi-step-ahead forecasting (Zhang, Patuwo, and Michael Y., 1998; Palit and Popovic, 2005; Kock and Teräsvirta, 2011; Crone, Hibon, and Nikolopoulos, 2011; Zhang, 2012). The lack of research in that area was the main motivation for a number of forecasting competitions, that aim at comparing the accuracy of machine learning algorithms with standard statistical forecasting

methods under different data conditions (e.g. the Santa Fe, NN3, NN5, and the annual ESTSP competitions Weigend and Gershenfeld, 1994; Crone, Hibon, and Nikolopoulos, 2011; Lendasse, Honkela, and Simula, 2010).

The considered machine learning algorithms and methodologies have often provided poor forecasts compared to simple statistical forecasting methods (Makridakis and Hibon, 2000). However, a large number of machine learning algorithms have not yet been considered in the forecasting literature. Also, many studies have often involved a limited amount of time series when comparing machine learning algorithms with traditional forecasting methods. In consequence, the results cannot be conclusive and recent debates about the process of mining the past to determine the future (see Hand, 2009a; Price, 2009; Crone, 2009a) confirm the importance of investigating further the role of data mining and machine learning in time series forecasting. Furthermore, Varian (2014) pointed out the lack of research at the intersection of econometrics and machine learning, and Wasserman (2014) use a similar argument for the general field of statistics. In consequence, a lot remains to be done to investigate further the role and potential benefits of machine learning in dealing with time series forecasting for the ever increasing amount of data.

1.2 Motivations and aims

The primary goal of this thesis is to contribute to the study and development of multi-step-ahead forecasting strategies based on machine learning algorithms. Traditionally, multi-step-ahead forecasting has been handled recursively, where a single time series model is estimated and each forecast is computed using previous forecasts. Another approach builds a separate time series model for each horizon, and forecasts are computed directly by the estimated model (Chevillon, 2007; Sorjamaa, Hao, Reyhani, et al., 2007; Kock and Teräsvirta, 2011).

In the forecasting literature, the recursive and direct strategies have often been considered with linear statistical models, with few research works involving nonlinear models and even less attention paid to machine learning algorithms. Also, the few in-depth studies that have considered machine learning algorithms have often either considered only a neural network algorithm or have been limited to one-step-ahead forecasting (see Berardi and Zhang (2003); Ahmed et al. (2010)). Forecasting multi-step-ahead, rather than one-step-ahead, is more prevalent in the majority of applications. In spite of this, it has been much less studied, partly because it is a more challenging problem. In fact, further steps ahead are typically harder to forecast, and also there is a potentially complex dependence between the observations at different steps ahead. Furthermore, one of the main assumptions of many machine learning algorithms is that the data set is composed of a set of independent observations, while one of the main features of a time series is the time dependence between the different observations.

This thesis will address these different issues by developing and comparing different multi-step forecasting strategies based on machine learning algorithms under different conditions, including time series length, forecast horizons and data generating processes. The different strategies will be compared from the perspective of the bias and variance components of the mean squared forecast errors.

Analyzing the behavior of bias and variance is paramount to understanding the different sources of errors and inner mechanics of the strategies over the forecast horizon. These are fundamental concepts that reveal the relation with model complexity, model misspecification, and data adequacy. An in-depth study could therefore give insights into the different interactions between time series length, model complexity, forecast horizon (i.e. number of steps ahead), and the strategy's performance. Finally, the different strategies will also be evaluated on real-world time series from two different forecasting competitions.

Multi-step forecasts can be produced recursively by iterating a one-step model, or directly using a specific model for each horizon. In practice, should the forecaster use the recursive or the direct strategy? The best between these two strategies involves a trade-off between bias and estimation variance that depends on many interacting factors, including the model complexity, the underlying data generating process, the time series length and the forecast horizon. A good trade-off can be difficult to achieve in practice with a limited amount of data. Using machine learning algorithms makes the trade-off even more difficult. So, choosing between the recursive and direct strategies is not an easy task in real-world forecasting problems.

This thesis will address this issue by proposing multi-stage forecasting strategies that do not consider the recursive and direct strategies as competitors, but seek to combine their best properties. The main idea is to generate recursive linear forecasts, and then adjust these forecasts by modeling the multi-step forecast residuals with direct nonlinear models at each horizon, called rectification models.

We will propose a first multi-stage strategy, called the rectify strategy, that estimates the rectification models with the nearest neighbors model. Because recursive linear forecasts often need small adjustments with real-world time series, we will also consider a second multi-stage strategy, called the boost strategy, that estimates the rectification models using gradient boosting algorithms that involve the so-called weak learners.

Recursive forecasts with nonlinear models can suffer from the amplification of errors over the horizon. Also, minimizing one-step-ahead forecast errors does not guarantee the minimum for multi-step-ahead errors with nonlinear models, and as a result, the recursive strategy is asymptotically biased. In consequence, the direct strategy is often adopted to generate multi-step forecasts with machine learning algorithms.

However, the direct strategy has a problem in generating forecasts from potentially very different models at different forecast horizons. In fact, because every model is selected independently at each horizon, it is possible for consecutive forecasts to be based on different conditioning information and different model forms. This can lead to irregularities in the forecast function. These irregularities are manifest as a contribution to the forecast variance. The problem is exacerbated when each of the models is allowed to be nonlinear and nonparametrically estimated, as with machine learning models.

This thesis will address this issue by proposing multi-horizon forecasting strategies that take advantage of the information contained in other forecast horizons to select the lag order and the hyperparameters of each model. In particular, multi-horizon strategies minimize forecast errors over multiple horizons rather than just over the horizon of interest.

There is often a gap between the theory and the practice of forecasting. Although forecasting theory is of paramount importance, real-world forecasting problems often involve tasks that are typically neglected in theoretical studies. For example, a typical forecasting problem will require a data preprocessing step which includes data cleansing and feature engineering as well as finding a good balance between computational complexity and statistical accuracy.

To experiment on a real-world forecasting problem in a real competition setting, we took part in the Load Forecasting track of the Global Energy Forecasting Competition 2012. The competition involved a hierarchical load forecasting problem for a US utility with 20 geographical zones. The available data consisted of the hourly loads for the 20 zones and hourly temperatures from 11 weather stations, for four and a half years. We were required to backcast and forecast hourly loads for 21 time series. Our team, TinTin, ranked fifth out of 105 participating teams, and we have been awarded an IEEE Power & Energy Society award.

1.3 Contributions

1.3.1 Publications and conferences

In the following, the related publications and conferences are given for each chapter.

- Chapter 4: Bias and variance analysis for multi-step forecasting
  - S Ben Taieb and AF Atiya (2014). "A bias and variance analysis for multi-step time series forecasting." Submitted to IEEE Transactions on Neural Networks and Learning Systems (under revision)
  - Oral presentation at the *26st European Conference on Operational Research (EURO)*, Rome, Italy, July 1-4, 2013.
- Chapter 5: Multi-stage forecasting strategies
  - S Ben Taieb and RJ Hyndman (2014a). Boosting multi-step autoregressive forecasts. In: *Proceedings of the 31th International Conference on Machine Learning (ICML)*, pp.109–117
  - S Ben Taieb and RJ Hyndman (2014b). "Recursive and direct multi-step forecasting: the best of both worlds." Submitted to International Journal of Forecasting (under revision)
  - Oral presentation at the *International Conference on Machine Learning (ICML)*, Beijing, China, June 21-26, 2014.
  - Oral presentation at the *33st Annual International Symposium on Forecasting (ISF)*, Seoul, Korea, June 23-26, 2013.
  - Oral presentation at the *32st Annual International Symposium on Forecasting (ISF)*, Boston, USA, June 24-27, 2012.
- Chapter 6: Multi-horizon forecasting strategies
  - S Ben Taieb, G Bontempi, A Atiya, et al. (2012). A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *Expert Systems with Applications* **39**(8), 7067–7083.
  - G Bontempi and S Ben Taieb (January 2011). Conditionally dependent strategies for multiple-step-ahead prediction in local learning. *International Journal of Forecasting* **27**(3), 689–699.
  - S Ben Taieb, A Sorjamaa, and G Bontempi (2010). Multiple-output modeling for multi-step-ahead time series forecasting. *Neurocomputing* **73**(10-12), 1950–1957.
  - Oral presentation at the *31st Annual International Symposium on Forecasting (ISF)*, Prague, Czech Republic, June 26-29, 2011.
  - Poster presentation at the *IEEE International Joint Conference on Neural Networks (IJCNN)*, Atlanta, Georgia, USA, June 14-19, 2009.
- Chapter 7: The Global Energy Forecasting Competition 2012
  - S Ben Taieb and RJ Hyndman (August 2013). A gradient boosting approach to the Kaggle load forecasting competition. *International Journal of Forecasting*, 1–19.
  - Oral presentation at the *IEEE Power and Energy Society General Meeting (PESGM)*, Vancouver, Canada, July 21-25, 2013. **[Invited]**
  - Oral presentation at *The second "Workshop on Industry & Practices for Forecasting" (WIPFOR)*, Paris, France, June 2013.

In the following, we list the publications that have not been considered in this thesis as well as some other conferences we attented:

- Other publications:
  - G Bontempi, S Ben Taieb, and YA Le Borgne (2013). Machine Learning Strategies for Time Series Forecasting. In: *Business Intelligence*. Ed. by MA Aufaure and E Zimányi. Vol. 138. Lecture Notes in Business Information Processing. Springer, pp.62–77.
  - A Vaccaro et al. (November 2012). Adaptive local learning techniques for multiple-step-ahead wind speed forecasting. *Electric Power Systems Research* **83**(1), 129–135.
  - S Ben Taieb and G Bontempi (December 2011). Recursive Multi-step Time Series Forecasting by Perturbing Data. In: *Proceedings of the 11th IEEE International Conference on Data Mining (ICDM)*. IEEE, pp.695–704.
  - S Ben Taieb, G Bontempi, A Sorjamaa, et al. (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 (IJCNN)*. Atlanta, U.S.A., pp.3054–3061
- Other conferences:
  - The *IEEE International Conference on Data Mining (ICDM)*, Brussels, Belgium, December 2012
  - The *International Conference on Data Mining (ICDM)*, Vancouver, Canada, December 2011.
  - The *18th European Symposium on Artificial Neural Networks (ESANN)*, Bruges, Belgium, April 2010.

1.3.2 Research activities

In addition to the conferences, I have attended the following three summer schools:

- *Machine Learning Summer School (MLSS)*, RUC, Beijing, China, June 2014.
- *Machine Learning Summer School (MLSS)*, Purdue University, USA, June 2011.
- *Mobility, Data Mining and Privacy (MODAP)*, Rhodos Island, Greece, August 2010.

During my PhD studies, I have made the following two research visits:

- *The Swiss AI Lab IDSIA*, Lugano, Switzerland, February 2013 (hosted by Prof. Giorgio Corani).
- *Business & Economic Forecasting Unit*, Monash University, Australia, September 2011 – January 2012 (hosted by Prof. Rob J. Hyndman).

1.3.3 Software development

Although the `forecast` package for R (Hyndman and Khandakar, 2008) allows to apply many forecasting methods, a package for applying different multi-step-ahead forecasting strategies does not seem to be yet available, especially for machine learning algorithms. To fill this gap, I have developed a new R package that allows to apply all the strategies presented in this thesis. The package should be available online soon.

Part I

Overview

Chapter 2

Background

The questions addressed in this thesis are at the intersection of machine learning and time series forecasting. The goal of this chapter is to present different concepts that will be frequently used throughout the thesis, from both domains.

Machine learning and time series forecasting are both large fields of research, so we will focus on concepts and methods that we will later build upon in future chapters. Specifically, we will first introduce the broad concept of learning from data, together with some learning algorithms that will be considered in the subsequent chapters. A second part will be devoted to the concepts and methods in time series forecasting including time series decomposition and autoregressive models.

For a broader introduction to time series forecasting, a number of references are available, e.g. Hyndman and Athanasopoulos (2014); Teräsvirta, Tjostheim, and Granger (2010); Fan and Yao (2003); Brockwell and Davis (2002); Chatfield (2000). For machine learning, key references include James et al. (2013); Abu-Mostafa, Magdon-Ismail, and Lin (2012); Mohri, Rostamizadeh, and Talwalkar (2012); Murphy (2012); Hastie, Tibshirani, and Friedman (2009).

2.1 Learning from data

In several scientific disciplines, we want to better understand or make predictions about a certain phenomenon under study. The process typically involves observing the phenomenon and constructing a model of that phenomenon by finding relations between several variables.

When the phenomenon has been studied for a long time or if it involves few variables, we can often build analytical solutions about the system or quantity of interest.

However, in many cases, the system is complex and depends on a large number of variables. In those cases, finding an analytical solution may not be possible.

In many fields such as science, engineering and economics, we can collect observations about some phenomenon, called *data*, that may reveal a lot about the underlying phenomenon. In particular, from this data, we can learn a model that approximates the true underlying phenomenon.

Learning from data is very powerful since the process can be automated and will not require a case by case analysis of the phenomenon under study. However, depending on the field, learning from data can be more or less powerful. For example, learning from financial data can be more difficult than learning from electricity consumption data.

One generally builds a model for one of two purposes: *explanation* or *prediction*. Shmueli (2010) provides a discussion about the distinction between explanatory and predictive modeling. An explanatory model gives an explanation of the behavior of the phenomenon, notably the interactions between the different components, while a predictive model provides predictions for new or future observations. In this work, we are mainly concerned with predictive models.

2.1.1 Different views and types of learning

Learning from data has been considered in different fields of research at different times. Each field has developed its own jargon and methods. However, when comparing these different fields, we can see that they have a common goal: *extracting knowledge from data*. Among the different fields dedicated to the subject, the main ones are *Machine learning*, *Statistics* and *Data mining*.

Machine learning is a relatively young field that has evolved in the computer science community and has a name that distinguishes it from human learning. Statistics is an older field that has a lot in common with Machine learning but has evolved in the field of mathematics. The difference between Machine learning and Statistics has motivated interesting debates including the article "Statistical Modeling: The Two Cultures" by Leo Breiman (Breiman, 2001). The main difference between these two fields is the assumption about the data. In Machine Learning, we often assume that the data has been generated from some unknown data generating process and different learning algorithms are used to approximate it. In Statistics, we usually assume a data model is the true data generating process and we try to estimate the parameters of this data model. So, Machine learning deals with learning algorithms or procedures while Statistics deals with models.

Compared to Machine learning and Statistics, Data mining is more focused on data analysis than on prediction. In particular, it focuses on extracting patterns and anomalies from large data sets. Hand (1998) and Friedman (1998) provide excellent discussions about the differences between Data mining and the other fields.

However, the gaps and differences between these three communities are shrinking and we expect these fields to combine in the future. Recently, the term *Data Science* has emerged as a general field that includes the different fields related to the analysis of data. Lin, Genest, et al. (2014) provide a broad overview of the past, present and future of the general field of *Statistical science*.

Learning from data consists in using a set of observations (also called *examples*) to uncover an underlying process. This broad premise is not easy to fit into a single framework. In consequence, different types of learning have been considered in the literature depending on the type of data involved in the learning process. Examples include unsupervised learning, reinforcement learning, active learning and online learning (Abu-Mostafa, Magdon-Ismail, and Lin, 2012).

In this work, we will be concerned with a typical learning scenario called *supervised learning* where each example contains an input and a target output. More precisely, we will focus on the regression learning problem, that is where the output is a quantitative variable rather than a qualitative variable as in classification.

2.1.2 The regression learning problem

In practice, we often have a data set $\mathcal{D}$ composed of N examples $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$ where $\boldsymbol{x}_i \in \mathbb{R}^d$ is a vector of d input variables and $y_i \in \mathbb{R}$ is the corresponding target or output variable. Each data point can be seen as a realization from the joint distribution $F(\boldsymbol{x}, y) = F(y|\boldsymbol{x})F(\boldsymbol{x})$ where $\boldsymbol{x}$ is a vector of d random input variables and y is a random output or target variable.

The regression learning problem often consists in using the data set $\mathcal{D}$ to build a model that will be used to predict the output for some new inputs $\boldsymbol{x}$ sampled from $F(\boldsymbol{x}, y)$, that are not necessarily in the data set $\mathcal{D}$. The problem involves estimating the stochastic dependence between $\boldsymbol{x}$ and y; in other words, we need to estimate the conditional distribution $F(y|\boldsymbol{x})$. However, for prediction purposes, it is often sufficient to estimate the conditional expectation given by:

$$f(\boldsymbol{x}) = \mathbb{E}[y|\boldsymbol{x}], \tag{2.1.1}$$

which is easier to estimate than the complete distribution.

We will assume the regression problem consists in learning the unknown target function $f : \mathbb{R}^d \to \mathbb{R}$ from the data set $\mathcal{D}$. Given a set of candidates $\mathcal{M}$ (called the *hypothesis set*[1]), a *learning algorithm* $\mathcal{A}$ will use the data set $\mathcal{D}$ to pick a hypothesis $m : \mathbb{R}^d \to \mathbb{R}$ from $\mathcal{M}$, that best approximates f, according to a *loss function* $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^+$. In this work, we will limit ourselves to the quadratic loss function defined as $L_2(\hat{y}, y) = (\hat{y} - y)^2$.

Le us denote $m_{\mathcal{D}}$ the hypothesis selected by the learning algorithm $\mathcal{A}$ from the hypothesis set $\mathcal{M}$ based on the data set $\mathcal{D}$. Then, we can define the *generalization error* (also called the *risk* or the *out-of-sample* error) of the hypothesis $m_{\mathcal{D}}$ as

$$E_{\text{out}}(m_{\mathcal{D}}) = \mathbb{E}_{\boldsymbol{x}}[(m_{\mathcal{D}}(\boldsymbol{x}) - y)^2], \tag{2.1.2}$$

which measures the cost of using $m_{\mathcal{D}}$ to predict y from $\boldsymbol{x}$ for new examples, that were not available in the data set $\mathcal{D}$. In the following, we will use m and $m_{\mathcal{D}}$, interchangeably.

We want to select the best hypothesis that generalizes well outside of the data set $\mathcal{D}$; in other words, the hypothesis with the smallest generalization error, which can be computed as follows:

$$m^* = \text{argmin}_{m \in \mathcal{M}} E_{\text{out}}(m)$$

where $\mathcal{M}$ is the hypothesis set and E_{out} is defined in (2.1.2).

However, in practice, explicitly computing the generalization error is infeasible since we would need to know the distribution $F(\boldsymbol{x}, y)$, that is usually not available. In fact, we only have access to the data set $\mathcal{D}$ which contains N independently and identically distributed (i.i.d.) examples that have been sampled from $F(\boldsymbol{x}, y)$.

What we can do is to try to estimate the generalization error by computing the *in-sample* error (also called the *empirical risk*), defined as

$$E_{\text{in}}(m) = \frac{1}{N} \sum_{i=1}^{N} (m(\boldsymbol{x}_i) - y_i)^2$$

which represents the average loss over the available data points.

Then, we can minimize the empirical risk to select the best hypothesis as follows:

$$m^* = \text{argmin}_{m \in \mathcal{M}} E_{\text{in}}(m), \tag{2.1.3}$$

which is also called *empirical risk minimization*.

[1] This should not be confused with statistical hypothesis testing (Lehmann and Romano, 2005).

In practice, we hope that empirical risk minimization performs similarly to true risk minimization. In others words, we would like to have

$$\text{argmin}_{m\in\mathcal{M}} E_{\text{in}}(m) \approx \text{argmin}_{m\in\mathcal{M}} E_{\text{out}}(m).$$

A natural question that arises is how well E_{in} estimates E_{out}?

We can find some hints to answer this question by making an analogy with a professor that tries to gauge how well the students have learned the course material. Typically, the question exams are new exercises that the students have not yet seen during the class. This allows the professor to obtain a good measure of how well the students have learned the course material. However, if the students receive as exam exercises the same questions they have used during the class, then the scores will not faithfully represent what the students have learned.

We can make the same distinction between the out-of-sample and the in-sample errors. In fact, the out-of-sample error E_{out} represents how well the hypothesis will generalize to new examples that have not been seen before. In expression (2.1.2), we can see that E_{out} is computed on the entire space but intuitively if we want to compute it on a finite sample data set, we must use new data points that have not been used during the training process. This is similar to the new exam questions that have not been used during the class.

By contrast, the in-sample error E_{in} is computed on data points that have been used for training. So, it may not reflect the true performance in a real scenario with new examples that have not been used in training. In consequence, the in-sample error E_{in} can be a biased estimate of the out-of-sample error E_{out} with an optimistic bias since the same examples have been used for both training and testing the generalization ability of the hypotheses.

The discrepancy between E_{in} and E_{out} will depend on several factors including the the hypothesis set $\mathcal{M}$ and the size of the data set $\mathcal{D}$.

Of course, the ideal hypothesis set is $\mathcal{M} = \{f\}$ that contains only the target function. But, f is exactly the function we are trying to estimate!

If the hypothesis set $\mathcal{M}$ is too complex, it will allow us to estimate a large set of functions and we can obtain a small in-sample error. However, we may fail to generalize well outside the training data since we did not learn but memorized the training data. This phenomenon is referred to as *overfitting*.

If the hypothesis set $\mathcal{M}$ is too simple, we may fail to approximate f and will have a large in-sample error. The empirical risk minimization will be closer to the true risk minimization in that case but it will also worsen the minimum of the true risk. This phenomenon is referred to as *underfitting*.

In practice, the choice of the hypothesis set $\mathcal{M}$ needs to find a good trade-off between approximating f on the training examples and generalizing on new examples.

One way to look at this trade-off is through the Vapnik-Chervonenkis (VC) theory, a subbranch of statistical learning theory (Vapnik, 1998). In particular, the VC generalization bound[2] provides a bound for the out-of-sample error E_{out} using the in-sample error E_{in} plus a penalty term Ω that depends on the complexity (or capacity) of the hypothesis set $\mathcal{M}$ and the size of the data set $\mathcal{D}$.

With finite hypothesis sets, the cardinality is typically used as a measure of the capacity of a hypothesis set $\mathcal{M}$. However, the VC bound uses the concept of VC dimension that can be computed for infinite hypothesis sets. If the reader is familiar with measures of complexity such as the

[2]For the reader's information, generalization bounds are often derived using concentration inequalities such as Hoefdding and McDiarmid's inequalities (Boucheron, Lugosi, and Massart, 2013).

dimensionality or the number of free parameters, the VC dimension can be seen as a more elaborated measure of complexity. Other measures of complexity have also been proposed in the literature such as the covering numbers and the Rademacher complexity (Mohri, Rostamizadeh, and Talwalkar, 2012).

The main argument is that high capacity (i.e. large VC dimension) and good generalization are at odds. In fact, if $\mathcal{M}$ has a high capacity that allows to explain every possible dataset, we should not expect a good generalization. On the other hand, if $\mathcal{M}$ has a small capacity, but at the same time provides small in-sample errors, then we can expect a good generalization. In other words, the VC bound suggests seeking a trade-off between reducing in-sample errors versus controlling the capacity of the hypothesis set.

One nice property of the VC generalization bound is that it is distribution independent and so it holds for all possible distributions. However, its generality is also its weakness since the bound may be too loose to give any useful information of practical use. In addition, the exact VC dimension is not easy to compute for many hypothesis sets. Finally, it is important to note that the VC analysis only depends on the hypothesis set $\mathcal{M}$, but not on the function f or the learning algorithm $\mathcal{A}$.

Another way to look at the trade-off between approximating f on the training examples and generalizing on new examples is through the so-called *bias and variance trade-off*, from the estimation theory (Geman, Bienenstock, and Doursat, 1992).

Let us assume $y = f(\boldsymbol{x}) + \varepsilon$, $\mathbb{E}[\varepsilon] = 0$ and $\mathbb{E}[\varepsilon^2] = \sigma^2$. By taking the expectation of expression (2.1.2) over all data sets $\mathcal{D}$, the expected out-of-sample error for our learning model can be written as

$$E_{\text{out}} = \mathbb{E}_{\mathcal{D},\varepsilon}[\mathbb{E}_{\boldsymbol{x}}[(y - m_{\mathcal{D}}(\boldsymbol{x}))^2]].$$

which can be decomposed as follows:

$$\begin{aligned} E_{\text{out}} &= \mathbb{E}_{\mathcal{D},\varepsilon,\boldsymbol{x}}[(y - m_{\mathcal{D}}(\boldsymbol{x}))^2] \\ &= \mathbb{E}_{\mathcal{D},\varepsilon,\boldsymbol{x}}[(y - f(\boldsymbol{x}))^2] + \mathbb{E}_{\mathcal{D},\varepsilon,\boldsymbol{x}}[(f(\boldsymbol{x}) - m_{\mathcal{D}}(\boldsymbol{x}))^2] \\ &= \mathbb{E}_{\varepsilon,\boldsymbol{x}}[(y - f(\boldsymbol{x}))^2] + \mathbb{E}_{\mathcal{D},\boldsymbol{x}}[(\bar{m}(\boldsymbol{x}) - f(\boldsymbol{x}))^2] + \mathbb{E}_{\mathcal{D},\boldsymbol{x}}[(m_{\mathcal{D}}(\boldsymbol{x}) - \bar{m}(\boldsymbol{x}))^2] \\ &= \underbrace{\mathbb{E}_{\varepsilon,\boldsymbol{x}}[(y - f(\boldsymbol{x}))^2]}_{\text{Noise variance } \sigma^2} + \underbrace{\mathbb{E}_{\boldsymbol{x}}[(\bar{m}(\boldsymbol{x}) - f(\boldsymbol{x}))^2]}_{\text{Squared bias}} + \underbrace{\mathbb{E}_{\mathcal{D},\boldsymbol{x}}[(m_{\mathcal{D}}(\boldsymbol{x}) - \bar{m}(\boldsymbol{x}))^2]}_{\text{Estimation variance}} \end{aligned} \tag{2.1.4}$$

where $\bar{m}(\boldsymbol{x}) = \mathbb{E}_{\mathcal{D}}[m_{\mathcal{D}}(\boldsymbol{x})]$.

We can see in expression (2.1.4) that E_{out} can be decomposed into three components, namely the noise variance, the squared bias and the estimation variance.

The bias component represents the consistent offset of the average predictions $\bar{m}(\boldsymbol{x})$, away from $f(\boldsymbol{x})$. The variance component relates to the stability of the predictions built on different data sets $\mathcal{D}$ sampled from the joint distribution $F(\boldsymbol{x}, y)$, and represents the variation of the predictions $m_{\mathcal{D}}(\boldsymbol{x})$ around their mean, $\bar{m}(\boldsymbol{x})$.

In case there is no bias and no variance, that is $\bar{m}(\boldsymbol{x}) = f(\boldsymbol{x})$ and $m_{\mathcal{D}}(\boldsymbol{x}) = \bar{m}(\boldsymbol{x})$, we can see in (2.1.4) that E_{out} will simply be equal to the noise variance σ^2, which is the irreducible error and the lowest value that E_{out} can achieve. However, in practice, this case is rarely met since any learning model is typically prone to bias and variance.

The ideal configuration is to have both a low bias and a low variance. However, making the learning model more flexible so that the bias decreases will induce an increase in variance, and vice versa. So, this ideal configuration is never achievable in practice. Instead, it is important to find a good trade-off between bias and variance to achieve the smallest E_{out}.

In contrast to the VC theory, a bias and variance analysis takes into account the function f we are trying to estimate as well as the learning algorithm. This can be important since different learning algorithms can have a different E_{out} for the same hypothesis set $\mathcal{M}$. In addition, the bias and variance decomposition applies to regression problems with squared errors. In consequence, we choose it as the main tool to study generalization throughout this work.

2.1.3 The cure for overfitting

We have seen that minimizing in-sample errors does not necessarily lead to a good estimate of the out-of-sample error, especially if the hypothesis set has high capacity and/or if the data set is small. This is because the same examples are used for both training the hypothesis and assessing its generalization error. In consequence, we cannot trust the in-sample error as an estimate of the out-of-sample error since it can lead to the selection of models that do not generalize well outside of the data set.

By taking that information into account, the out-of-sample error of a hypothesis m, $E_{\text{out}}(m)$, can be written as

$$E_{\text{out}}(m) = E_{\text{in}}(m) + \underbrace{\Omega(m)}_{\text{overfit penalty}}. \tag{2.1.5}$$

So, minimizing the in-sample error essentially disregards the overfitting phenomenon.

There are essentially two main classes of methods that address overfitting: *regularization* and *validation* (Abu-Mostafa, Magdon-Ismail, and Lin, 2012). Regularization attempts to compute an estimate $\hat{\Omega}(m)$ of the overfit penalty $\Omega(m)$ by making some assumptions on the hypothesis set $\mathcal{M}$. Then an estimate of the out-of-sample error $E_{\text{out}}(m)$ is given by $\hat{E}_{\text{out}}(m) = E_{\text{in}}(m) + \hat{\Omega}(m)$. Validation, on the other hand, directly computes $\hat{E}_{\text{out}}(m)$, typically using resampling methods.

Regularization consists in penalizing the in-sample error by an overfit penalty term that represents an estimate of the optimism or the bias of the in-sample error with respect to the out-of-sample error. Hypotheses that are too complex will typically have a small in-sample error but a large overfit penalty term. Too simple hypotheses will have a large in-sample error but a small overfit penalty term. The best hypotheses will have a good trade-off between these two terms and so will have a good generalization property.

Different methods for computing the overfit penalty have been proposed in the literature. Among those, there are the model-dependent penalties (Claeskens and Hjort, 2008) such as Akaike information criterion (AIC) (Akaike, 1969) and Bayesian information criterion (BIC) (Schwarz, 1978). Many other criteria have been proposed in the literature such as Mallow's C_p (Mallows, 2000) and the Minimum description length (MDL) (Rissanen, 1986), which is based on an optimal coding viewpoint (Cover and Thomas, 2012) and is closely related to BIC.

The VC analysis that allows us to understand the overfitting phenomenon can also be used to deal with overfitting. The idea is to have a measure of the complexity of a class of models and to penalize all models in that class. This is in contrast to model-based penalties that penalize individual models. This is called *structural risk minimization*.

As an alternative to regularization which makes an adjustment to the in-sample error, the validation approach directly estimates the out-of-sample error by evaluating the model performance on new examples that have not been used to fit the model. In the following, we will present a general overview of the main validation methods. Arlot and Celisse (2010) provide a detailed survey of different validation methods, including their statistical properties.

Ideally, in order to estimate the true generalization error of a hypothesis m, we would need an independent and very large data set, called a test set. However, in practice, such data set is rarely available and we only have the data set $\mathcal{D}$. A number of techniques have been proposed to simulate the true out-of-sample setting using the available data set $\mathcal{D}$. The general idea is to holdout a subset of examples from the fitting process and to evaluate the hypothesis on the held out examples.

A very simple approach, called the validation set approach, consists in randomly splitting the data set $\mathcal{D}$ containing N observations into a *training* set $\mathcal{D}_{\text{train}}$ with $N-M$ observations and a *validation* set or *hold-out* set $\mathcal{D}_{\text{val}}$ with M observations. To estimate the out-of-sample error, we run the learning algorithm and use the training set $\mathcal{D}_{\text{train}}$ to find the final hypothesis $m^- \in \mathcal{M}$, where we indicate with the minus superscript that some examples of $\mathcal{D}$ have not been used. Then, we can predict the output of the examples in the validation set $\mathcal{D}_{\text{val}}$ with the final hypothesis and compute the validation error:

$$E_{\text{val}}(m^-) = \frac{1}{M} \sum_{\boldsymbol{x}_i \in \mathcal{D}_{\text{val}}} (m^-(\boldsymbol{x}_i) - y_i)^2. \tag{2.1.6}$$

We know that $E_{\text{in}}(m^-)$ is a biased estimate of $E_{\text{out}}(m^-)$ but what can we say about $E_{\text{val}}(m^-)$? Because the validation set $\mathcal{D}_{\text{val}}$ is independent from the training set $\mathcal{D}_{\text{train}}$, we can write $\mathbb{E}_{\mathcal{D}_{\text{train}}}[E_{\text{val}}(m^-)] = E_{\text{out}}(m^-)$, that is the validation error is an unbiased estimate of the out-of-sample error. Also, what can we say about the reliability of E_{val} as an estimate of E_{out}?

We got an unbiased estimate of E_{out} by setting aside M examples from $\mathcal{D}$. However, there is a price to pay for that, and this price will depend on the value of M. In fact, if M is small, then m^- will be estimated using a training set with a larger size but E_{val} will be computed using few examples and so will be less reliable. On the other hand, if M is large then E_{val} will be more reliable since computed on more examples but $M-N$ will not be big enough to obtain a good hypothesis m^-.

With the validation approach, we are trying to estimate the out-of-sample error of m^- but the initial goal was to estimate that of m. In fact, m^- is the hypothesis computed on a subset of $\mathcal{D}$ with $N-M$ examples while m is computed on the entire data set $\mathcal{D}$ with N examples.

In addition, because m^- is based on a smaller data set, $E_{\text{out}}(m^-) > E_{\text{out}}(m)$ so m^- is not the best hypothesis. However, we don't need to output m^- as the final hypothesis. In fact, provided that M is large enough, $E_{\text{val}}(m^-)$ will likely still be better at estimating $E_{\text{out}}(m)$ than $E_{\text{in}}(m)$. So we can use this estimate but output m as the final hypothesis.

This shows again the importance of the value of M that should not be too large so that the hypothesis is based on a data set not too far from $\mathcal{D}$ and not too small to get a reliable E_{val}. In practice, one rule of thumb is to use $M = 0.2 \times N$ with the validation set approach (Abu-Mostafa, Magdon-Ismail, and Lin, 2012).

Instead of considering a single split of the data set $\mathcal{D}$, we can also have multiple splitting and combine the results from those different splits. This idea is used by the so-called *cross-validation* methods. We will consider two variants namely *Leave-One-Out* (LOO) cross-validation that considers N splits where each split uses $M = 1$ and *V-fold* cross-validation which uses V splits where each splits uses $M = \frac{N}{V}$. Key references for cross-validation are Stone (1977) and Allen (1974).

With LOO, the validation set contains a single example and the remaining $N-1$ examples form the training set. We can produce N such partitions by having a different example in the validation set. For all these partitions, we apply the same procedure as with the validation set approach to compute a validation error as in (2.1.6). Then, we average the results to obtain the LOO cross-validation

estimate, given by

$$E_{CV} = \frac{1}{N}\sum_{i=1}^{N} E_{\text{val}}(m_i^-), \tag{2.1.7}$$

with

$$E_{\text{val}}(m_i^-) = (m_i^-(\boldsymbol{x}_i) - y_i)^2$$

where m_i^- is the hypothesis learned from $\mathcal{D} \setminus \{(\boldsymbol{x}_i, y_i)\}$, that is the data set $\mathcal{D}$ from which we removed the ith example.

One advantage of LOO is that we use $M = 1$ and so m^- is very close to m. However, E_{val} is based on one example and so is not reliable. However, because we are applying the procedure N times, the hope is that it is close to the validation set approach that uses a validation set containing N observations.

LOO is a computationally demanding procedure since we need to run the learning procedure N times for a data set containing N examples, which can be time consuming when N is large. The only case where LOO can be computed quickly is with linear models which have an analytical solution that does not require to run the procedure N times (Allen, 1974). A key reference for LOO with linear model is Shao (1993), who showed that LOO is inconsistent.

An alternative to LOO is V-fold cross-validation which includes more observations in each validation set for each split but then has less splits. More precisely, V-fold cross-validation splits the data set $\mathcal{D}$ into V disjoint sets (or *folds*) $\mathcal{D}_1, \ldots, \mathcal{D}_V$ where each $\mathcal{D}_v$ has a size of N/V. For all these sets, we apply the same procedure as with the validation set approach, that is we learn the hypothesis on $\mathcal{D} \setminus \mathcal{D}_v$ and we compute the validation error on $\mathcal{D}_v$. Then, we average the results to obtain the V-fold cross-validation estimate as follows:

$$E_{CV} = \frac{1}{V}\sum_{v=1}^{V} E_{\text{val}}(m_{\mathcal{D}_v}^-), \tag{2.1.8}$$

with

$$E_{\text{val}}(m_{\mathcal{D}_v}^-) = \frac{V}{N}\sum_{\boldsymbol{x}_i \in \mathcal{D}_v} (m_{\mathcal{D}_v}^-(\boldsymbol{x}_i) - y_i)^2,$$

where $m_{\mathcal{D}_v}^-$ is the hypothesis learned from $\mathcal{D} \setminus \mathcal{D}_v$.

By comparing expression (2.1.8) with expression (2.1.7), we can see that LOO is equivalent to N-fold cross-validation. Compared to LOO, V-fold cross validation allows a decrease in computational time when $V << N$. However, less examples will be used to learn the hypothesis m^- which will increase the discrepancy between $E_{\text{out}}(m^-)$ and $E_{\text{out}}(m)$.

This brings us to the bias and variance trade-off with V-fold cross-validation methods. E_{CV} in expression (2.1.8) can be seen as an estimate of E_{out} and so has a bias and variance which will depend notably on the value of V.

If V is large, as with LOO, the hypothesis m^- is learned from $N-1$ examples and so is very close to the hypothesis m learned from the complete data set with N examples. So, in that case, the bias will be small. However, since each hypothesis will be learned from a training set that differs only on one example, the error of the different hypotheses will be highly correlated which in turn will induce a larger variance.

If V is small, the hypothesis m^- is learned from less examples compared to m and so the bias is large. However, since the different training sets will differ on many examples, the errors of the different hypothesis will be less correlated and so, in this case, the variance will be smaller.

In consequence, there is a bias-variance trade-off that depends on the value of V in V-fold cross-validation. In practice, a common choice is $V = 5$ or $V = 10$ since these values have been shown empirically to provide a good bias-variance trade-off (Kohavi, 1995).

Bootstrap (Efron, 1979) is another resampling method to estimate the out-of-sample errors. The idea is to sample the initial data set with replacements to form new datasets with the same size as the initial data set. To estimate the out-of-sample error, we learn a model for each bootstrap datasets and use the initial data set as testing set, but we only use the examples that are not in the bootstrap data set to compute the errors to avoid overfitting. We refer the reader to Efron and Tibshirani (1993) for an overview of the bootstrap. Also, Kohavi (1995) provides a comparison between cross-validation and bootstrap.

In this section, we have seen that there are mainly two approaches to estimate the out-of-sample error by taking into account the bias of the in-sample error: regularization and validation.

There is no best approach and the performance of each approach will depend on many factors, such as the size of the data set and the validity of the assumptions. In addition, the two approaches can be combined: for example some regularization methods use validation to find the right amount of regularization needed. Also, in some cases, it has been shown that some regularization methods are equivalent to validation methods. For example, Stone (1977) showed that AIC and LOO are asymptotically equivalent. For linear models, Shao (1997) showed that minimizing BIC is equivalent to leave-k-out cross-validation, that is where k examples are left out at each step instead of one example as in LOO.

One advantage of the regularization methods such as AIC and BIC is the decrease in computational time which can be useful when we need to compute it for a large number of hypotheses. Also, these methods are more suited for small sample data sets since the whole data set is used to compute the errors. On the other hand, these methods require to make some assumptions about the true underlying model, to compute the model degrees of freedom and to be able to estimate the error variance. In practice, the assumption may be too strong and it may be hard to compute the degrees of freedom or the error variance in some cases.

The main advantage of the validation methods is their generality. In fact, they make no assumptions about the underlying model and so can be applied with any model. One drawback of these methods is the loss of data when we holdout some examples from the training set. This can be unfeasible in practice when we only have few examples. Although these methods are computationally demanding, nowadays with fast computers, it is not anymore an issue. We refer the reader to Arlot and Celisse (2010) for a comprehensive survey about validation methods.

2.1.4 The learning procedure

In the previous section, we have discussed several alternatives to estimate the out-of-sample error of a given hypothesis. However, estimating the out-of-sample error is not our primary goal. Instead we want to select the best hypothesis among a hypothesis set and for that purpose, we needed a good estimate of the out-of-sample error. In this section, we will present the different steps of a *learning procedure* that allows us to select the best hypothesis from a hypothesis set.

The learning procedure can be decomposed in the following steps:

1. ***Model generation***. Given an hypothesis set $\mathcal{M}$, define a sequence of hypothesis sets with increasing complexity $\mathcal{M}_0 \subset \mathcal{M}_1 \subset \cdots \subset \mathcal{M}_S = \mathcal{M}$.

The hypothesis set $\mathcal{M}$ is typically an arbitrary choice when there is no prior knowledge about the true underlying function. Otherwise, the prior knowledge can be used to select a better hypothesis set.

We denote $m(\cdot;\boldsymbol{\beta},\boldsymbol{\psi})$ a hypothesis of the hypothesis set $\mathcal{M}$ where $\boldsymbol{\beta}\in\mathbb{R}^d$ is a set of parameters and $\boldsymbol{\psi}\in\{\boldsymbol{\psi}^{(0)},\boldsymbol{\psi}^{(1)},\ldots,\boldsymbol{\psi}^{(S)}\}$ is a set of hyperparameters.

Different values of the hyperparameters, i.e. $\boldsymbol{\psi}^{(0)},\boldsymbol{\psi}^{(1)},\ldots,\boldsymbol{\psi}^{(S)}$, define different classes of hypotheses, i.e. $\mathcal{M}_0,\mathcal{M}_1,\ldots,\mathcal{M}_S$. In other words, the hyperparameters $\boldsymbol{\psi}$ are directly related to the structure and complexity of the hypothesis m.

A given class of hypothesis, $\mathcal{M}_s$, or equivalently a given value of the hyperparameters $\boldsymbol{\psi}^{(s)}$, defines a subset of the hypothesis set $\mathcal{M}$ where all the hypotheses have the hyperparameters $\boldsymbol{\psi}=\boldsymbol{\psi}^{(s)}$, and each hypothesis has a different value of the parameters $\boldsymbol{\beta}$.

2. ***Parametric identification***. For each hypothesis set $\mathcal{M}_s$, apply the empirical risk minimization as in (2.1.3) to find the best hypothesis of that hypothesis set, $m(\cdot;\boldsymbol{\beta}^*,\boldsymbol{\psi}^{(s)})$. It is worth noting that some models do not have a parametric identification stage, e.g. the KNN model (see Section 2.2.3).

3. ***Model validation***. For each hypothesis set $\mathcal{M}_s$, compute $\hat{E}_{\text{out}}(m(\cdot;\boldsymbol{\beta}^*,\boldsymbol{\psi}^{(s)}))$, an estimate of the out-of-sample error of the hypothesis $m(\cdot;\boldsymbol{\beta}^*,\boldsymbol{\psi}^{(s)})$. For example, we can use 5-fold cross-validation to estimate E_{out}, as explained in the previous section.

   The parametric identification and the model validation stages form what is called the *structural identification* stage.

4. ***Model selection***. Select the hypothesis set with the lowest estimated out-of-sample error, that is $s^*=\operatorname{argmin}_s m(\cdot;\boldsymbol{\beta}^*,\boldsymbol{\psi}^{(s)})$ and return its best hypothesis $m(\cdot;\boldsymbol{\beta}^*,\boldsymbol{\psi}^{(s^*)})$.

   Model selection is also called *hyperparameters optimization* since we optimize the value of the hyperparameters. Instead of selecting the best model, we can also apply *model combination*, for example, by averaging several models. With a simple computation, one can show that the simple average of two unbiased estimators with a non zero variance returns a combined estimator with reduced variance. So, one of the main motivations of model combination is to obtain a model with a smaller variance.

2.2 Learning regression algorithms

In this section, we present several learning regression algorithms that will be considered in future chapters, namely the linear model, neural networks, nearest neighbors and gradient boosting. We will briefly describe each model and, if applicable, explain how the parametric identification is performed. Recall that we want to solve a regression learning problem as described in Section 2.1.2.

2.2.1 Linear model

The linear model is a parametric model that computes a linear combination of the input variables using a vector of parameters $\boldsymbol{\beta}$ (also called coefficients) as follows:

$$m(\boldsymbol{x})=\sum_{k=0}^{p}\beta_k x_k=\boldsymbol{\beta}'\boldsymbol{x}, \tag{2.2.1}$$

where $\boldsymbol{x}_0=1$ and $\boldsymbol{\beta}\in\mathbb{R}^p$.

Let us denote by $\boldsymbol{X}$ the $N \times (p+1)$ matrix whose rows contains the inputs $\boldsymbol{x}_i$ as row vectors, and denote by $\boldsymbol{y}$ the $N \times 1$ target vector whose components are the output values y_i. One of the most popular methods to estimate the parameters $\boldsymbol{\beta}$ is the *ordinary least squares* (OLS) in which the residual sum of squares is minimized. In other words, the parameters are estimated by minimizing the least squares criterion:

$$\mathcal{Q}(\boldsymbol{\beta}) = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}). \tag{2.2.2}$$

It can be shown that the unique solution to this problem can be easily computed analytically and is given by:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}\boldsymbol{y}, \tag{2.2.3}$$

assuming $(\boldsymbol{X}'\boldsymbol{X})$ is non-singular. If $(\boldsymbol{X}'\boldsymbol{X})$ is singular, then the parameters $\hat{\boldsymbol{\beta}}$ are not uniquely defined and can be computed using a pseudo-inverse.

After estimating the parameters, the fitted values at the training inputs can be computed as follows:

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}} = \underbrace{\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}}_{\boldsymbol{H}}\boldsymbol{y}, \tag{2.2.4}$$

where the matrix $\boldsymbol{H}$ is called the *hat matrix* because $\boldsymbol{H}$ "puts a hat" on $\boldsymbol{y}$.

In expression (2.2.1), the linear model is presented as an algorithm or a method for making predictions from an input $\boldsymbol{x}$. However, in the statistics literature, it is sometimes assumed that the linear model is the true underlying data generating process. In other words, the examples are assumed to be sampled from the following data generating process:

$$y = \sum_{k=0}^{p} \beta_k x_k + \varepsilon,$$

where ε is a noise term, and the function f defined in expression 2.1.1 is indeed linear in the x variables. Under the correct model and appropriate conditions, OLS enjoy several properties such as unbiasedness and consistency (Wasserman, 2004).

In expression (2.2.1), we can see that the model is linear not only in terms of the x_k variables but also in the parameters β_k. The OLS solution given in (2.2.3) suggests that the x_k are just constants as far as the algorithm is concerned, while the linearity in the parameters β_k is the key property for the linear model. This observation suggests that we do not have to use the initial inputs but we can also consider transformations of these inputs and still use all the linear machinery to estimate the parameters. For example, we can include nonlinear transformations of the input variables to model nonlinear phenomenon. In other words, the model given in (2.2.1) is linear in both the input variables and the parameters, but we can also consider models that are nonlinear in the input variables but linear in the parameters.

The main idea is to derive from the input vector $\boldsymbol{x}$, a new set of variables $\boldsymbol{z}$, called features, that are transformations of the raw input variables. In other words, we compute $B(\boldsymbol{x}) = (B_1(\boldsymbol{x}), \ldots, B_L(\boldsymbol{x}))$ to obtain $\boldsymbol{z} = (z_1, \ldots, z_L)$ where B_l is a basis function.

For example, if $\boldsymbol{x} = (1, x_1, x_2)$, we can define the polynomial basis functions $B_1(\boldsymbol{x}) = 1$, $B_1(\boldsymbol{x}) = x_1$, $B_2(\boldsymbol{x}) = x_2$, $B_3(\boldsymbol{x}) = x_1^2$, $B_4(\boldsymbol{x}) = x_1 x_2$ and $B_5(\boldsymbol{x}) = x_2^2$; the new features are then $z = (B_1(\boldsymbol{x}), B_2(\boldsymbol{x}), B_3(\boldsymbol{x}), B_4(\boldsymbol{x}), B_5(\boldsymbol{x})) = (1, x_1, x_2, x_1^2, x_1 x_2, x_2^2)$.

The beauty of this approach consists in the fact that we can model nonlinear phenomenon by using a linear model with nonlinear transformations of the input variables.

Let us consider a univariate regression case where we have only one input variable. After defining our basis functions, we can write the model as:

$$m(x) = \sum_{l=0}^{L} \beta_l B_l(x) \tag{2.2.5}$$

where B_l is the lth basis function.

We can see that expression (2.2.5) has the same form as expression (2.2.1). In fact, we can estimate the parameters β_l using OLS. More precisely, we first apply each basis function to each training example $\boldsymbol{x}_i$ and gather the results in a matrix $\boldsymbol{B}$ $(N \times L)$ where

$$B_{il} = B_l(x_i) \tag{2.2.6}$$

Then we can apply OLS using the matrix $\boldsymbol{B}$ in place of the usual matrix X as follows:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{B}'\boldsymbol{B})^{-1}\boldsymbol{B}\boldsymbol{y} \tag{2.2.7}$$

Among the different alternatives for the basis functions, there are the families of *splines*, which are piecewise-defined smooth polynomials. Splines have a high degree of smoothness at the points where the polynomial pieces connect, known as *knots*. For example, a *cubic spline* is a spline that has continuous first and second derivatives at the knots. The basis splines (B-splines) are the most popular notably due to their better numerical properties (Boor, 2001).

The basis expansion will typically increase the number of input variables, for example, with a polynomial of degree p, the number of variables will increase exponentially with p. In consequence, when there are too many basis functions, OLS will overfit the estimation of the parameters, which is a manifestation of the *curse of dimensionality* (Bellman and Bellman, 1961).

In Section 2.1.3, we have seen that we can add an overfit penalty to the in-sample errors to take into account the overfitting phenomenon (see expression (2.1.5)). The same idea can be applied to select the parameters of the linear model by minimizing *penalized* least squares:

$$\mathcal{Q}(\boldsymbol{\beta}) = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) + \lambda \mathcal{J}(\boldsymbol{\beta}) \tag{2.2.8}$$

where λ is the smoothing parameter and $\mathcal{J}(\boldsymbol{\beta})$ is a quadratic penalty of the form

$$\mathcal{J}(\boldsymbol{\beta}) = \boldsymbol{\beta}'\boldsymbol{P}\boldsymbol{\beta} \tag{2.2.9}$$

where $\boldsymbol{P}$ is a penalty matrix.

The smoothing parameter λ controls the degree of penalization. If $\lambda = 0$, then expression (2.2.8) is equivalent to the unpenalized least squares criterion in (2.2.2). Higher values of λ induce a larger amount of penalty.

As with OLS, the optimal solution can be computed analytically and is given by

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X} + \lambda \boldsymbol{P})^{-1}\boldsymbol{X}\boldsymbol{y} \tag{2.2.10}$$

Different penalty matrices lead to different methods and different estimates of the parameters $\boldsymbol{\beta}$. For example, $\boldsymbol{P} = \boldsymbol{I}$ is called the *ridge penalty* where all parameters are evenly shrunken toward zero (Hoerl and Kennard, 1970).

If we combine the idea of basis expansion in (2.2.5) with regularization in (2.2.8), we can build very flexible estimators with good generalization property. In particular, in the following, we will focus on *penalized regression splines (P-Splines)*.

Penalized regression splines (P-Splines)

Penalized regression splines (P-Splines) combine the idea of B-splines with a regularization term that enforces smoothness by penalizing large differences of the parameters of adjacent knots (Eilers and Marx, 1996).

With regression splines, after defining B-splines as basis functions, the parameters of the linear model in (2.2.5) can be estimated by OLS as in (2.2.7), where the matrix $\boldsymbol{B}$ defined in (2.2.6) replaces the usual matrix $\boldsymbol{X}$ in (2.2.3).

With P-splines, we use the same idea but we add a d-order difference penalty on the parameters $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_L)'$ given by

$$\sum_{l=d+1}^{L} (\Delta^d \beta_l)^2, \tag{2.2.11}$$

where the difference operator Δ^d is defined recursively as

$$\Delta^d \beta_l = \Delta(\Delta^{d-1} \beta_l),$$

with $\Delta\beta_l = \beta_l - \beta_{l-1}$ and $\Delta^0 \beta_l = \beta_l$.

In other words, we minimize a penalized least squares criterion as in (2.2.8) with $\boldsymbol{X} = \boldsymbol{B}$ where $\boldsymbol{B}$ is defined in (2.2.6), and the matrix $\boldsymbol{P}$ in (2.2.9) is given by

$$\boldsymbol{P} = \boldsymbol{D}'_{(d)} \boldsymbol{D}_{(d)},$$

where $\boldsymbol{D}_{(d)}$ is a matrix representation of the difference operator defined in (2.2.11). For example, a first order difference matrix is given by

$$\boldsymbol{D}_{(1)} = \begin{pmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{pmatrix},$$

while a second order difference is given by

$$\boldsymbol{D}_{(2)} = \begin{pmatrix} 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \end{pmatrix}.$$

Then, the parameters $\boldsymbol{\beta}$ can then be computed using expression (2.2.10), as follows

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{B}'\boldsymbol{B} + \lambda \boldsymbol{P})^{-1} \boldsymbol{B}\boldsymbol{y}$$

P-splines are characterized by a number of parameters that have to be specified: the degree of the B-spline bases, the number of knots, the order of the difference penalty d and the smoothing parameter λ.

Cubic B-splines (i.e. of order 3) are the most commonly used B-spline bases since they offer the best trade-off between flexibility and computational simplicity.

With P-Splines, the knots are typically fixed at quantiles of the input variable $\boldsymbol{x}_l$ and we only need to select the number of knots. In addition, Ruppert (2002) has shown that the number of knots does not have much effect on the estimation, provided enough knots are used.

The difference order is generally specified to $d = 2$, that is deviations from linearity are penalized. Also, when $d = 2$, the penalty can be seen as a discrete approximation of the smoothing spline penalty.

The smoothing parameter λ is the main hyperparameter of the P-splines that controls the trade-off between the fit and the amount of penalty. One can for example use cross-validation to select its best value. Another approach consists in specifying the *degree of freedom* for the fit as follows.

As with the (unpenalized) least squares in (2.2.4), we can write the fitted values at the training inputs as

$$\hat{\boldsymbol{y}} = \boldsymbol{B}\hat{\boldsymbol{\beta}} = \underbrace{\boldsymbol{B}(\boldsymbol{B}'\boldsymbol{B} + \lambda\boldsymbol{P})^{-1}\boldsymbol{B}}_{\boldsymbol{S}_\lambda}\boldsymbol{y},$$

where $\boldsymbol{S}_\lambda$ is known as the *smoother matrix*, which is also an hat matrix. The *degree of freedom* can then be defined as $\mathrm{df}_\lambda = \mathrm{tr}(\boldsymbol{S}_\lambda)$, that is the sum of the diagonal elements of $\boldsymbol{S}_\lambda$ (Hastie and Tibshirani, 1990). One advantage of this definition is that it allows a more natural way to parametrize the P-spline estimator. Of course, for a given degree of freedom, there is a corresponding smoothing parameter λ that can be computed.

In order to consider bivariate interactions, it is possible to extend univariate P-splines to bivariate P-splines by computing a tensor product of two univariate B-spline bases. To enforce smoothness as for univariate P-splines, a bivariate penalty matrix can be constructed from separate univariate difference penalties in order to penalize variations in both directions. For more details on tensor product basis functions, we refer the reader to Eilers and Marx (2003); Marx and Eilers (2005); Wood (2006).

2.2.2 Neural networks

The Multi-layer perceptron (MLP), also called *neural network*, is one of the most successful machine learning algorithms that allows to model complex nonlinear relationships between a set of input variables and an output variable.

A neural network is a network of nodes organized in layers including an input layer made up with the input variables, intermediate layers called hidden layers that contain hidden nodes, and 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.

We consider the standard neural network with one-hidden layer which is given by

$$m(\boldsymbol{x}) = \alpha_0 + \sum_{j=1}^{NH} \alpha_j g(\boldsymbol{w}_j^T \boldsymbol{x}'),$$

where $\boldsymbol{x}'$ is the input vector $\boldsymbol{x}$, augmented with 1, i.e., $\boldsymbol{x}' = (1, \boldsymbol{x}^T)^T$, $\boldsymbol{w}_j$ is the weight vector for the jth hidden node, $\alpha_0, \alpha_1, \ldots, \alpha_n$ are the weights for the output node and NH is the number of hidden nodes. The function g is the output of the hidden node, and is generally given by the logistic function: $g(u) = \frac{1}{(1+e^{-u})}$.

The number of hidden nodes (NH) controls the complexity of the model. In particular, when $NH = 0$, the MLP reduces to the linear model given in expression (2.2.1).

For a given number of hidden nodes, the weights are generally estimated using some specific optimization procedure, the most popular one being the backprogpagation procedure (Rumelhart, Hinton, and Williams, 1986). Usually, the weights are chosen to be random values near zero to begin with, and the backprogpagation procedure updates the weights so that the prediction errors are minimized.

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.

Because the MLP is a heavily parametrized model, it has a high chance of overfitting the data. In particular, large weights can induce a large variance of the output (Geman, Bienenstock, and Doursat, 1992). A very effective regularization method, called *weight decay*, consists in penalizing large weights by adding a penalty term to the error function. As in ridge regression with linear models (see Section 2.2.1), the usual penalty is the sum of squared weights and a parameter λ is used to control the trade off between fitting errors and weight size.

In the learning procedure presented in Section 2.1.4, for the MLP model, we have $\boldsymbol{\beta} = [\alpha_1, \ldots, \alpha_n; \boldsymbol{w}_1, \ldots, \boldsymbol{w}_{NH}]$ and $\boldsymbol{\psi} = [NH, \lambda]$. Finally, our implementation of the MLP model is based on the `nnet` package for R (Venables and Ripley, 2002).

2.2.3 K-Nearest neighbors

The K-Nearest neighbors (KNN) model is a nonlinear and nonparametric model where the prediction for a data point is obtained by averaging the target outputs of the K nearest neighbor points of the given point (Altman, 1992; Atkeson, Moore, and Schaal, 1997). In other words, the prediction for the input $\boldsymbol{x}$ is computed as follows

$$m(\boldsymbol{x}) = \frac{1}{K} \sum_{k=1}^{K} y_{\mathrm{NN}(\boldsymbol{x},k)} \tag{2.2.12}$$

where $y_{\mathrm{NN}(\boldsymbol{x},k)}$ is the output of the k-th nearest neighbor of the input vector $\boldsymbol{x}$ in the data set.

In the learning procedure presented in Section 2.1.4, for the KNN model we have $\boldsymbol{\psi} = K$ and because there is no parameters, $\boldsymbol{\beta} = \emptyset$.

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 $\boldsymbol{x}$ but the most common choice is the standard Euclidean distance. Despite its simplicity, the KNN model yields good results in practice, and is often used to provide a benchmark to more complex models such as neural networks.

Instead of computing a simple average in (2.2.12), we can also compute a weighted average to give more weights to the closer points. In this work, we shall use a weighted KNN model where the weights are a function of the Euclidean distance between the query point and the neighboring points (we used the biweight function of Atkeson, Moore, and Schaal, 1997). Finally, our implementation of the KNN model is based on a modified version of the `kknn` package for R (Hechenbichler, 2014).

2.2.4 Gradient Boosting

Boosting is a learning algorithm based on the idea of creating an accurate learner by combining many so-called "weak learners", i.e. learners that have a high bias/low variance property (Schapire, 1990). At the end of the nineties, Freund and Schapire (1996) and Freund and Schapire (1997) designed *AdaBoost*, the well-known classification algorithm based on the property of weak learnability. In AdaBoost, a weight is assigned to each training example which indicates its relative importance. These weights are used to compute the error of a hypothesis on the training data. At each iteration, examples are reweighted with larger weights given to instances that are not correctly classified by the last hypothesis. Thus, learning can focus on those instances that have higher errors as the process continues.

Schapire, Freund, et al. (1998) have focused on understanding the good performance of voting algorithms by bounding the generalisation error via the VC dimension and the distribution of so-called margins. Breiman (1999) states that the VC-type bounds are misleading and was the first to establish the link between AdaBoost and statistical learning. He shows that AdaBoost can be seen as an optimization algorithm in function space. An additional breakthrough of boosting in the statistical community was achieved by Friedman and Hastie (2000) who showed that AdaBoost can be seen as a *forward stagewise additive model*, that is a model formed by an additive combination of weak learners. From a classification algorithm, Boosting has evolved to a general modeling algorithm that can be interpreted as a functional gradient descent optimization algorithm (Friedman, 2001).

Since its inception in 1990, Boosting has attracted much attention in the literature due to its excellent prediction performance in a wide range of applications both in machine learning and statistics. The reason of its good performance seems to be associated with its resistance to overfitting, which is still under investigation (Mease and Wyner, 2008; Mease, 2008). An overview of the first developments of boosting can be found in Ridgeway (1999). Recent developments of boosting in the statistical community is given by Bühlmann and Hothorn (2007). Finally, we refer to Schapire and Freund (2012) for a general overview on boosting.

Gradient boosting models compute the prediction for the input $\boldsymbol{x}$ as follows

$$m(\boldsymbol{x}) = \sum_{j=0}^{J} \nu l^{[j]}(\boldsymbol{x}; \hat{\boldsymbol{\beta}}),$$

where $l^{[j]}(\boldsymbol{x}; \hat{\boldsymbol{\beta}})$ is the base learner estimate at the jth stage with parameters $\hat{\boldsymbol{\beta}}$ and $\nu \in [0, 1]$ is a shrinkage factor. Note that $\hat{l}^{[j]}(\boldsymbol{x})$ will be used as shorthand for $l^{[j]}(\boldsymbol{x}; \hat{\boldsymbol{\beta}})$.

Gradient boosting models are estimated in a stagewise manner. In this work, we will focus on gradient boosting models with the quadratic loss, also called L_2Boost.

We let $m^{[j]}(\boldsymbol{x})$ denote the estimation at the jth stage, where $j = 0, 1, \ldots, J$. The process begins with $m^{[0]}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} y_i$. Then the model is updated using

$$m^{[j]}(\boldsymbol{x}) = m^{[j-1]}(\boldsymbol{x}) + \nu l^{[j]}(\boldsymbol{x}; \hat{\boldsymbol{\beta}}).$$

Given an estimation $m^{[j-1]}$, each additional term $\hat{l}^{[j]}(\boldsymbol{x})$ is obtained as follows. First, the negative gradient (which gives the steepest descent direction) is computed as follows

$$u_i^j = -\frac{\frac{1}{2}\partial(y_i - m(\boldsymbol{x}))^2}{\partial m(\boldsymbol{x})}\bigg|_{m(\boldsymbol{x})=\hat{f}^{[j-1]}(\boldsymbol{x}_i)} = -(y_i - \hat{f}^{[j-1]}(\boldsymbol{x}_i)),$$

where $i = 1,\ldots,N$. Because of the quadratic loss, the pseudo-residuals $u_i^{[j]}$ are simply equal to residuals from the previous function estimates $m^{[j-1]}(\boldsymbol{x}_i)$.

After computing the pseudo-residuals, a regression is applied on $\{(\boldsymbol{x}_i, u_i^{[j]})\}_{i=1}^N$ by the base learner $l(\boldsymbol{x};\boldsymbol{\beta})$, i.e. the parameters of the base learner are estimated as follows:

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \sum_{i=1}^{N} [u_i^{[j]} - l^{[j]}(\boldsymbol{x}_i;\boldsymbol{\beta})]^2. \tag{2.2.13}$$

In other terms, $l^{[j]}(\boldsymbol{x};\hat{\boldsymbol{\beta}})$ is selected to best predict the residuals from the previous function estimates $m^{[j-1]}(\boldsymbol{x})$.

Finally, the final estimate is given by

$$m(\boldsymbol{x}) = m^{[J]}(\boldsymbol{x}) = m^{[0]}(\boldsymbol{x}) + \sum_{j=1}^{J} \nu l^{[j]}(\boldsymbol{x};\hat{\boldsymbol{\beta}}) \tag{2.2.14}$$

where the estimation is continuously improved by an additional component (or boost) $\nu l^{[j]}(\boldsymbol{x};\hat{\boldsymbol{\beta}})$ at stage j and we can prevent overfitting by limiting the number of components J.

As can be seen from expression (2.2.13), the previously described boosting algorithm estimates one single base-learner that uses all the variables in $\boldsymbol{x}$, at each boosting iteration. Bühlmann and Yu (2003) developed a better algorithm, called *component-wise* gradient boosting, that includes automatic base-learners selection. The key difference is that the new algorithm estimates B different base-learners instead of one single base-learner and selects at each iteration the base-learner with the largest contribution to the fit. In addition, these base-learners will typically depend on a subset of variables of the vector $\boldsymbol{x}$. One advantage of this algorithm is that it can handle high-dimensional regression problems (Bühlmann, 2006).

Algorithm 1 gives the different steps of the *component-wise* gradient boosting algorithm, including a description for each step.

From line (10) in Algorithm 1, we can see that the boosting procedure depends on two hyperparameters: ν, the shrinkage parameter and J, the number of components (or number of boosting iterations). The value of ν affects the best value for J, i.e. decreasing the value of ν requires a higher value for J. Since they can both control the degree of fit, we should ideally find the best value for both of them by minimising some model selection criterion. However, Friedman (2001) shows that small values of ν are better in terms of less overfitting of the boosting procedure. Hence, there is only one hyperparameter remaining for which the best value needs to be selected (Bühlmann and Yu, 2003).

Several weak learners have been used in the boosting literature, notably stumps (trees with two terminal nodes) (Friedman, 2001) and smoothing splines (Bühlmann and Yu, 2003). Penalised regression splines (P-splines) (Eilers and Marx, 1996) have also been considered in Schmid and Hothorn (2008) as a better alternative to smoothing splines in terms of computational time.

In this work, our implementation of the gradient boosting algorithm will be based on P-splines, which have been described in Section 2.2.1. P-splines require the selection of two additional parameters: the number of knots and the smoothing parameter. However, Ruppert (2002) showed that the number of knots does not have much effect on the estimation provided enough knots are used. The weakness of the P-spline is measured by its degree of freedom (df). Bühlmann and Yu (2003) and Schmid and Hothorn (2008) proposed that the smoothing parameter should be set to

Algorithm 1 Component-wise gradient boosting with quadratic loss

$\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ where $(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$: dataset
$\{l_1(\boldsymbol{x}), \ldots, l_B(\boldsymbol{x})\}$: set of B base-learners
J: number of boosting iterations
$0 < \nu \leq 1$: shrinkage factor

1: Initialize the function estimates: $m^{[0]}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^N y_i$
2: **for** $j \leftarrow 1, \ldots, J$ **do**
3: Compute the negative gradient of the quadratic loss function evaluated at the function values of the previous iteration $m^{[j-1]}(\boldsymbol{x}_i)$:

$$u_i^{[j]} = -\frac{\frac{1}{2}\partial(y_i - m(\boldsymbol{x}))^2}{\partial m(\boldsymbol{x})}\bigg|_{m(\boldsymbol{x})=m^{[j-1]}(\boldsymbol{x}_i)} = -(y_i - m^{[j-1]}(\boldsymbol{x}_i)) \,, \quad i = 1, \ldots, N.$$

4: **for** $b \leftarrow 1, \ldots, B$ **do**
5: Fit the bth base learner with the training set $\{(\boldsymbol{x}_i, u_i^{[j]})\}_{i=1}^N$, which gives $\hat{l}_b^{[j]}(\boldsymbol{x})$.
6: **end for**
7: Select the the base-learner $l_{b^*}^{[j]}(\boldsymbol{x})$ with the largest contribution to the fit, i.e. the base-learner that minimizes the sum of squared errors:

$$b^* = \underset{b}{\operatorname{argmin}} \sum_{i=1}^N (u_i^{[j]} - \hat{l}_b^{[j]}(\boldsymbol{x}_i))^2$$

8: Update the current function estimate by adding the best base-learner estimate of the current iteration (j) to the function estimate of the previous iteration $(j-1)$:

$$m^{[j]}(\boldsymbol{x}) = m^{[j-1]}(\boldsymbol{x}) + \nu \hat{l}_{b^*}^{[j]}(\boldsymbol{x})$$

9: **end for**
10: $m(\boldsymbol{x}) = m^{[0]}(\boldsymbol{x}) + \sum_{j=1}^J \nu \hat{l}_{b^*}^{[j]}(\boldsymbol{x})$

give a small value of df (i.e., df $\in [3, 4]$), and that this number should be kept fixed in each boosting iteration.

With univariate P-splines, we have $B = d$ and each base learner depends on one variable. In particular, the final solution in line (10) can be written as an additive model (Hastie and Tibshirani, 1990). If we consider bivariate P-splines, then $B = \binom{d}{2}$ and each base learner depends on a pair of variables. Kneib, Hothorn, and Tutz (2009) gives an introduction to bivariate P-splines to model spatial effects in the context of boosting algorithms. In this work, gradient boosting with univariate and bivariate P-splines will be denoted as BST1 and BST2, respectively. Finally, our implementation of the gradient boosting algorithms depends on the `mboost` package for R (Hothorn et al., 2010).

2.3 Time series forecasting

2.3.1 Introduction

In many different areas, we collect observations about some phenomenon or quantity at regular intervals to form what is called a *time series*. Predicting future observations of a collected sequence of historical observations is called *time series forecasting*. Forecasts are vital since they guide decisions in many areas of scientific, industrial and economic activity such as in meteorology, telecommunication and finance.

Depending on the specific application, forecasts can be required for short-term, medium-term and long-term horizons. Short-term forecasts are the most common required forecasts and include for example demand forecasting. Long-term forecasts are typically useful for strategic planning where

decisions must be taken for a long-term horizon. Forecasts can also be classified into *one-step* or *multi-step-ahead* forecasts where forecasts are required for the next or multiple future observations, respectively.

In this work, we will not make a distinction between the different types of forecasts. Instead, we will consider the general multi-step-ahead forecasting problem defined as follows. Given a univariate time series $\{y_1,\dots,y_T\}$ comprising T observations, the goal is to forecast the next H observations $\{y_{T+1},\dots,y_{T+H}\}$ where H is the forecast horizon.

In order to produce meaningful forecasts, we need to make some assumptions about the underlying dynamics and the information contained in the observed time series. In fact, if the future is completely independent on the past observations then there is no hope to produce good forecasts based on historical observations.

One assumption that is typically assumed is that the time series contains patterns that can be captured and these patterns are informative about its future. The goal of a forecasting method is to learn these patterns in order to produce forecasts. Note that forecasts can be produced in a changing environment as long as there are regularities in the way in which the environment changes and these regularities will continue into the future.

Some time series are easier to forecast than others. How far and how well we can forecast depends on the time series at hand. For example, forecasting the electricity load is much easier than forecasting exchange rates. However, irrespective of the application, the further we want to forecast the more difficult it is. For example, if the temperature of today is 27°C, it is unlikely that the tomorrow's temperature will be 2°C notably because of the continuity of physical phenomena. So, we expect the forecast of tomorrow's temperature to be relatively accurate. However, if we try to forecast the temperature for the day in three months, then the possible values are larger and so the forecast is expected to be poor compared to tomorrow's temperature. In other words, the further ahead we forecast, the more difficult it is to be accurate.

2.3.2 Time series decomposition

Time series data can exhibit many different patterns over time, that can be extracted to better understand the time series or to improve the forecasts. Two types of patterns can be generally identified in a time series: *trend* and *seasonal*.

The trend pattern can be described as a long-term increase or decrease in the level of the time series. The term *trend-cycle* is used when the time series exhibit fluctuations that are not of fixed period (also called a cyclic pattern). In this work, we will use the term "trend", even though it may contain a cyclic pattern.

The seasonal pattern can be described as periodic fluctuations influenced by seasonal factors that are of fixed period.

A time series y_t can be decomposed in three different components: a trend-cycle component Tr_t, a seasonal component S_t and a reminder pattern E_t that contains unidentified patterns.

The final decomposition can take two forms, either an additive or a multiplicative form. In this work, we focus on the additive form, where the time series is decomposed as

$$y_t = \mathrm{Tr}_t + S_t + E_t.$$

Various methods have been proposed in the literature to compute the different components of the decomposition (Cleveland et al., 1990; Ladiray and Quenneville, 2001). Most of these methods are

based on the classical decomposition, a relatively simple procedure. First, the trend is computed using moving averages. Second, after computing the detrended series, the seasonal component for each period is computed by simply averaging the detrended values for that period. Finally, the reminder term is computed by subtracting the computed trend and seasonal components from the initial time series.

A very general and robust method called the STL (Seasonal and Trend decomposition using Loess) decomposition has been developed by Cleveland et al. (1990). STL has many advantages over the other methods: (i) it can handle any type of seasonality, (ii) the flexibility of the trend and the seasonal component can be controlled by the user and (iii) it can be robust to outliers. One limitation of STL is that it can only handle additive decompositions. In this work, we will always use STL for decomposing time series.

In addition to better understand and extract the patterns of a time series, decomposition can also be useful for forecasting. In fact, instead of directly forecasting the initial time series, we can forecast each component separately and then sum the forecasts of each component to obtain the final forecast (Theodosiou, 2011). One advantage of this approach is that the different components are expected to be easier to forecast separately.

Given a time series $\{y_1, \ldots, y_T\}$ and assuming an additive decomposition, the forecasting procedure using decomposition involves the following steps:

- Decompose the time series $\{y_t\}$ into trend component $\{T_t\}$, seasonal component $\{S_t\}$ and reminder term $\{E_t\}$ where $y_t = \mathrm{Tr}_t + S_t + E_t$.
- Compute the deaseasonalized series $\{A_1, \ldots, A_T\}$ where $A_t = y_t - S_t = \mathrm{Tr}_t + E_t$.
- Forecast the deaseasonalized series to obtain $\{\hat{A}_{T+1}, \ldots, \hat{A}_{T+H}\}$.
- Forecast the seasonal component to obtain $\{\hat{S}_{T+1}, \ldots, \hat{S}_{T+H}\}$.
- Combine the forecasts of the deaseasonalized series and the seasonal component to obtain the final forecasts, that is $\{\underbrace{\hat{A}_{T+1} + \hat{S}_{T+1}}_{\hat{y}_{T+1}}, \ldots, \underbrace{\hat{A}_{T+H} + \hat{S}_{T+H}}_{\hat{y}_{T+H}}\}$.

2.3.3 The statistical forecasting perspective

The observed value y_t at time t can be considered as a realization of an underlying random variable. Similarly, a time series $\{y_1, \ldots, y_T\}$ can be seen as a finite realization of a stochastic process which can be described as a family of random variables indexed by time. The term *data generating process* (DGP) is also used to refer to the underlying stochastic process that has generated the observed time series. In practice, the goal is to use the observed time series to infer some properties of the underlying process (if it exists) that will allow us to produce forecasts for future observations.

In particular, the future observations $\{y_{T+1}, \ldots, y_{T+H}\}$ we want to forecast can be considered as random variables since they are unknown quantities. Also, we typically produce forecasts based on some observations, such as some historical data about the quantity we want to forecast. Suppose we denote all that information as $\mathcal{I}$, then we can define the conditional distribution as

$$F(y_t | \mathcal{I}_{t-1}),$$

which represent the distribution of y_t given the knowledge of $\mathcal{I}_{t-1}$.

A process can be said *unpredictable* with respect to $\mathcal{I}_{t-1}$ if

$$F(y_t|\mathcal{I}_{t-1}) = F(y_t),$$

which means that the knowledge of $\mathcal{I}_{t-1}$ has no influence on the uncertainty of y_t. This notion is similar to the conditional independence of random variables.

Modeling the conditional distribution from a single realization of the process is not an easy task and is at an early stage in the forecasting literature. Instead we typically try to estimate the first two moments of this distribution: the conditional mean and the conditional variance. In particular, we will assume the problem of forecasting consists in estimating the conditional mean defined as

$$\mathbb{E}[y_t|\boldsymbol{x}_{t-1}].$$

The forecast produced when estimating the conditional mean represents the average of the possible values the random variable y_t could take given $\boldsymbol{x}_{t-1}$.

This is similar to the problem of regression learning defined in Section 2.1.2. One notable difference however is that the training examples are not independent. The dependence between training examples marks the fundamental difference between a forecasting problem and a general regression learning problem.

In this section, we have seen that the time series can be considered as a realization of an underlying stochastic process, and we can produce forecasts after estimating a *forecasting model* from the time series. More generally, we will refer to a *forecasting method* for any procedure that computes forecasts from historical observations and this procedure does not necessarily need to assume an underlying model. This distinction is similar to the distinction between the two statistical modeling cultures, i.e. data modeling versus algorithmic modeling (Breiman, 2001).

2.3.4 Autoregressive models

In Section 2.1.2, we have seen that the regression model estimates the function f defined in (2.1.1) that maps a vector $\boldsymbol{x}$ of input variables to an output variable y, where each variable can represent different quantities.

With time series data, the input vector and the output variable represent the same quantity at different time instant (also called *lags*), and so the model is called an *autoregressive* model (or process).

An autoregressive process has the form

$$y_t = f(\underbrace{y_{t-1}, \ldots, y_{t-d}}_{\boldsymbol{x}_{t-1}}) + \varepsilon_t, \tag{2.3.1}$$

where the state at time t is represented as a function of the d immediate past values plus a noise term for time t. The process is defined by a function f, a lag order d and a noise term ε_t, which is a stochastic iid process with $\mathbb{E}[\varepsilon_t] = 0$ and $\mathbb{E}[\varepsilon_t] = \sigma^2$.

Several classes of processes arise depending on the type of function f. For example, in the forecasting literature, the function f is often assumed linear and gives rise to the (linear) autoregressive (AR) process defined as

$$y_t = \phi_0 + \phi_1 y_{t-1} + \cdots + \phi_d y_{t-d} + \varepsilon_t \tag{2.3.2}$$

where the state at time t is represented as a *linear combination* of the d immediate past values plus a noise term for time t. With linear models, a weaker condition for the noise term ε_t is used

where it is assumed that the noise is an uncorrelated noise (or white noise) process instead of an independent noise process, as in (2.3.1).

Another popular class of models is the *moving average* (MA) models that use past forecast errors as lagged variables instead of past values of the forecast variable. These models are particularly useful to define a general class of models called the *autoregressive moving average* (ARMA) models that include both an autoregression and a moving average model. The ARMA models can be better alternatives to either pure AR models or pure MA models since they can model a time series with a more parsimonious model that would induce better parameter estimates. In this work, we will not consider MA models and will focus on autoregressive models. One reason for this choice is that it is easier to consider different nonparametric and nonlinear models with autoregressive models. For more details about ARMA models, we refer the reader to Holan, Lund, and Davis (2010).

Another class of models called *nonlinear autoregressive* (NAR) models is included in the general form given in expression (2.3.1) when the function f is nonlinear.

In particular, if the function f is piecewise linear, this leads to the *threshold autoregressive* (TAR) model (Tong and Lim, 1980). The idea is to have a different AR model in different regions. A better alternative to the TAR model is the Smooth Threshold autoregressive (STAR) model (Haggan and Ozaki, 1981; Teräsvirta, 1994) which allows a smooth transition from one AR model to the other. Tong (2011) provides a recent selective review of the development of the threshold model over the past 30 years.

The function f can also be an additive function (Stone, 1985; Hastie and Tibshirani, 1990), which gives rise to an *additive autoregressive* model defined as

$$y_t = f_0 + f_1(y_{t-1}) + \cdots + f_d(y_{t-d}) + \varepsilon_t.$$

Finally, the function f can be any learning model such as a neural network model.

A useful property of a stochastic process is the *stationarity* property which requires the statistical properties of the process to be stable through time. Stationarity is an important property since it implies some regularities in the stochastic process and thus also in the time series realizations of this process. There are two types of stationarity, namely *weak stationarity* and *strict stationarity*.

A process $\{y_t\}$ is *weak stationary* (or *second-order stationary*) if the first and second moments are finite and do not change over time. More formally, a process $\{y_t\}$ is weak stationary if $\mathbb{E}[y_t]$ is a constant that does not depend on time t, and $\mathrm{Cov}(y_t, y_{t+\tau})$ does not depend on t but only on τ.

A process $\{y_t\}$ is *strictly stationary* if $(y_1, \ldots, y_n)$ and $(y_{1+\tau}, \ldots, y_{n+\tau})$ have the same joint distribution for any integer $n \geq 1$ and any integer τ.

Checking whether a linear model is stationary is relatively straightforward. However, it is much harder to show that a nonlinear model is strictly stationary.

A large part of the forecasting theory is based on stationary models. However, in practice, time series are often non-stationary since they exhibit some persistent increase or decrease over time.

For some time series, called *difference-stationary*, it is possible to make it stationary by applying a transformation called *differencing*, which consists in computing the differences between consecutive observations (Hyndman and Athanasopoulos, 2014). Being able to make the time series stationary is important if we want to take advantage of the theory of stationary models.

The *first-order differencing* of a series provides a new series that represents the change between consecutive observations in the original series and is defined as

$$y'_t = y_t - y_{t-1}.$$

Similarly, the *second-order differencing* is the differencing applied to an already first-order differenced series. In practice, first-order differencing will already transform the original series into a stationary series. Sometimes, a second-order differencing is required but it is very rare to need to go beyond second-order differences.

For a given time series, a visual inspection can help us to decide whether the time series is stationary or if differencing is needed. Except for some obvious cases, such as a clear trend, deciding if the time series needs differencing is not an easy task. In addition, the number of time series to process can be huge and therefore the visual inspection can be unfeasible in practice. To reduce the subjectivity in the differencing process and to make it more automatic, one can use a *unit root* test to determine if differencing is required (Baltagi, 2001).

Unit root tests are hypothesis tests of stationarity that can be used to determine whether differencing is required. Different unit root tests are based on different assumptions and have different null-hypotheses. Two popular test are the *Augmented Dickey-Fuller* (ADF) test and the *Kwiatkowski-Phillips-Schmidt-Shin* (KPSS) test.

The null-hypothesis for the ADF test is that the series is non-stationary while for the KPSS test it is reversed, and the null-hypothesis is that the series is stationary. Both tests can be used to decide whether differencing is required but the KPSS test is often used to complement the ADF test.

In this work, we have focused on autoregressive models but there are other methods that can be used for forecasting, including *exponential smoothing* methods (Gardnerjr, 2006). These methods are not based on an autoregression and produce forecasts by taking a weighted average of past observations with the weights decaying exponentially as the observations get older. Autoregressive models and exponential smoothing methods are the most popular forecasting methods.

It is worth noting that a large class of forecasting models can be represented in a *state space* form including exponential smoothing and autoregressive models (Durbin and Koopman, 2001; Hyndman, Koehler, et al., 2008). There are many advantages in writing forecasting methods in a state space form, including the possibility of studying their properties in a common mathematical framework and the easier software implementation that can include a large spectra of models.

We will not go into more details about the other existing forecasting methods, but we refer the interested reader to the following key references: Hyndman and Athanasopoulos (2014); Teräsvirta, Tjostheim, and Granger (2010); Fan and Yao (2003); Brockwell and Davis (2002); Chatfield (2000).

2.3.5 Autoregressive model selection

Let us assume a time series $\{z_1, \ldots, z_T\}$ with T observations is sampled from the process given in (2.3.1), and we need to estimate the function f and the lag order d.

In the forecasting literature, the linear model given in expression (2.2.1) has often been adopted to estimate the function f. In particular, the parameters are often estimated by OLS (see (2.2.3)) and the lag order d is often selected by model-dependent penalties such as AIC or BIC (see Section 2.1.3). If the function f is effectively linear as in expression (2.3.2), and under some conditions, OLS enjoys a number of properties such as unbiasedness and consistency (Wooldridge, 2012).

In this work, we have considered the different learning algorithms presented in Section 2.2 to estimate the function f, and the lag order and the hyperparameters have been selected using the learning procedure presented in Section 2.1.4.

For a given lag order p and forecast horizon h, the time series $\{z_1,\ldots,z_T\}$ can be embedded into a dataset $\mathcal{D}$ made up of N input-output pairs as follows:

$$\mathcal{D} = \{(\boldsymbol{x}_i, y_i) : \boldsymbol{x}_i = [z_i,\ldots,z_{p+i-1}] \text{ and } y_i = z_{p+i+h-1}\}_{i=1}^{N=T-p-h+1}.$$

To select the lag order d and the hyperparameters, we can use different validation methods ss explained in Section 2.1.3. For example, we can adopt an holdout approach where the data set $\mathcal{D}$ is split into a training set $\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{M}$ and a validation set $\mathcal{D}_{\text{val}} = \{(\boldsymbol{x}_i, y_i)\}_{i=M+1}^{N}$ for a given value of M.

In Section 2.1.3, we have seen that cross-validation is a better alternative to the holdout approach since it involves multiple splitting. However, because the examples from a time series are not independent, we cannot use the standard cross-validation (see expression (2.1.8)) to select the lag order and the hyperparameters since it can lead to a biased model selection.

For example, Hart and Wehrly (1986) and Opsomer, Wang, and Yang (2001) show that there is overfitting when cross-validation is used to select the bandwidth in kernel regression. Modifications of cross-validation have been proposed where the main idea has been to withhold some of the neighbors of the current validation point to eliminate the dependence effect (see for example Chu and J Marron, 1991; Burman, Chow, and Nolan, 1994; Hart, 1994; Racine, 2000; Carmack et al., 2009; Brabanter and Brabanter, 2011). However, it is not easy to find the correct number of neighbors to withhold in real-world applications. On the other hand, Burman and Nolan (1992) showed that cross-validation can be applied when the errors follow a stationary Markov process in a specific framework, and Bergmeir and Benítez (2012) did not find practical consequences of correlated errors on cross-validation in a large experimental study.

An alternative to the standard cross-validation is the time series cross-validation approach (also called forecast evaluation with a rolling origin (Tashman, 2000)) where the first part of the (embedded) dataset $\mathcal{D}$ is used as training set and the remaining part is used as validation/rolling set. In particular, the model is fitted using the training set and the errors are computed using the validation set. We repeat this procedure multiple times, and each time one example moves from the validation set to the training set. The procedure can be summarized as follows:

1. Split the data set $\mathcal{D}$ into training set $\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{j}$ and validation set $\mathcal{D}_{\text{val}} = \{(\boldsymbol{x}_i, y_i)\}_{i=j+1}^{N}$.
2. Fit the model using $\mathcal{D}_{\text{train}}$, then compute the predictions $\hat{y}_i$ and the corresponding errors ($e_i = y_i - \hat{y}_i$) for all the validation examples (i.e. $i = j+1,\ldots,N$).
3. Compute the MSE from $e_{j+1},\ldots,e_N$.
4. Repeat steps $1-3$ for $j = M,\ldots,N-1$ where M is the minimum number of examples required to fit a model. The final cross-validation error is the average of the errors computed in step 3 for the different runs.

Instead of repeating the steps $1-3$ for $j = M,\ldots,N-1$, it is also possible to increase the training set $\mathcal{D}_{\text{train}}$ by several values at a time. In other words, we can run step 4 for $j = M, M+k, M+2k,\ldots,N-1$ for $k \neq 1$. This can be useful for large data sets to reduce the computational time of the procedure.

2.3.6 Evaluating forecasts accuracy

We often need to compare the forecast accuracy of different forecasting methods on several data sets, notably in forecasting competitions. Many different accuracy measures have been proposed to evaluate the performance of forecasting methods. Hyndman and Koehler (2006) give a complete overview and compare different forecast accuracy measures, while Tashman (2000) provides a review of out-of-sample tests for comparing forecasting methods. In the following, we present the two accuracy measures used in the real-world experiments of this work to evaluate the performance of forecasting methods.

Let us denote by $\{\hat{y}_{T+1},\ldots,\hat{y}_{T+H}\}$ the forecasts and by $\{y_{T+1},\ldots,y_{T+H}\}$ the true future observations of the time series $\{y_1,\ldots,y_T\}$. The forecast error at horizon h is defined as $e_{T+h} = \hat{y}_{T+h} - y_{T+h}$ where $h \in \{1,\ldots,H\}$. The two commonly used error measures that are based on e_{T+h} are the absolute errors $|e_{T+h}|$ and the squared errors e^2_{T+h}. However, because the error e_{T+h} is on the same scale as the data, it cannot be used to compare series that are on different scales.

There are mainly two alternatives for scale-independent error measures: *percentage* and *scaled* errors. Let us assume we have produced forecasts for M different time series and let us define the forecast error for the mth time series at horizon h as $e^m_{T+h} = \hat{y}^m_{T+h} - y^m_{T+h}$ where $m \in \{1,\ldots,M\}$ and $h \in \{1,\ldots,H\}$. We will present one error measure from each type.

One example of the percentage error is the symmetric mean absolute percentage error (sMAPE) which is defined as

$$\text{sMAPE}_h = \text{mean}(\text{sAPE}^m_h), \tag{2.3.3}$$

where

$$\text{sAPE}^m_h = 200 * \frac{|e^m_{T+h}|}{(\hat{y}^m_{T+h} + y^m_{T+h})}.$$

To avoid sMAPE takes negative values, we can define it with absolute values in the denominator, that is

$$\text{sAPE}^m_h = 200 * \frac{|e^m_{T+h}|}{(|\hat{y}^m_{T+h}| + |y^m_{T+h}|)}.$$

One example of the scaled error is the mean absolute scaled error (MASE) proposed by Hyndman and Koehler (2006). It is defined as

$$\text{MASE}_h = \text{mean}(\text{ASE}^m_h), \tag{2.3.4}$$

where

$$\text{ASE}_h = \frac{|e^m_{T+h}|}{\frac{1}{T-1}\sum_{t=2}^{T}|y_t - y_{t-1}|}.$$

Chapter 3

An overview of strategies for multi-step-ahead time series forecasting

3.1 Preamble

Multi-step-ahead time series forecasting has been mainly studied in econometrics and statistics. In the last two decades, there have been few studies involving algorithms and methodologies from machine learning. However, the lack of communication between these different fields made it difficult to have a broad overview of the different developments in this area. This chapter addresses this issue by providing an overview of the research on multi-step-ahead forecasting undertaken in the different fields.

When facing a multi-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, applied iteratively for the desired number of steps. With the direct strategy, an horizon-specific model is estimated and forecasts are computed directly by the estimated model for each forecast horizon. We describe the two strategies in Section 3.3, including how the different models are selected.

A large part of the literature has focused on comparing these two strategies and investigating under which conditions one strategy is better than the other. We provide an overview of the different comparisons in Section 3.4 for both linear and nonlinear models.

In Section 3.5, we present the alternative strategies that have been proposed, either by improving recursive or direct forecasts, or by developing hybrid strategies that combine properties of both recursive and direct strategies.

In Section 3.6, we review the few studies that considered machine learning algorithms for time series modeling and forecasting. Finally, we provide a brief summary with concluding remarks in Section 3.7.

3.2 Multi-step forecasting

Given a univariate time series $\{y_1, \ldots, y_T\}$ comprising T observations, we want to forecast the H next observations of the time series, $\{y_{T+1}, \ldots, y_{T+H}\}$.

Time series of different fields or applications can have different resolutions (e.g. yearly, monthly, daily, hourly, etc), and that could lead to different time series lengths T. Also, depending on the required horizon H, forecasts can be typically classified into short, medium or long term forecasts.

Typically, the further in the future we attempt to forecast the harder it can be because of the larger uncertainty.

We will assume the time series $\{y_1,\dots,y_T\}$ is a realization of an autoregressive process of the form

$$y_t = f(\boldsymbol{x}_{t-1}) + \varepsilon_t \;\text{ with }\; \boldsymbol{x}_{t-1} = [y_{t-1},\dots,y_{t-d}]', \tag{3.2.1}$$

which is specified by a function f, a lag order (or number of lagged variables) d and a noise term $\{\varepsilon_t\}$, which is a stochastic iid noise process with $\mathbb{E}[\varepsilon_t] = 0$ and $\mathbb{E}[\varepsilon_t^2] = \sigma^2$.

Different forms or values of these three components can produce time series with very different characteristics. The autoregressive process in (3.2.1) is also called the data generating process (DGP) for the time series $\{y_1,\dots,y_T\}$. In particular, time series generated by different DGPs can have very different forecastability properties.

In practice, we do not have access to the true DGP (if it exists). The only information we have is one time series realization from that DGP. The goal is to produce the best forecasts (according to an accuracy measure) based on this time series realization.

We will consider the mean squared error (MSE) as forecast error measure. Let us denote $g(\boldsymbol{x}_t;\hat{\boldsymbol{\theta}}_T;h)$ the h-step ahead forecast from input $\boldsymbol{x}_t$ using the set of parameters $\hat{\boldsymbol{\theta}}_T$, that have been estimated from the time series $\boldsymbol{Y}_T = \{y_1,\dots,y_T\}$. Then, the MSE at horizon h is defined as

$$\text{MSE}_h(\boldsymbol{x}_t) = \mathbb{E}_{\varepsilon,\boldsymbol{Y}_T}\left[(y_{t+h} - g(\boldsymbol{x}_t;\hat{\boldsymbol{\theta}}_T;h))^2 \mid \boldsymbol{x}_t\right],$$

It can be shown that the optimal h-step ahead forecast, i.e. the forecast that has the minimum MSE at horizon h, is the conditional expectation given by $\mu_{t+h|t} = \mathbb{E}[y_{t+h}|\boldsymbol{x}_t]$ (see, for example, Clements and Hendry, 1998, p.44). For the reminder of the thesis, we will assume the goal of multi-step-ahead forecasting is to estimate $\mu_{t+h|t}$ for $h = 1,\dots,H$ using one time series realization $\{y_1,\dots,y_T\}$.

For one-step-ahead forecasts, that is $h = 1$, we have the expression $\mu_{t+1|t} = f(\boldsymbol{x}_t)$. So, the problem of forecasting reduces to the estimation of the function f and the lag order d, given in expression (3.2.1).

For multi-step forecasts, that is $h > 1$, the problem is more difficult and does not necessarily reduces to the estimation of the function f and the lag order d. In fact, to produce multi-step-ahead forecasts, we need a forecasting strategy which typically involves estimating one or more models which are not necessarily of the same form as f and may not have the same lag order d as the function f.

3.3 The recursive and direct forecasting strategies

Multi-step-ahead forecasting can be handled recursively, where a single time series model is estimated and each forecast is computed using previous forecasts. Another approach builds a separate time series model for each forecasting horizon, and forecasts are computed directly by the estimated model (Chevillon, 2007; Sorjamaa, Hao, Reyhani, et al., 2007).

The *recursive* strategy centers on building a model of the same form as (3.2.1), aiming to minimize the one-step-ahead forecast error variance. In others words, it estimates a model of the form

$$y_t = m(\boldsymbol{z}_{t-1};\boldsymbol{\phi}) + e_t\,, \tag{3.3.1}$$

where $\boldsymbol{z}_{t-1} = [y_{t-1}, \ldots, y_{t-p}]'$, p is an estimate of the true lag order d (i.e. $p = \hat{d}$), $\boldsymbol{\phi} = [\boldsymbol{\psi}, \boldsymbol{\beta}]$ is the model parameters where $\boldsymbol{\psi}$ is a set of hyperparameters and $\boldsymbol{\beta}$ is a set of parameters, and $e_t = f(\boldsymbol{x}_{t-1}) - m(\boldsymbol{z}_{t-1}; \boldsymbol{\phi}) + \varepsilon_t$ is the forecast error of the model m with $\mathbb{E}[e_t] = 0$.

The lag order and the hyperparameters are estimated by minimizing one-step-ahead errors as follows

$$(p, \hat{\boldsymbol{\psi}}) = \underset{p,\boldsymbol{\psi}}{\operatorname{argmin}} \sum_{(\boldsymbol{z}_{t-1}, y_t) \in \mathcal{D}_{\text{val}}} [y_t - m(\boldsymbol{z}_{t-1}; \boldsymbol{\psi}, \hat{\boldsymbol{\beta}})]^2, \tag{3.3.2}$$

where $\mathcal{D}_{\text{val}}$ is the validation set and $\hat{\boldsymbol{\beta}}$ is estimated using the training set $\mathcal{D}_{\text{train}}$[1], as described in Section 2.3.5.

The recursive strategy ensures that the fitted model m matches the assumed data generating process f as closely as possible. However, the minimization of one-step forecast errors does not necessarily guarantee the minimum for h-step ahead errors.

Multi-step forecasts are obtained dynamically by repeatedly iterating the model and by plugging in the missing future values with their respective forecasts. More precisely, the forecasts $\hat{\mu}_{T+h|T}$ are computed as follows

$$m^{(h)}(\boldsymbol{z}_T; \hat{\boldsymbol{\phi}}) = \begin{cases} m(\boldsymbol{z}_T; \hat{\boldsymbol{\phi}}) & \text{if } h = 1, \\ m(m^{(h-1)}(\boldsymbol{z}_T; \hat{\boldsymbol{\phi}}), \ldots, m^{(1)}(\boldsymbol{z}_T; \hat{\boldsymbol{\phi}}), y_T, \ldots, y_{T-p+h}; \hat{\boldsymbol{\phi}}) & \text{if } 1 < h \leq p, \\ m(m^{(h-1)}(\boldsymbol{z}_T; \hat{\boldsymbol{\phi}}), \ldots, m^{(h-p)}(\boldsymbol{z}_T; \hat{\boldsymbol{\phi}}); \hat{\boldsymbol{\phi}}) & \text{if } h > p. \end{cases} \tag{3.3.3}$$

With linear models, the forecasts of the recursive strategy are also sometimes called "iterated multi-step" (IMS) forecasts (e.g., Chevillon, 2007). When used with parametric nonlinear models, the recursive strategy is called the deterministic method (see, for example, Clements and Hendry, 1998, p.94) or the naive method (see, for example, Teräsvirta, Tjostheim, and Granger, 2010, p.348). Recursive forecasts can also be computed using the exact, the Monte Carlo and the bootstrap methods, as will be explained in Section 3.4.2. In this thesis, "recursive forecasts" will refer to the recursive strategy defined in expression (3.3.3).

A variation on the recursive strategy is to use a different set of parameters for each forecasting horizon:

$$(p_h, \hat{\boldsymbol{\psi}}_h) = \underset{p,\boldsymbol{\psi}}{\operatorname{argmin}} \sum_{(\boldsymbol{z}_{t-h}, y_t) \in \mathcal{D}_{\text{val},h}} [y_t - m^{(h)}(\boldsymbol{r}_{t-h}; \boldsymbol{\psi}, \hat{\boldsymbol{\beta}}_h)]^2, \tag{3.3.4}$$

where $\mathcal{D}_{\text{val},h}$ is the validation set for horizon h, $\hat{\boldsymbol{\beta}}_h$ is estimated using $\mathcal{D}_{\text{train},h}$, the training set for horizon h, and $m^{(h)}$ is defined in (3.3.3). In other words, we select the parameters $\hat{\boldsymbol{\beta}}_h$, the hyperparameters $\hat{\boldsymbol{\psi}}_h$ and the lag order p_h that when used recursively h times minimize h-step-ahead errors. In this variant, forecasts are still obtained recursively but the model parameters are allowed to change with the horizon.

When the learning algorithm is nonparametric (i.e. $\boldsymbol{\beta} = \emptyset$), expression (3.3.4) can be easily computed, but if the learning algorithm depends on some parameters $\boldsymbol{\beta}$, the parametric identification requires a nonlinear optimization problem.

This variant has also been considered with linear models (i.e. when $\boldsymbol{\psi} = \emptyset$), where the parameters (or coefficients) are estimated so that they minimize h-step-ahead errors. For example, Bhansali (2002) discusses this variant of the recursive strategy in Section 9.5 (see expression 9.24) to select the parameters of an AR(1) model by minimizing two-step-ahead errors.

[1] To simplify the notations for all the strategies, we will consider a holdout approach but in practice we can adopt a time series cross-validation approach as described in Section 2.3.5.

The *direct* strategy tailors the forecasting model directly to the forecast horizon. In other words, different forecasting models are used for each forecast horizon:

$$y_t = m_h(\boldsymbol{r}_{t-h}; \boldsymbol{\gamma}_h) + e_{t,h}, \tag{3.3.5}$$

where, in contrast to the recursive strategy, m_h is a direct model, $\boldsymbol{r}_{t-h} = [y_{t-h}, \ldots, y_{t-h-p_h}]'$, p_h is the lag order for horizon h, $\boldsymbol{\gamma}_h = [\boldsymbol{\psi}_h, \boldsymbol{\beta}_h]$ is the model parameters for horizon h where $\boldsymbol{\psi}_h$ is a set of hyperparameters and $\boldsymbol{\beta}_h$ is a set of parameters, $e_{t,h}$ is the forecast error of the model m_h at horizon h and $h = 1, \ldots, H$.

For each model m_h, the lag order and the hyperparameters are estimated by minimizing direct h-step-ahead errors as follows:

$$(p_h, \hat{\boldsymbol{\psi}}_h) = \underset{p,\boldsymbol{\psi}}{\operatorname{argmin}} \sum_{(\boldsymbol{r}_{t-h}, y_t) \in \mathcal{D}_{\text{val},h}} [y_t - m_h(\boldsymbol{r}_{t-h}; \boldsymbol{\psi}, \hat{\boldsymbol{\beta}}_h)]^2. \tag{3.3.6}$$

where $\mathcal{D}_{\text{val},h}$ is the validation set for horizon h and $\hat{\boldsymbol{\beta}}_h$ is estimated using $\mathcal{D}_{\text{train},h}$, the training set for horizon h.

In contrast to the recursive strategy, the direct strategy does not match the model used for forecasting with the assumed model, given in expression (3.2.1). However, the direct strategy minimizes the h-step ahead forecast errors instead of one-step errors as the recursive strategy. Also, because the direct strategy estimates H models rather than one model as the recursive strategy, it involves a heavier computational load than the recursive strategy.

Multi-step forecasts are obtained for each horizon from the corresponding model, $\hat{\mu}_{T+h|T} = m_h(\boldsymbol{r}_T; \hat{\boldsymbol{\gamma}}_h)$. This is sometimes also known as "direct multi-step" (DMS) forecasting (e.g., Chevillon, 2007).

Throughout this thesis, to highlight the difference between recursive and direct forecasts, we will use $\boldsymbol{\phi}$ to denote the model parameters of the recursive strategy and $\boldsymbol{\gamma}$ for the direct strategy. It is worth noting that when $\boldsymbol{\psi} = \emptyset$, for example with the LIN model (see Section 2.2.1), then $\boldsymbol{\phi}$ and $\boldsymbol{\gamma}$ will only contain the parameters $\boldsymbol{\beta}$. Therefore, in both expressions (3.3.2) and (3.3.6), we will only select the lag order p. When there is no parameters, i.e. $\boldsymbol{\beta} = \emptyset$, for example with the KNN model (see Section 2.2.3), then $\boldsymbol{\phi}$ and $\boldsymbol{\gamma}$ will not contain the parameters $\boldsymbol{\beta}$, and therefore, there will be no parametric identification step.

3.4 Recursive or direct forecasts?

When we want to generate multi-step-ahead forecasts, we have to decide whether to use the recursive or direct forecasting strategy.

If we ignore the estimation variance, i.e. we have infinite amount of data ($T = \infty$), and if we know the function f and the true lag order d in expression (3.2.1), then the recursive strategy and the direct strategy would be equivalent when f is linear, but not when f is nonlinear (Fan and Yao, 2003). Because of minimizing the one-step-ahead prediction errors, when f is nonlinear the recursive strategy is asymptotically biased (Brown and Mariano, 1984; Lin and Granger, 1994) while, under appropriate conditions, the direct strategy achieves the optimal mean squared error (Atiya et al., 1999).

In practice, because multi-step forecasts require the estimation of one or multiple models from a finite-sample time series, as can be seen in Eq. (3.3.1) and (3.3.5), the choice between recursive and direct multi-step forecasts involves a trade-off between bias and estimation variance of the

forecasts. Which component, the bias or the estimation variance, contributes most to the mean squared forecast error depends on many interacting factors including the model complexity, the (unknown) underlying DGP, the time series length and the forecast horizon. Therefore, the question of which multi-step strategy is best is an empirical one.

Although the performance of recursive and direct forecasts depends on many factors, a number of studies have compared the two strategies in specific settings under certain conditions. In particular, a large part of the econometrics literature has focused on linear models. Some work involved nonlinear models but were limited compared to linear models. In the following two sections, we provide an overview of the literature for both linear and nonlinear models.

3.4.1 Linear models

The importance of multi-step forecasting in econometrics has contributed to the large number of studies that compare the recursive and direct strategies. A large part of the literature often involves comparing recursive and direct linear forecasts for some specific DGP with a specific estimation method, and discussing the conditions under which one or the other is better. A summary of the findings is given by Bhansali (1999) and Chevillon (2007).

The common practice has been to assume that the postulated model is the true model (i.e., that m and f are asymptotically equivalent). In that case, generating multi-step forecasts reduces to estimate the model parameters and use it recursively. Multi-step recursive forecasts is often advocated in standard time series and econometrics textbooks. One reason is that often time series models are estimated and producing multi-step forecasts from these models is straightforward with the recursive strategy.

However, important contributions to the theoretical literature on recursive and direct forecasting have been done when misspecified models (i.e., that m and f are not asymptotically equivalent) have been considered, as explained below.

The idea of "direct" forecasting can be at least traced back to Cox (1961) who considered the estimation of an exponentially weighted moving average model and an AR model when the true DGP is either AR or ARMA. The author found that multi-step forecasts can be made more robust if the smoothing parameter is allowed to change with the forecast horizon.

What is meant by "direct" is that the parameters are selected by minimizing h-step ahead errors as follows

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \sum_{t} [y_t - m^{(h)}(\boldsymbol{z}_{t-h}; \boldsymbol{\beta})]^2.$$

where $m^{(h)}$ is defined in (3.3.3).

However, this is one way to apply "direct" multi-step estimation where the parameters of the postulated model are computed by minimizing the implied h-step-ahead errors instead of the traditional one-step-ahead errors. The other approach consists in having a different model at each horizon, which is generally called the direct strategy. Lin and Tsay (1996) and Bhansali (1999) have pointed out these ways to proceed with "direct" multi-step estimation.

Findley (1983) proposed to mach the estimation and forecasting criterion functions by showing that the optimal values of the parameters of misspecified AR models depend on the forecast horizon.

Stoica and Nehorai (1989) extended the work on multi-step estimation of Findley (1983) for ARMA models. The authors proposed various algorithms for the multi-step parameter estimation that involves a nonlinear optimization problem. Also, the authors confirmed the importance of

model misspecification as a justification for multi-step estimation and have found that under-parametrization of the model can benefit multi-step estimation.

Weiss (1991) showed that, with misspecified models, it is asymptotically optimal to minimize the sum of squared in-sample multi-step forecast errors when the forecast evaluation criterion is the expected sum of squared multi-step forecast errors.

Multi-step estimation for ARMA processes was considered in Tiao and Xu (1993) and Clements and Hendry (1996). Tiao and Xu (1993) showed that more efficient forecasts can be obtained with multi-step estimation when the model is misspecified. Clements and Hendry (1996) performed an analysis of multi-step estimation for stationary and integrated processes.

Findley, Pötscher, and Wei (2004), Ing (2003) and Ing (2004) considered very general settings. Findley, Pötscher, and Wei (2004) provides an estimation theory for fitting misspecified models by multi-step error minimization. Under mild conditions, Ing (2003) shows for stationary cases that the MSE of the recursive forecasts is asymptotically greater than direct forecasts. Ing (2004) shows that correctly identifying the true model order for autoregressive processes does not guarantee obtaining the optimal forecasts in a MSE sense.

Chevillon and Hendry (2005) provide a useful review of some of the literature comparing recursive and direct forecasting strategies, and explore in detail the differences between the strategies for vector autoregressive (VAR) models applied to stationary and difference-stationary time series. They show that for these models, when f, m and m_h are all assumed multivariate linear, and under various mis-specification conditions, then the recursive MSE is greater than the direct MSE.

Chevillon and Hendry (2005), Chevillon (2008) and Proietti (2011) have considered misspecified (AR)IMA processes. Some researchers - such as Tiao and Tsay (1994), Bhansali (1996); Bhansali (1997), Brodsky and Hurvich (1999) and Bhansali and Kokoszka (2002) - have focused on long memory processes. A frequency domain approach for multi-step estimation has also been proposed in Haywood and Wilson (1997) and McElroy and Wildi (2013).

Despite the practical importance of multi-step forecasting, there have been limited empirical studies comparing recursive and direct strategies with linear models. Kang (2003) studied univariate AR models on nine U.S. economic time series and found mixed results by concluding that the direct strategy "may or may not improve forecast accuracy" compared to the recursive strategy. Marcellino, Stock, and Watson (2006) compared the two strategies in a "large-scale experiment" using data on 170 U.S. macroeconomic time series and found strong evidence in favor of recursive forecasts. Proietti (2011) finds there are no significant gains in predictive accuracy arising from the direct strategy when dealing with the level of economic time series in real terms and confirms the findings of Marcellino, Stock, and Watson (2006). Pesaran, Pick, and Timmermann (2011) use the same data set as Marcellino, Stock, and Watson (2006) and found that the recursive forecasts are better than direct forecasts with short time series and for long forecast horizons.

In summary, with linear models, the theoretical literature tends to conclude that if the model is correctly specified, the recursive strategy benefit from more efficient parameter estimates, while the direct strategy is more robust to model misspecification. However, empirical studies often found superior performance of recursive forecasts, especially for long horizons. One explanation is that with short time series, the efficiency of the recursive forecasts are more important than the robustness of direct forecasts.

3.4.2 Nonlinear models

In contrast to the large number of studies comparing recursive and direct forecasts with linear models, there have been less work with nonlinear models. More generally, nonlinear forecasting is

still in its infancy compared to linear forecasting notably due the additional complexity and the heavier computational load when analyzing nonlinear forecasting models (Gooijer and Hyndman, 2006).

As with the linear model, nonlinear direct forecasts are obtained by regressing the h-step ahead variable with past lagged variables. However, there are four alternatives to compute recursive forecasts: the traditional recursive strategy (also called naive or deterministic), the exact, the Monte Carlo and the bootstrap methods (see, for example, Teräsvirta, Tjostheim, and Granger, 2010, p.348).

Nonlinear direct forecasts have been considered in Stock and Watson (1999) where the authors compared linear and nonlinear models for forecasting macroeconomic time series. Nonlinear recursive forecasts with the bootstrap methods have been considered in Teräsvirta, Dijk, and Medeiros (2005) also for forecasting macroeconomic time series.

Brown and Mariano (1984) considered nonlinear parametric models for f and show that the bias of the recursive strategy with large samples is $\mathcal{O}(1)$, i.e. there is asymptotic bias. The authors also showed that the exact and bootstrap forecasts have bias $\mathcal{O}(1/T)$ where T is the length of the time series. Lin and Granger (1994) extended the results of Brown and Mariano (1984) to non-parametric models. They also compared the different recursive approaches including the direct strategy in a simulation study with different DGPs. The authors concluded that the kernel bootstrap approach is a better alternative to the recursive strategy.

Atiya et al. (1999) have also compared the MSE of both strategies with general nonlinear models for two-step ahead forecasting and show that the MSE of the direct strategy is asymptotically smaller than the MSE of the recursive strategy.

Compared to the recursive strategy, the exact, the Monte Carlo and the bootstrap methods require good knowledge of the function f and/or the distribution of the noise term ε_t (Teräsvirta, Tjostheim, and Granger, 2010). In practice, the (naive) recursive strategy is often used due to its simplicity both in implementation and computational resources.

As with the linear models, the number of empirical studies comparing recursive and direct forecasts with nonlinear models is also limited.

Atiya et al. (1999) found direct forecasts to perform better than recursive forecasts with neural networks applied to the problem of river flow forecasting. Kline (2004) found empirical evidence in favor of the direct strategy compared to the recursive strategy on a subset of the quarterly time series from the M3 competition (see Appendix A.1.1).

Sorjamaa, Hao, Reyhani, et al. (2007) compared recursive and direct forecasts with the KNN model for electric load forecasting, and found better forecasts with the direct strategy. They also pointed out the heavier computational time of the direct strategy when input selection is applied.

Hamzaçebi, Akay, and Kutay (2009) compared recursive and direct forecasts with neural networks on the six time series of Box and Jenkins (1976), and found superior performance of the direct strategy.

Kock and Teräsvirta (2011) applied a small simulation study to compare recursive and direct forecasts where the DGP is a simple neural network model. They found recursive forecasts to be explosive with a neural network that include a linear unit, as explained by Teräsvirta, Dijk, and Medeiros (2005). Since, in that case, recursive and direct forecasts cannot be compared for long horizons, they applied an "insanity filter" as explained by Swanson (1995). They found that the recursive strategy is superior to the direct strategy for sufficiently long horizons.

Finally, in addition to the forecasting strategy, there have also been an interest in determining whether nonlinear models yield better forecasts than linear models, especially for economic time series. The answer to that question has been negative in many cases. One argument put forward is that economic time series are often weakly nonlinear which does not justify the use of nonlinear models since linear models already provide a good approximation. Also, because nonlinear behavior seems not to occur very frequently, there is no benefit in using nonlinear models. Finally, another argument is that even if the time series is nonlinear, overfitting can prevent nonlinear models to perform better than linear ones, particularly with short time series. We refer the reader to Teräsvirta (2006) for more details about the comparison between linear and nonlinear models.

In summary, the recursive strategy with nonlinear models is known to be asymptotically biased. Alternatives, such as the exact, the Monte Carlo and the bootstrap methods have been proposed but these methods require additional knowledge either about the function f or the noise distribution. The direct strategy is often preferred over the recursive strategy with nonlinear models. In particular, empirical studies often found superior performance of the direct strategy compared to the recursive strategy. Many studies also found that linear models perform better than nonlinear models with real-world time series, that are typically short and weakly nonlinear.

3.5 Alternative forecasting strategies

In addition to comparing recursive and direct strategies, alternative strategies have been proposed to improve multi-step forecasts or decrease the computational time, for example by changing the way the parameters are selected and the way the forecasts are generated.

3.5.1 Improving recursive forecasts

For the recursive strategy, the main improvement has been to consider h-step ahead forecast errors instead of the usual one-step-ahead errors.

With linear models, we have seen in Section 3.4.1 that the loss of forecasting performance when using an incorrect model with parameters tuned for multi-step forecasts is expected to be smaller than parameters tuned for one-step-ahead forecasts.

With neural networks, h-step ahead criteria have been considered with recurrent neural networks (Rumelhart, Hinton, and Williams, 1986; Werbos, 1988; Elman, 1990) using backpropagation through time (Werbos, 1990). Atiya et al. (1999) found superior performance for recursive forecasts with recurrent neural networks compared to the usual one-step-ahead neural network for the problem of river flow forecasting.

With local models, McNames (1998) proposed to find the nearest trajectory segments, rather than the conventional nearest neighbors. The method has been successfully used in the K.U. Leuven time series competition (Suykens and Vandewalle, 2000) for forecasting (noisy-free) chaotic time series (McNames, Suykens, and Vandewalle, 1999). More recently, Wichard (2010) considered an ensemble of models including nearest trajectory models to forecasting the time series from the NN5 forecasting competition (see Section A.1.2).

Bontempi, Birattari, and Bersini (1999) proposed an extension of the Predicted REsidual Sum of Squares (PRESS) leave-one-out statistic (Allen, 1974), called iterated PRESS, to select local models for multi-step forecasting. The authors found superior performance of the iterated PRESS compared to the non-iterated approach on three (noise-free) chaotic time series, from the Santa Fe (Weigend and Gershenfeld, 1994) and the KU Leuven competitions (Suykens and Vandewalle, 2000).

3.5.2 Improving direct forecasts

For the direct strategy, the focus has been on how to exploit the fact that the errors for different models are serially correlated.

Lee and Billings (2003) suggested to estimate the multi-step forecast errors and include them as inputs for the next model. The goal is to obtain a smaller covariance of the parameter estimates which is directly related to the forecast accuracy. With both linear and nonlinear simulated time series, the authors show that the proposed strategy outperforms both the recursive and direct strategies.

Chen, Yang, and Hafner (2004) proposed a multi-stage direct strategy with nonparametric smoothing techniques. The rationale is to exploit substantial information about the conditional mean function that is contained in the intermediate horizons. In particular, the authors proposed to use a smoother version of the output variable for horizon h by replacing it with predictions generated by the estimated function. Under various conditions, the authors showed that the new strategy has smaller asymptotic MSE than the traditional direct strategy.

Pesaran, Pick, and Timmermann (2011) proposed to view the H regressions of the direct strategy as a set of seemingly unrelated regression equations (Fiebig, 2001) that create non-overlapping blocks from the data which are spaced apart by the length of the forecast horizon. The authors also proposed modified versions of the AIC that take into account the autocorrelated residuals in the forecast models. Monte Carlo experiments show the gain in efficiency with the proposed method.

Kline (2004) and Franses and Legerstee (2009) have considered the problem of jointly estimating all the direct models. Kline (2004) considered multi-output neural networks where each output node corresponds to one forecast horizon. After learning the multi-output neural network with backpropagation, direct forecasts can be generated in "one shot". The authors found superior performance of the multi-output approach compared to the direct approach on the quarterly time series from the M3 competition (See appendix A.1.1).

Franses and Legerstee (2009) applied the Partial Least Squares (PLS) regression method (Wold, 2006) to jointly estimate the direct models. The authors compared the proposed method with the recursive and direct strategies on the quarterly index of US industrial production, for the period January 1945 to April 2000. They found mixed results with better forecasts for the recursive and the PLS methods compared to the direct strategy.

3.5.3 Hybrid forecasts

There have been also some research work that developed hybrid strategies between the recursive and the direct strategy.

Zhang and Hutchinson (1994) proposed to mix the recursive and direct strategies by having a different model for each horizon as with the direct strategy, but to include the forecasts from all the previous horizons (i.e. $h-1$ to 1) into the inputs of the model for horizon h .

In other words, instead of estimating the parameters as in (3.3.6), they are estimated as follows

$$(p_h, \hat{\boldsymbol{\psi}}_h) = \underset{p,\boldsymbol{\psi}}{\operatorname{argmin}} \sum_t \Big[y_t - [m_h(\hat{m}_{h-1}, \ldots, \hat{m}_1, \boldsymbol{r}_{t-h}; \boldsymbol{\psi}, \hat{\boldsymbol{\beta}}_h)] \Big]^2 .$$

where $\hat{m}_h$ is shorthand for $m_h(\boldsymbol{r}_{t-h}; \hat{\boldsymbol{\gamma}}_h)$ with $\hat{\boldsymbol{\gamma}}_h = [\hat{\boldsymbol{\psi}}_h, \hat{\boldsymbol{\beta}}_h]$. Then the forecasts are obtained for each horizon from the corresponding model, that is $\hat{\mu}_{T+h|T} = m_h(\hat{m}_{h-1}, \ldots, \hat{m}_1, \boldsymbol{r}_T; \hat{\boldsymbol{\gamma}}_h)$.

The authors considered neural networks with backpropagation for the models m_h. Because the number of lagged variables grow linearly with the horizon, this strategy suffers from the curse of dimensionality. So the authors used this strategy only for the first few horizons and considered traditional direct models for subsequent horizons. The authors did not compare their strategy with the recursive and direct strategies.

Sorjamaa and Lendasse (2006) considered a similar strategy with the KNN model including a variable selection method to deal with the increasing input vector. The authors found that the strategy produces smaller errors than both the recursive and direct strategies for forecasting a time series from the Santa Fe competition (Weigend and Gershenfeld, 1994) and another electric load time series.

Zhang, Zhou, et al. (2013) proposed another hybrid strategy, called MSVR, that estimates the first S ($S < H$) direct models and use them recursively to generate the H required forecasts. In other words, we learn the models as in (3.3.5) but only for horizons $h = 1,\ldots,S$ rather than $h = 1,\ldots,H$.

Assuming $H = n \times S$ (for simplicity), the MSVR strategy will produce n blocks of S forecasts by using n times the S direct models. The first block containing the first S forecasts is computed with the direct strategy and the remaining $(n-1)$ blocks are computed using the intermediate forecasts as input for the S models. So, the MSVR strategy does not change the way the model are estimated but only the way the forecasts are computed from direct models. The authors focused on support vector regression models but the strategy can be applied to any models.

The major advantage of the MSVR strategy is the reduced computational time when $S << H$. The major disadvantage is the possible loss of forecast accuracy due to the forecasts used as inputs for the direct models. In addition, The MSVR strategy require the selection of an additional parameter S that controls the trade-off between recursive and direct forecasts.

3.6 Time series forecasting with machine learning

In the previous sections, we have presented research that focuses on multi-step forecasting strategies, i.e. how to generate multi-step forecasts. In the machine learning literature, there have been also some research works on time series modeling and forecasting using machine learning models where the forecasting strategies is not the main focus.

The literature on machine learning for time series forecasting is rather sparse. The book by Palit and Popovic (2005) entitled *Computational intelligence in time series forecasting* is the only reference related to that subject. However, the authors only considered the neural network model and the book's focus is on computational intelligence methods based on fuzzy logic (Zadeh, 1975).

More recently, Ahmed et al. (2010) presented an empirical comparison of machine learning models for time series forecasting. Models that have been considered include neural networks, kernel regression, K-nearest neighbors, regression trees, support vector machines and Gaussian processes. The authors have compared these different models on the time series from the M3 competition data. However, the study has been limited to the problem of one-step-ahead forecasting.

The main machine learning model that has been considered in the forecasting literature is the neural network model. An overview of forecasting with neural network is given by Zhang, Patuwo, and Michael Y. (1998) and Zhang (2012). Crone, Hibon, and Nikolopoulos (2011) gives empirical evidence of the neural network ability to deal with complex time series data, including short and seasonal time series. Other references on the subject include Zhang and Qi (2005); Medeiros, Teräsvirta, and Rech (2006); Zhang and Kline (2007); Kock and Teräsvirta (2011). Recently, a particular type of neural networks called Extreme Learning Machine (Huang, Zhu, and Siew, 2006)

has also been considered for time series forecasting by Grigorievskiy et al. (2014). Finally, the top computational intelligence entries for the NN3 and NN5 competitions have been obtained with neural network models (Adeodato et al., 2009; Andrawis, Atiya, and El-Shishiny, 2011).

Kim (2003) considered support vector machines (SVM) for financial time series forecasting with a focus on stock price index forecasting. The author shows that SVM is a promising alternative to neural networks and case-based reasoning. Thissen (2003) used SVM for time series forecasting with applications in the field of Chemometrics. The authors compared SVM with both ARMA models and recurrent neural networks. Crone (2006) has applied an exhaustive empirical comparison of support vector machines and neural networks with traditional statistical methods. The least squares support vector machines (LS-SVM) (Suykens and Vandewalle, 1999; Suykens, Van Gestel, et al., 2002) has also been considered for forecasting problems (Van Gestel et al., 2001; Spinoza and Falck, 2008; Rubio et al., 2010). A survey of time series forecasting with support vector machines is provided by Sapankevych and Sankar (2009).

Nearest neighbors and more general local models have also been successfully applied in time series forecasting. The development of nearest neighbor models for time series forecasting has started with nonlinear systems (Casdagli, 1989). An extension of the nearest neighbor model, called the nearest neighbor trajectory model (McNames, 1998) has been proposed to improve forecast accuracy. Bontempi, Birattari, and Bersini (1999) and Sorjamaa, Hao, and Lendasse (2005) also considered local models for forecasting chaotic time series. Recently, nearest neighbors models have been used in combination with other machine learning models to forecast the time series of the NN5 forecasting competition (Wichard, 2010). A discussion about forecasting with nonparametric models can be found in Teräsvirta, Tjostheim, and Granger, 2010, Chap. 14.3.

A number of other machine learning algorithms have received little attention in the forecasting literature. Regression trees have been considered by Tran, Yang, and Tan (2009) for forecasting the operating conditions of machines. Self-organizing maps (SOM) (Kohonen, Schroeder, and Huang, 2001) have been considered by Simon et al. (2004); Simon et al. (2005); Simon, Lee, et al. (2007) for various forecasting tasks.

Gaussian processes have been considered by Girard, Rasmussen, and Candela (2003); Brahim-Belhouari and Bermak (2004). An overview of Bayesian time series models can be found in Barber, Cemgil, and Chiappa (2011).

Boosting has only recently been considered in the forecasting literature. See for example Assaad, Boné, and Cardot (2008); Audrino and Bühlmann (2009); Buchen and Wohlrabe (2011); Robinzonov, Tutz, and Hothorn (2012). Some research with bagging can be found in Inoue and Kilian (2008); Cordeiro (2009).

In his article "Mining the past to determine the future: Problems and possibilities", Dr. Hand provides an interesting discussion about "data mining" applied to forecasting. Price (2009) and Crone (2009a) provide useful comments about the paper, and Hand (2009b) concludes the discussion.

3.7 Summary and concluding remarks

This chapter has presented an overview of the literature that has considered multi-step forecasting strategies with both linear and nonlinear models, including machine learning models.

The majority of those studies have focused on comparing recursive and direct forecasts generated with linear models. Less work has been done with nonlinear models, notably due the additional

complexity and the heavier computational load. In particular, except neural networks (Zhang, 2012), multi-step forecasts generated with machine learning models have received little attention.

Alternative forecasting strategies have been proposed, including variants of recursive forecasts (McNames, 1998; Bontempi, Birattari, and Bersini, 1999; Atiya et al., 1999), variants of direct forecasts (Lee and Billings, 2003; Chen, Yang, and Hafner, 2004; Franses and Legerstee, 2009), and hybrid strategies (Zhang and Hutchinson, 1994; Sorjamaa and Lendasse, 2006; Zhang, Zhou, et al., 2013). However, these alternative strategies have received little attention in the literature, notably due to the additional complexity or the limited increase in performance compared to the traditional recursive and direct strategies.

The findings about the comparison between the recursive and the direct strategy can be summarized as follows.

With linear models, the theoretical literature tends to conclude that model misspecification plays an important role in the relative performance between the recursive and direct strategy (Chevillon, 2007). If the model is correctly specified, the recursive strategy benefit from more efficient parameter estimates, while the direct strategy is more robust to model misspecification. However, empirical studies often found superior performance of recursive forecasts compared to direct forecasts, especially for long horizons (Marcellino, Stock, and Watson, 2006; Pesaran, Pick, and Timmermann, 2011).

With nonlinear models, recursive forecasts are known to be asymptotically biased since they do not consider the innovation terms (Brown and Mariano, 1984; Lin and Granger, 1994). Alternatives, such as the exact, the Monte Carlo and the bootstrap methods have been proposed but these methods require additional knowledge either about the function f or the noise distribution (Teräsvirta, Tjostheim, and Granger, 2010). The direct strategy is often preferred over the recursive strategy since it avoids the accumulation of errors. In particular, empirical studies often found superior performance of the direct strategy compared to the recursive strategy (Atiya et al., 1999; Kline, 2004; Sorjamaa, Hao, Reyhani, et al., 2007).

When comparing linear with nonlinear models, it has been found that nonlinear models have poorer performance than their linear counterparts, especially with economic time series. Arguments put forward include the weak nonlinearity of the time series, the rare occurrence of nonlinear features and the overfitting with short time series (Stock and Watson, 1999; Teräsvirta, 2006; Teräsvirta, Tjostheim, and Granger, 2010).

There are several limitations to the comparisons that have been performed in the literature. First, all studies have considered the scenario where both the model and the DGP are linear or nonlinear. In particular, a study with the scenario where the model is linear and the DGP is nonlinear (and vice versa) does not seem to be available yet. Such study would be very valuable to study the behavior of recursive and direct forecasts when the model is more complex or too simple compared to the DGP.

Second, many studies have focused on nonlinear parametric models but there have been very limited studies involving machine learning algorithms. Among the limited studies that have considered machine learning algorithms, many have focused on the neural network model and/or to the problem of one-step-ahead forecasting.

Finally, the majority of the studies have often included only a limited number of models and time series in their comparisons. Also, many studies have considered (noisy-free) chaotic time series that do not reflect the type of time series we encounter in real-world forecasting applications.

We address these different issues in the next chapter with an in-depth bias and variance study of the recursive and direct strategies using different machine learning algorithms for both linear and

nonlinear DGPs. Our study will consider different scenarios depending on whether the model and the DGP are linear or nonlinear. Also, we will investigate the role of the time series length and the forecast horizon in the performance of the recursive and direct strategies.

Part II

Contributions

Chapter 4

Bias and variance analysis for multi-step forecasting

This chapter is partly based on the following publication: S Ben Taieb and AF Atiya (2014). "A bias and variance analysis for multi-step time series forecasting." Submitted to IEEE Transactions on Neural Networks and Learning Systems (under revision).

4.1 Introduction

In the previous section, we have seen that a large part of the literature has often compared the recursive and direct strategies for some specific DGPs with specific estimation methods, and discussing the conditions under which one or the other is better. In particular, many of the studies have mainly focused on the linear setting, i.e. where both the model and the DGP are linear (Chevillon, 2007). There have been few studies involving nonlinear or machine learning models (Atiya et al., 1999; Kline, 2004; Sorjamaa, Hao, Reyhani, et al., 2007; Kock and Teräsvirta, 2011), notably due to the analytical challenges and the heavier computational load.

Most empirical studies have focused on specific forecasting problems with either limited data or very short forecast horizons. Also, the few in-depth studies that have considered time series forecasting with machine learning models have been limited to the case of one-step-ahead forecasting (Berardi and Zhang, 2003; Ahmed et al., 2010). Furthermore, the studies have often compared the recursive and direct strategies based on overall accuracy measures such as mean squared error. Although these measures reflect the general forecasting performance, they do not allow to separate the different sources of errors, which is essential to gain a better understanding of the different strategies and to provide guidance for how to improve forecast accuracy.

One step towards achieving this goal has been taken by the development of a number of forecast-error taxonomies to understand the different sources of forecast errors for econometric models: see for example Clements and Hendry (1998), Hendry (2000), Clements and Hendry (2006) and Hendry and Mizon (2013). Although these taxonomies allow to reveal major sources of errors for econometric models, they do not allow to understand the behavior of different strategies with different learning algorithms in a real-world setting with limited data.

In this chapter, we address these issues by decomposing the multi-step mean squared forecast errors of the recursive and direct strategies into the so-called bias and variance components. The bias represents the consistent offset of the forecasts, away from the optimal forecasts, and the variance represents the variation of the forecasts around their mean. The bias and variance decomposition

is a very useful tool for analyzing the source of errors of an estimator (Geman, Bienenstock, and Doursat, 1992).

A theoretical bias and variance analysis for the general case of h-step ahead forecasts with machine learning models is not an easy task, notably because of the large number of potential factors that could affect the two components over the forecast horizon. In consequence, our bias and variance analysis will be decomposed into two parts. In the first part, we will apply a theoretical analysis for the case of two-step ahead forecasts to simplify the derivations. We extend the work of Atiya et al. (1999) who only considered the bias component in their analysis. In the second part, we consider the general case of h-step ahead forecasts by using Monte Carlo simulations to effectively compute the bias and variance components for the different strategies over the forecast horizon. In particular, we will study the behavior of the recursive and direct strategies with different learning models and different time series length on different DGPs.

To the best of our knowledge, this study is the first to apply a bias and variance analysis for multi-step forecasts with different machine learning models. Berardi and Zhang (2003) also conducted a bias and variance study for time series forecasting. However, there are many distinctive differences between their study and ours. First, we consider the problem of multi-step-ahead forecasting instead of one-step-ahead forecasting, which is a fundamental difference. Second, they have focused on neural networks while we consider various machine learning models (including neural networks). Finally, our focus is not on the learning model but on the forecasting strategy, that is the way we generate multi-step forecasts.

As a result of the bias and variance analysis, we will be able to give more insight about which component contributes more to degraded forecast accuracy of the recursive and direct strategies over the forecast horizon. Also, the observations made about the recursive and direct strategies in this chapter will guide the development of new forecasting strategies in subsequent chapters.

In the next section, we derive the decomposition of the multi-step mean squared forecast error into the bias and variance components. In Section 4.3, we present the general methodology for our bias and variance analysis, including the bias and variance estimation. Finally, we apply our methodology in Section 4.4 to compare the recursive and direct strategies. Note that the methodology developed in this chapter will also be used in Chapters 5 and 6 to compare the new strategies we propose.

4.2 Mean squared multi-step forecast error decomposition

Let $\boldsymbol{Y}_T = \{y_1, \ldots, y_T\}$, a time series comprising T observations, denotes a realization of a stationary autoregressive process of the form

$$y_t = f(\boldsymbol{x}_{t-1}) + \varepsilon_t \quad \text{with} \quad \boldsymbol{x}_{t-1} = [y_{t-1}, \ldots, y_{t-d}]', \tag{4.2.1}$$

where $\{\varepsilon_t\}$ is a stochastic iid noise process with $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$.

Given a realization $\boldsymbol{Y}_T = \{y_1, \ldots, y_T\}$, we denote $g(\boldsymbol{x}_t; \hat{\boldsymbol{\theta}}_T; h)$, the h-step ahead forecasts of a given strategy for the input $\boldsymbol{x}_t$. For each realization, a lag order p and a set of parameters $\hat{\boldsymbol{\theta}}_T$ are estimated, and could possibly be different from one realization to the other. In particular, the estimated lag order p could possibly be different from the "real" lag order d defined in (4.2.1).

Similarly to the bias and variance decomposition given in expression (2.1.4), we can decompose the conditional mean squared forecast error (MSFE or MSE)[1] of a given strategy for the input $\boldsymbol{x}_t$ at

[1] In this work we will use the terms MSE and MSFE interchangeably.

horizon h as follows

$$\begin{aligned}
&\mathrm{MSE}_h(\boldsymbol{x}_t) \\
&= \mathbb{E}_{\varepsilon,\boldsymbol{Y}_T}\left[\left(y_{t+h} - g(\boldsymbol{x}_t;\hat{\boldsymbol{\theta}}_T;h)\right)^2 \mid \boldsymbol{x}_t\right] \\
&= \underbrace{\underbrace{\mathbb{E}_{\varepsilon}\left[\left(y_{t+h} - \mu_{t+h|t}\right)^2 \mid \boldsymbol{x}_t\right]}_{\text{Noise variance } N_h(\boldsymbol{x}_t)} + \underbrace{\mathbb{E}_{\boldsymbol{Y}_T}\left[\left(g(\boldsymbol{x}_t;\hat{\boldsymbol{\theta}}_T;h) - \mathbb{E}_{\boldsymbol{Y}_T}\left[g(\boldsymbol{x}_t;\hat{\boldsymbol{\theta}}_T;h)\right]\right)^2 \mid \boldsymbol{x}_t\right]}_{\text{Estimation variance } V_h(\boldsymbol{x}_t)}}_{\text{Forecast variance } N_h(\boldsymbol{x}_t)+V_h(\boldsymbol{x}_t)} + \underbrace{\left(\mu_{t+h|t} - \mathbb{E}_{\boldsymbol{Y}_T}\left[g(\boldsymbol{x}_t;\hat{\boldsymbol{\theta}}_T;h)\right]\right)^2}_{\text{Squared bias } B_h(\boldsymbol{x}_t)}
\end{aligned} \tag{4.2.2}$$

where $\mathbb{E}_x$ and $\mathbb{E}[\cdot|x]$ denote the expectation over x and the expectation conditional on x, respectively, and $\mu_{t+h|t} = \mathbb{E}[y_{t+h}|\boldsymbol{x}_t]$ is the conditional mean.

The unconditional MSE can be computed by integrating over the input $\boldsymbol{x}_t$ as follows

$$\begin{aligned}
&\mathrm{MSE}_h \\
&= \mathbb{E}_{\boldsymbol{x}_t}\left[\mathrm{MSE}_h(\boldsymbol{x}_t)\right] \\
&= \underbrace{\underbrace{\mathbb{E}_{\boldsymbol{x}_t,\varepsilon}\left[(y_{t+h} - \mu_{t+h|t})^2 \mid \boldsymbol{x}_t\right]}_{\text{Noise variance } N_h} + \underbrace{\mathbb{E}_{\boldsymbol{x}_t,\boldsymbol{Y}_T}\left[(g(\boldsymbol{x}_t;\hat{\boldsymbol{\theta}}_T;h) - \mathbb{E}_{\boldsymbol{Y}_T}\left[g(\boldsymbol{x}_t;\hat{\boldsymbol{\theta}}_T;h)\right])^2 \mid \boldsymbol{x}_t\right]}_{\text{Estimation variance } V_h}}_{\text{Forecast variance } N_h+V_h} + \underbrace{\mathbb{E}_{\boldsymbol{x}_t}\left[(\mu_{t+h|t} - \mathbb{E}_{\boldsymbol{Y}_T}\left[g(\boldsymbol{x}_t;\hat{\boldsymbol{\theta}}_T;h)\right])^2\right]}_{\text{Squared bias } B_h}
\end{aligned} \tag{4.2.3}$$

(4.2.4)

We can see in (4.2.4) that the MSE of the forecasts at horizon h can be decomposed into three different components, the noise variance N_h, the estimation variance V_h and the (squared) bias B_h. In the following, we will simplify the terms by referring to the noise variance with noise and to the estimation variance with variance. It is worth noting that this decomposition is similar to the bias and variance decomposition given in expression (2.1.4).

The noise component N_h is the irreducible error that can only be attained by the optimal forecasts, that is the conditional mean $\mu_{t+h|t}$. In fact, if $g(\boldsymbol{x}_t;\hat{\boldsymbol{\theta}}_T;h) = \mu_{t+h|t}$, we can easily see that $\mathrm{MSE}_h = N_h$. Also, N_h does not depend on the forecasting strategy but only on the DGP. In practice, that is with a finite sample time series, it is impossible for the forecasts to be equal to $\mu_{t+h|t}$. Instead, we have an estimate of $\mu_{t+h|t}$ which is prone to bias and variance.

The variance component, V_h, relates to the stability of the forecasts built on different time series sampled from the same DGP and represents the variation of the forecast around its mean. The bias component, B_h, represents the consistent offset of the average forecast $\mathbb{E}_{\boldsymbol{Y}_T}\left[g(\boldsymbol{x}_t;\hat{\boldsymbol{\theta}}_T;h)\right]$, away from the optimal forecast, that is the conditional mean $\mu_{t+h|t}$.

Factors that can affect the bias and variance components include notably the model complexity and the time series length T. A complex model will tend to have a low bias, as it is powerful enough to be able to produce any shape of the fit. On the other hand, a simple model will be less flexible and will produce large bias. Concerning the variance, complex models will tend to be much more volatile, because they are more sensitive to the data and the random terms. Simple models, on the other hand, have less sensitivity, because they have less parameters and hence less "knobs" that can be used to tune any solution.

The ideal configuration is to have both a low bias and a low variance. However, this ideal configuration is never achievable in practice as decreasing the bias will increase the variance and vice

versa. Instead, it is important to obtain forecasts with a good trade-off between bias and variance to achieve a smaller MSE, as stated by the so-called *bias and variance trade-off* (Geman, Bienenstock, and Doursat, 1992). In particular, different forecasting strategies will generate forecasts with different bias and variance components, some achieving a better bias and variance trade-off than others.

4.2.1 Further decompositions

The bias and variance components can be further decomposed to illustrate other factors affecting the bias and variance trade-off.

Let us consider the case where the forecasts $g(\boldsymbol{x}_t; \hat{\boldsymbol{\theta}}_T; h)$ can be written as a sum of two quantities, that is $g(\boldsymbol{x}_t; \hat{\boldsymbol{\theta}}_T; h) = g_1(\boldsymbol{x}_t; \hat{\boldsymbol{\theta}}_{1;T}; h) + g_2(\boldsymbol{x}_t; \hat{\boldsymbol{\theta}}_{2;T}; h)$. For example, if the forecasts g represent the average of forecasts generated by two different models, g_1 (g_2) will represent the forecasts from the first (second) model divided by two.

In that case, the total variance V_h in (4.2.4) can be further decomposed into subcomponents as follows

$$\begin{aligned} V_h &= \mathbb{E}_{\boldsymbol{x}_t, \boldsymbol{Y}_T}\left[(g(\boldsymbol{x}_t; \hat{\boldsymbol{\theta}}_T; h) - \mathbb{E}_{\boldsymbol{Y}_T}\left[g(\boldsymbol{x}_t; \hat{\boldsymbol{\theta}}_T; h)\right])^2 \mid \boldsymbol{x}_t\right] \\ &= \underbrace{E_{\boldsymbol{x}_t, \boldsymbol{Y}_T}\left[(g_1(\boldsymbol{x}_t; \hat{\boldsymbol{\theta}}_{1;T}; h) - g_1(\boldsymbol{x}_t; \boldsymbol{\theta}_{1;T}; h))^2 \mid \boldsymbol{x}_t\right]}_{V_{1;h}} + \underbrace{E_{\boldsymbol{x}_t, \boldsymbol{Y}_T}\left[(g_2(\boldsymbol{x}_t; \hat{\boldsymbol{\theta}}_{2;T}; h) - g_2(\boldsymbol{x}_t; \boldsymbol{\theta}_{2;T}; h))^2 \mid \boldsymbol{x}_t\right]}_{V_{2;h}} \\ &\quad + \underbrace{2E_{\boldsymbol{x}_t, \boldsymbol{Y}_T}\left[(g_1(\boldsymbol{x}_t; \hat{\boldsymbol{\theta}}_{1;T}; h) - g_1(\boldsymbol{x}_t; \boldsymbol{\theta}_{1;T}; h))(g_2(\boldsymbol{x}_t; \hat{\boldsymbol{\theta}}_{2;T}; h) - g_2(\boldsymbol{x}_t; \boldsymbol{\theta}_{2;T}; h)) \mid \boldsymbol{x}_t\right]}_{\mathrm{COV}_h}, \end{aligned} \tag{4.2.5}$$

where $V_{1;h}$ is the variance of $g_1(\boldsymbol{x}_t; \hat{\boldsymbol{\theta}}_{1;T}; h)$, $V_{2;h}$ is the variance of $g_2(\boldsymbol{x}_t; \hat{\boldsymbol{\theta}}_{2;T}; h)$ and COV_h is the covariance of $g_1(\boldsymbol{x}_t; \hat{\boldsymbol{\theta}}_{1;T}; h)$ and $g_2(\boldsymbol{x}_t; \hat{\boldsymbol{\theta}}_{2;T}; h)$. This is also known as the *variance-covariance* decomposition (Ueda and Nakano, 1996) and can easily be generalized to an arbitrary number of models.

The bias term B_h in (4.2.4) can also be further decomposed into subcomponents as follows

$$\begin{aligned} B_h &= \mathbb{E}_{\boldsymbol{x}_t}\left[\left(\mu_{t+h|t} - \mathbb{E}_{\boldsymbol{Y}_T}\left[g(\boldsymbol{x}_t; \hat{\boldsymbol{\theta}}_T; h)\right]\right)^2\right] \\ &= \mathbb{E}_{\boldsymbol{x}_t}\left[\left(\underbrace{\mu_{t+h|t} - g(\boldsymbol{x}_t; \boldsymbol{\theta}^*; h)}_{A} + \underbrace{g(\boldsymbol{x}_t; \boldsymbol{\theta}^*; h) - \mathbb{E}_{\boldsymbol{Y}_T}\left[g(\boldsymbol{x}_t; \hat{\boldsymbol{\theta}}_T; h)\right]}_{B}\right)^2\right]. \end{aligned}$$

The A term represents the discrepancy between the conditional mean and the best potential of the model family we consider. For example, consider that $\mu_{t+h|t}$ is nonlinear, while we consider a linear forecasting model, then $g(\boldsymbol{x}_t; \boldsymbol{\theta}^*; h)$ is the linear forecast that provides the best approximation of the nonlinear function. In such a case the A term is a measure of the limitation of the model family, not taking into account the learning algorithm. The B term represents the error due to using a specific learning algorithm that estimate the parameters using a time series limited to T observations. For example, even if $\mu_{t+h|t}$ can be correctly estimated with $g(\boldsymbol{x}_t; \boldsymbol{\theta}^*; h)$ (the A term cancels), the B term will still remain if the learning algorithm is not able to obtain the best parameters, i.e. if $\boldsymbol{\theta}_T \neq \boldsymbol{\theta}^*$. Recall that $\mathbb{E}_{\boldsymbol{Y}_T}\left[g(\boldsymbol{x}_t; \hat{\boldsymbol{\theta}}_T; h)\right] = \mathbb{E}_{\boldsymbol{Y}_T}\left[g(\boldsymbol{x}_t; \hat{\boldsymbol{\theta}}_T; h)\right]$.

4.3 Methodology

We present our methodology to compare different forecasting strategies from the perspective of the bias and variance components of their respective mean squared forecast errors.

In the previous section, we have seen that the bias and variance components depend on a number of interacting factors including the model complexity, the underlying DGP, the time series length and the forecast horizon. Therefore, a theoretical analysis for the general case of h-steps ahead forecasting that takes into account all these factors is not an easy task.

Consequently, we will limit our theoretical analysis to the case of two-steps ahead forecasts to simplify the derivations, as in Atiya et al. (1999). We will also consider the general case of h-step-ahead forecasts by using Monte Carlo simulations to effectively estimate the bias and variance components.

Because the underlying DGP, the learning model and their discrepancy play an important role in the bias and variance components, we will consider different scenarios depending on whether the DGP or the learning model are linear or nonlinear. So, we will end up with four different scenarios: **(A)** a linear model with a linear DGP, **(B)** a linear model with a nonlinear DGP, **(C)** a nonlinear model with a linear DGP, and **(D)** a nonlinear model with a nonlinear DGP.

The scenario A, where both the model and the DGP are linear, has received much attention from the forecasting and econometrics literature, as explained in Section 3.4.1. For this scenario, we will consider both a well-specified and a misspecified linear model where the misspecification will come from omitted lagged variables.

The scenario B, where the model is linear and the DGP is nonlinear, will allow us to analyze the performance of the strategies when the class of model is limited and does not include the true nonlinear DGP. This scenario is particularly useful since, in practice, linear models are widely used while we expect real-world phenomena to behave nonlinearly.

The scenario C, where the model is nonlinear and the DGP is linear, will give us some insight about the cost of extending the class of models beyond the class of the true DGP. For example, using complex machine learning models when the DGP is linear.

The scenario D, where both the model and the DGP are nonlinear, allows us to compare the strategies when everything is nonlinear.

In the next section, we give more details about our theoretical analysis for two-step-ahead forecasting, while Section 4.3.2 presents the general case of h-step ahead forecasting.

4.3.1 Theoretical analysis for two-step ahead forecasts

Recall the observed time series are assumed to be generated from the autoregressive process given in (4.2.1). We want to analyze the bias and variance components $B_2(\boldsymbol{x}_t)$ and $V_2(\boldsymbol{x}_t)$ of the two-step ahead forecast errors for a given forecasting strategy. More precisely, we want to compute for different forecasting strategies the sum of the bias and variance components at horizon $h = 2$, given by

$$\begin{aligned} & B_2(\boldsymbol{x}_t) + V_2(\boldsymbol{x}_t) \\ & = (\mu_{t+2|t} - g(\boldsymbol{x}_t; \boldsymbol{\theta}_T; 2))^2 \\ & \quad + \mathbb{E}_{\boldsymbol{Y}_T}\left[(g(\boldsymbol{x}_t; \hat{\boldsymbol{\theta}}_T; 2) - g(\boldsymbol{x}_t; \boldsymbol{\theta}_T; 2))^2 \mid \boldsymbol{x}_t\right], \end{aligned} \tag{4.3.1}$$

where $B_2(\boldsymbol{x}_t)$ and $V_2(\boldsymbol{x}_t)$ are defined in expression (4.2.2).

In order to compute expression (4.3.1), we need to compute the conditional expectation $\mu_{t+2|t}$ for the process defined in (4.2.1). Aditionally, we need to obtain an expression for the two-step ahead forecasts $g(\boldsymbol{x}_t; \hat{\boldsymbol{\theta}}_T; 2)$ of each strategy.

We compute the conditional expectation $\mu_{t+2|t} = \mathbb{E}[y_{t+2}|\boldsymbol{x}_t]$ as in Atiya et al. (1999). First we compute y_{t+2} using the Taylor series approximation and keep up to second-order terms. This yields the following expression:

$$\begin{aligned} y_{t+2} &= f(f(\boldsymbol{x}_t) + \varepsilon_{t+1}, y_t, \ldots, y_{t-d+2}) + \varepsilon_{t+2} \\ &\approx f(f(\boldsymbol{x}_t), y_t, \ldots, y_{t-d+2}) + \varepsilon_{t+1} f_{x_1} + \tfrac{1}{2}(\varepsilon_{t+1})^2 f_{x_1 x_1} + \varepsilon_{t+2}, \end{aligned}$$

where the function f is computed at point $[f(\boldsymbol{x}_t) + \varepsilon_{t+1}, y_t, \ldots, y_{t-d+2})]$ and the Taylor series approximation is computed around the point $[f(\boldsymbol{x}_t), y_t, \ldots, y_{t-d+2})]$, f_{x_1} and $f_{x_1 x_1}$ are the first and second derivatives of f with respect to its first argument, respectively.

The conditional expectation $\mu_{t+2|t}$ is then given as

$$\mu_{t+2|t} = \mathbb{E}[y_{t+2}|\boldsymbol{x}_t] \approx f(f(\boldsymbol{x}_t), y_t, \ldots, y_{t-d+2}) + \tfrac{1}{2}\sigma^2 f_{x_1 x_1}. \tag{4.3.2}$$

The previous expression is valid for the autoregressive process defined in (4.2.1) with a possibly nonlinear function f. However, if the function f is linear and is defined as

$$f(y_{t-1}, \ldots, y_{t-d}) = \varphi_0 + \sum_{j=1}^{d} \varphi_j y_{t-j}, \tag{4.3.3}$$

where φ_j are the coefficients of the linear combination, then the DGP given in (4.2.1) becomes

$$y_t = \varphi_0 + \varphi_1 y_{t-1} + \cdots + \varphi_d y_{t-d} + \varepsilon_t. \tag{4.3.4}$$

Since $f_{x_1 x_1} = 0$, the expression (4.3.2) for the conditional expectation at horizon $h = 2$ reduces to

$$\mu_{t+2|t} = f(f(\boldsymbol{x}_t), y_t, \ldots, y_{t-d+2}).$$

and using expression (4.3.3), can be rewritten as

$$\mu_{t+2|t} = (\varphi_0 + \varphi_1\varphi_0) + (\varphi_1^2 + \varphi_2) y_t + \cdots + (\varphi_1\varphi_{d-1} + \varphi_d) y_{t-d+2} + (\varphi_1\varphi_d) y_{t-d+1}. \tag{4.3.5}$$

For the expression of $g(\boldsymbol{x}_t; \hat{\boldsymbol{\theta}}_T; 2)$, we will model the forecasts of each strategy as a sum of three terms: the true function value we are trying to estimate, which is the conditional mean $\mu_{t+2|t} = \mathbb{E}[y_{t+2}|\boldsymbol{x}_t]$, an offset term denoted by $\delta(\boldsymbol{z}_t; \boldsymbol{\theta})$ and a variability term denoted by $\eta(\boldsymbol{z}_t; \boldsymbol{\theta})\varepsilon_\eta$, where $\eta(\boldsymbol{z}_t; \boldsymbol{\theta})$ is a deterministic factor giving the standard deviation of the term, and $\varepsilon_\eta \sim \mathcal{N}(0, 1)$.

The offset term $\delta(\boldsymbol{z}_t; \boldsymbol{\theta})$ is the discrepancy from the conditional mean $\mu_{t+2|t}$ arising notably from (i) the lack of flexibility of the considered forecasting model (i.e. the model, with its parameters $\boldsymbol{\theta}$, is not powerful enough to reconstruct $\mu_{t+2|t}$ accurately), (ii) potential missing variables in the inputs ($\boldsymbol{z}_t$ not equal to $\boldsymbol{x}_t$), and (iii) an inadequate estimation algorithm for the parameters $\boldsymbol{\theta}$ (i.e. even if the model is powerful, the training algorithm may fall short of finding the right parameters).

The variability term $\eta(\boldsymbol{z}_t; \boldsymbol{\theta})\varepsilon_\eta$ represents the variability of the forecasts, and it arises notably due to (i) the finite-sampledness of the time series $\boldsymbol{Y}_T = \{y_1, \ldots, y_T\}$ used to estimate $\boldsymbol{\theta}$, (ii) the number of input variables in $\boldsymbol{z}_t$ potentially including redundant or meaningless variables, and (iii) the chosen model complexity that might make it too flexible.

Finally, since we have computed the conditional expectation of the autoregressive process defined in (4.2.1), we can now compute the noise component $N_2(\boldsymbol{x}_t)$, given in (4.2.4), as follows

$$\begin{aligned} N_2(\boldsymbol{x}_t) &= \mathbb{E}_\varepsilon\left[(y_{t+2} - \mu_{t+2|t})^2\right] \\ &\approx \mathbb{E}_\varepsilon\left[\left(\varepsilon_{t+1} f_{x_1} + \tfrac{1}{2}\varepsilon_{t+1}^2 f_{x_1x_1} + \varepsilon_{t+2} - \tfrac{1}{2}\sigma^2 f_{x_1x_1}\right)^2\right] \\ &= \sigma^2[1 + f_{x_1}^2] + \tfrac{1}{2}\sigma^4 f_{x_1x_1}^2. \end{aligned} \tag{4.3.6}$$

where we used the fact that $\mathbb{E}[\varepsilon^3] = 0$ and $\mathbb{E}[\varepsilon^4] = 3\sigma^4$ for the standard normal distribution.

We can see in (4.3.6) that the noise component does not depend on the forecasting strategy, but only on the DGP. One can also observe that the noise term is larger than $N_1(\boldsymbol{x}_t) = \sigma^2$, the noise term at horizon $h = 1$ and becomes larger for strongly nonlinear DGPs (because, then $|f_{x_1x_1}|$ will be larger).

For the linear DGP defined in (4.3.4), since $f_{x_1} = f_{y_{t-1}} = \varphi_1$ and $f_{x_1x_1}^2 = 0$, expression (4.3.6) for the noise component at horizon $h = 2$ reduces to

$$N_2(\boldsymbol{x}_t) = \sigma^2[1 + \varphi_1^2].$$

We refer to the Proposition 3.4 in Fan and Yao (2003) for the noise component of h-step ahead forecasts.

4.3.2 Monte Carlo simulations for h-step ahead forecasts

As a complement to the theoretical analysis for two-step ahead forecasts, we will consider Monte Carlo simulations to perform a bias and variance analysis for the general case of h-step ahead forecasts. In particular, it will allow us to compare the bias and variance components of different forecasting strategies over the forecast horizon for various DGPs, learning models and time series lengths.

Data generating processes

In the simulation study, we will consider three different autoregressive processes where one is linear and the two others are nonlinear. Simulated time series of the different processes are available in Appendix B.

The first process is a linear stationary autoregressive (AR) process given by

$$y_t = 1.32y_{t-1} - 0.52y_{t-2} - 0.16y_{t-3} + 0.18y_{t-4} - 0.26y_{t-5} + 0.19y_{t-6} + \varepsilon_t, \tag{4.3.7}$$

where $\varepsilon_t \sim \mathcal{N}(0,1)$. This process exhibits cyclic behavior and was selected by fitting an AR(6) model to the famous annual sunspot series. Because it is a linear process, the variance of ε_t simply scales the resulting series. Consequently, we set the error variance to one, without loss of generality.

The second process is the Smooth Transition AutoRegressive (STAR) process given by

$$y_t = 0.3y_{t-1} + 0.6y_{t-2} + (0.1 - 0.9y_{t-1} + 0.8y_{t-2})[1 + \exp\{(-10y_{t-1})\}]^{-1} + \varepsilon_t,$$

where exp is the exponential function and $\varepsilon_t \sim \mathcal{N}(0, 0.1^2)$. We set the error variance to a value which allows enough predictability to obtain meaningful results.

This process has been considered by Berardi and Zhang (2003) in a bias and variance study for one-step-ahead forecasting with neural network models. Several other simulation studies have

used the STAR process for the purposes of model selection, model evaluation as well as model comparison. See Teräsvirta, Tjostheim, and Granger (2010), Teräsvirta and Anderson (1992), Tong and Lim (1980) and Tong (1995) for some examples as well as related theoretical background and applications.

The third process is a nonlinear autoregressive (NAR) process given by

$$y_t = -0.17 + 0.85y_{t-1} + 0.14y_{t-2} - 0.31y_{t-3} + 0.08y_{t-7} + 12.80\ G_1(\boldsymbol{y}_{t-1}) + 2.44\ G_2(\boldsymbol{y}_{t-1}) + \varepsilon_t$$

with

$$G_1(\boldsymbol{y}_{t-1}) = (1 + \exp\{-0.46(0.29y_{t-1} - 0.87y_{t-2} + 0.40y_{t-7} - 6.68)\})^{-1},$$
$$G_2(\boldsymbol{y}_{t-1}) = (1 + \exp\{-1.17 \times 10^3(0.83y_{t-1} - 0.53y_{t-2} - 0.18y_{t-7} + 0.38)\})^{-1}$$

where $\varepsilon_t \sim \mathcal{N}(0,1)$.

Medeiros, Teräsvirta, and Rech (2006) built this process by fitting an artificial neural network with two hidden units to the annual sunspot series. In Kock and Teräsvirta (2011), this process has been used to compare different forecasting methods in a nonlinear setting.

Bias and variance estimation

The different components of the MSE decomposition given in (4.2.3), namely the noise variance N_h, the bias B_h and the variance V_h, can be estimated by replacing expectations with averages over a large number of samples as follows.

For a given DGP, we generate L independent time series $\boldsymbol{Y}^{(i)} = \{y_1, \ldots, y_T\}$, $i \in \{1, \ldots, L\}$, each composed of T observations using different randomly generated numbers for the noise terms. These generated time series represent samples of the DGP. Note that we discarded the first three hundred simulated values for each simulated series to stabilize the time series, as suggested by Law and Kelton (2000).

To measure the bias and variance components, we use an independent time series $\{u_1, \ldots, u_R\}$ from the same DGP for testing purposes. We represent this independent testing time series by a set of input/output pairs $\{(\boldsymbol{x}_j, \boldsymbol{y}_j)\}_{j=1}^{R-(d+H)}$ where $\boldsymbol{x}_j$ is a sequence of d consecutive observations that will yield the lagged input variables and the vector $\boldsymbol{y}_j$ comprises the next H observations that need to be forecasted, i.e. $\boldsymbol{x}_j = [u_j, \ldots, u_{j+d-1}]$ and $\boldsymbol{y}_j = [u_{j+d}, \ldots, u_{j+d+H-1}]$.

Let $g(\boldsymbol{x}_j; \boldsymbol{\theta}_{\boldsymbol{Y}^{(i)}}; h)$ denote the forecast of a given strategy for the input $\boldsymbol{x}_j$ at horizon h using dataset $\boldsymbol{Y}^{(i)}$ and let $y_j(h)$ denote the hth element of the vector $\boldsymbol{y}_j$, then the MSE at horizon h, given in expression (4.2.3), can be estimated as follows

$$\begin{aligned}
\widehat{MSE}_h &= \frac{1}{R}\sum_{j=1}^{R} \underbrace{\frac{1}{L}\sum_{i=1}^{L}\left(y_j(h) - g(\boldsymbol{x}_j; \boldsymbol{\theta}_{\boldsymbol{Y}^{(i)}}; h)\right)^2}_{\widehat{\mathrm{MSE}}_h(\boldsymbol{x}_t)} \\
&= \frac{1}{R}\sum_{j=1}^{R}\left[\hat{N}_h(\boldsymbol{x}_j) + \hat{B}_h(\boldsymbol{x}_j) + \hat{V}_h(\boldsymbol{x}_j)\right] \\
&= \underbrace{\frac{1}{R}\sum_{j=1}^{R}\hat{N}_h(\boldsymbol{x}_j)}_{\hat{N}_h} + \underbrace{\frac{1}{R}\sum_{j=1}^{R}\hat{B}_h(\boldsymbol{x}_j)^2}_{\hat{B}_h} + \underbrace{\frac{1}{R}\sum_{j=1}^{R}\hat{V}_h(\boldsymbol{x}_j)^2}_{\hat{V}_h}
\end{aligned} \tag{4.3.8}$$

with

$$\hat{N}_h(\boldsymbol{x}_j) = \left(y_j(h) - \hat{\mu}_{j+h|j}\right)^2,$$
$$\hat{B}_h(\boldsymbol{x}_j) = \left(\hat{\mu}_{j+h|j} - g(\boldsymbol{x}_j;\boldsymbol{\theta};h)\right)^2,$$
$$\hat{V}_h(\boldsymbol{x}_j) = \frac{1}{L}\sum_{i=1}^{L}\left[g(\boldsymbol{x}_j;\boldsymbol{\theta}_{\boldsymbol{Y}^{(i)}};h) - g(\boldsymbol{x}_j;\boldsymbol{\theta};h)\right]^2,$$

where $g(\boldsymbol{x}_j;\boldsymbol{\theta};h) = \frac{1}{L}\sum_{i=1}^{L} g(\boldsymbol{x}_j;\boldsymbol{\theta}_{\boldsymbol{Y}^{(i)}};h)$ and $\hat{\mu}_{j+h|j} = \text{Avg}[y_{t+h} \mid \boldsymbol{x}_t = \boldsymbol{x}_j]$ is the Monte Carlo estimate of the conditional expectation at input $\boldsymbol{x}_j$ for horizon h.

The conditional expectation $\mu_{j+h|j}$ can be calculated analytically for linear DGPs and some nonlinear DGPs (Teräsvirta, 2006). More generally, for nonlinear DGPs, we can use simulations to estimate it as follows.

For every $\boldsymbol{x}_j$ there are different possible subsequent patterns y_j that depend on the realization of the error term. From a fixed starting vector $\boldsymbol{x}_j$, we generate S different h-step realizations (also called *paths*), denoted $y_s^{(h)}(\boldsymbol{x}_j)$ with $s = 1,\ldots,S$. Then we compute the average of these S different realizations to obtain $\hat{\mu}_{j+h|j} = \frac{1}{S}\sum_{s=1}^{S} y_s^{(h)}(\boldsymbol{x}_j)$.

Note that the law of large numbers ensures that asymptotically the quantities $\hat{N}_h$, $\hat{B}_h$ and $\hat{V}_h$ in expression (4.3.8) will converge to the true values N_h, B_h and V_h in expression (4.2.4). In our simulations, we used the values $L = 1000$, $R = 2000$ and $S = 10,000$ which have been found to provide statistically significant results.

Model selection and estimation

In our experiments, we consider the linear (LIN) model and three nonlinear models namely the neural network (MLP) model, the K-nearest neighbors (KNN) model and the gradient boosting (BST2) model with bivariate P-splines as base learners. More details about these models are given in Section 2.2.

For the linear model (see Section 2.2.1), the parameters $\boldsymbol{\beta}$ are estimated by ordinary least squares (OLS).

For the MLP model (see Section 2.2.2), the parameters $\boldsymbol{\beta}$ are estimated by backpropagation, the number of hidden nodes NH are allowed to range in the set $\{0,1,2,3,5\}$ (where 0 means no hidden neurons, i.e. effectively a linear model) and the weight decay λ possible values are from the following choices $\{0.005, 0.01, 0.05, 0.1, 0.2, 0.3\}$.

For the KNN model (see Section 2.2.3), the number of neighbors is allowed to range in the set $1,\ldots,N$ where N is the number of examples in the dataset.

For the BST model (see Section 2.2.4), we limit the maximum number of iterations J to 500. The shrinkage parameter is fixed to $\nu = 0.3$ and the number of knots are set to 20 and 5 for the univariate and bivariate P-splines, respectively.

In addition to the model parameters, the lag order p is an important parameter that should be carefully selected. In our simulations, we allow p to range in the set $\{1,\ldots,7\}$ for the AR DGP, in $\{1,\ldots,5\}$ for the STAR DGP, and in $\{1,\ldots,10\}$ for the NAR DGP.

To validate the different parameters of the models (including the lag order p), we use a time-series cross-validation approach with five origins (see Section 2.3.5). More precisely, the first part of the

dataset (70%) is used as training set and the remaining part (30%) is used as validation/rolling set. For model selection, we use nested loops with one loop for each hyperparameter plus an additional loop at the top for the lag order p.

Finally, we considered a maximum forecasting horizon $H = 10$ for the AR DGP and the NAR DGP, and $H = 15$ for the STAR DGP. To show the importance of the length of the time series T for each strategy, we compare different sizes, namely $T \in \{50, 100, 400\}$.

4.4 Analysis of the recursive and direct strategies

In this section, we will apply the methodology described in Section 4.3 to compare the recursive and the direct strategies. Recall that we will consider four different scenarios depending on whether the learning model and the DGP are linear or nonlinear. For each scenario, we will compare the bias and variance components of the two strategies for two-step ahead forecasts. We will also perform Monte Carlo simulations to consider the general case of h-step ahead forecasts.

Before considering the four different scenarios, we first derive the expressions of the two-step ahead forecasts for both the recursive and direct strategies. Then, we analyze the results of some Monte Carlo simulations to make general observations about the behavior of the different components of the MSE decomposition.

Let us first state the two-step ahead forecasts of the recursive and direct strategies using the terminology defined in Section 4.3.1. To simplify notations, we will remove the dependence on the size of the time series T.

Forecasts of the recursive strategy. To produce two-step ahead forecasts, the recursive strategy first estimates a one-step-ahead model as in (3.3.1) and then produces forecasts recursively.

A possibly nonlinear model can be written as

$$m(\boldsymbol{z}_t;\hat{\boldsymbol{\phi}}) = \underbrace{\underbrace{f(\boldsymbol{x}_t)}_{\mu_{t+1|t}} + \delta(\boldsymbol{z}_t;\boldsymbol{\phi}) + \eta(\boldsymbol{z}_t;\boldsymbol{\phi})\varepsilon_\eta}_{m(\boldsymbol{z}_t;\boldsymbol{\phi})}, \tag{4.4.1}$$

where $m(\boldsymbol{z}_t;\boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{Y}_T}[m(\boldsymbol{z}_t;\hat{\boldsymbol{\phi}})]$ is the expectation over all time series datasets of size T, and $\delta(\boldsymbol{z}_t;\boldsymbol{\phi})$ and $\eta(\boldsymbol{z}_t;\boldsymbol{\phi})$ are defined in Section 4.3.1.

Two-step forecasts are obtained recursively and can be computed, after some simplifications using a Taylor series expansion, as follows

$$\begin{aligned} &g(\boldsymbol{x}_t;\hat{\boldsymbol{\phi}};2) \\ &= m(m(\boldsymbol{z}_t;\hat{\boldsymbol{\phi}}), y_t, y_{t-1}, \ldots, y_{t-p+2};\hat{\boldsymbol{\phi}}) \\ &\approx f(f(\boldsymbol{x}_t), y_t, y_{t-1}, \ldots, y_{t-p+2}) \\ &\quad + \delta(f(\boldsymbol{x}_t), y_t, y_{t-1}, \ldots, y_{t-p+2};\boldsymbol{\phi}) + \delta(\boldsymbol{z}_t;\boldsymbol{\phi}) m_{z_1} + \frac{1}{2}[\delta(\boldsymbol{z}_t;\boldsymbol{\phi})]^2 m_{z_1 z_1} \\ &\quad + \eta(f(\boldsymbol{x}_t), y_t, y_{t-1}, \ldots, y_{t-p+2};\boldsymbol{\phi})\varepsilon_{\eta_2} + \eta(\boldsymbol{z}_t;\boldsymbol{\phi})\varepsilon_{\eta_1} m_{z_1} + \frac{1}{2}[\eta(\boldsymbol{z}_t;\boldsymbol{\phi})\varepsilon_{\eta_1}]^2 m_{z_1 z_1}, \end{aligned} \tag{4.4.2}$$

where ε_{η_1} and ε_{η_2} are the stochastic components of the variability term for input $\boldsymbol{z}_t$ and $[f(\boldsymbol{z}_t), y_t, y_{t-1}, \ldots, y_{t-p+2}]$ respectively, and m_{z_1} and $m_{z_1 z_1}$ are respectively the first and second derivatives of the model m with respect to its first argument.

For the multi-step recursive strategy, defined in (3.3.4), the two-step-ahead forecasts can also be represented using (4.4.2) since the forecasts are also obtained recursively. However, because the parameters are selected by minimizing the h-step-ahead error to take into account the propagation of errors, it will have an impact on the offset and variability terms, $\delta(\boldsymbol{z}_t;\boldsymbol{\phi})$ and $\eta(\boldsymbol{z}_t;\boldsymbol{\phi})\varepsilon_\eta$. Note that we will only consider the multi-step recursive strategy for nonlinear models.

When the model is linear, expression (4.4.1) can be rewritten as

$$m(\boldsymbol{z}_t;\hat{\boldsymbol{\phi}}) = \phi_0 + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \eta(\boldsymbol{\phi})\varepsilon_\eta, \tag{4.4.3}$$

where we used the fact that, for linear models, the offset and the variability terms are independent of the inputs and only depend on the set of parameters $\boldsymbol{\phi} = [\phi_0, \phi_1, \ldots, \phi_p]$.

The expression for two-step ahead forecasts, in (4.4.2), for linear models simplifies to

$$\begin{aligned} & g(\boldsymbol{x}_t; \hat{\boldsymbol{\phi}}; 2) \\ & = (\phi_0 + \phi_1\phi_0) + (\phi_1^2 + \phi_2)y_t + \cdots + (\phi_1\phi_{p-1} + \phi_p)y_{t-p+2} + (\phi_1\phi_p)y_{t-p+1} \\ & \quad + (\phi_1 + 1)\eta(\boldsymbol{\phi})\varepsilon_\eta, \end{aligned} \tag{4.4.4}$$

where we used the fact that, in expression (4.4.2), if the model m is linear as in (4.4.3), then we have $m_{z_1} = \phi_1$, $m_{z_1 z_1} = 0$, and $\varepsilon_{\eta_1} = \varepsilon_{\eta_2} = \varepsilon_\eta$.

Forecasts of the direct strategy. A model is estimated to directly produce two-step ahead forecasts. For a possibly nonlinear model it can be written as

$$\begin{aligned} & g(\boldsymbol{x}_t; \hat{\boldsymbol{\gamma}}; 2) \\ & = \underbrace{\mu_{t+2|t} + \delta(\boldsymbol{r}_t; \boldsymbol{\gamma})}_{m_2(\boldsymbol{r}_t; \boldsymbol{\gamma})} + \eta(\boldsymbol{r}_t; \boldsymbol{\gamma})\varepsilon_\eta. \end{aligned} \tag{4.4.5}$$

In contrast to the forecasts of the recursive strategy in (4.4.2), we can see that the conditional mean $\mu_{t+2|t}$ appears in the previous expression.

If the model is linear, then the previous expression can be rewritten as

$$\begin{aligned} & g(\boldsymbol{x}_t; \hat{\boldsymbol{\gamma}}; 2) \\ & = \underbrace{\gamma_0 + \gamma_1 y_t + \cdots + \gamma_p y_{t-p+1}}_{m_2(\boldsymbol{r}_t; \boldsymbol{\gamma})} + \eta(\boldsymbol{\gamma})\varepsilon_\eta, \end{aligned} \tag{4.4.6}$$

where $\boldsymbol{\gamma} = [\gamma_0, \gamma_1, \ldots, \gamma_p]$.

The different expressions derived here will be used in the next sections to compute the bias and variance components for two-step ahead forecasts.

In addition to the theoretical analysis, we will use Monte Carlo simulations to estimate the different component of the MSE decomposition as explained in Section 4.3.2.

We will consider different learning models with both the recursive (REC) and direct (DIR) strategies. The two strategies will be denoted as REC-LIN or DIR-LIN with the linear model, REC-KNN or DIR-KNN with the KNN model, REC-MLP or DIR-MLP with the MLP model and REC-BST2 or DIR-BST2 with the BST2 model. The multistep recursive strategy will be denoted as RTI.

Before comparing the recursive and direct strategies, we will first make general observations about the different components of the MSE decomposition by considering the results of the STAR DGP. The results for the recursive and direct strategies are given in Figures 4.1 and 4.2, respectively.

Each figure presents stacked area plots giving the relative contribution of the noise component N_h, the bias component B_h and the variance component V_h to the MSE over the forecast horizon ($h = 1, \ldots, H$) for different learning models (in column) and different time series lengths T (in row). The noise component is given in grey, the bias in cyan and the variance in yellow. These three components are defined in (4.2.4) and are estimated as explained in Section 4.3.2.

In both Figures 4.1 and 4.2, we can see that the noise component does not depend on the forecasting strategy, the learning model or the time series length. In fact, the noise component only depends on

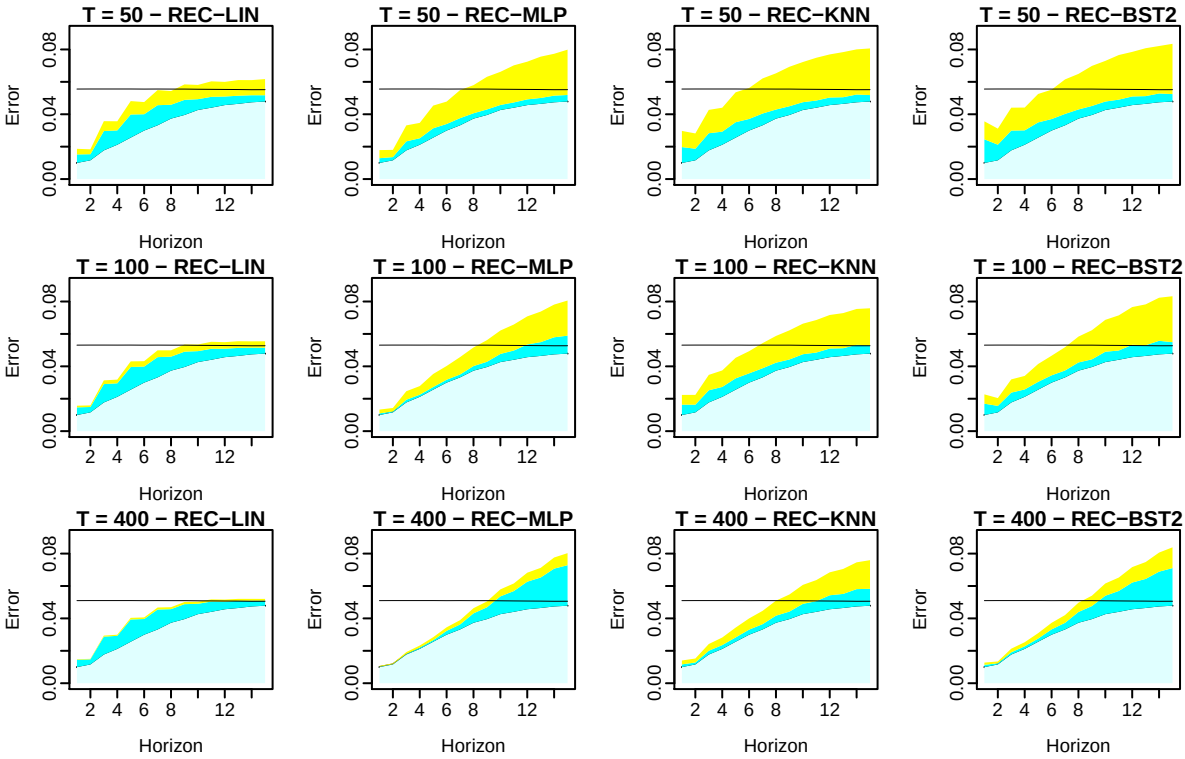

Figure 4.1: *STAR DGP. The MSE of recursive forecasts generated with different learning models (by column) and different time series lengths (by row) is decomposed into noise (in grey), bias (in cyan) and variance (in yellow). The stacked area plots show the relative contribution of each component to the total MSE over the forecast horizon.*

the DGP and represents the MSE of the optimal forecasts, that is the conditional mean. In addition, the noise component is increasing with the forecast horizon, which is consistent with expression (4.3.6) in Section 4.3.1. Finally, the noise component dominates the other components of the MSE decomposition, which is particularly noticeable for long horizons. The horizontal line corresponds to the MSE of the (unconditional) mean forecasts computed with T observations. This line allows us to have a measure of the predictability of the time series relative to the mean forecasts. In fact, we can see in the first column that the difference between the MSE of the optimal forecasts and the mean forecasts is decreasing with the horizon. This shows that the longer the horizon, the less predictable the DGP is.

Similarly to the noise component, the variance component tends to increase with the forecast horizon. Furthermore, independently from the forecasting strategy, it tends to decrease with longer time series (compare $T = 50$ in the first row with $T = 400$ in the last row). If we compare the first column with the other three columns, that is the linear model with nonlinear models, we can observe that the linear model has a smaller variance than the nonlinear models. Also, among the nonlinear models, we can notice a difference in both the bias and the variance components which can be explained by the difference in flexibility among these models.

Concerning the bias component, we can see that it is high for the linear model and low for the nonlinear models. This can be explained by the fact that the DGP is nonlinear and so the linear model cannot capture this nonlinearity. Also, the behavior of the bias component is changing with the model and the strategy. Increasing the sample size is also decreasing the bias, especially at short horizons, but is less pronounced than with the variance component. Finally, we can also

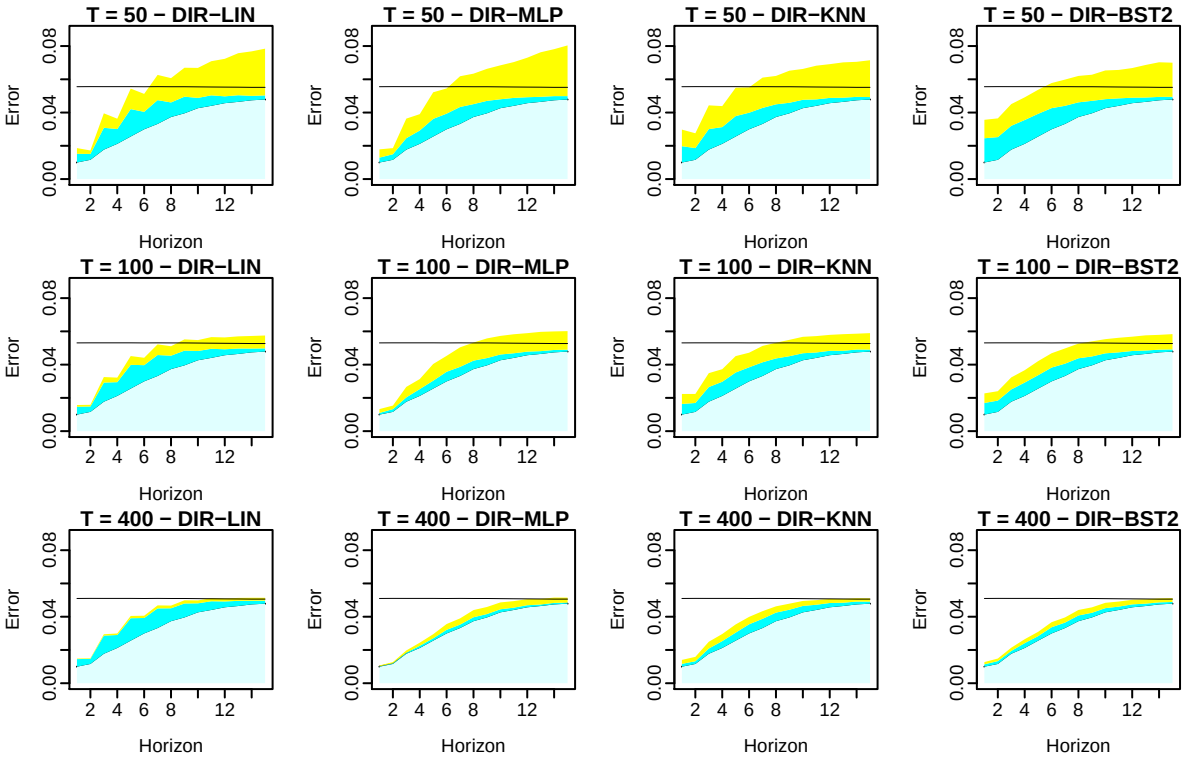

Figure 4.2: *STAR DGP. The MSE of direct forecasts generated with different learning models (in column) and different time series lengths (in row) is decomposed into noise (in grey), bias (in cyan) and variance (in yellow). The stacked area plots show the relative contribution of each component to the total MSE over the forecast horizon.*

notice that the bias and variance components have comparable values for short horizons but the variance tends to dominate the bias for longer horizons.

From these first observations, we have seen that the bias and variance components of each strategy depend on many interacting factors including the learning model, the DGP, the time series length as well as the forecast horizon.

In the following sections, we will compare the recursive and direct strategies in the four different scenarios presented in Section 4.3.2. For each scenario, we will first compute the bias and variance components of the two-step ahead forecast errors, as given in expression (4.3.1). For the expressions of the two-step ahead forecasts, we will use expressions (4.4.4) and (4.4.6) for a linear model and expressions (4.4.2) and (4.4.5) for a nonlinear model. For the conditional mean at horizon $h = 2$, we will use expression (4.3.5) for a linear DGP and (4.3.2) for a nonlinear DGP.

For each scenario, we will also compare the recursive and direct strategies for the general case of h-step ahead forecasts with Monte Carlo simulations. To allow a better visualization and a deeper comparison between the strategies, the different components will be displayed differently than in Figures 4.1 and 4.2. Each figure will give for different values of T, namely $\{50, 100, 400\}$, the Monte Carlo estimations of the unconditional MSE (first column), the bias (second column), the variance (third column) and the bias plus variance (fourth column). In the first column, corresponding to the MSE, the bias and variance components of the different strategies are hidden by substantial noise, making comparisons between strategies difficult. Consequently, we will consider the three other columns to compare the strategies and use the MSE as a measure of the predictability of the time series relative to the mean (the red line). The MSE of the conditional expectation, which also represents the noise variance, will be plotted in grey.

4.4.1 Scenario A: Linear model and linear DGP

For the recursive strategy, expression (4.3.1) is given by

$$B_2^{\text{REC}}(\boldsymbol{x}_t) + V_2^{\text{REC}}(\boldsymbol{x}_t) \tag{4.4.7}$$
$$= \Big[\Big((\varphi_0 + \varphi_1\varphi_0) + (\varphi_1^2 + \varphi_2)y_t + \cdots + (\varphi_1\varphi_{d-1} + \varphi_p)y_{t-d+2} + (\varphi_1\varphi_p)y_{t-d+1}\Big) \tag{4.4.8}$$
$$-\Big((\phi_0 + \phi_1\phi_0) + (\phi_1^2 + \phi_2)y_t + \cdots + (\phi_1\phi_{p-1} + \phi_p)y_{t-p+2} + (\phi_1\phi_p)y_{t-p+1}\Big)\Big]^2 \tag{4.4.9}$$
$$+ (\phi_1 + 1)^2\eta(\boldsymbol{\phi})^2. \tag{4.4.10}$$

For the direct strategy, expression (4.3.1) is given by

$$B_2^{\text{DIR}}(\boldsymbol{x}_t) + V_2^{\text{DIR}}(\boldsymbol{x}_t) \tag{4.4.11}$$
$$= \Big[\Big((\varphi_0 + \varphi_1\varphi_0) + (\varphi_1^2 + \varphi_2)y_t + \cdots + (\varphi_1\varphi_{d-1} + \varphi_d)y_{t-d+2} + (\varphi_1\varphi_d)y_{t-d+1}\Big) \tag{4.4.12}$$
$$-\Big(\gamma_0 + \gamma_1 y_t + \cdots + \gamma_{p-1}y_{t-p+2} + \gamma_p y_{t-p+1}\Big)\Big]^2 \tag{4.4.13}$$
$$+ \eta(\boldsymbol{\gamma})^2.$$

The recursive strategy first estimates a linear model, as in (4.4.3), for the linear DGP, as in (4.3.4); then two-step forecasts are produced recursively using the estimated model. In contrast, the direct strategy estimates a linear model that directly estimates two-step ahead dependencies where the parameters γ estimate a nonlinear transformation of the coefficients φ. Then, two-step ahead forecasts can be directly produced from the model.

If the model is well-specified, that is $p = d$, and if the parameters are correctly identified ($\varphi_j = \phi_j$ for all $j = 1,\ldots,p$), then the two-step ahead forecasts of the recursive strategy will be unbiased as can be seen in (4.4.8)-(4.4.9). The two-step direct forecasts will also be unbiased provided that the direct parameters γ correctly identify the nonlinear transformation of the coefficients φ as can be seen in (4.4.12)-(4.4.13). In other words, if $p \geq 3$, we need to have $\gamma_0 = \varphi_0 + \varphi_1\varphi_0$, $\gamma_j = \varphi_1\varphi_j + \varphi_{j+1}$ for $j = 1,\ldots,p-1$, and $\gamma_p = \varphi_1\varphi_p$. If the model is well-specified, the recursive strategy is known to provide more efficient parameter estimates than the direct strategy, notably because the recursive strategy uses more data points (see Section 3.4).

If the model is misspecified, that is $p \neq d$ or if the parameters are not correctly identified ($\varphi_j \neq \phi_j$ for any $j = 1,\ldots,p$), then we can see in (4.4.8)-(4.4.9) that two-step ahead forecasts will be biased. However, since the parameters are selected by minimizing 2-step ahead errors, as can be seen in (3.3.6) with $h = 2$, direct forecasts are more robust to misspecifications.

Finally, it is worth noting that a model is an approximation of the reality and thus in practice a model is almost always misspecified. With the previous discussion, one might think that direct linear forecasts provide better forecasts than recursive linear forecasts in practice. However, even if biased, recursive linear forecasts can have a smaller variance than direct linear forecasts especially for long horizons and subsequently obtain a smaller MSE.

Let us now consider Monte carlo simulations for the general case of h-step ahead forecasts. We will consider both a well-specified and a misspecified model for the AR DGP defined in (4.3.7). For the well-specified model, the lag order p is allowed to range in $\{1,\ldots,7\}$, which includes the true lag order $d = 6$. For the misspecified model, the lag order p is only allowed to range in $\{1,2,3\}$ and thus cannot be equal to the true lag order $d = 6$. We will use LIN to denote a well-specified model and LINMIS to denote a misspecified model. The results are given in Figure 4.3.

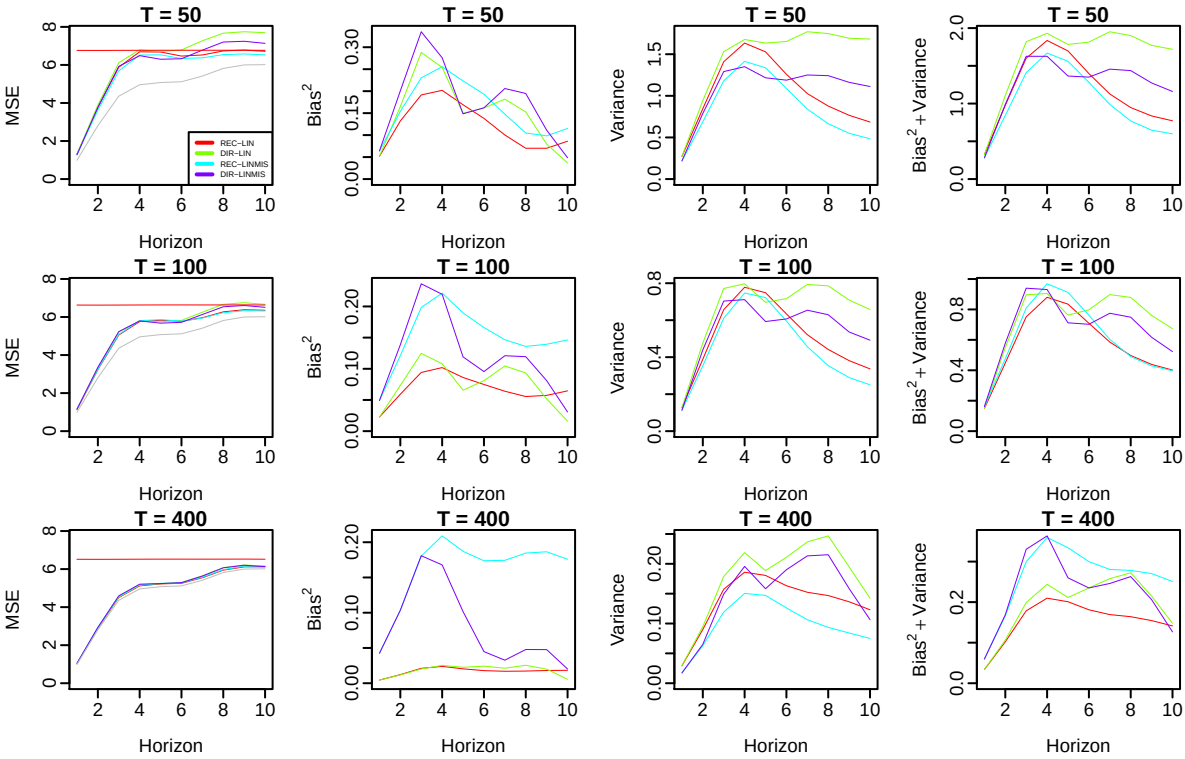

Figure 4.3: *AR DGP. MSE decomposition of recursive (REC) and direct (DIR) forecasts with a well-specified linear model (LIN) and a misspecified linear model (LINMIS).*

Let us first compare REC-LIN with DIR-LIN. In the last column, we can see for all values of T that after the first few horizons, REC-LIN has smaller errors than DIR-LIN. For $T = 50$, we can see in the second and the third column that REC-LIN has both a smaller bias and a smaller variance, with a larger difference in variance. For $T = 100$ and $T = 400$, the difference is mainly in the variance component. This confirms the theoretical findings that, when the model is well-specified, the recursive strategy produces more efficient parameter estimates that lead to better forecasts. Finally, in the first column, we can see that the linear forecasts become closer and closer to the optimal forecasts (grey line) as T increases. Recall that under appropriate conditions, REC-LIN and DIR-LIN are asymptotically equivalent (see Section 3.4).

Let us now compare REC-LINMIS with DIR-LINMIS. In the last column, we can see for $T = 50$ and $T = 100$ that REC-LINMIS has smaller errors than DIR-LINMIS for long horizons, due to a smaller variance as can be seen in third column. For $T = 100$ and $T = 400$, we can see a smaller bias for DIR-LINMIS in the second column, which confirms the robustness to misspecification of the direct strategy.

One limitation of this comparison is that the variance component is also affected by the variability of the lag order set. In fact, the lag order p for LIN ranges in the set $\{1, 2, 3\}$ while for LINMIS, it ranges in the set $\{1, \ldots, 7\}$. Ideally, to see the difference between a well-specified and a misspecified model, we should set the value of the lag order to a fixed value and analyze the bias and variance components. However, we decided to keep the variability of the lag order set to mimic a realistic scenario where the lag order must also be selected.

Let us now compare REC-LINMIS with REC-LIN and DIR-LINMIS with DIR-LIN. For $T = 50$, we can see in the last column that both REC-LINMIS and DIR-LINMIS have smaller errors than REC-LIN and DIR-LIN, respectively. The error decrease is mainly due to the decrease in the variance component, particularly for long horizons. For $T = 100$, the increase in bias becomes

important as can be seen in the second column and so the performances of the strategies become closer. For $T = 400$, the increase in bias becomes more important than the decrease in variance and so REC-LIN and DIR-LIN end up having smaller errors than REC-LINMIS and DIR-LINMIS.

4.4.2 Scenario B: Linear model and nonlinear DGP

For the recursive strategy, expression (4.3.1) is given by

$$B_2^{\text{REC}}(\boldsymbol{x}_t) + V_2^{\text{REC}}(\boldsymbol{x}_t) \tag{4.4.14}$$

$$= \Bigg[\underbrace{f(f(\boldsymbol{x}_t), y_t, \ldots, y_{t-d+2}) + \frac{1}{2}\sigma^2 f_{x_1 x_1}}_{\mu_{t+2|t}} \tag{4.4.15}$$

$$- \Big((\phi_0 + \phi_1\phi_0) + (\phi_1^2 + \phi_2) y_t + \cdots + (\phi_1\phi_{p-1} + \phi_p) y_{t-p+2} + (\phi_1\phi_p) y_{t-p+1} \Big) \Bigg]^2 \tag{4.4.16}$$

$$+ (\phi_1 + 1)^2 \eta(\boldsymbol{\phi})^2$$

For the direct strategy, expression (4.3.1) is given by

$$B_2^{\text{DIR}}(\boldsymbol{x}_t) + V_2^{\text{DIR}}(\boldsymbol{x}_t) \tag{4.4.17}$$

$$= \left[\underbrace{f(f(\boldsymbol{x}_t), y_t, \ldots, y_{t-d+2}) + \frac{1}{2}\sigma^2 f_{x_1 x_1}}_{\mu_{t+2|t}} - \Big(\gamma_0 + \gamma_1 y_t + \cdots + \gamma_p y_{t-p+1} \Big) \right]^2 \tag{4.4.18}$$

$$+ \eta(\boldsymbol{\gamma})^2$$

Because the DGP is nonlinear, we have $f_{x_1 x_1} \neq 0$ in (4.4.15) and (4.4.18) and as a result, the two-step ahead recursive and direct linear forecasts are biased since linear models cannot capture the nonlinearity of the function f. In particular, the bias will be high if $|f_{x_1 x_1}|$ is large; that is when f has large curvatures or if $p \neq d$.

In the scenario A (previous section), the difference between the bias component of the recursive and direct strategies was mainly due to the different way of generating the forecasts, since there were no discrepancies between the model and the DGP. In this scenario, there is an additional source of bias stemming from the non-captured nonlinearity. The bias due to the nonlinearity is expected to be similar for the recursive and direct strategies, and therefore the difference is expected to come from the different way of generating the forecasts.

The performance of the recursive and direct strategies in this scenario will depend on the nonlinearity of the function f, that is on $|f_{x_1 x_1}|$. In particular, if the function is weakly nonlinear, that is $|f_{x_1 x_1}|$ is small, linear forecasts will provide a good first order approximation and then this scenario will reduce to scenario A. For strongly nonlinear function f, the bias from both strategies will be high but the reduced variance of the linear model can induce a smaller MSE.

In fact, we expect the variance component of the recursive strategy to be smaller than the direct strategy especially with short time series and at long horizons. The higher variance of the direct strategy should come from its additional flexibility (e.g. different lag orders at each horizon) and the reduced data set at each horizon.

Let us now consider Monte Carlo simulations for the general case of h-step ahead forecasts. The results for the STAR and the NAR DGP are given in Figures 4.4 and 4.5, respectively.

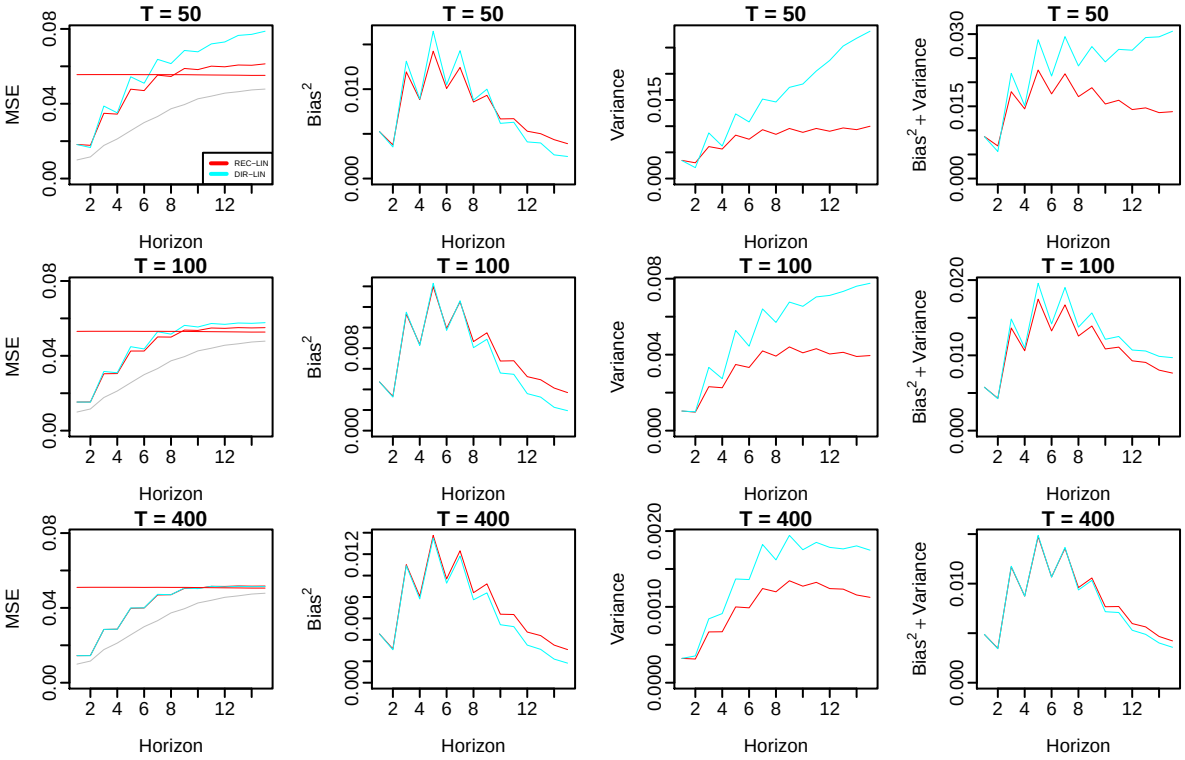

Figure 4.4: *STAR DGP. MSE decomposition of recursive (REC) and direct (DIR) forecasts with the LIN model.*

In both Figures 4.4 and 4.5, we can see in the last column that REC-LIN has smaller errors than DIR-LIN for both $T = 50$ and $T = 100$ consistently over the horizon. The third column shows that this is due to a smaller variance that dominates the errors.

4.4.3 Scenario C: Nonlinear model and linear DGP

This scenario is closely related to scenario A with the difference that the model is allowed to be nonlinear even if the true DGP is linear. The term "allow" is important since, for certain class of models, the final model can be a linear model, e.g. a neural network with zero hidden nodes.

For the recursive strategy, expression (4.3.1) is given by

$$B_2^{\text{REC}}(\boldsymbol{x}_t) + V_2^{\text{REC}}(\boldsymbol{x}_t) \tag{4.4.19}$$

$$= \Big[\Big(\delta(f(\boldsymbol{x}_t), y_t, y_{t-1}, \ldots, y_{t-p+2}; \boldsymbol{\phi}) + \delta(\boldsymbol{z}_t; \boldsymbol{\phi}) m_{z_1} \tag{4.4.20}$$

$$+ \frac{1}{2}[\delta(\boldsymbol{z}_t; \boldsymbol{\phi})]^2 m_{z_1 z_1} + \frac{1}{2}[\eta(\boldsymbol{z}_t; \boldsymbol{\phi})]^2 m_{z_1 z_1}\Big)\Big]^2 \tag{4.4.21}$$

$$+ [\eta(f(\boldsymbol{x}_t), y_t, y_{t-1}, \ldots, y_{t-p+2}; \boldsymbol{\phi})]^2 + [\eta(\boldsymbol{z}_t; \boldsymbol{\phi}) m_{z_1}]^2 \tag{4.4.22}$$

$$+ \frac{1}{2}[\eta(\boldsymbol{z}_t; \boldsymbol{\phi})]^4 m_{z_1 z_1}^2$$

$$+ 2\eta(f(\boldsymbol{x}_t), y_t, y_{t-1}, \ldots, y_{t-p+2}; \boldsymbol{\phi}) \eta(\boldsymbol{z}_t; \boldsymbol{\phi}) m_{z_1} \mathbb{E}[\varepsilon_{\eta_1} \varepsilon_{\eta_2}]$$

$$+ \eta(\boldsymbol{z}_t; \boldsymbol{\phi})^2 \eta(f(\boldsymbol{x}_t), y_t, y_{t-1}, \ldots, y_{t-p+2}; \boldsymbol{\phi}) m_{z_1 z_1} \mathbb{E}[\varepsilon_{\eta_1}^2 \varepsilon_{\eta_2}], \tag{4.4.23}$$

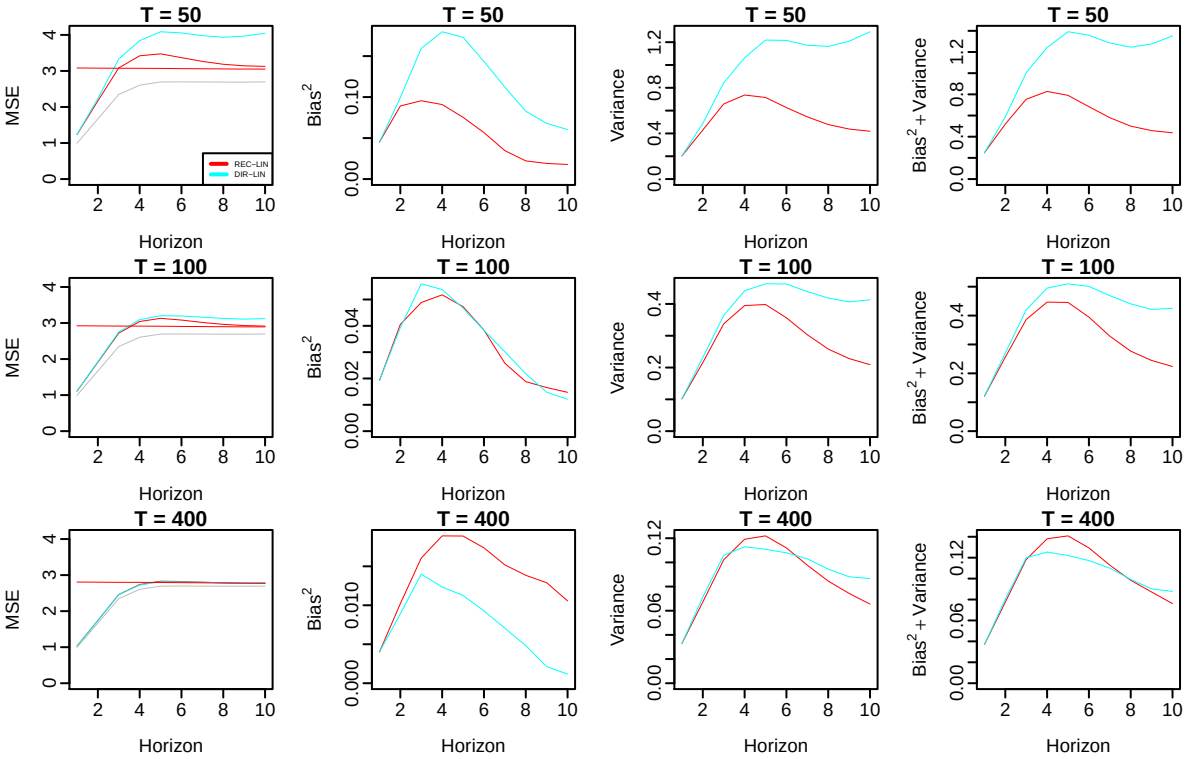

Figure 4.5: *NAR DGP. MSE decomposition of recursive (REC) and direct (DIR) forecasts with the LIN model.*

where we used the fact that $\mathbb{E}[\varepsilon_\eta^3] = 0$ and $\mathbb{E}[\varepsilon_\eta^4] = 3$ for the standard normal distribution.

For the direct strategy, expression (4.3.1) is given by

$$B_2^{\text{DIR}}(\boldsymbol{x}_t) + V_2^{\text{DIR}}(\boldsymbol{x}_t) \tag{4.4.24}$$

$$= \left[\underbrace{(\varphi_0 + \varphi_1\varphi_0) + (\varphi_1^2 + \varphi_2)y_t + \cdots + (\varphi_1\varphi_{d-1} + \varphi_d)y_{t-d+2} + (\varphi_1\varphi_d)y_{t-d+1}}_{\mu_{t+2|t}} - m_2(\boldsymbol{r}_t;\boldsymbol{\gamma}) \right]^2$$

$$+ \eta(\boldsymbol{r}_t;\boldsymbol{\gamma})^2. \tag{4.4.25}$$

In expressions (4.4.20)-(4.4.21) for the bias component of the recursive strategy, we can see that the offset $\delta(\cdot;\boldsymbol{\phi})$ and the variability term $\eta(\boldsymbol{z}_t;\boldsymbol{\phi})$ of the first horizon are propagated to the second horizon with an amplification when the model m produces a function that has large variations (i.e. m_{z_1} and $m_{z_1 z_1}$ are large in magnitude). In particular, increasing or decreasing the complexity of the model ($m_{z_1 z_1}$) will not prevent the amplification of the errors except if the model becomes linear, that is $m_{z_1 z_1} = 0$. A similar behavior is observed for the variability terms in the variance component of the recursive strategy, as can be seen in (4.4.22)–(4.4.23).

In contrast to the recursive strategy, the direct strategy does not suffer from the accumulation of errors and, provided that the model is flexible enough, the bias can be made arbitrarily small. However, the direct strategy can have a higher variance due to the irregularities that can be caused by having consecutive forecasts generated by potentially very different models. If these models are allowed to be nonlinear, then the increase in variance will be exacerbated, especially if they have a large flexibility or if the input includes several lagged variables.

In this scenario, because the DGP is linear, nonlinear models that have large variations will only increase the errors and will not bring any benefit. In particular, having $m_{z_1 z_1} \neq 0$ is not necessary and instead, $m_{z_1 z_1}$ should be made equal to zero to avoid the amplification of errors for the recursive strategy and to decrease the variance of the direct models.

In consequence, this scenario will be favorable to nonlinear models that effectively switch to a linear model. In that case, we will have $m_{z_1 z_1} = 0$ and $m_{z_1} = \phi_1$. For the bias component, we will obtain $(\phi_1 + 1)^2 \delta(\boldsymbol{\phi})^2$, which is a different way of writing expressions (4.4.8)-(4.4.9). In fact, the term $\delta(\boldsymbol{\phi})$ includes the errors due to having $p \neq d$ or $\phi_j \neq \varphi_j$ for some $j = 1, \ldots, p$. For the variance component, we will obtain $(\phi_1 + 1)^2 \eta(\boldsymbol{\phi})^2$, the same variance component as in expression (4.4.10).

For the direct strategy, if the model switches to a linear model, then (4.4.24) would be similar to (4.4.11). One notable difference with the recursive strategy is that for the direct forecasts with nonlinear models to be exactly equal to the direct forecasts of scenario A, all the models should switch to linear models. Because of the noise and the limited size of real-world time series, requiring all nonlinear direct models to switch to linear models can be more difficult than requiring a one-step recursive model to switch to a linear model. Nevertheless, if both the recursive and direct models switch to linear models, then this scenario becomes similar to the scenario A.

Let us now consider the multi-step recursive strategy, where the parameters are estimated by minimizing multi-step-ahead errors instead of one-step-ahead errors but where the forecasts are still obtained recursively as with the recursive strategy. For this strategy, we need to find to select the parameters of a model, that when applied recursively several times is able to estimate the conditional mean.

Compared to the recursive strategy, both the bias and variance can significantly be reduced especially for long horizons, notably due to the limitation of the error amplification; but for short horizons, the difference between the two strategies is not expected to be high.

Compared to the direct strategy which also deals with multi-step errors, this strategy incurs restrictions on the model that may somewhat limit its ability to fit the true DGP, leading to some bias. However, since the DGP is linear in this scenario, the model parameters might be easier to select. The two strategies can a have close variance component for short time series but the difference should increase with a longer time series, notably because the multi-step recursive strategy still produces forecasts recursively.

Let us now consider Monte Carlo simulations for the general case of h-step ahead forecasts. Figure 4.6 gives the results for the KNN and MLP models with the AR DGP.

Let us first consider the KNN model. In the last column of Figure 4.6, we can see that DIR-KNN has smaller errors than REC-KNN consistently over the horizon. If we look at the bias and variance components in the second and the third columns, we can see that the smaller errors of DIR-KNN are due to its smaller variance since it does not suffer from the amplification of errors as with REC-KNN. In fact, both strategies have a comparable bias component but DIR-KNN has a significantly smaller variance.

For the MLP model, we obtained different results than for the KNN model. In the last column of Figure 4.6, we see that REC-MLP has smaller errors than DIR-MLP. Similarly to the KNN model, both strategies have a comparable bias component but REC-MLP has a smaller variance than DIR-MLP. This can be explained by the fact that, in contrast to the KNN model, the MLP model has the ability to switch or at least to be close to a linear model. In this case, since the DGP is indeed linear, the MLP model can match the DGP and so REC-MLP can benefit from more efficient parameter estimates than DIR-MLP. The fact that the MLP model is closer to the linear DGP than

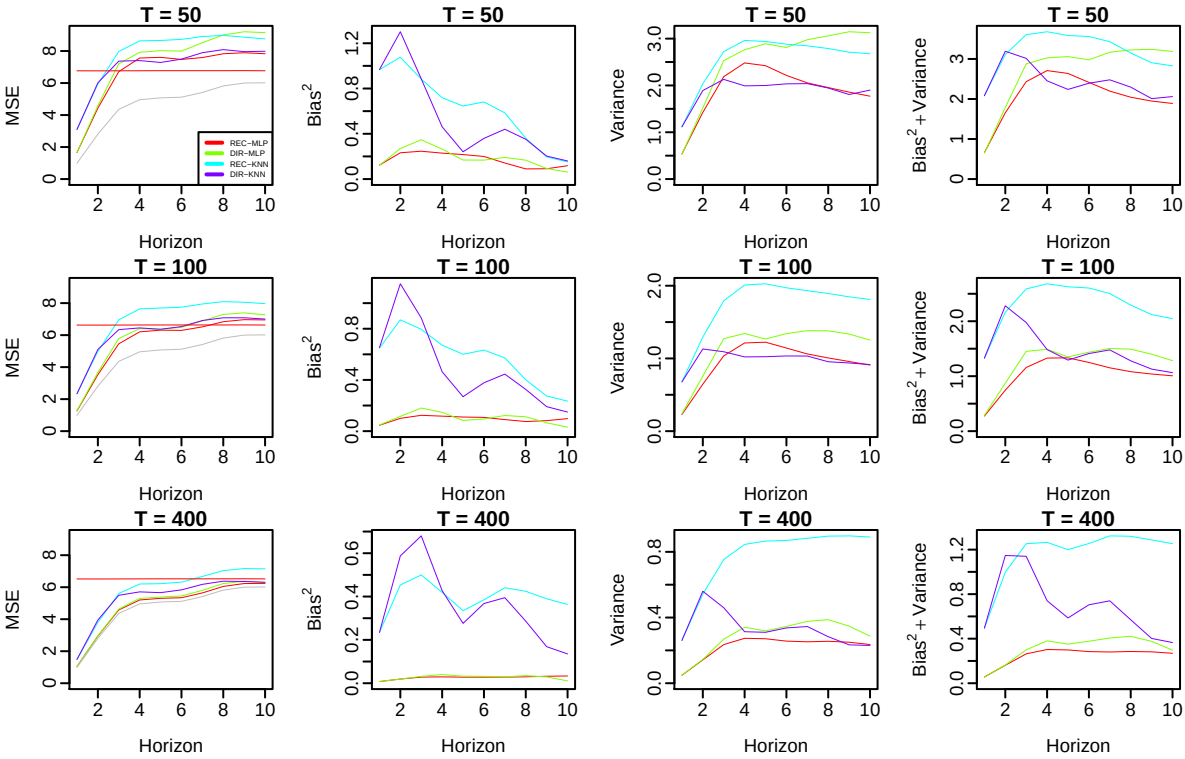

Figure 4.6: *AR DGP. MSE decomposition of recursive (REC) and direct (DIR) forecasts with the MLP and the KNN model.*

the KNN model is also confirmed by the smaller bias component for both strategies as can be seen in the second column of Figure 4.6.

To illustrate the cost of extending the model space beyond the DGP, Figure 4.7 gives the results for the MLP and the LIN model.

As expected, we see in the last column of Figure 4.7 that the LIN model has a smaller error than the MLP model for both recursive and direct strategies. In the third column, we can see that the gain with the LIN model with respect to the MLP model is mainly in terms of variance. For $T = 50$ and $T = 100$, both models have a close bias component. For $T = 400$, the relative gap in bias is more noticeable.

Ideally, we want the MLP model to always switch to the LIN model since the true DGP is linear. However, because of the limited amount of data and the overfitting problem, it is hard to achieve that setting in practice. In other words, the MLP model will not necessarily select zero hidden nodes, and as a result, the MLP model will suffer from a higher variance due to the additional flexibility. However, if a small number of hidden nodes is used such as one hidden node, then the MLP model will not be too far from the LIN model and the increase in variance will be small.

Figure 4.8 shows the results for the multi-step recursive strategy with the KNN model.

In the last column, we can see that RTI-KNN has significantly reduced the errors of REC-KNN. The third column shows that this is due to the high decrease in variance and in the second column, we can see that the different strategies have a comparable bias component. This suggests that the main advantage of RTI-KNN compared to REC-KNN is the smaller variance of the forecasts.

If we compare RTI-KNN with DIR-KNN in the last column, we can see that DIR-KNN dominates RTI-KNN consistently over the horizon. This is also observed in the second and third columns for

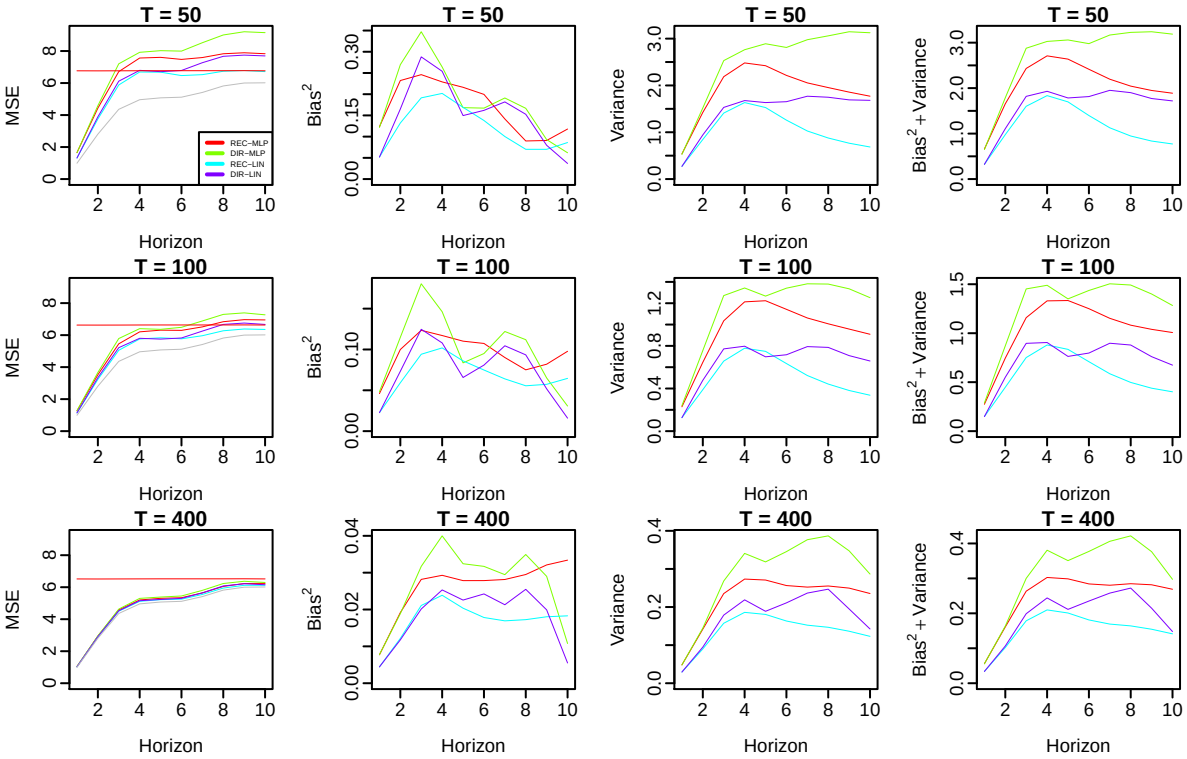

Figure 4.7: *AR DGP. MSE decomposition of recursive (REC) and direct (DIR) forecasts with the MLP and the LIN model.*

the bias and variance components. We can explain this behavior by the restrictions incurred by the RTI-KNN on the model that may somewhat limit its ability to fit the true DGP, leading to some bias. Also, although it allows a different set of parameters at each horizon, the forecasts are still produced recursively, which can explain the higher variance of RTI-KNN compared to DIR-KNN.

4.4.4 Scenario D: Nonlinear model and nonlinear DGP

In this scenario, the DGP is nonlinear as in scenario B (Section 4.4.2), but now the model is nonlinear and can fit a nonlinear function.

For the recursive strategy, expression (4.3.1) is given by

$$
\begin{aligned}
& B_2^{\mathrm{REC}}(\boldsymbol{x}_t) + V_2^{\mathrm{REC}}(\boldsymbol{x}_t) && (4.4.26)\\
&= \Bigg[\tfrac{1}{2}\sigma^2 f_{x_1 x_1} - \Big(\delta(f(\boldsymbol{x}_t), y_t, y_{t-1}, \ldots, y_{t-p+2}; \boldsymbol{\phi}) + \delta(\boldsymbol{z}_t; \boldsymbol{\phi}) m_{z_1} && (4.4.27)\\
&\qquad + \frac{1}{2}[\delta(\boldsymbol{z}_t; \boldsymbol{\phi})]^2 m_{z_1 z_1} + \frac{1}{2}[\eta(\boldsymbol{z}_t; \boldsymbol{\phi})]^2 m_{z_1 z_1 z_1}\Big)\Bigg]^2 && (4.4.28)\\
&+ [\eta(f(\boldsymbol{x}_t), y_t, y_{t-1}, \ldots, y_{t-p+2}; \boldsymbol{\phi})]^2 + [\eta(\boldsymbol{z}_t; \boldsymbol{\phi}) m_{z_1}]^2 && (4.4.29)\\
&\quad + \frac{1}{2}[\eta(\boldsymbol{z}_t; \boldsymbol{\phi})]^4 m_{z_1 z_1}^2\\
&\quad + 2\eta(f(\boldsymbol{x}_t), y_t, y_{t-1}, \ldots, y_{t-p+2}; \boldsymbol{\phi}) \eta(\boldsymbol{z}_t; \boldsymbol{\phi}) m_{z_1} \mathbb{E}[\varepsilon_{\eta_1} \varepsilon_{\eta_2}]\\
&\quad + \eta(\boldsymbol{z}_t; \boldsymbol{\phi})^2 \eta(f(\boldsymbol{x}_t), y_t, y_{t-1}, \ldots, y_{t-p+2}; \boldsymbol{\phi}) m_{z_1 z_1} \mathbb{E}[\varepsilon_{\eta_1}^2 \varepsilon_{\eta_2}], && (4.4.30)
\end{aligned}
$$

where we used the fact that $\mathbb{E}[\varepsilon_\eta^3] = 0$ and $\mathbb{E}[\varepsilon_\eta^4] = 3$ for the standard normal distribution.

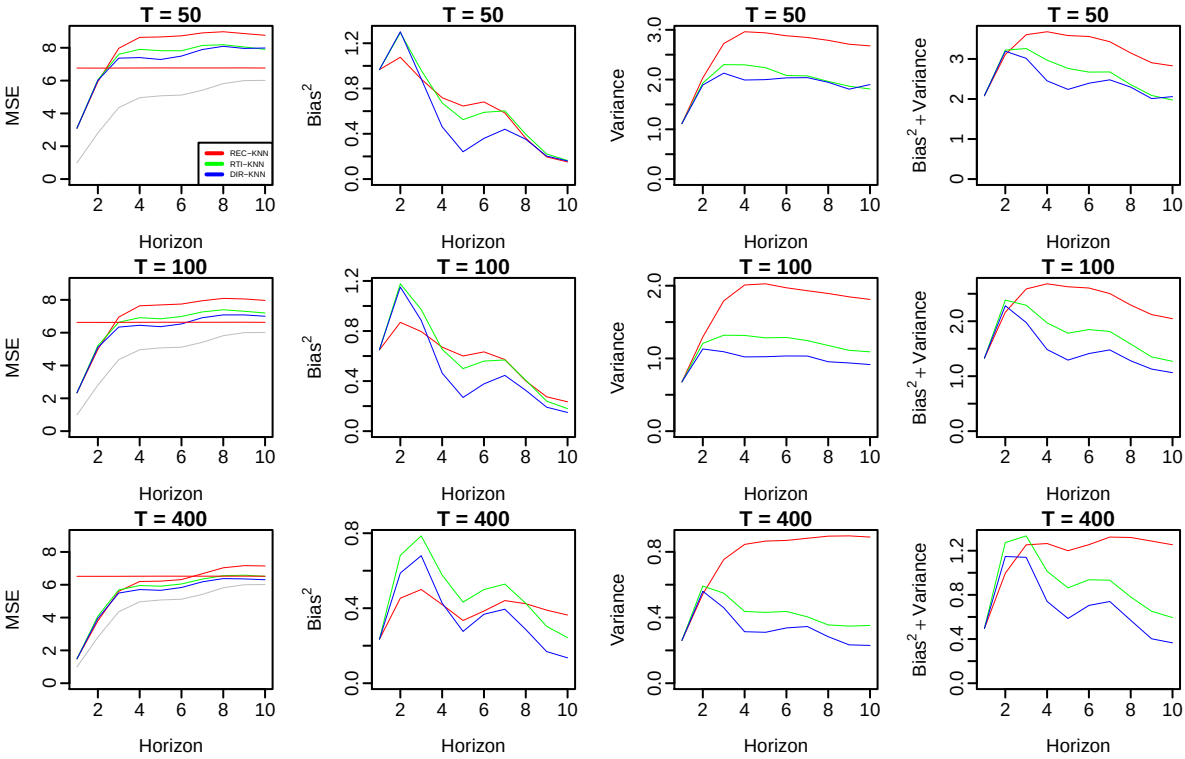

Figure 4.8: *AR DGP. MSE decomposition of recursive (REC), multi-step recursive (RTI) and direct (DIR) forecasts with the KNN model.*

For the direct strategy, expression (4.3.1) is given by

$$
\begin{aligned}
&B_2^{\text{DIR}}(\boldsymbol{x}_t) + V_2^{\text{DIR}}(\boldsymbol{x}_t) \\
&= \Bigg[\underbrace{f(f(\boldsymbol{x}_t), \ldots, y_{t-d+2}) + \tfrac{1}{2}\sigma^2 f_{x_1 x_1}}_{\mu_{t+2|t}} - m_2(\boldsymbol{r}_t; \boldsymbol{\gamma}) \Bigg]^2 \\
&\quad + \eta(\boldsymbol{r}_t; \boldsymbol{\gamma})^2.
\end{aligned} \tag{4.4.31}
$$

Similarly to scenario C (Section 4.4.3), we see for the bias component of the recursive strategy in (4.4.27)-(4.4.28), an amplification of both the offset $\delta(\cdot;\boldsymbol{\phi})$ and the variability term $\eta(\cdot;\boldsymbol{\phi})$ when the model m produces a function that has large variations (i.e. m_{z_1} and $m_{z_1 z_1}$ are large in magnitude). However, in contrast to scenario C, since the DGP is nonlinear, i.e. $f_{x_1 x_1} \neq 0$, we now have an additional term $\frac{1}{2}\sigma^2 f_{x_1 x_1}$ that will remain even if the model is able to perfectly estimate the function f, i.e. even if $\delta(\cdot;\boldsymbol{\phi}) = 0$ and $\eta(\cdot;\boldsymbol{\phi}) = 0$. In fact, nonlinear recursive forecasts for a nonlinear DGP are known to be asymptotically biased even if the model is well-specified (Brown and Mariano, 1984; Lin and Granger, 1994). In particular, the bias will be large whenever $|f_{x_1 x_1}|$ is large; that is when f is highly nonlinear or has high curvatures.

An amplification of the variability terms is also observed in the variance component as can be seen in (4.4.29)–(4.4.30). One difference with scenario C is that now the DGP is nonlinear and hence the estimated model is expected to be more complex than with a linear DGP, which will in turn induce a higher amplification of the variability terms.

The direct strategy would generally be preferred in that case since it estimates the conditional mean directly and therefore does not suffer from the accumulation of errors, which can be particularly high in this scenario, where both the model and the DGP are nonlinear. Also, since the model m_2 is nonlinear and hence has a high flexibility, the bias can be arbitrarily small provided that $m_2 \asymp \mu_{t+2|t}$ where $\asymp$ means "has the same form". Although the direct strategy does not suffer from the accumulation of errors, it can suffer from a high variance at long horizons. In fact, the direct strategy has a dataset reduced by $h-1$ points at horizon h which can be particularly large for short time series. Also, we can see in (4.4.25) that the variance depends notably on the variability induced by the dimensionality of the input vector $\boldsymbol{r}_t$ and the model parameters $\boldsymbol{\gamma}$. This variability can be particularly large for highly nonlinear models which involve many interacting variables in $\boldsymbol{r}_t$ or have a large set of parameters $\boldsymbol{\gamma}$.

In scenario B, we have seen that multi-step forecasts with linear models are biased because they cannot capture the nonlinearity of the DGP. For the recursive strategy, allowing the model to be nonlinear does not remove the bias for long horizons and can possibly make the error worse than using linear models, due to the increased variance. For the direct strategy, having nonlinear models allow to capture the nonlinearity of the forecast function. However, even if the DGP is nonlinear, the low variance of linear models can make the MSE smaller than the high variance of nonlinear models, particularly if the DGP is weakly nonlinear or if the time series is short.

Let us now consider the multi-step recursive strategy. Because both the model and the DGP are nonlinear, this scenario is the worst scenario for the recursive strategy which suffers from the amplification of errors. So, the multi-step recursive strategy is expected to significantly reduce the errors by taking into account the propagation of errors when selecting the model parameters. Compared to the direct strategy, the multi-step recursive strategy incurs restrictions on the model that may limit its ability to fit the true DGP. In particular, because the DGP is nonlinear, finding the set of parameters might be more difficult than in scenario C, which can lead to some bias.

Let us now consider Monte Carlo simulations for the general case of h-step ahead forecasts. Figures 4.9 and 4.10 show the results for the NAR and the STAR DGP, respectively.

In the last column of Figure 4.9, the amplification of errors can be clearly seen for both REC-MLP and REC-KNN. In fact, we can see the errors of the recursive strategy growing linearly with the horizon while the errors of the direct strategy are stabilizing. The same behavior is observed for both the bias and variance components of the recursive strategy in the second and third columns. This is consistent with the theoretical analysis performed above. For $T = 50$, the difference between the recursive and direct strategies is small because of the high variance that dominates the errors with short time series.

In the last column of Figure 4.10, we can see that REC-MLP has a smaller error than DIR-MLP consistently over the horizon but the relative difference decreases with longer time series. In the third column, we can see that the component dominating the errors for REC-MLP and DIR-MLP is the variance component.

For the KNN model, we observe the opposite, that is REC-KNN and DIR-KNN have close performance for $T = 50$ but as the time series length increases, DIR-KNN outperforms REC-KNN. Again, in the third column, we can see the importance of the variance component compared to the bias component, with DIR-KNN having a smaller variance than REC-KNN for $T = 100$ and $T = 400$.

If we compare the results for the KNN model and the MLP model in Figure 4.10, we can see in the second column a higher bias for the KNN model compared to the MLP model, particularly for the first few horizons. This can be explained by the fact that the NAR DGP has an MLP model as underlying process (see Section 4.3.2) while the KNN model is a locally constant model. The fact

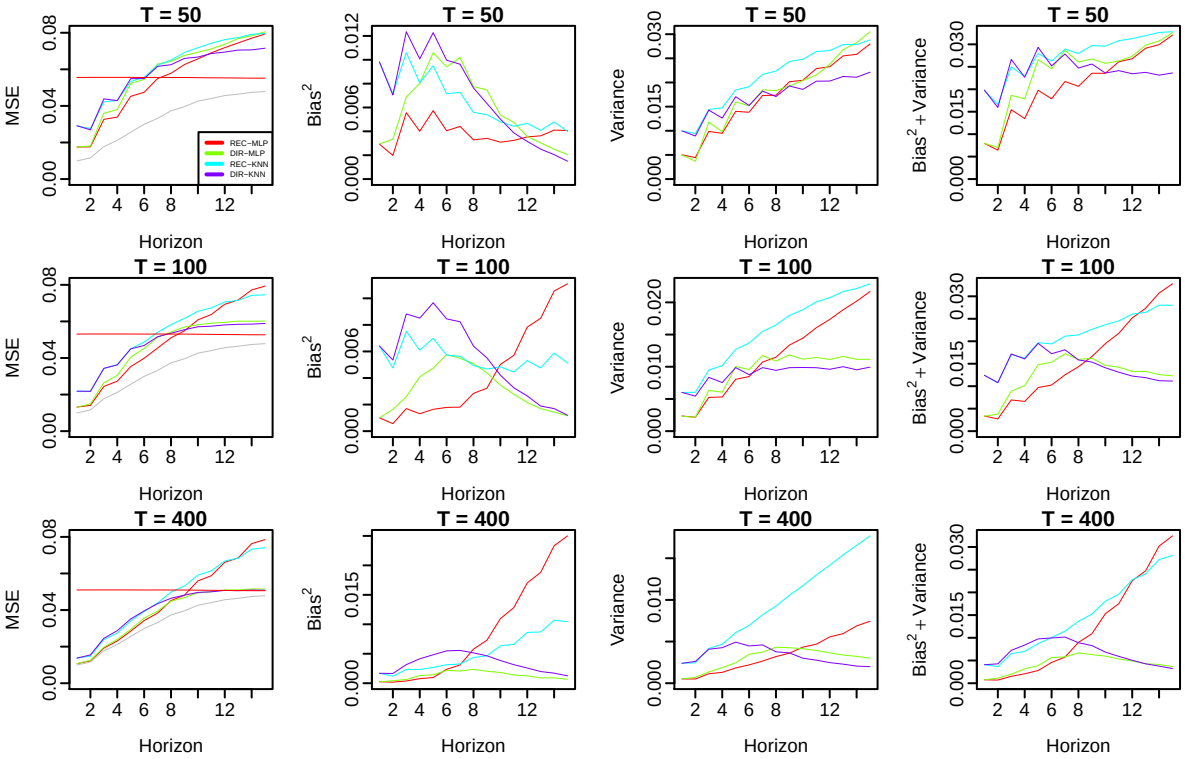

Figure 4.9: *STAR DGP. MSE decomposition of recursive (REC) and direct (DIR) forecasts with the MLP and the KNN model.*

that the bias reduces with the horizon can possibly be explained by the fact that the conditional mean becomes closer to the (unconditional) mean, which is easier to estimate.

In the scenario B, we have seen that the linear model is biased for nonlinear DGPs since it cannot capture the nonlinearity of the function. To shed some light on the differences between linear and nonlinear models when forecasting nonlinear DGPs, Figure 4.11 and 4.12 give the results for both the LIN and MLP models with the STAR and NAR DGPs, respectively.

For the STAR DGP, we can see in the third column of Figure 4.11 that both REC-LIN and DIR-LIN benefit from a lower variance compared to REC-MLP and DIR-MLP for the different values of T. A notable exception is DIR-LIN that has similar performance to REC-MLP and DIR-MLP for $T = 50$. In the second column however, we see a larger bias component for REC-LIN and DIR-LIN compared to REC-MLP and DIR-MLP. For $T = 100$ and $T = 400$, REC-MLP has a higher bias than all strategies at long horizons.

In the last column of Figure 4.11, we can see the sum of the bias and variance components. For $T = 50$, REC-LIN has smaller errors than all strategies particularly for long horizons. For all values of T, we can see that REC-MLP has smaller errors than both REC-LIN and DIR-LIN for short horizons but gets worse for long horizons due to the amplification of errors. DIR-MLP has a comparable error to REC-LIN and DIR-LIN for $T = 100$, and outperforms the LIN model for $T = 400$.

These results are interesting since they show that even if the linear model is biased for the nonlinear DGP, the smaller variance greatly reduces the total MSE for short time series. For longer time series, the increase in bias becomes more important than the decrease in variance.

For the NAR DGP, we can see in the last column of Figure 4.11 that the LIN model outperforms the MLP model and again, the third column shows that this is due to the smaller variance of the LIN

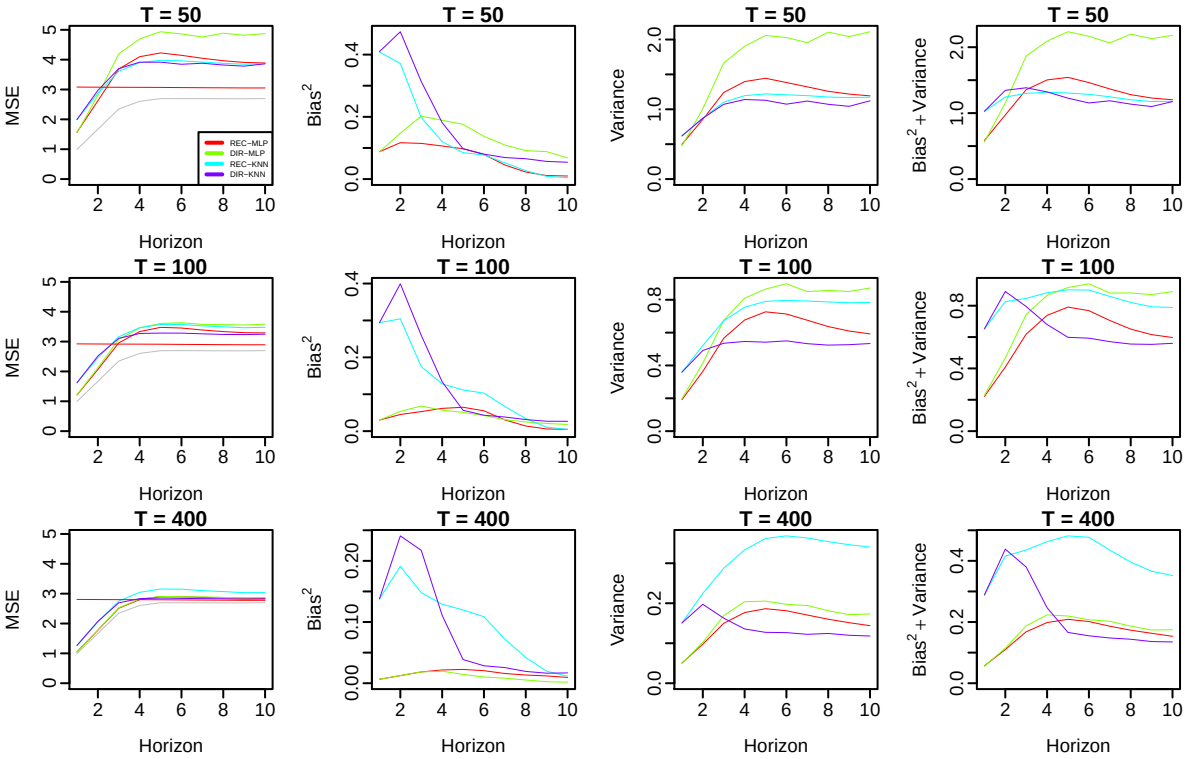

Figure 4.10: *NAR DGP. MSE decomposition of recursive (REC) and direct (DIR) forecasts with the MLP and the KNN model.*

model. In terms of bias, recursive forecasts have smaller errors than direct forecasts for $T = 50$, but as T increases, direct forecasts become better than recursive forecasts as can be seen for $T = 400$.

Figures 4.13 and 4.14 show the results for the multi-step recursive strategy with the KNN model (RTI-KNN).

In both Figures, we can see in the third column that RTI-KNN has a smaller variance than REC-KNN, especially for long horizons. Also, the reduction in variance becomes more important for long time series. In the second column, we can see that REC-KNN has a smaller bias component than RTI-KNN for short horizons, and vice versa for long horizons. The last column shows that RTI-KNN is reducing the errors of REC-KNN for long horizons and has slightly higher errors for short horizons.

If we compared RTI-KNN with DIR-KNN in Figure 4.13, we can see that DIR-KNN has close performance to RTI-KNN for $T = 50$ and outperforms RTI-KNN for $T = 100$ and $T = 400$. In Figure 4.14, DIR-KNN and RTI-KNN have always close performance. Finally, RTI-KNN is always closer to DIR-KNN than REC-KNN.

4.4.5 Summary

We can summarize the observations of the comparison between the recursive and direct strategies for the different scenarios as follows:

- The noise variance dominates the bias and variance components, and the variance component tends to dominate the bias component, especially for long horizons. As the time series length increases, both the bias and variance decrease with a larger decrease for the variance.

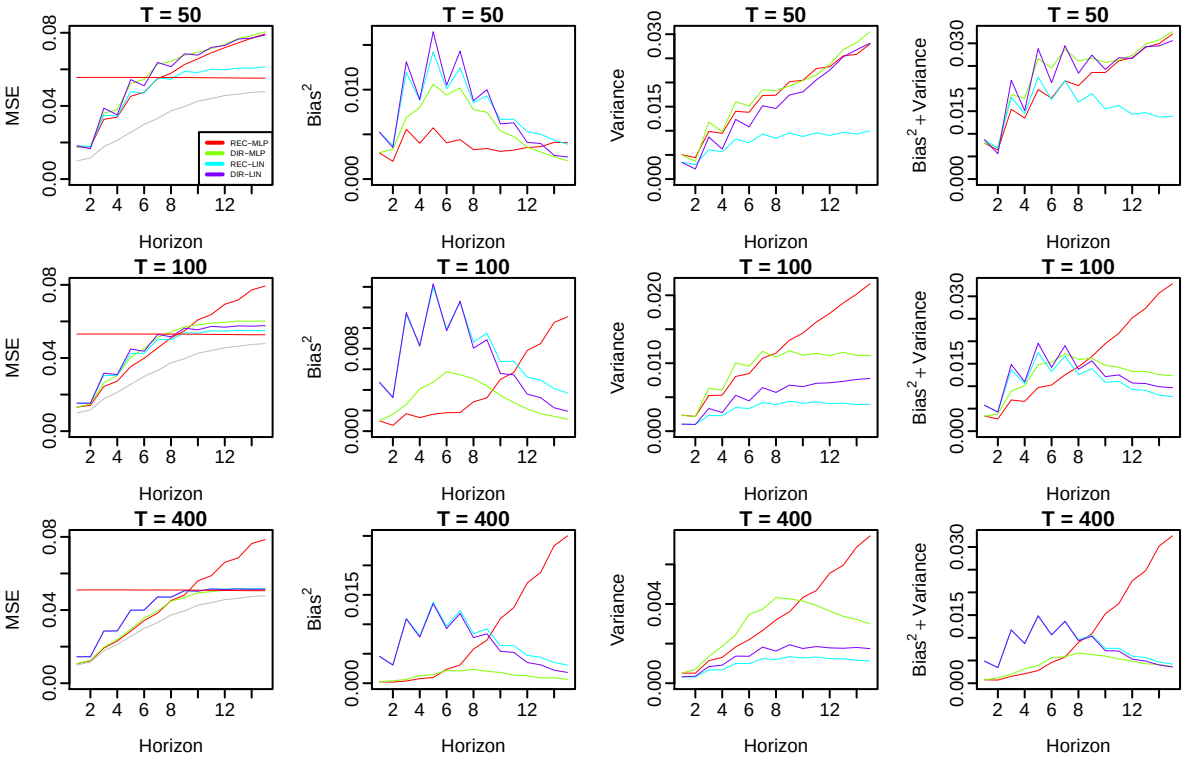

Figure 4.11: *STAR DGP. MSE decomposition of recursive (REC) and direct (DIR) forecasts with the MLP and the LIN model.*

- **Scenario A: Linear model and linear DGP.** With a well-specified model, recursive forecasts have smaller errors than direct forecasts, especially with short time series at long horizons. As the time series length increases, the difference in performance between the two strategies decreases, and asymptotically they become equivalent. We have confirmed the advantage of recursive forecasts with a well-specified model and the robustness of direct forecasts for misspecification.

- **Scenario B: Linear model and nonlinear DGP.** Both recursive and direct forecasts are biased since they cannot generate nonlinear forecasts. In particular, the bias is dominated by the non-captured nonlinearity rather than the difference in the way the forecasts are generated.

  For short time series and long horizons, recursive forecasts are favored because of their reduced variance. For longer time series, recursive and direct forecasts have the same performance.

- **Scenario C: Nonlinear model and linear DGP.** The considered model form plays an important role in the difference between recursive and direct forecasts. If the model is able to effectively switch to a linear model, this scenario becomes equivalent to scenario A.

  If the model is nonlinear, the recursive and direct strategies will generate nonlinear forecasts which will bring no benefits over linear forecasts since the DGP is linear. Because of the additional flexibility of nonlinear models, both recursive and direct forecasts will have a higher variance than linear forecasts.

  In particular, because of the propagation of errors, recursive forecasts will have a higher variance than direct forecasts, which benefit from a higher decrease in variance than recursive forecasts for an increasing time series length.

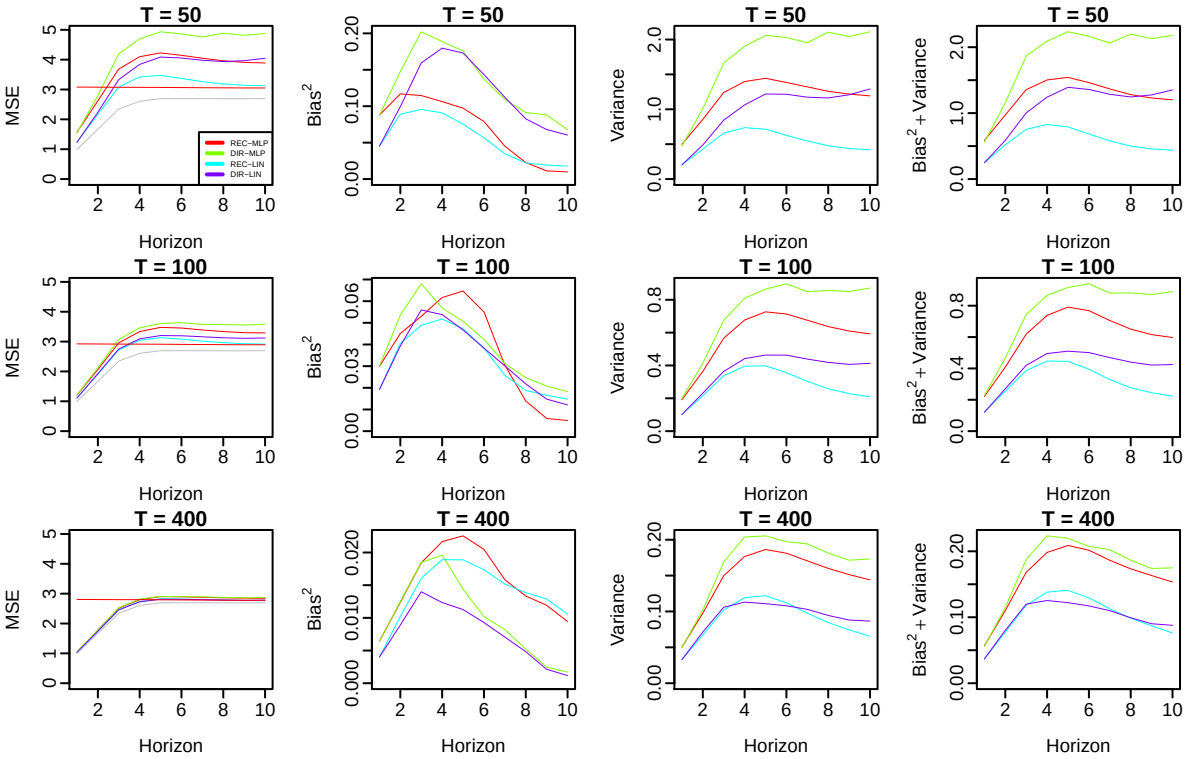

Figure 4.12: *NAR DGP. MSE decomposition of recursive (REC) and direct (DIR) forecasts with the MLP and the LIN model.*

- **Scenario D: Nonlinear model and nonlinear DGP.** Recursive forecasts suffer from the propagation of errors which leads to both a higher bias and variance component compared to direct forecasts especially for long horizons. Also, recursive forecasts are known to be asymptotically biased even if the model is well-specified.

  Although direct forecasts do not suffer from the propagation of errors, they can have a high variance with short time series at long horizons. However, with long time series, direct forecasts have typically lower errors than recursive forecasts, especially for long horizons.

  The linear forecasts in scenario B provide similar or better forecasts than nonlinear forecasts with short time series, particularly with recursive forecasts. This suggests that the lower variance of linear models with short time series can benefit the forecasts even if the forecasts are biased for the nonlinear DGP. Finally, the benefit from nonlinear models will also depend on the degree of nonlinearity of the DGP.

- The main benefit of the multi-step recursive strategy seems to stem from its smaller variance for long horizons compared to the recursive strategy, notably because it takes into account the propagation of errors when estimating the model parameters.

  The multi-step recursive strategy has closer performance to the direct strategy than the recursive strategy, but the direct strategy has always smaller errors since the forecasts of the multi-step recursive strategy are still generated recursively even if a different set of parameters is used at each horizon.

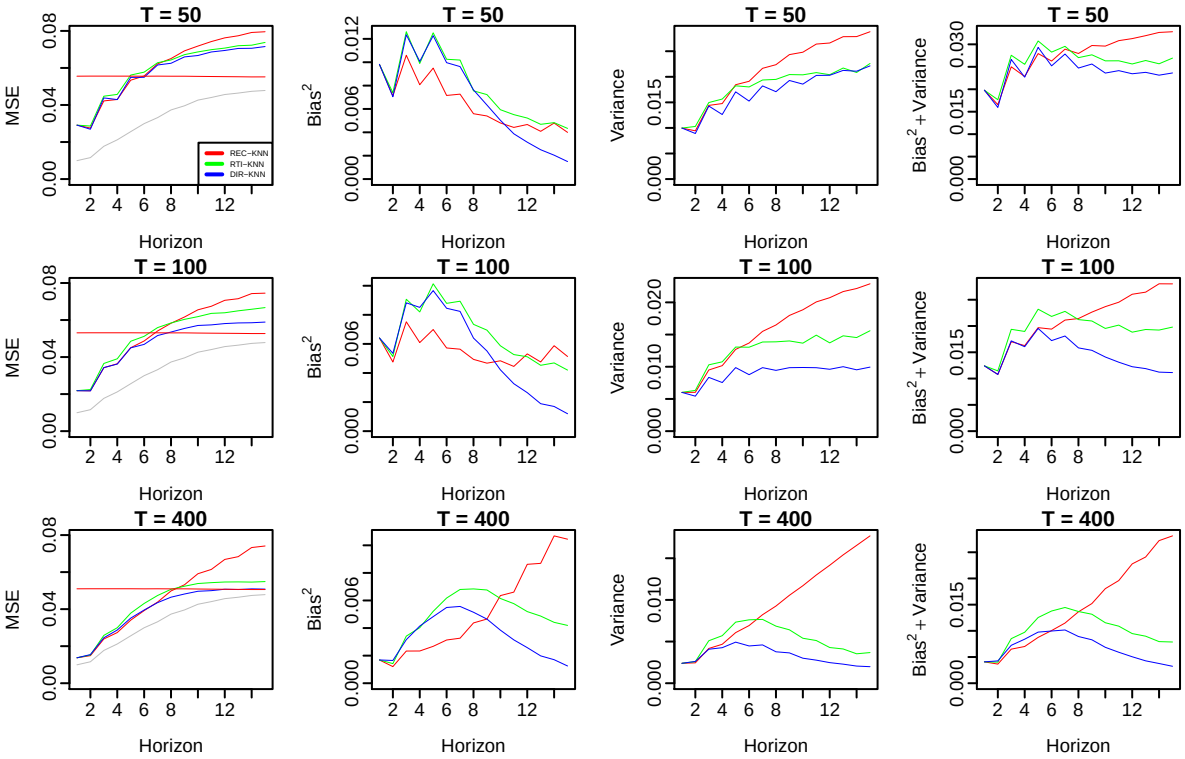

Figure 4.13: *STAR DGP. MSE decomposition of recursive (REC), multi-step recursive (RTI) and direct (DIR) forecasts with the KNN model.*

4.5 Concluding remarks

There have been limited in-depth studies comparing between recursive and direct forecasts generated with machine learning models. In particular, many studies have focused on the linear setting, where both the model and the DGP are linear. We conducted an in-depth study that compares recursive and direct forecasts generated with different learning algorithms for different DGPs. More precisely, we have decomposed the multi-step mean squared forecast errors into the bias and variance components, and have analyzed their behavior over the forecast horizon for different time series lengths.

Overall, the results have shown that the accuracy of recursive and direct forecasts depend on the different considered factors, including the learning algorithm, the DGP, the time series length and the forecast horizon. This emphasizes the difficulty of choosing between recursive and direct forecasts in real-world applications.

When both the model and the DGP are linear, we confirmed that model misspecification plays an important role in the relative performance between the recursive and direct strategy (Chevillon, 2007).

When both the model and the DGP are nonlinear, we observed a propagation of errors with recursive forecasts and smaller errors with direct forecasts, especially with long time series.

When the DGP is nonlinear and the nonlinear model can switch to a linear model or if the model is linear, recursive forecasts have lower errors than direct forecasts especially with short time series. Also, although they are biased, we found that linear forecasts have lower errors than nonlinear forecasts particularly with short time series. These results emphasize the fact that weakly nonlinear

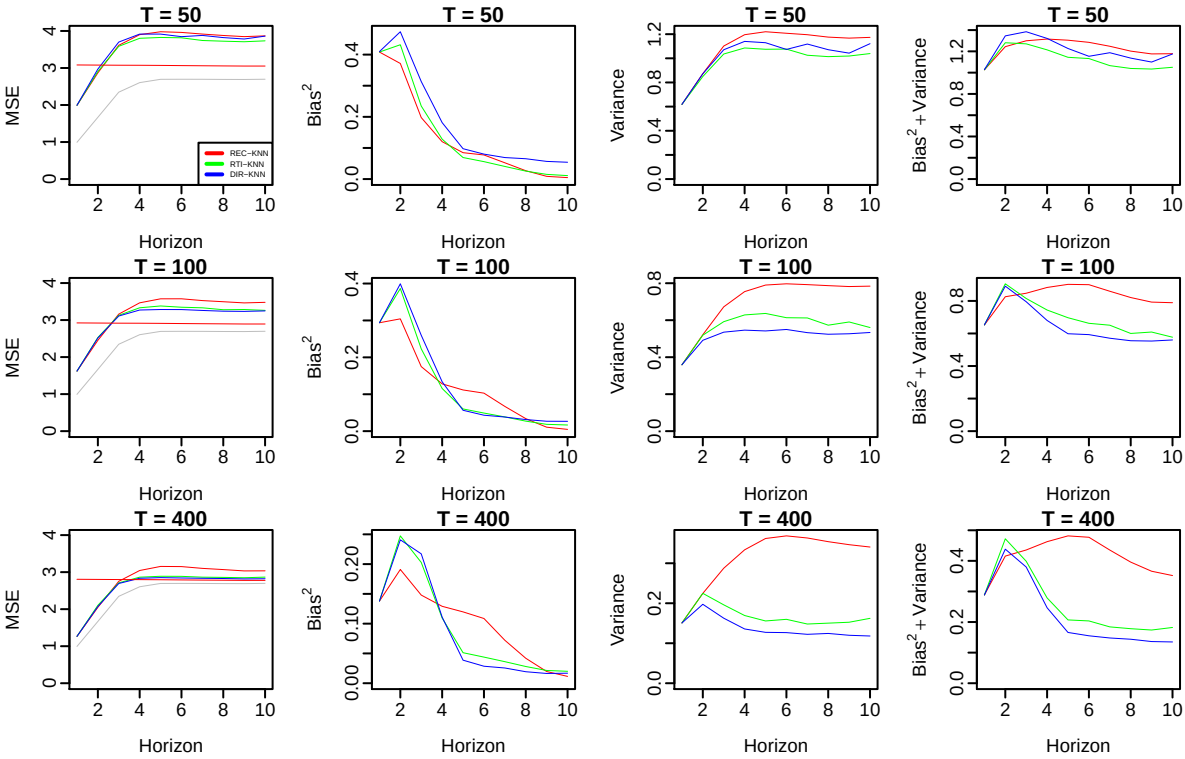

Figure 4.14: *NAR DGP. MSE decomposition of recursive (REC), multi-step recursive (RTI) and direct (DIR) forecasts with the KNN model.*

time series and/or short time series will favor linear forecasts over nonlinear forecasts, notably due to overfitting (Teräsvirta, 2006).

Finally, when the model is nonlinear and the DGP is linear, both recursive and direct forecasts will bring no benefits over linear forecasts since the DGP is linear. Recursive forecasts will suffer from a higher variance with long time series and long horizons while direct forecasts will have a higher variance for short time series and long horizons. If the model can switch to a linear model, we found that recursive forecasts have lower errors than direct forecasts.

In the next chapter, we develop a new forecasting strategy by using the fact that recursive linear forecasts often provide the best forecasts with short time series even if the DGP is nonlinear, and choosing between recursive and direct forecasts is not an easy task in real-world applications.

In Chapter 6.3, we also develop a new forecasting strategy that aims at reducing the large variance of direct forecasts generated with machine learning models, especially for short time series and long horizons.

For the next two chapters, we also use the following observations: (i) real-world time series are often short and weakly nonlinear and (ii) overfitting is often the main weakness of forecasts generated with machine learning models.

Chapter 5

Multi-stage forecasting strategies

This chapter is heavily based on the following publications

- S Ben Taieb and RJ Hyndman (2014a). Boosting multi-step autoregressive forecasts. In: *Proceedings of the 31th International Conference on Machine Learning (ICML)*, pp.109–117
- S Ben Taieb and RJ Hyndman (2014b). "Recursive and direct multi-step forecasting: the best of both worlds." Submitted to International Journal of Forecasting (under revision)

5.1 Introduction

In both Sections 3.4 and 4.4, we have seen that the accuracy of recursive and direct forecasts depends on many interacting factors such as the learning model, the underlying DGP, the time series length and the forecast horizon.

Although different studies have contributed to better understand under which conditions one strategy is better than the other, choosing between recursive and direct forecasts still remains a challenging problem in real-world applications since it involves finding the best trade-off between bias and estimation variance of the forecasts over the horizon.

In this chapter we aim at developing a new forecasting strategy that avoids choosing between recursive and direct forecasts, and at the same time has better, or at least close performance to the best of the recursive and direct forecasts. In particular, rather than treating the recursive and direct strategies as competitors, we seek to combine the best properties of both strategies.

Some hybrid strategies that combine the architectures of both recursive and direct strategies have been proposed in the literature (Zhang and Hutchinson, 1994; Sorjamaa and Lendasse, 2006; Zhang, Zhou, et al., 2013). However, these strategies have received little attention notably due to the additional complexity or the limited increase in performance compared to the traditional recursive or direct forecasts.

In the bias and variance analysis of Section 4.4, we have found that recursive linear forecasts often provide the best forecasts with short time series even if the DGP is nonlinear, notably due to their reduced variance over the forecast horizon. With longer time series, we have seen that recursive linear forecasts can suffer from a higher bias component since they cannot capture the nonlinearity, but still provide a good first approximation. In consequence, recursive linear forecasts can be used as a starting point and only need some adjustment to capture the potential nonlinearity.

From these observations, we develop multi-stage forecasting strategies, that first generate recursive linear forecasts, and then adjust these forecasts by modeling the multi-step forecast residuals with direct nonlinear models at each horizon, called rectification models.

We considered a first strategy, called the *rectify* strategy, that estimates the rectification models with the nearest neighbors model, but any nonlinear learning model can be used. Related methods have been considered in Judd and Small (2000) where "correctors" have been applied to a short-term predictor to improve long-term prediction of nonlinear models. However, the multi-stage strategies differ in many ways, notably in terms of the models and estimation method considered, as well as the evaluation procedure.

Because real-world time series often possess slightly or moderately nonlinear behavior, we often only need few adjustments of the recursive linear forecasts at each horizon. In particular, by considering complex nonlinear machine learning models for the rectification models, there is a chance of overfitting, especially with short time series at long horizons. Therefore, we considered a second strategy, called the *boost* strategy, that estimates the rectification models using gradient boosting algorithms that involve the so-called weak learners.

The boost strategy is particularly useful since it provides a procedure for applying boosting algorithms for multi-step forecasting problems. In fact, although boosting algorithms have proven to be very powerful prediction algorithms, they have received little attention in the forecasting community, except in few recent publications (Assaad, Boné, and Cardot, 2008; Audrino and Bühlmann, 2009; Shafik and Tutz, 2009; Bai and Ng, 2009; Buchen and Wohlrabe, 2011; Robinzonov, Tutz, and Hothorn, 2012). Furthermore, when boosting algorithms have been applied to forecasting, the direct strategy has generally been adopted. One advantage of the boost strategy is that it links all the direct models together with the same unifying base model, thus possibly reducing the forecast variance.

The multi-stage forecasting strategies have many advantages: (i) they avoid the difficult choice between recursive and direct forecasts; and (ii) they balance flexibility and robustness since the linear recursive forecasts serve as a first robust approximation and flexibility is added with the nonlinear rectification models at each forecast horizon.

In the sections, we evaluate and compare the rectify and the boost strategies with both the recursive and direct strategies. In Section 5.3, we first apply a bias and variance analysis of the multi-step mean squared forecast errors of the different strategies, As in Section 4.4. Then, in Section 5.4, we compare the different strategies with real-world time series from the M3 and NN5 competitions.

5.2 Multi-stage forecasting strategies

Rather than treating the recursive and direct strategies as competitors, we take a different approach and seek to combine the best properties of both the recursive and direct strategies. We propose a new strategy that first produce recursive linear forecasts, then adjust these forecasts by modelling the multi-step forecast errors using a direct strategy with a nonlinear model at each horizon. We call this new strategy "rectify" because it begins with recursive forecasts and adjusts (or rectifies) them so they have smaller error.

In other words, we first model the time series using an autoregressive linear model (also called base model) given by

$$y_t = \underbrace{\phi_0 + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p}}_{m(\boldsymbol{z}_{t-1};\boldsymbol{\phi})} + e_t. \tag{5.2.1}$$

where $\boldsymbol{\phi} = [\phi_0, \ldots, \phi_p]$ are the parameters of the linear model, that can be for example estimated by OLS (see Section 2.2.1). For the selection of the lag order p, a time series cross-validation approach can be adopted, as explained in Section 2.3.5. Then forecasts are produced recursively from the estimated model: $m^{(h)}(\boldsymbol{z}_t; \hat{\boldsymbol{\phi}})$.

At this stage, the forecasts are equivalent to those from the traditional linear AR model and can be considered as a first approximation of the conditional expectation $\mu_{t+h|t}$. Then, we add an additional stage to adjust for data features which cannot or are not represented by the linear autoregressive fit. More precisely, we adjust the recursive base forecasts by applying direct nonlinear models to the (in-sample) residuals from the linear recursive forecasts; that is, we fit the models m_h given by

$$\underbrace{y_t - m^{(h)}(\boldsymbol{z}_{t-h}; \hat{\boldsymbol{\phi}})}_{\tilde{y}_t} = m_h(\boldsymbol{r}_{t-h}; \boldsymbol{\gamma}_h) + e_{t,h} \tag{5.2.2}$$

where $\tilde{y}_t$ is the new response variable of the regression, $\boldsymbol{z}_{t-h} = [y_{t-h}, \ldots, y_{t-h-p}]'$, $\boldsymbol{r}_{t-h} = [y_{t-h}, \ldots, y_{t-h-p_h}]'$, $\boldsymbol{\gamma}_h$ are the parameters of the (direct) rectification models and $h = 1, \ldots, H$.

In contrast to the parameters of the direct models in (3.3.5), the parameters of the rectification models $\boldsymbol{\gamma}_h$ in (5.2.2) depend on the values of the parameters $\hat{\boldsymbol{\phi}}$. They can be estimated as in (3.3.6), but with the new response variable $\tilde{y}_t$. Then, the final forecasts are obtained for each horizon by adding the rectifications to the forecasts from the base model: $\hat{\mu}_{T+h|T} = m^{(h)}(\boldsymbol{z}_T; \hat{\boldsymbol{\phi}}) + m_h(\boldsymbol{r}_T; \hat{\boldsymbol{\gamma}}_h)$. Algorithm 2 summarizes the different steps of the rectify strategy.

Algorithm 2 multi-step-ahead forecasting with the rectify strategy.

$\{y_1, \ldots, y_T\}$: Time series with T observations.
H: Forecast horizon.

1: Fit the AR(p) model given in (5.2.1) to obtain $m(\cdot; \hat{\boldsymbol{\phi}})$.
2: **for** $h \leftarrow 1, \ldots, H$ **do**
3: Compute the h-step ahead recursive forecasts from the AR(p) model $m^{(h)}(\boldsymbol{z}_{t-h}; \hat{\boldsymbol{\phi}})$ to obtain a first (possibly crude) estimate of $\mu_{t+h|t}$.
4: Compute the new response variable $\tilde{y}_t$ as defined in (5.2.2).
5: Learn the (direct) rectification model $m_h(\boldsymbol{r}_{t-h}; \hat{\boldsymbol{\gamma}}_h)$ as defined in (5.2.2) from the dataset $\mathcal{D} = \{(\boldsymbol{r}_{t-h}, \tilde{y}_t)\}_{t=1}^{T}$.
6: **end for**
7: The final forecasts are given by $\hat{\mu}_{T+h|T} = m^{(h)}(\boldsymbol{z}_T; \hat{\boldsymbol{\phi}}) + m_h(\boldsymbol{r}_T; \hat{\boldsymbol{\gamma}}_h)$ for $h = 1, \ldots, H$.

It is worth noting that the rectify strategy is different from a simple equally weighted forecast combination of the recursive and the direct strategy where forecasts are generated as follows:

$$\hat{\mu}_{T+h|T} = \frac{m^{(h)}(\boldsymbol{z}_T; \hat{\boldsymbol{\phi}})}{2} + \frac{m_h(\boldsymbol{r}_T; \hat{\boldsymbol{\gamma}}_h)}{2} \tag{5.2.3}$$

where $\hat{\boldsymbol{\phi}}$ and $\hat{\boldsymbol{\gamma}}_h$ are the parameters of the models in (3.3.1) and (3.3.5), respectively. In fact, with the rectify strategy, the parameter values of the (direct) rectification models $\hat{\boldsymbol{\gamma}}_h$ depend on the parameters of the recursive base model $\hat{\boldsymbol{\phi}}$ and hence, the (direct) rectification models are *complementary* to the (recursive) base model. However, with the equally weighted combination in expression (5.2.3), the forecasts are obtained by a simple average of recursive and direct forecasts, independently generated.

Compared to the recursive strategy, the rectify strategy does not suffer from the amplification of errors since recursive forecasts are generated with a linear model. Another advantage of the linear model is that it allows the underlying process to be estimated in areas where the data is sparse.

An important property of the rectify forecasts is that they are equivalent to the forecasts of the traditional AR model when there are no rectifications, that is when $m_h(\boldsymbol{r}_T; \hat{\gamma}_h) = 0$. The same phenomenon happens with the MLP model when it has zero hidden nodes (see Section 2.2.2).

Compared to the direct strategy, the rectification models are linked together as they all operate on the forecast errors of the same linear model. This will reduce the irregularities that can arise with independent models, and in turn reduce the forecast variance.

Of course, it is still possible for the rectification models to be different from each other, but these differences are likely to be much smaller when modelling the residuals from the recursive forecasts than using a pure direct strategy.

Also, because the linear model will allow part of the signal to be modelled, simpler rectification models will be needed since the residuals will be less dependent on the conditioning variables and the function to be estimated will be smoother. Therefore, the rectify strategy is attractive since it avoids having too complex direct models as with the pure direct strategy.

Finally, in the extreme case where the recursive forecasts are null, that is $m^{(h)}(\boldsymbol{z}_T; \hat{\boldsymbol{\phi}}) = 0$, we can see in (5.2.2) that the rectify strategy is equivalent to the direct strategy.

When the direct rectification models in expression (5.2.2) are nonlinear machine learning models, they can potentially include high-order interaction terms between the lagged variables in $\boldsymbol{r}_{t-h}$. Having complex nonlinear rectification models can induce a large variance in the final forecasts of the rectify strategy. Of course, the model selection procedure can select a small value of the lag orders p_h which will implicitly induce a lower-order interaction and a smaller variance. Nevertheless, it could still happen that a bad model is selected with noisy data.

Instead of applying one potentially high-variance rectification, we propose a second strategy that apply several *small-variance* rectifications at each horizon where the number of rectifications needs to be selected. In particular, we propose to *boost* the base forecasts by applying a gradient boosting procedure (see Section 2.2.4) on the residuals from the linear recursive model. We call it the *boost* strategy.

The *small-variance* adjustments are ensured by the so-called weak learner; that is a learner with large bias relative to variance (see Section 2.2.4). Also, since we expect real-world time series to depend on lower-order interactions, we only allow bivariate interactions between the input variables for each model.

So, the rectifications models m_h in (5.2.2) can be written as a sum of *small-variance* adjustments or boosting components as follows:

$$\underbrace{y_t - m^{(h)}(\boldsymbol{z}_{t-h}; \hat{\boldsymbol{\phi}})}_{\tilde{y}_t} = \sum_{j=1}^{J_h} \nu \cdot l^{[j]}(y_{t-a}, y_{t-c}; \psi) + e_{t,h}, \tag{5.2.4}$$

where J_h is the number of boosting iterations at horizon h, $\nu \cdot l^{[j]}$ is the weak adjustment at iteration j with $0 < \nu \leq 0$ being a shrinkage factor, and $y_{t-a}, y_{t-c} \in \boldsymbol{r}_{t-h}$.

One should note that the weak adjustments in (5.2.4) are ensured to be *weak* because of three reasons: (i) the shrinkage factor ν shrinks the estimate towards zero, (ii) the restriction to bivariate interactions between variables, and (iii) the weak learner $l(\cdot; \psi)$.

Algorithm 3 gives the different steps of the boost strategy to generate H-step forecasts for a given time series $\{y_1, \ldots, y_T\}$ with T observations. We assume the number of boosting iterations at each horizon $[J_1, \ldots, J_H]$ has already been selected and is given.

Algorithm 3 Multi-step forecasting with the boost strategy

$\{y_1,\ldots,y_T\}$: Time series with T observations.
H: Forecasting horizon.
$\{l_1(y_{t-1},y_{t-2}),\ldots,l_B(y_{t-d+1},y_{t-d})\}$: set of $B=\binom{d}{2}$ base-learners where d is the maximum lag order allowed.
$\{J_1,\ldots,J_H\}$: number of boosting iterations for each horizon.
$0<\nu\leq 1$: shrinkage factor.

Fit the AR(p) model given in (5.2.1) to obtain $m(\cdot;\hat{\boldsymbol{\phi}})$.
for $h\leftarrow 1,\ldots,H$ **do**
    Compute the h-step ahead recursive forecasts from the AR(p) model $m^{(h)}(\boldsymbol{z}_{t-h};\hat{\boldsymbol{\phi}})$ to obtain a first (possible crude) estimate of $\hat{\mu}_{t+h|t}$.
    Denote by $\hat{F}_h^{[j]}$ the estimate of the conditional mean at horizon h for iteration j, and let $F_h^{[0]}=m^{(h)}(\boldsymbol{z}_{t-h};\hat{\boldsymbol{\phi}})$ be the estimate for iteration 0.
    for $j\leftarrow 1,\ldots,J_h$ **do**
        Compute the negative gradient of the quadratic loss function evaluated at the function values of the previous iteration $F_h^{[j-1]}$:

$$u_t^{[j]}=-\frac{\frac{1}{2}\partial(y_t-F_h)^2}{\partial F_h}\bigg|_{F_h=F_h^{[j-1]}}=-(y_t-F_h^{[j-1]}),\quad t=1,\ldots,T.$$

        for $b\leftarrow 1,\ldots,B$ **do**
            Fit the bth base learner with the training set $\{(\boldsymbol{r}_{t-h},u_t^{[j]})\}_{t=1}^T$, which gives $l_b^{[j]}(y_{t-a},y_{t-c};\hat{\boldsymbol{\gamma}})$, that will be also denoted as $\hat{l}_b^{[j]}(y_{t-a},y_{t-c})$.
        end for
        Select the base-learner $\hat{l}_{b^*}^{[j]}(y_{t-a},y_{t-c})$ with the largest contribution to the fit, i.e. the base-learner that minimizes the sum of squared errors:

$$b^*=\underset{b}{\operatorname{argmin}}\sum_{t=1}^T(u_t^{[j]}-\hat{l}_b^{[j]}(y_{t-a},y_{t-c}))^2$$

        Update the current function estimate by adding the best base-learner estimate of the current iteration (j) to the function estimate of the previous iteration ($j-1$):

$$\hat{F}_h^{[j]}=\hat{F}_h^{[j-1]}+\nu\cdot\hat{l}_{b^*}^{[j]}(y_{t-a},y_{t-c})$$

    end for

end for
The final forecasts are given by $[\hat{\mu}_{T+1|T},\ldots,\hat{\mu}_{T+H|T}]$ where $\hat{\mu}_{T+h|T}=\hat{F}_h^{[J_h]}=m^{(h)}(\boldsymbol{z}_T;\hat{\boldsymbol{\phi}})+\sum_{j=1}^{J_h}\nu\cdot l^{[j]}(y_{T-a},y_{T-c};\hat{\boldsymbol{\gamma}})$

Algorithm 3 is an extension of the component-wise boosting algorithm (see Algorithm 1 in page 24) to the problem of multi-step forecasting. In particular, we can see that the boosting procedure is applied on the residuals from the AR(p) model (see line 3) while in Algorithm 1 (page 24), the mean is used as first approximation (see line 1).

The final forecasts are composed of two parts: the AR(p) model and a set of weak rectifications (line 14). We can see that we can have a different number of boosting iterations at each horizon. In particular, if $J_h = 0$ then the forecasts at horizon h are not adjusted and are reduced to the forecasts of the traditional AR(p) forecasts.

The boost strategy depends on different parameters: (i) the number of boosting iterations, (ii) the weak learner and (iii) the shrinkage factor. The number of boosting iterations J is the main hyperparameter that needs to be selected and is particularly important as it controls the trade-off between the bias and variance of the estimation (see Section 2.2.4). Cross-validation methods can be used to select the best number of iterations. The weak learner estimates are generally obtained with a least square type method. The shrinkage factor is generally fixed to a small value, such as $\nu = 0.1$.

In our implementation of the boost strategy, we have considered P-splines as in Schmid and Hothorn (2008). P-splines require the selection of two additional parameters: the number of knots and the smoothing parameter or the associated degree of freedom (see Section 2.2.1). However, Ruppert (2002) has shown that the number of knots does not have much effect on the estimation provided enough knots are used. The weakness of the P-spline is measured by its degrees of freedom (df). Bühlmann and Yu (2003) and Schmid and Hothorn (2008) proposed that the smoothing parameter be set to give a small value of df (i.e., $\text{df} \in [3, 4]$), and that this number be kept fixed in each boosting iteration.

5.3 Bias and variance analysis

As in Section 4.4, we will use the terminology described in Section 4.3 to compare the rectify and boost strategies with both the recursive and direct strategies from the perspective of the bias and variance components of the multi-step mean squared forecast errors. In addition, we will also compare the rectify and boost strategies with averaging strategies, i.e. strategies that compute a simple average of recursive and direct forecasts, as in expression (5.2.3).

As explained in Section 4.3, we will consider four different scenarios depending on whether the DGP and the learning model of the recursive and direct strategies are linear or nonlinear. Of course, for the rectify and the boost strategies, the base model is always linear and the rectification models are always nonlinear. For each scenario, we will perform a theoretical bias and variance analysis for the case of two-step ahead forecasts. The general case of h-step ahead forecasts will be considered using Monte Carlo simulations.

Let us first write the two-step ahead forecasts of the rectify and boost strategies using the terminology defined in Section 4.3.1. To simplify the notations, we will remove the dependence on the size of the time series T.

Forecasts of the rectify strategy. In the first stage, a linear autoregressive model is fitted as in (4.4.3) and two-step ahead forecasts are computed recursively as in (4.4.4). In the second stage, the residuals of the linear model for two-step ahead forecasts are modeled using a direct rectification model. The final forecasts are composed of the two-step ahead forecasts from the linear model and the adjustments from the rectification model as follows.

$$
\begin{aligned}
&g(\boldsymbol{x}_t;[\hat{\boldsymbol{\phi}};\hat{\boldsymbol{\gamma}}];2) \qquad (5.3.1)\\
&= (\phi_0 + \phi_1\phi_0) + (\phi_1^2 + \phi_2)y_t + \cdots + (\phi_1\phi_{p-1} + \phi_p)y_{t-p+2} + (\phi_1\phi_p)y_{t-p+1} + (\phi_1 + 1)\eta(\boldsymbol{\phi})\varepsilon_{\eta_0}\\
&\quad + m_2(\boldsymbol{r}_t;\boldsymbol{\gamma}) + \eta(\boldsymbol{r}_t;\boldsymbol{\gamma})\varepsilon_{\eta_1}
\end{aligned}
$$

where where ε_{η_0} and ε_{η_1} are the stochastic components of the variability term for the linear base model and the rectification model respectively.

Forecasts of the boost strategy. Similarly to the rectify strategy, the final forecasts are composed of the recursive and direct components, but the direct adjustments are computed with a boosting procedure, as follows

$$
\begin{aligned}
&g(\boldsymbol{x}_t;[\hat{\boldsymbol{\phi}};\hat{\boldsymbol{\gamma}}];2) \qquad (5.3.2)\\
&= (\phi_0 + \phi_1\phi_0) + (\phi_1^2 + \phi_2)y_t + \cdots + (\phi_1\phi_{p-1} + \phi_p)y_{t-p+2} + (\phi_1\phi_p)y_{t-p+1} + (\phi_1 + 1)\eta(\boldsymbol{\phi})\varepsilon_{\eta_0}\\
&\quad + \sum_{j=1}^{J_2} \nu \cdot [\underbrace{l^{[j]}(y_{t-a}, y_{t-c};\boldsymbol{\gamma}_j) + \eta_j(y_{t-a}, y_{t-c};\boldsymbol{\gamma}_j) * \varepsilon_{\eta_j}}_{l^{[j]}(y_{t-a}, y_{t-c};\hat{\boldsymbol{\gamma}}_j)}] \qquad (5.3.3)
\end{aligned}
$$

where $\boldsymbol{\phi} = [\phi_1,\ldots,\phi_p]$, $\boldsymbol{\gamma} = [\boldsymbol{\gamma}_1,\ldots,\boldsymbol{\gamma}_J]$ and $a, c \in \{1,\ldots,p'\}$ where p' is the maximum lag order allowed for the rectification models.

Expression (5.3.3) can be also written in a different form where instead of summing over the iterations, it is written as a sum over the different bivariate interactions. In other words, we have

$$
\sum_{j=1}^{J_2} \nu \cdot [l^{[j]}(y_{t-a}, y_{t-c};\boldsymbol{\gamma}_j) + \eta_j(y_{t-a}, y_{t-c};\boldsymbol{\gamma}_j) * \varepsilon_{\eta_j}]
$$

$$
\begin{aligned}
&= \sum_{(u,v)} \sum_{S_{uv}=\{j|a=u\wedge c=v\}} \nu[I^{[j]}(y_{t-a}, y_{t-c}; \boldsymbol{\gamma}_j) + \eta_j(y_{t-a}, y_{t-c}; \boldsymbol{\gamma}_j) * \varepsilon_{\eta_j}] \\
&= \sum_{(u,v)} \underbrace{M_{uv}(y_{t-u}, y_{t-v}; \boldsymbol{\gamma}_{uv}) + \eta_{uv}(y_{t-u}, y_{t-v}; \boldsymbol{\gamma}_{uv}) * \varepsilon_{\eta_{uv}}}_{M_{uv}(y_{t-u}, y_{t-v}; \hat{\boldsymbol{\gamma}}_{uv})}
\end{aligned} \tag{5.3.4}
$$

where $(u,v) \in \{1,\ldots,p\} \times \{1,\ldots,p\}$ and $\hat{\boldsymbol{\gamma}}_{uv} = \{\hat{\boldsymbol{\gamma}}_j | j \in S_{uv}\}$.

For the expression of the variance component, we will use the representation given in expression (5.3.3) while for the bias component, we will use the representation given in (5.3.4).

In the Monte Carlo simulations, the rectify strategy with the KNN model will be denoted as RFY-KNN, while the boost strategy will be denoted as RFY-BST2. The averaging strategies that compute a simple average of REC-M1 and DIR-M2 will be denoted AVG-M1-M2, where where M1 and M2 can be any learning model such as LIN, MLP, KNN or BST2.

Finally, when comparing the rectify and the boost strategies with the averaging strategy, we will also decompose the total variance into the variances of each model plus the covariances as described in Section 4.2.1.

5.3.1 Scenario A: Linear model and linear DGP

We compare the rectify and boost strategies with the recursive and direct linear forecasts when the DGP is linear. The expression (4.3.1) for the recursive and direct strategies are given by expressions (4.4.7) and (4.4.11), respectively.

For the rectify strategy, we compute expression (4.3.1) using expression (5.3.1) as follows

$$
\begin{aligned}
&B_2^{\mathrm{RFY}}(\boldsymbol{x}_t) + V_2^{\mathrm{RFY}}(\boldsymbol{x}_t) \\
&= \Big[\Big((\varphi_0 + \varphi_1\varphi_0) + (\varphi_1^2 + \varphi_2)y_t + \cdots + (\varphi_1\varphi_{d-1} + \varphi_d)y_{t-d+2} + (\varphi_1\varphi_d)y_{t-d+1}\Big) \\
&\quad - \Big((\phi_0 + \phi_1\phi_0) + (\phi_1^2 + \phi_2)y_t + \cdots + (\phi_1\phi_{p-1} + \phi_p)y_{t-p+2} + (\phi_1\phi_p)y_{t-p+1}\Big) \\
&\quad - m_2(\boldsymbol{r}_t; \boldsymbol{\gamma})\Big]^2 \\
&\quad + (\phi_1 + 1)^2\eta(\boldsymbol{\phi})^2 + \eta(\boldsymbol{r}_t; \boldsymbol{\gamma})^2
\end{aligned} \tag{5.3.5}
$$

where we assume that $\varepsilon_{\eta_0} \perp \varepsilon_{\eta_1}$, that is the variability terms of the linear base model and the nonlinear rectification model are uncorrelated.

For the boost strategy, we compute expression (4.3.1) using expressions (5.3.2) and (5.3.4) as follows

$$
\begin{aligned}
&B_2^{\mathrm{RFY}}(\boldsymbol{x}_t) + V_2^{\mathrm{RFY}}(\boldsymbol{x}_t) \\
&= \Big[\Big((\varphi_0 + \varphi_1\varphi_0) + (\varphi_1^2 + \varphi_2)y_t + \cdots + (\varphi_1\varphi_{d-1} + \varphi_d)y_{t-d+2} + (\varphi_1\varphi_d)y_{t-d+1}\Big) \\
&\quad - \Big((\phi_0 + \phi_1\phi_0) + (\phi_1^2 + \phi_2)y_t + \cdots + (\phi_1\phi_{p-1} + \phi_p)y_{t-p+2} + (\phi_1\phi_p)y_{t-p+1}\Big) \\
&\quad - \sum_{uv} M_{uv}(y_{t-u}, y_{t-v}; \boldsymbol{\gamma}_{uv})\Big]^2 \\
&\quad + (\phi_1 + 1)^2\eta(\boldsymbol{\phi})^2 + \underbrace{\sum_{j=1}^{J} \frac{\eta_j(y_{t-u}, y_{t-v}; \boldsymbol{\gamma}_j)^2}{\tau^2}}_{\sum_{uv}\eta_{uv}(y_{t-u}, y_{t-v}; \boldsymbol{\gamma}_{uv})^2}
\end{aligned} \tag{5.3.6}
$$

where $\nu = \frac{1}{\tau}$ and we assumed $\varepsilon_{\eta_0} \perp \varepsilon_{\eta_j}$ for all j, that is the variability terms of the linear model and the boosting components are uncorrelated, and $\varepsilon_{\eta_j} \perp \varepsilon_{\eta_{j'}}$ for $j \neq j'$, that is the variability terms of the different boosting components are also uncorrelated.

Let us consider the case of a well-specified linear model (i.e. $p = d$) for the recursive and direct strategies as well as for the base model of the rectify and boost strategies. In that case, we have seen in Section 4.4.1 that the recursive strategy benefits from more efficient parameter estimates than the direct strategy.

For the rectify and the boost strategies, the direct rectification models will not bring any benefit to the recursive base forecasts from a well-specified model since the residuals from this model will essentially contain noise. In consequence, both the bias and variance components will be affected by the additional rectification terms as can be seen in (5.3.5) and (5.3.6). More precisely, the linear terms will disappear and the nonlinear terms will remain in both the bias and variance components.

We would like the rectification terms to be small and ideally to be equal to zero. This will make the rectify and boost forecasts equivalent to the recursive linear forecasts, that generate better forecasts in that setting.

However, in practice, the rectification terms will rarely be exactly equal to zero especially when they are estimated from a short and noisy time series. In consequence, we expect the rectify and boost forecasts to have a higher error than the recursive linear forecasts due to these additional terms.

The increase in error will depend on the the complexity of the rectification models over the forecast horizon. In particular, if the rectification models of the rectify strategy include many lagged variables to interact in a nonlinear way, we expect the error increase to be high. The boost strategy can be a better alternative since it limits the complexity of the rectification models by only allowing bivariate interactions between the lagged variables and by benefiting from the reduced variance of the weak components.

Direct forecasts are expected to have a higher variance than recursive forecasts in this scenario, particularly for long horizons. However, because the rectification models of the rectify and the boost strategies are allowed to be nonlinear, they can have a higher variance than direct linear forecasts. Of course, the rectify and boost forecasts may generate better forecasts than direct linear forecasts if they have no rectification terms, or equivalently, if the forecasts becomes identical to the recursive linear forecasts.

Recall however that a well-specified model is not a realistic assumption since a model is an approximation of the reality and thus is almost always misspecified.

Let us consider the case where the recursive and direct forecasts are generated with a misspecified linear model, and the linear base model of the rectify and boost strategies is also misspecified. As in Section 4.4.1, we will assume the misspecification is due to omitted lagged variables.

We have seen in Section 4.4.1 that with a misspecified linear model, recursive forecasts are biased and direct forecasts are more robust to misspecification since the models are selected by minimizing h-step ahead forecasts directly.

Because recursive forecasts from a misspecified base model are biased, the residuals from the base model will contain the unmodeled signal. The rectification models of the rectify and boost strategies will be able to adjust these forecasts and reduce the bias, provided their inputs include the omitted variables.

However, since the DGP is linear, the signal contained in the residuals is expected to be small. Fitting these residuals with nonlinear direct models can significantly increase the total variance especially with short time series. Again, the boost strategy can be a better alternative in this case due to the possibly reduced variance of the rectification models.

Direct linear forecasts are known to be more robust to misspecification than recursive forecasts. In addition, because of the reduced variance of the linear models, direct linear forecasts are expected to provide smaller errors than the rectify and the boost strategies which can suffer from the higher variance of the nonlinear rectification models.

Let us now consider Monte Carlo simulations for the general case of h-step ahead forecasts. As in Section 4.4.1, we will consider both a well-specified (LIN) and a misspecified (LINMIS) linear model. In particular, RFY-KNN and RFY-BST2 will refer to the strategies that involve the LIN model while RFYMIS-KNN and RFYMIS-BST2 will refer to the strategies that involve the LINMIS model. Also, for both RFY and RFYMIS strategies, the nonlinear rectification models will include the true lag order in the set of possible lags. The results are given in Figure 5.1 where we have also included the results for the recursive and direct strategies, from Figure 4.3.

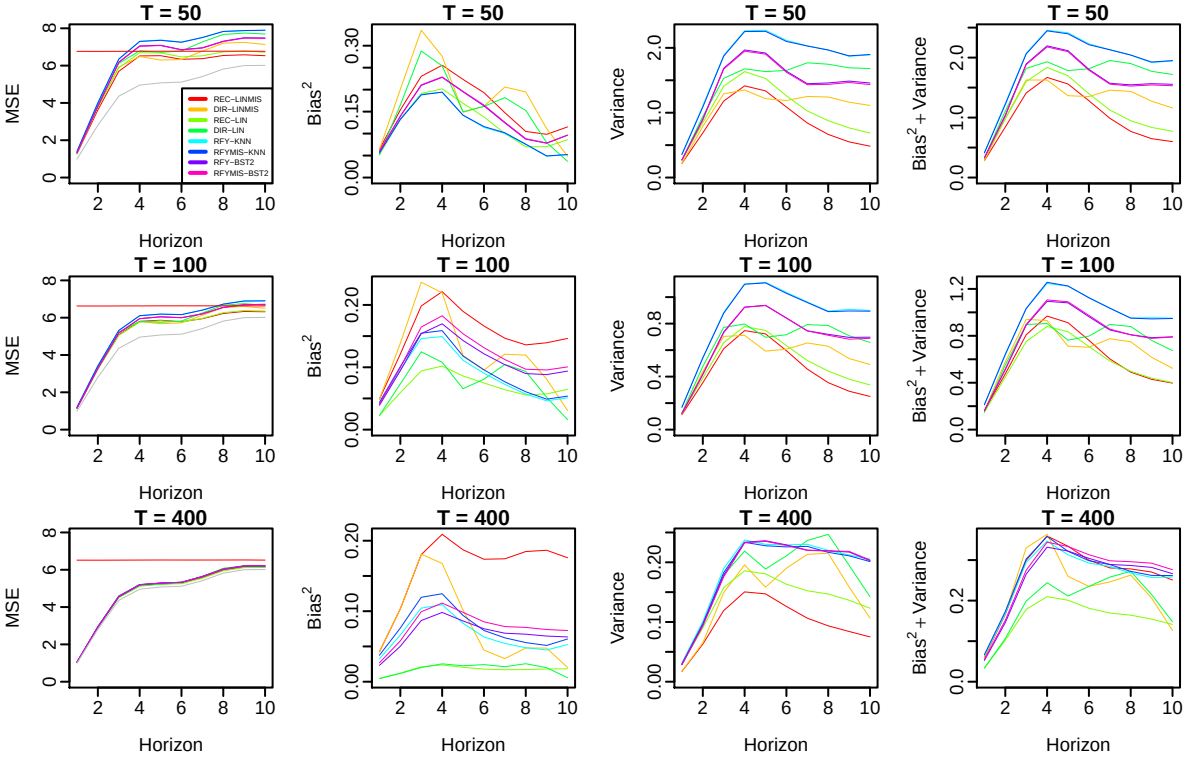

Figure 5.1: *AR DGP. MSE decomposition of rectify and boost forecasts with a well-specified linear base model (RFY-KNN and RFY-BST2, respectively), and a misspecified linear base model (RFYMIS-KNN and RFYMIS-BST2, respectively). The results of Figure 4.3 are also included.*

Let us first consider a well-specified model and compare REC-LIN and DIR-LIN with RFY-KNN and RFY-BST2. In the last column of Figure 5.1, we can see that the RFY strategies have higher errors for all values of T. As previously explained, this is particularly due to the higher variance of the nonlinear rectification terms when the DGP is linear, as can be seen in the third column.

If we compare RFY-KNN with RFY-BST2, we can see that RFY-BST2 has smaller errors than RFY-KNN for $T = 50$ and $T = 100$. This can be explained by the lower variance of the rectification terms for RFY-BST2. For $T = 400$, RFY-KNN and RFY-BST2 have a comparable performance.

If we consider a misspecified linear model, we can see in the last column that the difference between RFYMIS and RFY is small. No difference is visible in terms of the variance component. A small difference is visible in terms of the bias component for $T = 100$ and $T = 400$ that shows the advantage of the well-specified model over the misspecified model, but not for $T = 50$.

By comparing REC-LINMIS with RFYMIS strategies, we can see that although the linear model is misspecified, recursive forecasts have smaller errors than RFYMIS strategies. This makes sense since forecasts from a misspecified linear model are expected to have smaller errors than forecasts which include nonlinear terms with a higher variance. However, in the second column, we can see that the bias component of the RFYMIS strategies is between the bias component of REC-LINMIS and REC-LIN. This shows that the bias has effectively been reduced by the rectification models. However, the reduction in bias is smaller than the increase in variance.

We can make the same conclusion in terms of the variance component by comparing DIR-LINMIS with RFYMIS strategies. In terms of the bias component, the RFYMIS strategies have smaller errors for the first few horizons and mixed results for longer horizons. So, even if the models are misspecified, their linearity reduces significantly the variance and makes them more attractive than RFY strategies when the DGP is linear.

All in all, this scenario is not favorable for the rectify and the boost strategies since they are penalized by the additional complexity induced by the nonlinear rectification terms. Between the rectify and the boost strategies, the boost strategy is more attractive in this scenario notably for the reduced variance of the rectification terms.

Although in this scenario the RFY strategies have higher errors than REC-LIN, we will see that the RFY strategies are very attractive compared the REC and DIR strategies with nonlinear models, in other scenarios.

5.3.2 Scenario B: Linear model and nonlinear DGP

We compare the rectify and boost strategies with recursive and direct linear forecasts when the DGP is nonlinear. The expression (4.3.1) for the recursive and direct strategies are given by expressions (4.4.14) and (4.4.17), respectively.

For the rectify strategy, we compute expression (4.3.1) using expression (5.3.1) as follows

$$B_2^{\mathrm{RFY}}(\boldsymbol{x}_t) + V_2^{\mathrm{RFY}}(\boldsymbol{x}_t) \tag{5.3.7}$$

$$= \Bigg[\underbrace{f(f(\boldsymbol{x}_t), y_t, \ldots, y_{t-d+2}) + \frac{1}{2}\sigma^2 f_{x_1 x_1}}_{\mu_{t+2|t}} \tag{5.3.8}$$

$$- \Big((\phi_0 + \phi_1\phi_0) + (\phi_1^2 + \phi_2) y_t + \cdots + (\phi_1\phi_{p-1} + \phi_p) y_{t-p+2} + (\phi_1\phi_p) y_{t-p+1} \Big)$$

$$- m_2(\boldsymbol{r}_t; \boldsymbol{\gamma}) \Bigg]^2 \tag{5.3.9}$$

$$+ (\phi_1 + 1)^2 \eta(\boldsymbol{\phi})^2 + \eta(\boldsymbol{r}_t; \boldsymbol{\gamma})^2 \tag{5.3.10}$$

For the boost strategy, we compute expression (4.3.1) using expressions (5.3.2) and (5.3.4) as follows

$$B_2^{\mathrm{RFY}}(\boldsymbol{x}_t) + V_2^{\mathrm{RFY}}(\boldsymbol{x}_t) \tag{5.3.11}$$

$$= \Bigg[\underbrace{f(f(\boldsymbol{x}_t), y_t, \ldots, y_{t-d+2}) + \frac{1}{2}\sigma^2 f_{x_1 x_1}}_{\mu_{t+2|t}} \tag{5.3.12}$$

$$- \Big((\phi_0 + \phi_1\phi_0) + (\phi_1^2 + \phi_2) y_t + \cdots + (\phi_1\phi_{p-1} + \phi_p) y_{t-p+2} + (\phi_1\phi_p) y_{t-p+1} \Big)$$

$$- \sum_{uv} M_{uv}(y_{t-u}, y_{t-v}; \boldsymbol{\gamma}_{uv}) \Bigg]^2 \tag{5.3.13}$$

$$+ (\phi_1 + 1)^2 \eta(\boldsymbol{\phi})^2 + \sum_{j=1}^{J} \frac{\eta_j(y_{t-a}, y_{t-c}; \boldsymbol{\gamma}_j)^2}{\tau^2} \tag{5.3.14}$$

As explained in Section 4.4.2, because the DGP is nonlinear, both recursive and direct linear forecasts are biased since they cannot capture the nonlinearity $f_{x_1 x_1}$ of the function f. In particular, the bias will be high if $|f_{x_1 x_1}|$ is large; that is when f has large curvatures or if $p \neq d$.

However, because the rectify and the boost strategies include direct nonlinear terms, they can model nonlinear functions. In particular, if we compare expressions (5.3.8)-(5.3.9) and (5.3.12)-(5.3.13) with expressions (4.4.15)-(4.4.16), we can see that the role of the nonlinear rectification terms is to remove or decrease the bias of the recursive linear base forecasts.

The amount of rectification required will depend on the nonlinearity of the function f. If the DGP is weakly nonlinear, the recursive linear forecasts can already provide a good first approximation and few rectifications will be required for the non-captured (weak) nonlinearity. If the DGP is strongly nonlinear, then more rectifications can be required.

Because he rectify strategy do not impose any restriction on the rectification models, it can handle any nonlinear function provided the rectification models are flexible enough. However, the boost

strategy only allow bivariate interactions between lagged variables for the rectification models. As a result, if the function includes higher-order interactions, the boost strategy will provide biased forecasts, but the bias is expected to be smaller than the bias of linear models.

The rectify and boost strategies add a rectification term to the first (low-variance) approximation to reduce the bias of the recursive base forecasts. By doing so, an additional variance term is added in top of the variance of the base forecasts as can be seen in (5.3.10) and (5.3.14). In contrast, recursive and direct linear forecasts have only one variance term that will be typically low. Although the DGP is nonlinear, a lower variance can still induce a smaller MSE if the bias is not too high, especially with short time series. One advantage of the boost strategy is that it include nonlinear terms in the forecasts and, at the same time, limit overfitting with the reduced variance of the weak components and the limitation to bivariate interactions.

Let us now consider Monte Carlo simulations for the general case of h-step ahead forecasts. Figure 5.2 gives the results for the STAR DGP.

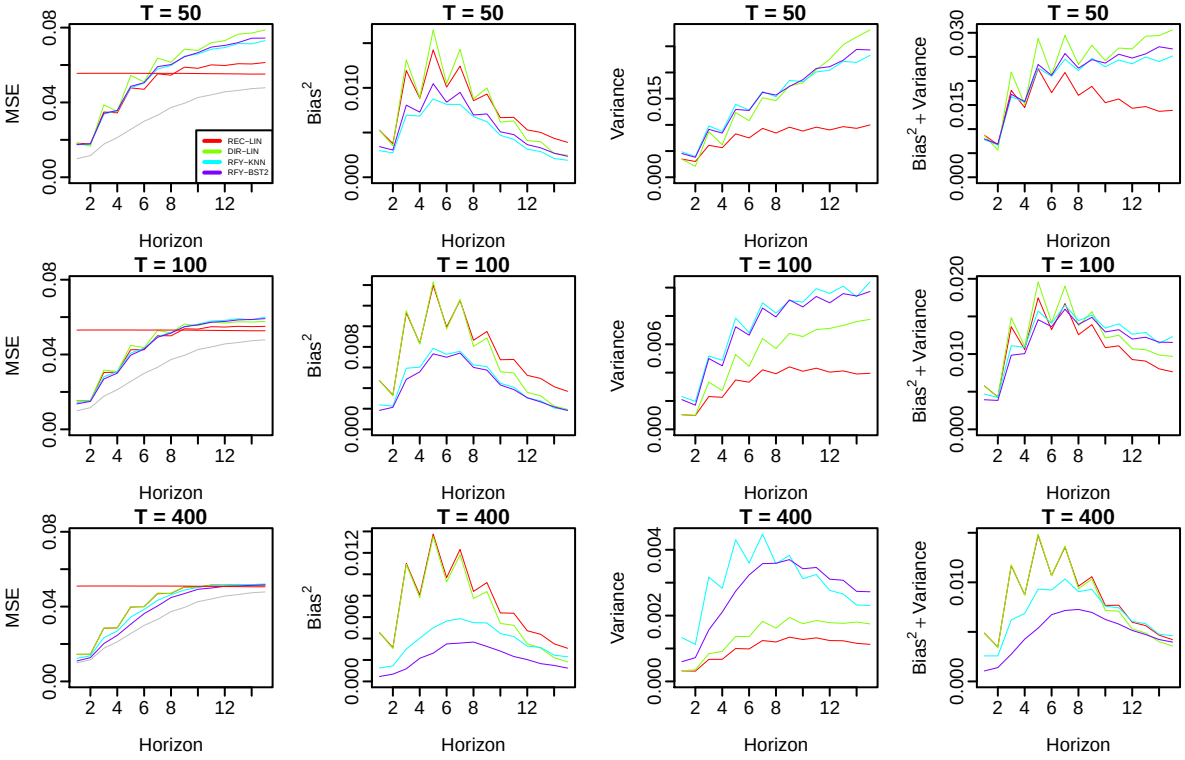

Figure 5.2: *STAR DGP. MSE decomposition of rectify (RFY-KNN) and boost (RFY-BST2) forecasts. The results for linear recursive (REC-LIN) and direct (DIR-LIN) forecasts are also included.*

In the second column of Figure 5.2, we can clearly see that the RFY strategies have a smaller bias component than REC-LIN and DIR-LIN, which shows the positive contribution of the rectification models of the RFY strategies. We also see that the relative decrease in bias gets larger with longer time series, i.e. with larger value of T.

Naturally, the decrease in bias has also induced an increase in variance as can be seen in the third column of Figure 5.2. In particular, we can see that REC-LIN and DIR-LIN have a lower variance than the RFY strategies, expect for $T = 50$ where DIR-LIN has a comparable bias component to the RFY strategies.

The last column of Figure 5.2 allows us to see which strategy has the best trade-off between bias and variance components. For a short time series as $T = 50$, we can see that it is hard to beat REC-LIN

even if the DGP is nonlinear. This is due to the huge decrease in variance (without a too much increase in bias) for REC-LIN, particularly at long horizons. For longer time series, the performance of both REC-LIN and DIR-LIN reduces compared to the RFY strategies. For $T = 100$, we observe higher errors for REC-LIN and DIR-LIN at short horizons due to their higher bias component. For $T = 400$, the RFY strategies have smaller errors compared to REC-LIN and DIR-LIN with RFY-BST2 dominating all the strategies consistently over the horizon.

For the the NAR DGP, in the second column of Figure 5.3, we do not observe a large reduction in the bias component of RFY strategies compared to REC-LIN and DIR-LIN, as with the STAR DGP in Figure 5.2. This suggests that the NAR DGP is a weakly nonlinear process compared to theSTAR DGP. With RFY-KNN, we can even see a high bias which suggests that the direct KNN rectification models are not well suited for the NAR DGP. This also shows that, in addition to reducing the variance, RFY-BST2 can also reduce the bias compared to RFY-KNN if an appropriate weak learner is used.

Concerning the variance component, the third column of Figure 5.3 shows the lower variance of REC-LIN as in Figure 5.3 with the STAR DGP. Also, we can see that the nonlinear rectification terms of the RFY strategies have not reduced the bias of REC-LIN, but instead have only increased the total variance.

If we compare the third and the fourth column of Figure 5.3, we see that they are very similar. This suggests that the variance component is dominating the bias all over the horizons. This can also be seen by analyzing the scale of the second and the third columns. Finally, we see that the relative performance of RFY-BST2 compared to REC-LIN and DIR-LIN increases with the time series length T while RFY-KNN is dominated by the other strategies due to its higher bias, particularly for $T = 100$ and $T = 400$.

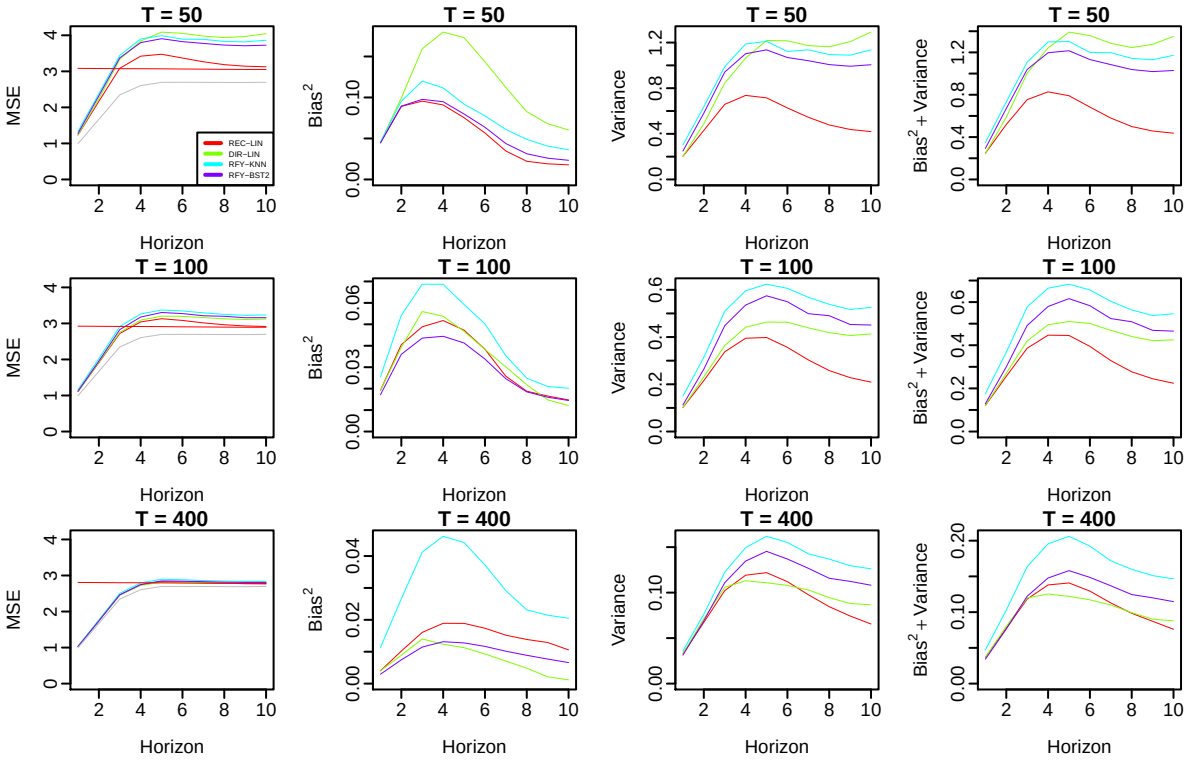

Figure 5.3: *NAR DGP. MSE decomposition of rectify (RFY-KNN) and boost (RFY-BST2) forecasts. The results for linear recursive (REC-LIN) and direct (DIR-LIN) forecasts are also included.*

5.3.3 Scenario C: Nonlinear model and linear DGP

In this section, we compare the rectify and boost strategies with the recursive and direct nonlinear forecasts when the DGP is linear.

If the nonlinear models of the recursive and direct strategies switch to linear models then this scenario becomes equivalent to scenario A (Section 5.3.1). In that case, because the DGP is linear, the ideal configuration for the rectify and boost forecasts is to have no rectification components, so that they are equivalent to recursive linear forecasts.

Let us consider the case where the nonlinear models of the recursive and direct strategies do not switch to linear models. The expression (4.3.1) for the two-step ahead bias and variance components is given by expression (5.3.5) and (5.3.6) for the rectify and the boost strategy, respectively.

If we compare expressions (5.3.5) and (5.3.6) with expression (4.4.19), we see that the rectify and boost forecasts do not suffer from the amplification of errors as nonlinear recursive forecasts since the recursive base forecasts are generated by a linear model. Of course, the direct rectifications will increase the variance of the linear base forecasts but we expect this variance to be smaller than the variance induced by the amplification of errors of nonlinear recursive forecasts, particularly when the model m produces a function that has large variations (i.e. m_{z_1} and $m_{z_1 z_1}$ are large in magnitude).

If we compare expressions (5.3.5) and (5.3.6) with expression (4.4.24), we can see that they both contain direct nonlinear terms.

However, because the rectify and boost strategies remove the signal modelled by the linear model, the direct nonlinear rectification terms are expected to be simpler than the nonlinear terms of the direct strategy. In particular, with a linear DGP, recursive linear forecasts plus direct nonlinear rectification terms are expected to have a lower variance than only direct nonlinear terms. Furthermore, the nonlinear rectification terms of the the boost strategy can benefit from a reduced variance due to the weak components and the limitation to bivariate interactions.

Let us now consider Monte Carlo simulations for the general case of h-step ahead forecasts. We compare RFY-KNN with REC-KNN and DIR-KNN in Figure 5.4, where the results for the LIN and the MLP models are also included for comparison. The same information is given in Figure 5.5 to compare RFY-BST2 with REC-BST2 and DIR-BST2.

In Figures 4.6 and 4.7, we have seen that REC-LIN and DIR-LIN have the best performance, followed by REC-MLP and DIR-MLP, which are followed by REC-KNN and DIR-KNN. In the last column of Figure 5.4, we see that RFY-KNN has smaller errors than REC-KNN and DIR-KNN as well as REC-MLP and DIR-MLP. In particular, RFY-KNN is in the middle between forecasts generated with the LIN model and both the KNN and MLP models. This suggests that although RFY-KNN suffer from higher errors than REC-LIN due to useless rectifications (since the DGP is linear), it still has lower errors than recursive and direct nonlinear forecasts. Finally, by comparing RFY-KNN with REC-KNN and DIR-KNN, we can see the benefits of the linear base forecasts.

In the last column of Figure 5.5, we can make the same observations as in Figure 5.4. In fact, we see that RFY-BST2 has smaller errors than recursive and direct forecasts from both the BST2 and the MLP models. Also, RFY-BST2 is in the middle between forecasts generated with the LIN model and both the BST2 and MLP models.

Overall, the RFY strategies achieve a good trade-off between the bias and variance components compared to recursive and direct nonlinear forecasts. Also, the smaller error of the RFY strategies compared to REC-MLP suggests that the RFY strategies are also competitive with recursive forecasts generated with a nonlinear model that can switch to a linear model such as the MLP model.

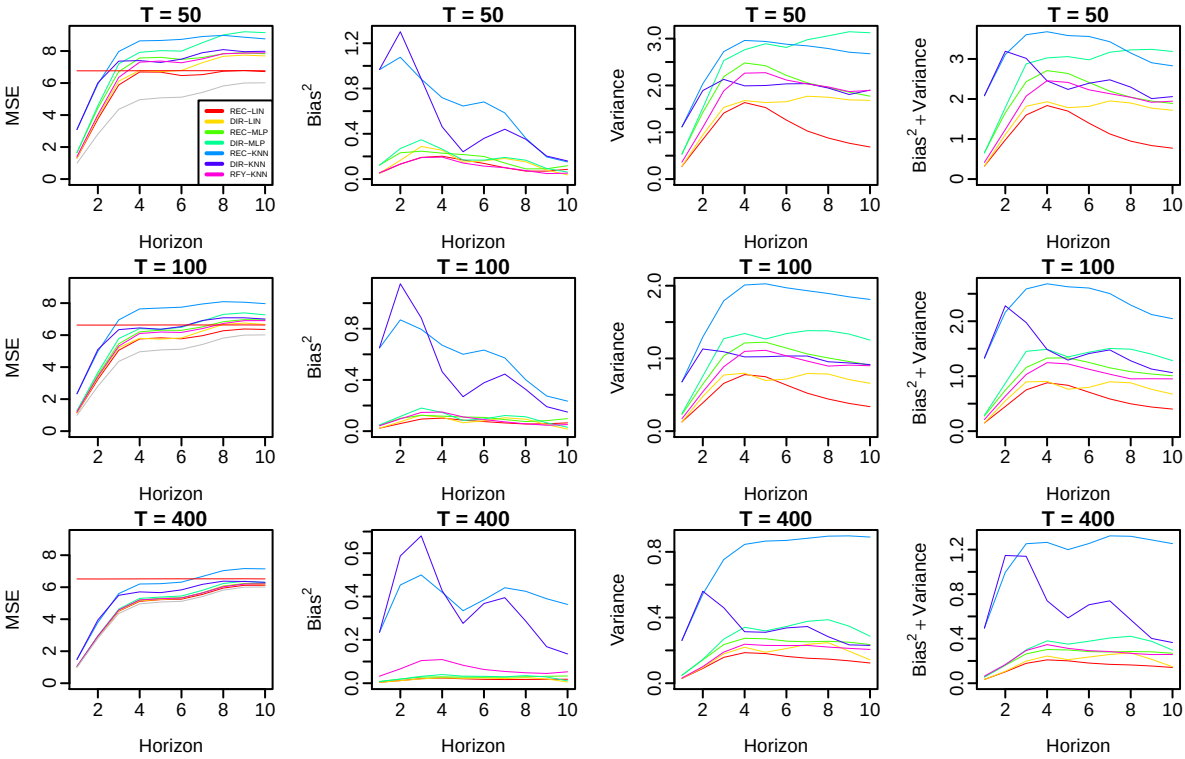

Figure 5.4: *AR DGP. MSE decomposition of rectify forecasts (RFY-KNN). The results for recursive (REC) and direct (DIR) forecasts with the LIN, MLP and KNN models are also included.*

5.3.4 Scenario D: Nonlinear model and nonlinear DGP

We compare the rectify and boost strategies with recursive and direct nonlinear forecasts when the DGP is nonlinear.

As in Scenario B (Section 5.3.2), the linear base forecasts of the rectify and boost strategies will generate biased forecasts since the DGP is nonlinear, and the direct nonlinear rectification models will remove or decrease the bias by considering nonlinear rectification terms over the forecast horizon. The difference with scenario B is that the recursive and direct strategies generate forecasts with a nonlinear model.

The expression (4.3.1) for the two-step ahead bias and variance components of the rectify and the boost strategy are given in expressions (5.3.7) and (5.3.11), respectively.

Let us first compare the rectify and the boost strategies with the recursive strategy by comparing expressions (5.3.7) and (5.3.11) with expression (4.4.26). In Section 4.4.4, we have seen that nonlinear recursive forecasts are asymptotically biased and suffer from an amplification of errors, particularly with highly nonlinear models.

Because the rectify and boost strategies compute recursive linear forecasts, they do not suffer from the amplification of errors as explained in Section 4.4.3. Also, although the recursive linear forecasts are biased when the DGP is nonlinear, we have seen in Scenario B that they can provide good first approximations especially for a weakly nonlinear DGP. Nevertheless, the rectify and the boost strategies will add additional rectification terms to remove or decrease the bias of the recursive linear forecasts, as can be seen in expressions (5.3.9) and (5.3.13).

Although the additional rectification terms of the rectify and boost strategies will increase the total variance, as can be seen in expressions (5.3.10) and (5.3.14), the amplified variance of nonlinear

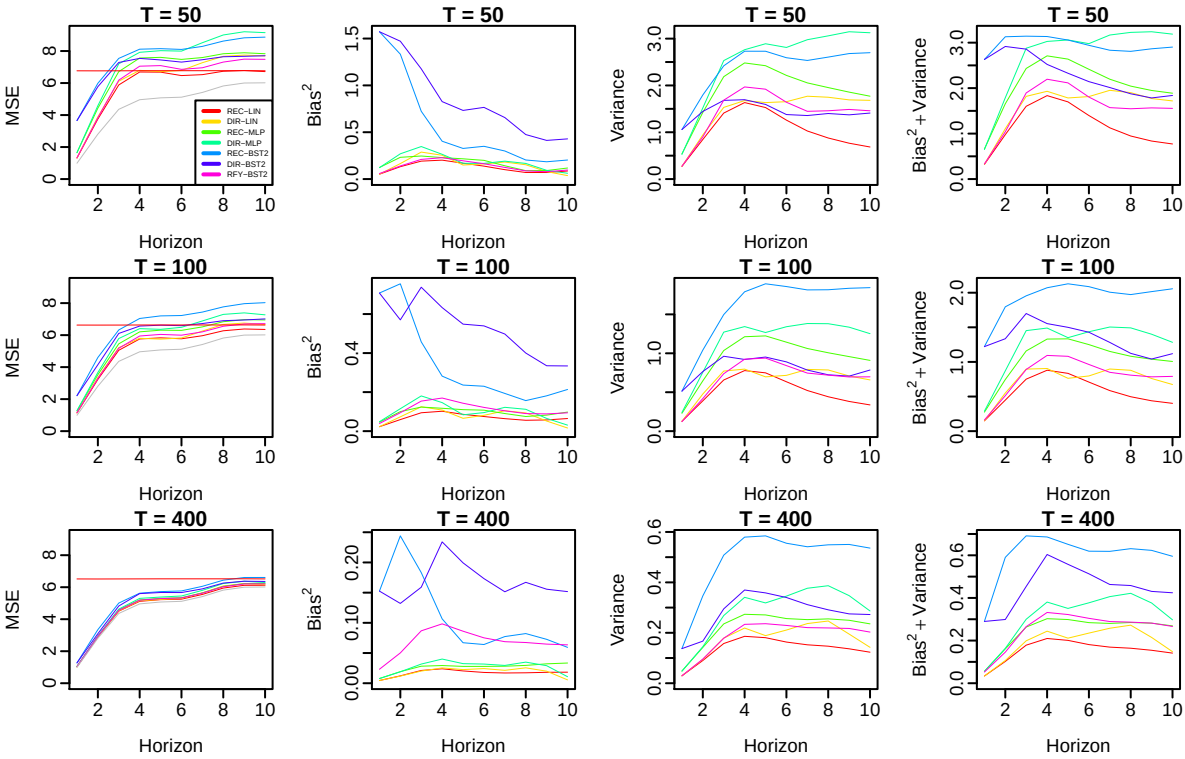

Figure 5.5: *AR DGP. MSE decomposition of boost forecasts (RFY-BST2). The results for recursive (REC) and direct (DIR) forecasts with the LIN, MLP and BST2 models are also included.*

recursive forecasts, given in (4.4.29)-(4.4.30), is expected to be much higher. In addition, the boost strategy will have an even smaller increase in variance notably due the limitation to bivariate interactions.

We now compare the rectify and the boost strategies with the direct strategy by comparing (5.3.7) and (5.3.11) with expression (4.4.31). In Section 4.4.4, we have seen that the direct strategy does not suffer from the amplification of errors as with the recursive strategy. However, because it selects a different model for each horizon, it can have a high variance at long horizons, especially with short time series.

The rectify and boost strategies also do not suffer from the amplification of errors and can have an arbitrarily small bias provided the rectification models are flexible enough to the estimate the residuals from the base forecasts. Also, because the base model has already modeled part of the regression function, we expect the residuals from the base model to be smoother and have less dependence on the inputs $\boldsymbol{r}_t$ (see (5.2.2)). A reduced dependence on the inputs allows simpler models with fewer lagged variables (less variables in $\boldsymbol{r}_t$) to be fitted. For example, with nonparametric models, reduced dependence on the inputs allows larger bandwidth. In other words, with a nonlinear DGP, recursive linear forecasts adjusted by simple nonlinear rectification terms is expected to have a lower variance than direct nonlinear forecasts, i.e. both (5.3.10) and (5.3.14) are expected to be smaller than (4.4.31).

Let us now consider Monte Carlo simulations for the general case of h-step ahead forecasts. We will compare RFY-KNN with REC-KNN and DIR-KNN, and RFY-BST2 with REC-BST2 and DIR-BST2.

The results for the STAR DGP are given in Figures 5.6 and 5.7, for RFY-KNN and RFY-BST2, respectively. We also include the results for the recursive and direct forecasts generated with the LIN and the MLP models.

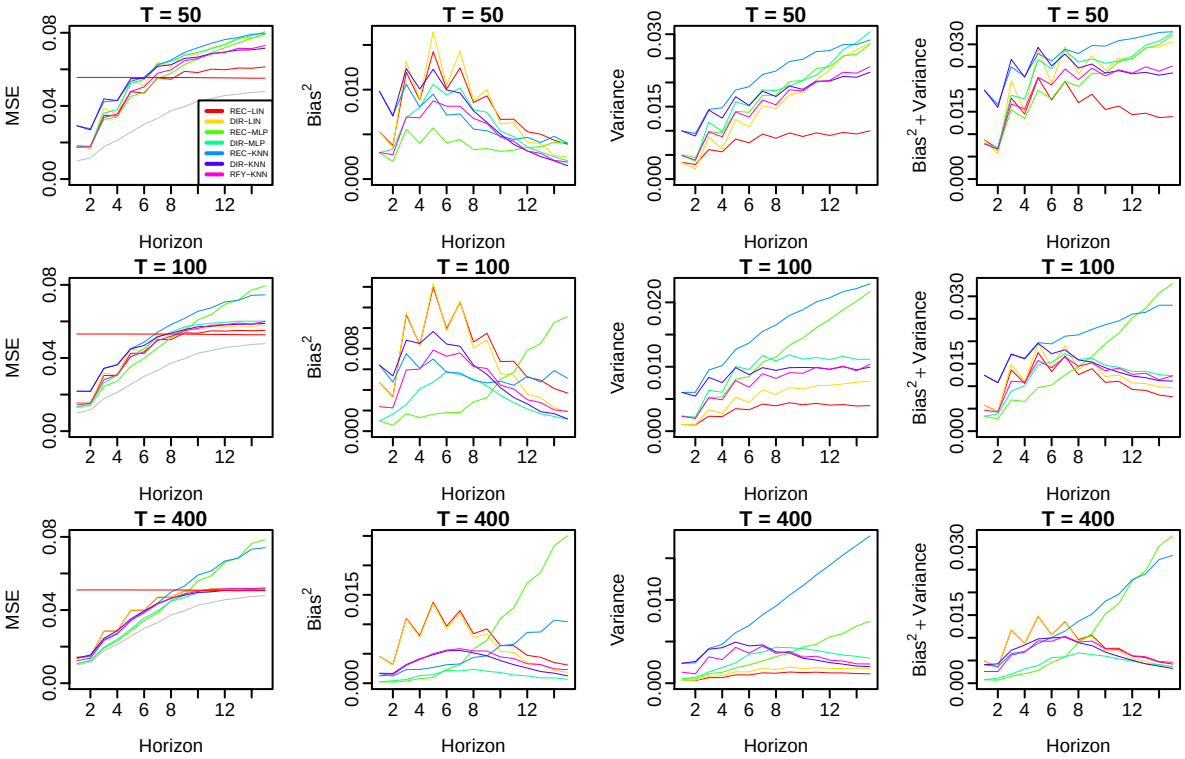

Figure 5.6: *STAR DGP. MSE decomposition of rectify forecasts (RFY-KNN). The results for recursive (REC) and direct (DIR) forecasts with the LIN, MLP and KNN models are also included.*

In the last column of Figure 5.6, we can see for $T = 50$ and $T = 100$ that RFY-KNN has smaller errors than both REC-KNN and DIR-KNN. For $T = 400$, RFY-KNN has comparable performance than DIR-KNN and better performance than REC-KNN for long horizons.

Compared to REC-MLP, RFY-KNN has smaller errors for long horizons but larger errors for short horizons. DIR-MLP and RFY-KNN have a comparable performance for $T = 50$ and $T = 100$ but DIR-MLP outperforms all strategies for $T = 400$.

Finally, RFY-KNN has performance close to REC-LIN for short horizons and smaller errors than DIR-LIN for $T = 50$. For $T = 100$, the three strategies have close performances over the horizon. For $T = 400$, RFY-KNN has smaller error than both REC-LIN and DIR-LIN, especially at short horizons. The same behavior can be observed with RFY-BST2 in Figure 5.7.

The results for the NAR DGP are given in Figures 5.8 and 5.9, for RFY-KNN and RFY-BST2, respectively.

In the last column of Figure 5.8, we can see that RFY-KNN has smaller errors than REC-KNN consistently over the horizon. Compared to DIR-KNN, it has smaller errors for short horizons and comparable performance for long horizons. In the second column, we can see that RFY-KNN has significantly reduced the bias component at short horizons compared to both REC-KNN and DIR-KNN. The third column also shows that RFY-KNN has a lower variance than REC-KNN consistently over the horizon and a lower variance than DIR-KNN for short horizons.

For $T = 50$ and $T = 100$, RFY-KNN outperforms both REC-MLP and DIR-MLP, while for $T = 400$, they have a comparable performance. The third column shows that the better performance of RFY-KNN is due to its lower variance component.

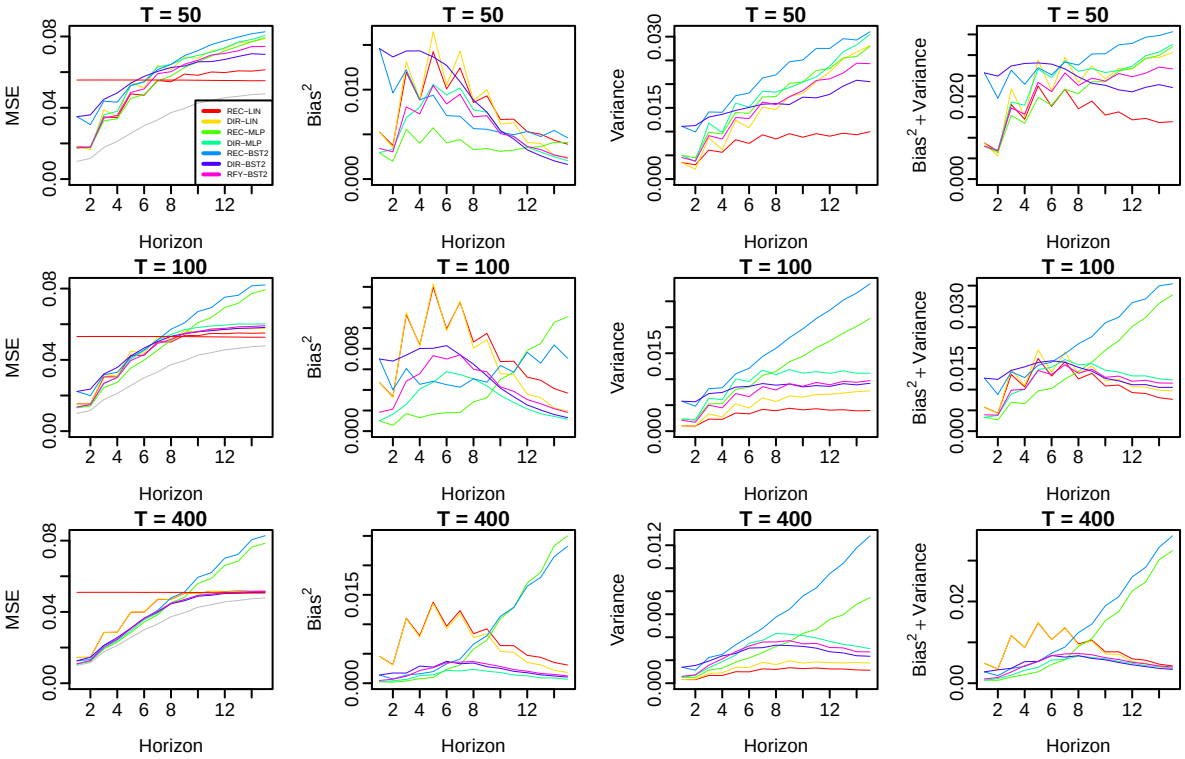

Figure 5.7: *STAR DGP. MSE decomposition of boost forecasts (RFY-BST2). The results for recursive (REC) and direct (DIR) forecasts with the LIN, MLP and BST2 models are also included.*

Except for $T = 50$, both REC-LIN and DIR-LIN have smaller errors than RFY-KNN consistently over the horizon. Finally, in contrast to REC-KNN and DIR-KNN, RFY-KNN does not have a huge difference in performance with REC-LIN and DIR-LIN. The same behavior can be observed with RFY-BST2 in Figure 5.9.

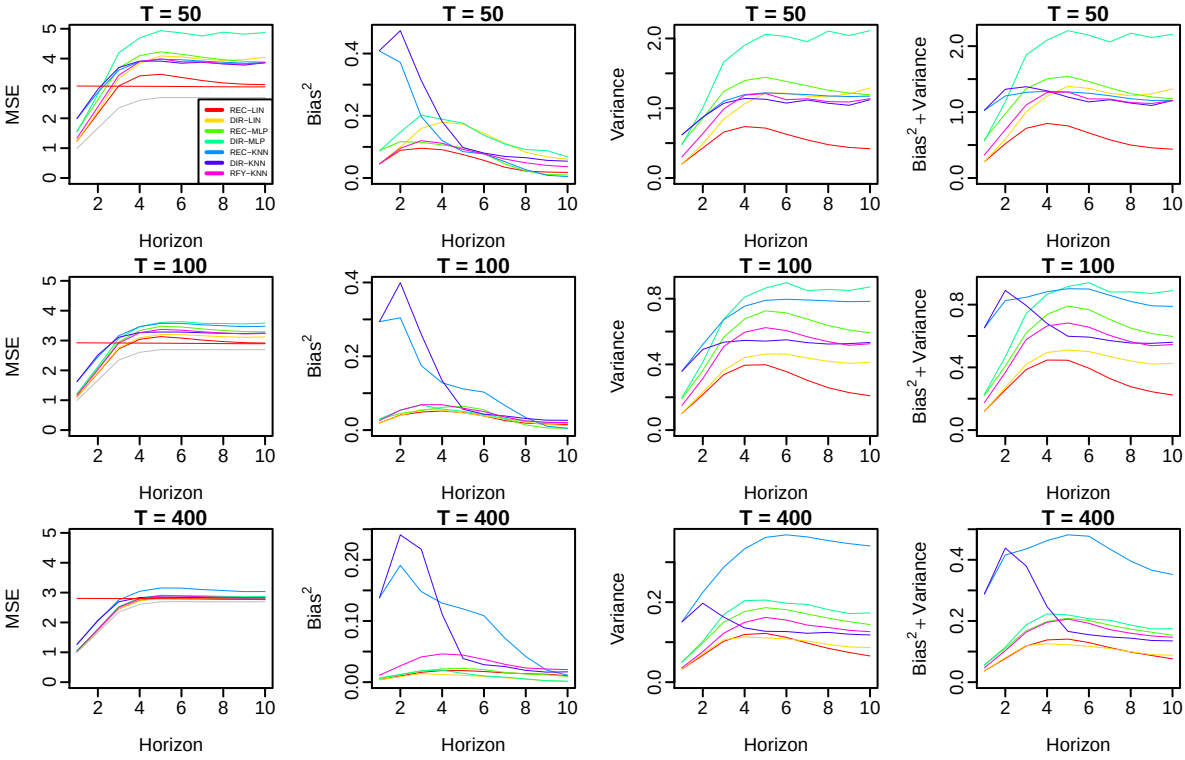

Figure 5.8: *NAR DGP. MSE decomposition of rectify forecasts (RFY-KNN). The results for recursive (REC) and direct (DIR) forecasts with the LIN, MLP and KNN models are also included.*

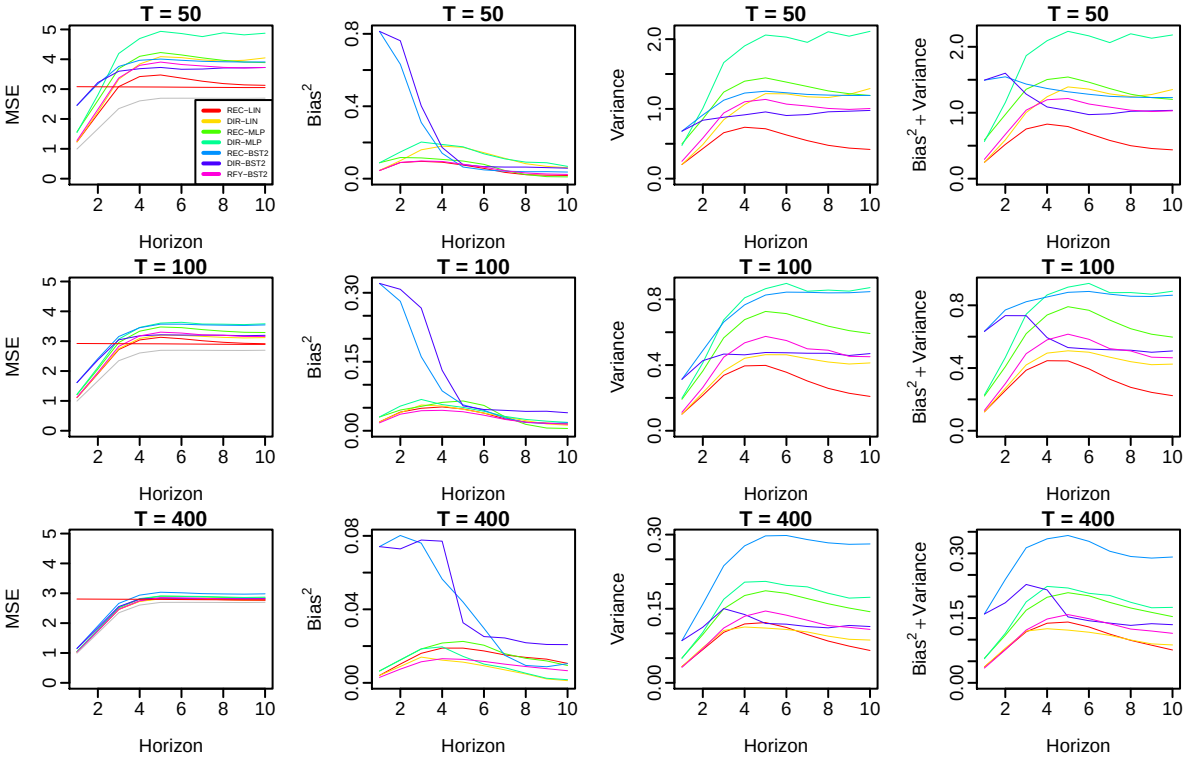

Figure 5.9: *NAR DGP. MSE decomposition of boost forecasts (RFY-BST2). The results for recursive (REC) and direct (DIR) forecasts with the LIN, MLP and BST2 models are also included.*

5.3.5 Averaging strategies

The RFY strategies are closely related to the strategies that compute a simple average of the recursive and direct forecasts, which we will call the AVG strategies. The main difference between the RFY and the AVG strategies is that the RFY strategies build the forecasts iteratively by adding direct nonlinear rectification terms to the recursive base forecasts, while the AVG strategies compute recursive and direct forecasts independently and then average them to obtain the final forecasts, as can be seen in expression (5.2.3).

In this section, we will compare RFY-KNN and RFY-BST2 with the following AVG strategies: (i) average of REC-LIN and DIR-KNN, denoted as AVG-LIN-KNN, (ii) average of REC-LIN and DIR-BST2, denoted as AVG-LIN-BST2, (iii) average of REC-KNN and DIR-KNN, denoted as AVG-KNN-KNN and (iv) average of REC-BST2 and DIR-BST2, denoted as AVG-BST2-BST2.

As explained in Section 4.2.1, the variance component of both the AVG and the RFY strategies can be further decomposed into subcomponents as in expression (4.2.5), to understand the different sources of variance. In fact, the forecasts g of both the AVG and the RFY strategies can be written as a sum of two parts g_1 and g_2. For the AVG strategies, g_1 and g_2 represent the recursive and direct forecasts respectively divided by two. For the RFY strategies, g_1 represents the REC-LIN forecasts and g_2 represents the rectification term. So, the total variance V_h of the forecasts g at horizon h can be decomposed into the variances $V_{1;h}$ and $V_{2;h}$ of each part g_1 and g_2 plus the covariance COV_h of the two parts.

This additional decomposition will be given in the form of stacked area plots with the following information. The total variance is given by the (hatched) yellow area under the red line and the total bias is given by the area in cyan between the black line and the red line. The variance of the recursive forecasts is given in grey and the variance of the direct forecasts or the rectification terms is given in orange. Finally, the covariance component is given in white if it is positive, otherwise, it is given by the area above the red line (not taking into account the blue area).

The results for the AR DGP are given in Figures 5.10 and 5.11.

Let us first compare the RFY-strategies with REC-LIN. Recall that the RFY strategies adjust the forecasts from REC-LIN by using direct nonlinear rectification models. Because, the LIN model is well-specified for the AR DGP, the residuals from the REC-LIN forecasts will only contain noise. In consequence, the rectifications models of the RFY strategies will contribute to the increase of the forecast variance as can be seen in the third column of Figure 5.10.

For both RFY-KNN and RFY-BST2, the variance of the rectification models can be seen in Figure 5.11 where the increase in variance compared to REC-LIN is particularly noticeable at long horizons, but is smaller with RFY-BST2. In addition, for the RFY strategies, the covariance is close to zero since the base model and the rectification models are estimating different quantities. This also justifies the assumptions we made about the variability terms for expressions (5.3.1) and (5.3.2). Finally, the difference in bias between REC-LIN and the RFY strategies increases with the time series length T as can be seen in the second column of Figure 5.10.

Let us now compare RFY-KNN and RFY-BST2 with AVG-LIN-KNN and AVG-LIN-BST2. In the last column of Figure 5.10, we see that the RFY strategies have higher errors than the AVG strategies, especially for long horizons. However, the relative difference between the RFY strategies and the AVG strategies decreases with an increasing time series length. In addition, for $T = 400$, we observe larger errors for AVG-LIN-KNN compared to RFY-KNN at short horizons.

The smaller errors of the AVG-LIN strategies can be explained by the fact that the DGP is linear, and as a result REC-LIN has both a smaller bias and variance component. Averaging REC-LIN

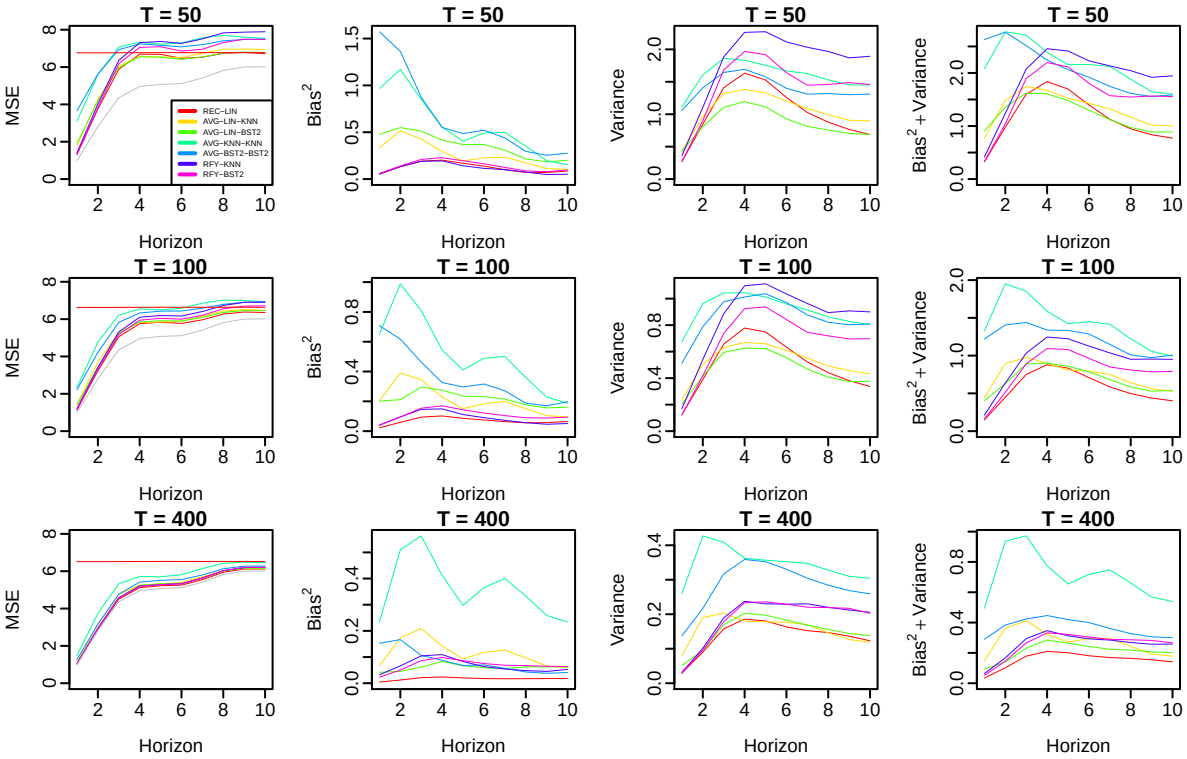

Figure 5.10: *AR DGP. MSE decomposition of recursive linear forecasts (REC-LIN), rectify forecasts (RFY-KNN), boost forecasts (RFY-BST2), and averaging (AVG) forecasts with LIN, MLP, KNN and BST2 models.*

with nonlinear direct forecasts will increase the errors for long time series but for short time series, the AVG-LIN strategies will still benefit from the smaller errors of REC-LIN.

The variance decomposition of the AVG-LIN strategies is given in Figure 5.11 where we can see that the variances of g_1 and g_2 are much smaller than those of the RFY strategies. This can be explained by the fact that the averaging operator is dividing each part by two, and as a result, the variance of each part is divided by four.

In contrast to the RFY strategies, we can see that the two parts g_1 and g_2 of the AVG-LIN strategies have a positive covariance. This can be explained by the fact that the two parts are estimating the same quantity, while for the RFY strategies, g_2 is estimating the residuals from g_1; and therefore they are complementary. In terms of the bias component, we can see in the second column of Figure 5.10 that the AVG-LIN strategies have a higher bias than the RFY strategies, particularly for $T = 50$ and $T = 100$.

Finally, let us compare the RFY strategies with AVG-KNN-KNN and AVG-BST2-BST2, that is strategies that compute an average of nonlinear recursive and direct forecasts. The results in the last column of Figure 5.10 suggest that these strategies do not improve the performance compared to the RFY strategies. In the second and third column, we can see that they have both a high bias and high variance compared to the RFY strategies. We can also see that the AVG-BST2-BST2 strategy has smaller errors than AVG-KNN-KNN for $T = 100$ and $T = 400$. One explanation is that the averaging procedure cannot reduce the high bias of REC-KNN and REC-BST2 (see Figures 5.4 and 5.5).

The results for the STAR DGP are given in Figures 5.12 and 5.13.

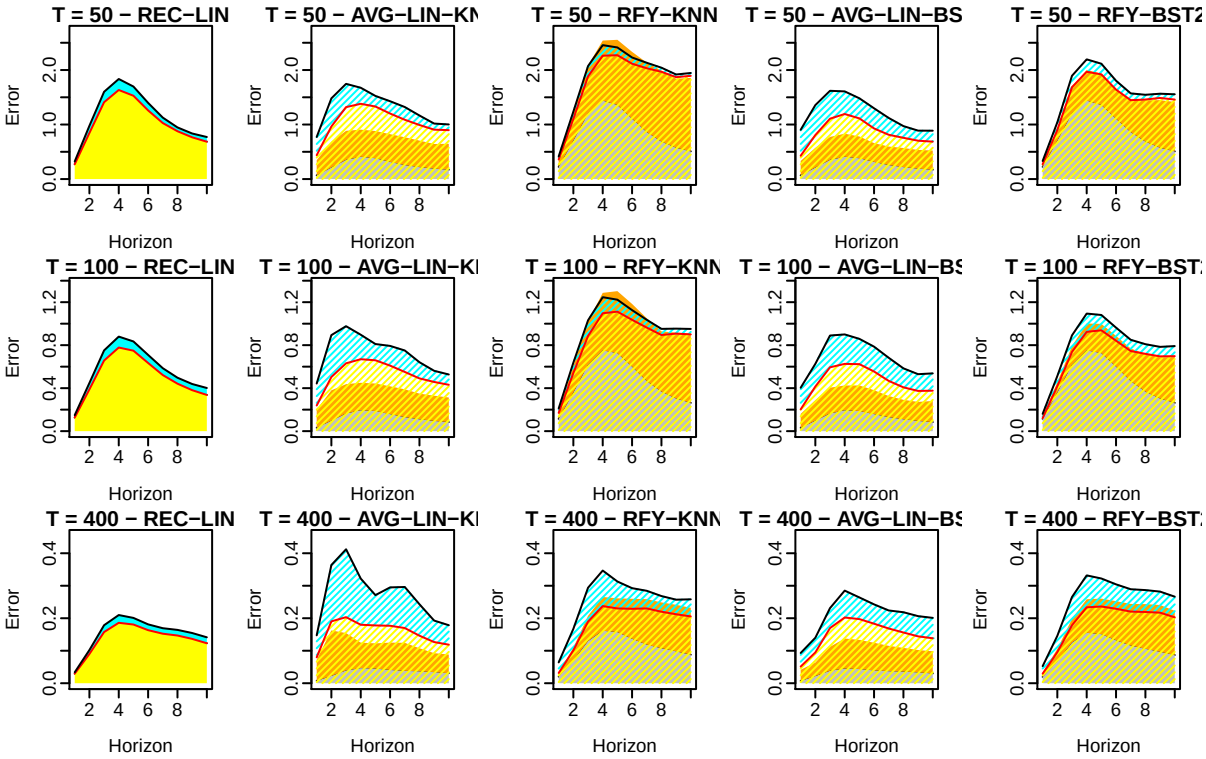

Figure 5.11: *AR DGP. The MSE of REC-LIN, AVG-LIN-KNN, RFY-KNN, AVG-LIN-BST2 and RFY-BST2 is decomposed into noise (in grey), bias (in cyan) and variance (in yellow) components. The stacked area plots show the relative contribution of each component to the total MSE over the forecast horizon.*

In Section 4.4.4, we have seen that, even if the DGP is nonlinear, REC-LIN has smaller errors than the strategies that involve nonlinear forecasts at long horizons due to a significantly lower variance (see Figure 4.11). In the third column of Figure 5.12, we can see that the AVG-LIN strategies also benefit from a significantly lower variance since the final forecasts are partly composed of REC-LIN forecasts.

In Section 4.4.4, we have also seen that REC-LIN is biased when the DGP is nonlinear. In Figure 5.12, we can see that the AVG-LIN strategies have also a higher bias compared to other strategies but a smaller bias than REC-LIN since they are partly composed of direct nonlinear forecasts.

Compared to the RFY strategies, the AVG-LIN strategies have a lower variance for all time series lengths. In Figure 5.13, we can see that the higher variance of the RFY strategies comes from the higher variance of the rectification models at long horizons. However, the AVG-LIN strategies have a higher bias than the RFY strategies for short horizons and a comparable bias component for longer horizons. In addition, the relative difference in the bias component between these strategies increases with the time series length T, especially with RFY-BST2.

If we compare AVG-KNN-KNN and AVG-BST2-BST2 with the corresponding RFY strategies, we can see in the last column of Figure 5.12 that AVG-KNN-KNN and AVG-BST2-BST2 suffer from higher errors compared to the RFY strategies for the first few horizons and for $T = 50$ and $T = 100$, due to both a higher bias and a higher variance. For longer horizons, the RFY strategies have higher errors when $T = 50$ and comparable performance when $T = 100$. In the third column, we can see for $T = 400$ that both AVG-KNN-KNN and AVG-BST2-BST2 have an increasing variance with the horizon. This can be explained by the fact that they are composed of nonlinear recursive forecasts.

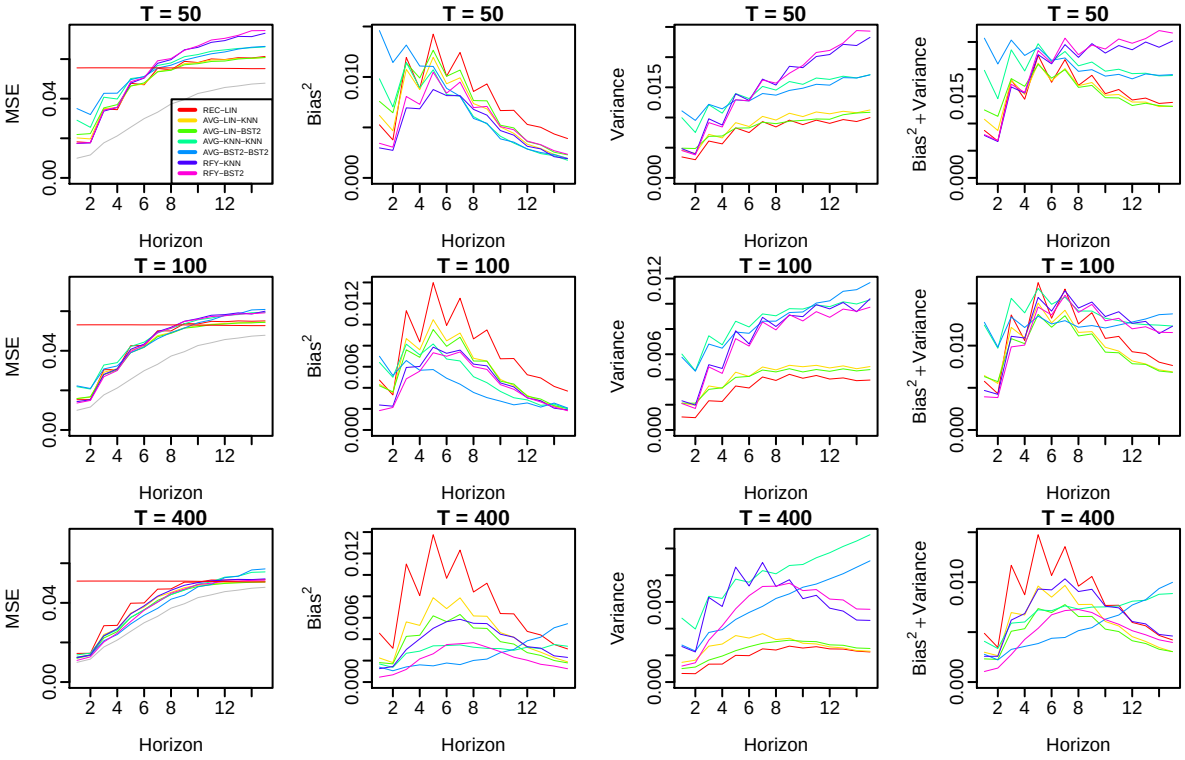

Figure 5.12: *STAR DGP. MSE decomposition of recursive linear forecasts (REC-LIN), rectify forecasts (RFY-KNN), boost forecasts (RFY-BST2), and averaging (AVG) forecasts with LIN, MLP, KNN and BST2 models.*

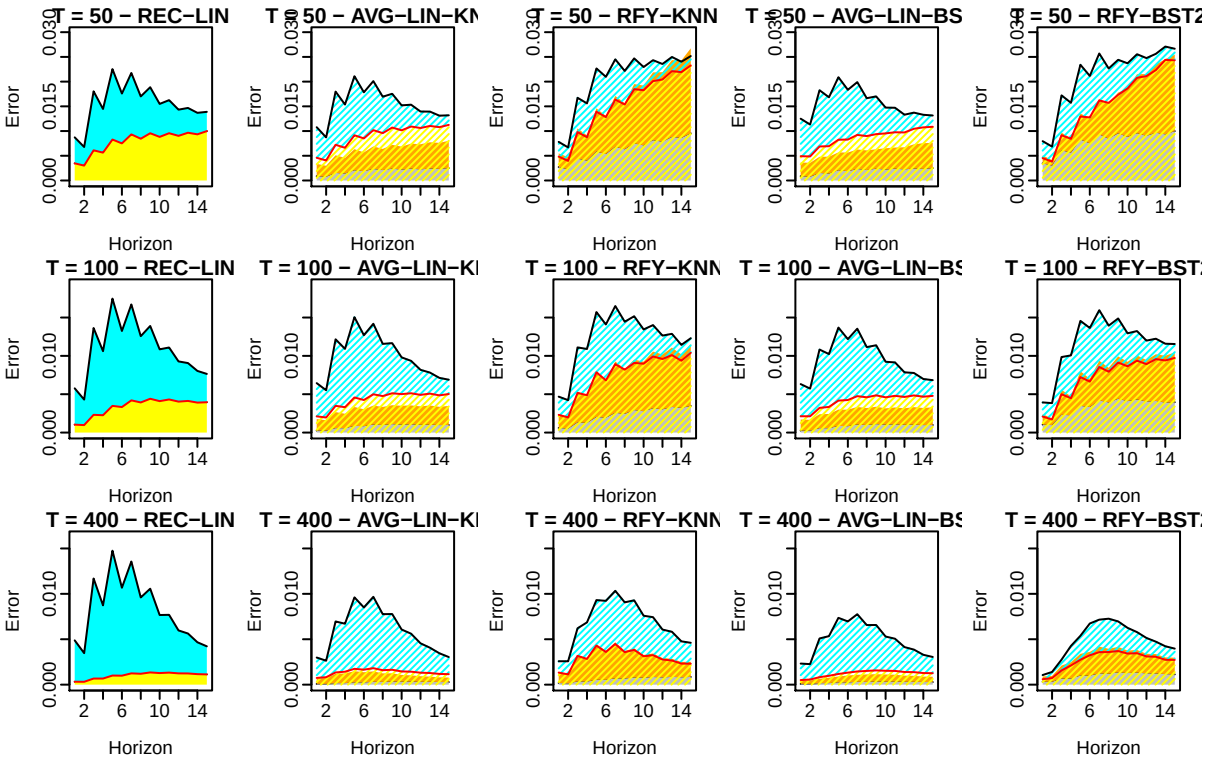

Figure 5.13: *STAR DGP. The MSE of REC-LIN, AVG-LIN-KNN, RFY-KNN, AVG-LIN-BST2 and RFY-BST2 is decomposed into noise (in grey), bias (in cyan) and variance (in yellow) components. The stacked area plots show the relative contribution of each component to the total MSE over the forecast horizon.*

5.3.6 Summary

- **Scenario A: Linear model and linear DGP.** If the base model is well-specified, the direct rectification terms of the rectify and boost strategies will not bring any benefit to the recursive base forecasts since the residuals will only contain noise. In that case, having small or no rectification terms is better in order not to increase the variance too much. In particular, with no rectification terms, the rectify and boost forecasts become equivalent to the recursive linear forecasts that has better properties in that setting.

  If the base model is misspecified, for example due to omitted variables, the rectification terms will remove or reduce the bias of the recursive base forecasts provided that the omitted variables are included into the inputs of the rectification models.

  Overall, because the DGP is linear, nonlinear rectification terms are likely to increase the variance of both rectify and boost forecasts compared to recursive and direct linear forecasts, especially with short time series. The boost strategy can be a better alternative since it limits the complexity of the rectification models by only allowing bivariate interactions between the lagged variables and by benefiting from the reduced variance of the weak learners.

- **Scenario B: Linear model and nonlinear DGP.** In contrast to the recursive and direct linear forecasts, the rectify and boost forecasts will generate unbiased forecasts provided that the rectification models are flexible enough to estimate the nonlinear function. However, even if the DGP is nonlinear, the recursive and direct linear forecasts can benefit from a significantly lower variance especially with short time series at long horizons.

  The performance of the rectify and boost forecasts compared to recursive and direct linear forecasts will also depend on the level of nonlinearity. For a weakly nonlinear DGP, recursive and direct linear forecasts can already provide good approximations and will have a low bias component, especially for short time series. The rectify and boost forecasts will reduce this bias and will have smaller errors provided that the rectification terms do not increase too much the variance.

  For a strongly nonlinear DGP, the bias of the recursive and direct linear forecasts is expected to be high, especially for long time series. The rectify and boost forecasts will have a lower bias since they include nonlinear rectification terms. However, because of the limitations to bivariate interactions, the boost strategy can have a higher error than the rectify strategy, for example if the nonlinear function contain higher-order interactions.

- **Scenario C: Nonlinear model and linear DGP.**

  If the nonlinear models of the recursive and direct strategies all switch to linear models, this scenario becomes equivalent scenario A. Otherwise, because the DGP is linear, the rectification terms of the rectify and boost strategies will not bring any benefit to the recursive base forecasts. In contrast, having additional rectification terms will only contribute to increase the errors, especially the variance with short time series.

  However, even if the rectify and boost strategies includes additional rectification terms, the forecast errors are expected to be smaller than the forecast errors of the recursive nonlinear forecasts, that suffer from the amplification of errors with the horizon. Compared to direct nonlinear forecasts, the rectify and the boost strategies can benefit from the lower variance of the recursive base forecasts provided that the rectification terms do not increase the variance too much. This is more likely to happen with the boost strategy due to the reduced variance of the rectification models.

- **Scenario D: Nonlinear model and nonlinear DGP.**

Because the rectify and boost strategies generate recursive linear forecasts, they do not suffer from the amplification of errors with the horizon as with the recursive nonlinear forecasts. Also, the rectification models of the rectify and boost strategies allow to reduce the bias of the recursive linear forecasts at each horizon.

Compared to direct nonlinear forecasts, the rectify and boost strategies model part of the function using the linear base model rather than directly estimate it. One advantage of this process is that simpler nonlinear direct rectification models can be used which can in turn decrease the total variance.

- **Comparison with averaging strategies**

  For both linear and nonlinear DGP, and for long time series, the rectify and boost strategies have comparable performance to the averaging strategies that are composed of recursive linear forecasts and direct nonlinear forecasts, but have higher errors for short time series. In particular, the rectification models of the rectify and boost strategies contribute significantly to the increase in variance at long horizons.

  The averaging strategies that are composed of recursive and direct nonlinear forecasts have higher errors than the averaging strategies which includes recursive linear forecasts. Compared to the rectify and boost strategies, they have higher errors when the DGP is linear and comparable performance when the DGP is nonlinear.

5.4 Real-data experiments

To shed some light on the performance of the different strategies with real-world time series, we carry out some experiments with the time series from the M3 and NN5 competitions. Details about the data and the methodology are given in Appendix A.

The tables that follow present the SMAPE and MASE forecast accuracy measures (see Section 2.3.6) for different strategies over the forecast horizon. In addition, the tables show average forecast error measures over the horizons 1 to h in the columns labelled 'Average $1-h$'. The average ranking for each strategy over all horizons is also given in the last column, labelled 'Avg. rank'. For each accuracy measure, the strategies are ranked according to the 'Avg. rank', i.e. the last column of each table. To compare the different strategies, we will consider the average rank, the forecast errors at each horizon as well as the average errors over the horizons 1 to h. The presentation of the results is inspired by the work of Athanasopoulos and Hyndman (2011) where the authors compared the performance of various methods for forecasting tourism data.

We first compare the results of the RFY strategies with the REC and DIR strategies for different learning algorithms.

The results for the M3 competition are presented in Table 5.1. Since we can draw the same conclusions from both the SMAPE and MASE accuracy measures, we will only consider the SMAPE measure to compare the strategies. The results can be summarized as follows:

- For all models, the REC strategies have comparable performance to the DIR strategies for short horizons (i.e. $h = 1, 2, 3$), but have smaller errors for longer horizons (i.e. $h = 6, 12, 18$). One reason is that the DIR strategies have a decrease of $h-1$ samples at horizon h which can be important in this case since the M3 time series are relatively short ($T \leq 126$) and the horizon ($H = 18$) is large compared to the time series length.
- REC-LIN and REC-MLP have a comparable performance. This suggests that the MLP model has often selected a structure close to the LIN model. We observe the same behavior when comparing DIR-LIN with DIR-MLP.
- RFY-BST2 improves the performance of REC-LIN, while RFY-KNN decreases it. We can explain the reduced performance of RFY-KNN compared to REC-LIN by the possible overfitting of the direct KNN rectification models with the short M3 time series. The better performance of RFY-BST2 can be explained by the reduced variance of the boosting components. In particular, we can see that RFY-BST2 has comparable performance to REC-LIN for short horizons (i.e. $h = 1, 2, 3$) but smaller errors for long horizons (i.e. $h = 12, 18$). This suggest that RFY-BST2 has applied few boosting iterations at short horizons compared to longer horizons.
- For both KNN and BST2 models, the RFY strategies outperform their respective REC and DIR strategies. In fact, we can see that RFY-KNN has smaller errors than both REC-KNN and DIR-KNN consistently over the horizon. The same behavior is observed for RFY-BST2 compared to REC-BST2 and DIR-BST2. This confirms the advantage of the rectify and boost strategies which model the residuals from the recursive linear forecasts using direct nonlinear models. In particular, RFY-BST2 has considerably reduced the errors over REC-BST2 and DIR-BST2 as already observed in Figure 5.5, which was mainly due to a large bias component.

The results for the NN5 competition are presented in Table 5.2 and can be summarized as follows:

- In contrast to the M3 competition, the DIR strategies outperform the REC strategies, with DIR-BST2 ranking first. One explanation can be that the NN5 time series are longer ($T = 735$)

Table 5.1: *M3 competition. SMAPE and MASE forecast accuracy measures for rectify (RFY-KNN) and boost forecasts (RFY-BST2) as well as recursive (REC) and direct (DIR) forecasts with LIN, KNN, MLP and BST2 models.*

Strategy	Forecast horizon (h)						Average			Avg. rank
	1	2	3	6	12	18	1-6	1-12	1-18	
SMAPE										
RFY-BST2	12.54	11.48	13.17	12.98	**15.11**	**19.78**	12.97	**13.91**	**15.34**	**2.39**
REC-LIN	12.52	**11.37**	13.18	13.02	15.38	20.12	**12.90**	13.93	15.41	3.17
REC-MLP	12.55	11.65	13.59	**12.97**	15.23	19.98	13.05	14.01	15.42	3.56
RFY-KNN	12.77	12.07	13.52	13.34	15.47	19.91	13.40	14.14	15.60	3.83
DIR-MLP	**12.43**	11.74	**13.14**	13.65	15.11	21.07	13.43	14.24	16.11	5.00
DIR-LIN	12.45	11.49	13.18	13.48	15.34	21.91	13.33	14.09	16.15	5.06
REC-KNN	13.10	12.20	13.54	14.05	15.94	20.07	13.82	14.66	16.05	6.36
DIR-KNN	13.10	12.21	13.65	14.26	16.58	20.72	13.83	14.86	16.42	7.25
NAIVE	15.49	14.45	15.86	14.68	15.99	20.86	15.69	16.16	17.37	8.56
REC-BST2	15.49	15.46	17.42	16.91	18.04	21.38	16.96	17.43	18.62	10.03
DIR-BST2	15.49	14.60	17.23	18.44	21.18	23.95	17.08	18.71	20.33	10.81
MASE										
RFY-BST2	2.56	2.51	2.91	**3.42**	4.43	6.06	**3.11**	**3.58**	**4.19**	**2.00**
REC-MLP	2.56	2.53	3.09	3.44	4.36	**6.01**	3.13	3.60	4.20	2.50
REC-LIN	2.55	2.51	2.94	3.46	4.47	6.10	3.12	3.61	4.23	3.44
RFY-KNN	2.63	2.62	3.14	3.46	4.40	6.26	3.25	3.66	4.31	4.33
DIR-MLP	**2.54**	**2.51**	**2.87**	3.56	4.44	6.74	3.14	3.67	4.50	5.56
DIR-LIN	2.55	2.54	2.97	3.45	**4.31**	7.26	3.20	3.65	4.55	5.72
REC-KNN	2.77	2.68	3.27	3.67	4.70	6.20	3.36	3.83	4.42	6.36
DIR-KNN	2.77	2.60	3.21	3.76	5.00	6.61	3.38	3.89	4.57	7.53
NAIVE	3.26	3.08	3.65	3.82	4.54	6.56	3.75	4.12	4.70	7.94
REC-BST2	3.37	3.41	4.08	4.62	5.45	6.68	4.14	4.63	5.16	9.75
DIR-BST2	3.37	3.11	4.05	5.01	6.54	7.39	4.23	5.05	5.75	10.86

and less noisy, and the forecast horizon is longer ($H = 56$). In that case, the DIR strategy can provide better forecasts than recursive forecasts since it does not suffer from the propagation of errors with the horizon, and the loss of data points with the horizon is less important since T is large relative to the horizon H.

- DIR-LIN and DIR-MLP have a comparable performance as in the M3 competition. However, REC-LIN has much lower errors than REC-MLP, which suggest that the MLP model has often selected structures that are quite different from the LIN model.
- When considering the averaged errors, RFY-KNN has larger errors than REC-LIN, while RFY-BST2 has comparable performance to REC-LIN. Again, the reduced variance of the boosting components can explain the better performance of RFY-BST2.
- The RFY strategies have higher errors than their respective DIR strategies. This suggests that the recursive linear forecasts do not provide any benefit and fitting direct models is a better alternative.

Table 5.2: *NN5 competition. SMAPE and MASE forecast accuracy measures for rectify forecasts (RFY-KNN) and boost forecasts(RFY-BST2) as well as recursive (REC) and direct (DIR) forecasts with LIN, KNN, MLP and BST2 models.*

Strategy	Forecast horizon (h)							Average								Avg. rank
	1	2	3	7	14	21	56	1-7	1-14	1-21	1-28	1-35	1-42	1-49	1-56	
SMAPE																
DIR-BST2	14.34	11.81	11.89	18.18	18.30	**40.02**	**16.52**	**15.99**	**17.19**	**19.99**	**19.70**	**21.91**	**22.41**	**22.00**	**21.31**	**2.38**
DIR-LIN	14.05	12.63	12.67	18.78	17.87	40.67	17.21	16.50	17.65	20.43	20.08	22.22	22.68	22.25	21.59	3.86
DIR-MLP	14.86	13.36	12.43	18.72	18.46	40.76	17.27	16.96	17.96	20.76	20.37	22.43	22.86	22.41	21.73	4.64
DIR-KNN	14.66	12.12	**11.64**	**17.78**	18.38	41.48	17.27	16.28	17.47	20.35	20.06	22.25	22.75	22.34	21.69	4.72
REC-BST2	14.34	**11.44**	11.99	18.86	18.65	41.47	18.05	16.07	17.45	20.30	20.10	22.48	22.85	22.39	21.75	5.04
RFY-BST2	**14.03**	12.40	12.56	18.55	18.31	40.92	16.75	16.45	17.75	20.54	20.24	22.44	22.92	22.52	21.81	5.27
REC-LIN	14.05	12.60	12.44	18.45	17.92	41.18	17.70	16.53	17.81	20.53	20.28	22.49	23.07	22.71	22.10	6.23
REC-KNN	14.66	12.40	12.46	18.45	19.15	41.39	17.15	16.88	18.24	21.06	20.76	22.80	23.23	22.76	22.08	6.24
RFY-KNN	14.84	13.63	12.86	18.99	18.32	40.77	17.87	17.17	18.48	21.06	20.78	22.86	23.45	23.09	22.53	8.16
MEAN	16.69	14.22	13.64	18.59	**17.83**	41.54	18.03	18.06	18.91	21.42	21.09	23.24	23.80	23.40	22.77	9.02
REC-MLP	14.90	13.47	13.79	19.61	19.73	43.01	18.84	17.69	19.03	21.85	21.68	23.96	24.39	23.95	23.29	10.43
MASE																
DIR-BST2	0.28	0.28	0.38	0.43	0.42	**0.79**	**0.40**	**0.48**	**0.52**	**0.59**	**0.57**	**0.61**	**0.64**	**0.62**	**0.61**	**2.60**
DIR-LIN	0.27	0.30	0.40	0.44	0.41	0.81	0.42	0.49	0.53	0.60	0.58	0.62	0.64	0.63	0.61	3.79
DIR-MLP	0.28	0.31	0.39	0.44	0.43	0.81	0.42	0.50	0.53	0.61	0.59	0.62	0.65	0.63	0.62	4.66
REC-BST2	0.28	**0.27**	0.38	0.45	0.44	0.83	0.44	0.48	0.52	0.60	0.58	0.62	0.64	0.63	0.61	4.78
DIR-KNN	0.28	0.29	**0.37**	**0.42**	0.43	0.83	0.42	0.49	0.53	0.60	0.58	0.62	0.65	0.63	0.62	4.83
RFY-BST2	**0.27**	0.29	0.39	0.43	0.43	0.82	0.41	0.49	0.53	0.61	0.59	0.62	0.65	0.64	0.62	5.62
REC-KNN	0.28	0.29	0.39	0.44	0.44	0.84	0.42	0.50	0.55	0.62	0.60	0.63	0.66	0.65	0.63	6.31
REC-LIN	0.27	0.30	0.39	0.43	0.42	0.82	0.43	0.49	0.54	0.61	0.59	0.62	0.66	0.65	0.63	6.50
RFY-KNN	0.29	0.31	0.41	0.44	0.43	0.81	0.43	0.51	0.56	0.62	0.60	0.64	0.67	0.66	0.65	8.39
REC-MLP	0.28	0.31	0.42	0.44	0.45	0.85	0.43	0.51	0.56	0.63	0.61	0.65	0.68	0.66	0.65	9.20
MEAN	0.33	0.34	0.44	0.43	**0.41**	0.82	0.44	0.54	0.57	0.64	0.62	0.65	0.68	0.67	0.65	9.32

We now compare the results of the RFY strategies with the AVG strategies. The main motivation for this comparison is to show the difference between averaging recursive and direct forecasts, and adjusting recursive linear forecasts with direct nonlinear models.

The results for the M3 competition are given in Table 5.3 and can be summarized as follows:

- When considering the averaged errors, RFY-BST2 has a comparable performance to AVG-LIN-MLP, AVG-LIN-KNN and AVG-MLP-MLP. This is also confirmed by the MASE accuracy measure.
- AVG-LIN-KNN has smaller errors than RFY-KNN. In other words, averaging recursive linear forecasts with direct KNN forecasts is better than adjusting recursive linear forecasts with direct KNN models. In contrast, RFY-BST2 outperforms AVG-LIN-BST2 which averages the best forecasts (REC-LIN) with the worst forecasts (REC-BST2) as can be seen in Table 5.1.
- The poor performance of AVG-KNN-KNN and AVG-BST2-BST2 can be explained by the fact that they both average recursive and direct forecasts that have high errors.

The results for the NN5 competition are given in Table 5.4 and can be summarized as follows:

- RFY-BST2 has larger errors than both AVG-BST2-BST2 and AVG-LIN-BST2, which benefit from the superior performance of DIR-BST2, as can be seen in Table 5.2. The larger error of

Table 5.3: *M3 competition. SMAPE and MASE forecast accuracy measures for rectify forecasts (RFY-KNN) and boost forecast (RFY-BST2) as well as averaging forecasts (AVG) with LIN, KNN, MLP and BST2 models.*

Strategy	Forecast horizon (h)						Average			Avg. rank
	1	2	3	6	12	18	1-6	1-12	1-18	
SMAPE										
AVG-LIN-MLP	12.41	**11.42**	**12.97**	13.02	14.80	20.04	12.95	**13.73**	15.26	**2.61**
AVG-LIN-KNN	12.69	11.43	12.97	13.07	15.38	**19.37**	**12.94**	13.89	**15.25**	2.67
AVG-MLP-MLP	**12.40**	11.55	13.27	**12.89**	**14.72**	19.97	13.01	13.79	15.30	2.89
RFY-BST2	12.54	11.48	13.17	12.98	15.11	19.78	12.97	13.91	15.34	3.00
RFY-KNN	12.77	12.07	13.52	13.34	15.47	19.91	13.40	14.14	15.60	4.67
AVG-KNN-KNN	13.10	11.84	13.22	13.54	15.70	19.56	13.45	14.35	15.73	5.28
AVG-LIN-BST2	13.01	11.88	14.22	14.35	16.52	20.36	13.97	15.01	16.37	6.89
AVG-BST2-BST2	15.49	14.86	16.85	17.26	19.04	22.32	16.73	17.67	18.95	8.00
MASE										
RFY-BST2	2.56	2.51	2.91	**3.42**	4.43	6.06	3.11	3.58	**4.19**	**2.28**
AVG-MLP-MLP	**2.54**	2.49	2.92	3.43	**4.23**	6.13	3.09	**3.53**	4.21	2.56
AVG-LIN-MLP	2.54	2.48	**2.84**	3.44	4.29	6.18	**3.09**	3.53	4.22	3.06
AVG-LIN-KNN	2.60	**2.47**	2.99	3.46	4.56	**6.06**	3.15	3.62	4.23	3.22
RFY-KNN	2.63	2.62	3.14	3.46	4.40	6.26	3.25	3.66	4.31	4.72
AVG-KNN-KNN	2.77	2.57	3.14	3.57	4.68	6.15	3.28	3.75	4.35	5.39
AVG-LIN-BST2	2.71	2.52	3.27	3.86	5.06	6.32	3.38	3.96	4.56	6.78
AVG-BST2-BST2	3.37	3.23	4.02	4.67	5.81	6.89	4.12	4.72	5.30	8.00

Table 5.4: *NN5 competition. SMAPE and MASE forecast accuracy measures for rectify forecasts (RFY-KNN) and boost forecasts (RFY-BST2) as well as averaging forecasts (AVG) with LIN, KNN, MLP and BST2 models.*

Strategy	Forecast horizon (h)							Average								Avg. rank
	1	2	3	7	14	21	56	1-7	1-14	1-21	1-28	1-35	1-42	1-49	1-56	
SMAPE																
AVG-BST2-BST2	14.34	**11.60**	11.89	18.38	18.09	40.76	**16.71**	**15.99**	**17.18**	**19.99**	**19.69**	**21.95**	**22.34**	**21.86**	**21.15**	**2.07**
AVG-LIN-BST2	14.08	12.10	11.89	18.19	18.05	**40.60**	16.98	16.07	17.34	20.12	19.86	22.09	22.63	22.25	21.58	3.12
AVG-KNN-KNN	14.66	12.09	11.92	**17.80**	18.60	41.25	16.86	16.46	17.73	20.60	20.28	22.38	22.84	22.38	21.70	3.86
AVG-LIN-MLP	14.41	12.84	12.22	18.43	**17.99**	40.91	17.34	16.63	17.76	20.49	20.18	22.32	22.84	22.43	21.78	4.38
AVG-LIN-KNN	**13.94**	11.98	**11.84**	17.87	18.06	41.34	17.44	16.18	17.48	20.33	20.07	22.29	22.84	22.45	21.82	4.82
RFY-BST2	14.03	12.40	12.56	18.55	18.31	40.92	16.75	16.45	17.75	20.54	20.24	22.44	22.92	22.52	21.81	4.84
AVG-MLP-MLP	14.85	13.25	12.64	18.59	18.54	41.63	17.58	17.08	18.20	20.98	20.69	22.79	23.22	22.77	22.07	5.89
RFY-KNN	14.84	13.63	12.86	18.99	18.32	40.77	17.87	17.17	18.48	21.06	20.78	22.86	23.45	23.09	22.53	7.02
MASE																
AVG-BST2-BST2	0.28	**0.28**	0.38	0.43	0.42	0.81	**0.40**	**0.48**	**0.51**	**0.59**	**0.57**	**0.61**	**0.63**	**0.62**	**0.60**	**2.12**
AVG-LIN-BST2	0.27	0.29	0.38	0.43	**0.42**	**0.80**	0.41	0.48	0.52	0.60	0.58	0.61	0.64	0.63	0.62	3.27
AVG-KNN-KNN	0.28	0.28	0.37	**0.42**	0.43	0.83	0.41	0.49	0.53	0.61	0.59	0.62	0.65	0.64	0.62	3.77
AVG-LIN-MLP	0.28	0.30	0.39	0.43	0.42	0.81	0.42	0.50	0.53	0.61	0.58	0.62	0.65	0.64	0.62	4.39
AVG-LIN-KNN	**0.27**	0.28	**0.37**	0.42	0.42	0.82	0.42	0.49	0.53	0.60	0.58	0.62	0.65	0.64	0.62	4.80
RFY-BST2	0.27	0.29	0.39	0.43	0.43	0.82	0.41	0.49	0.53	0.61	0.59	0.62	0.65	0.64	0.62	5.11
AVG-MLP-MLP	0.28	0.30	0.40	0.43	0.43	0.83	0.42	0.50	0.54	0.62	0.60	0.63	0.66	0.64	0.63	5.54
RFY-KNN	0.29	0.31	0.41	0.44	0.43	0.81	0.43	0.51	0.56	0.62	0.60	0.64	0.67	0.66	0.65	7.00

both RFY-BST2 and AVG-LIN-BST2 seems to stem from REC-LIN which increases the errors, especially for long horizons.

- As with RFY-BST2, RFY-KNN has larger errors than both AVG-KNN-KNN and AVG-LIN-KNN that benefit from the superior performance of DIR-KNN. Again, RFY-BST2 outperforms RFY-KNN due to the reduced variance of the boosting components.
- AVG-LIN-MLP has lower errors than AVG-MLP-MLP which is penalized by the large errors of REC-MLP, as can be seen in Table 5.2

All in all, we obtain different results for the M3 and the NN5 competitions. In fact, with the M3 competition, the recursive strategy outperforms the direct strategy for all learning algorithms, and vice versa for the NN5 competition. In particular, recursive linear forecasts provide good forecasts with the M3 competition but are outperformed by other learning algorithms for the NN5 competition. Consequently, the rectify and boost strategies have also provided good forecasts for the M3 competition but poorer forecasts for the NN5 competition. In fact, the main assumption of the multi-stage strategies is that recursive linear forecasts will provide a good first approximation that only needs few adjustments. This assumption seems to be met with the M3 competition but not with the NN5 competition where direct nonlinear models provide better forecasts. One explanation can be that the M3 time series are short ($T \leq 126$) and very noisy while the NN5 time series are longer ($T = 735$) and less noisy. Also, the forecast horizon ($H = 18$ for M3 and $H = 56$ for NN5) can play an important role in the performance of the direct strategy which suffer from a loss of $h-1$ points at horizon h. Another observation is that the boost strategy always provides better forecasts than the rectify strategy with the KNN model, which can be explained by the reduced variance of the boosting components.

Finally, for the M3 competition, we have seen that the boost strategy has similar performance than the best of the averaging strategies that involves recursive linear forecasts. For the NN5 competition, the boost strategy is penalized by the fact that recursive linear forecasts provide poor forecasts, but does not increase the errors too much compared to the rectify strategy with the KNN model.

5.5 Concluding remarks

A number of studies have focused on comparing recursive and direct forecasts, and have investigated under which conditions one strategy is better than the other. The accuracy of recursive and direct forecasts depends on many interacting factors, and choosing between these two strategies is a challenging task in real-world applications since it involves finding the best trade-off between bias and estimation variance of the forecasts over the horizon.

Hybrid forecasting strategies have been proposed but they received little attention notably due to the additional complexity or the limited increase in performance compared to recursive and direct forecasts.

Rather than treating the recursive and direct strategies as competitors, we proposed a new strategy, called rectify, that seek to combine the best properties of both strategies. The main idea is to produce recursive forecasts from a linear autoregressive model, and then adjust these forecasts using direct nonlinear models in which the linear forecast errors are modelled, thus allowing nonlinearity and interactions between lagged variables at each forecast horizon. In our implementation, we considered the nearest neighbors model to estimate the rectification models.

Because recursive linear forecasts often need few adjustments at each horizon with real-world time series, we considered a second strategy, called the boost strategy, that estimates the rectification

models using gradient boosting algorithms that involve the so-called weak learners. As a result, the boost strategy allows to reduce the overfitting phenomenon with weakly nonlinear time series, but also provides a procedure for applying boosting algorithms to multi-step forecasting problems.

We have compared the rectify and boost strategies with (i) the recursive strategy, (ii) the direct strategy and (iii) averaging strategies that compute an average of recursive and direct forecasts. We considered both a bias and variance study as well as real-world experiments.

Overall, we found that the boost strategy outperforms the rectify strategy, notably due to resistance to overfitting of the gradient boosting algorithms that involve weak learners as well as the limitation to bivariate interactions. Because we found similar results when comparing the rectify and the boost strategies with the recursive and direct strategies, we only summarize the results with the boost strategy.

Compared to recursive and direct nonlinear forecasts, we found that the boost forecasts have smaller errors for both linear and nonlinear DGP due to a lower variance, especially for short horizons. In particular, for long time series, the boost forecasts have smaller errors than recursive nonlinear forecasts that suffer from the accumulation of errors.

Compared to recursive and direct linear forecasts, the boost forecasts have lower errors at short horizons when the DGP is nonlinear because of a lower bias component since they allow nonlinear interactions between lagged variables. For longer horizons, linear forecasts have lower errors since the variance component dominates the errors. When the DGP is linear, if the boost forecasts include nonlinear rectification terms, they have larger errors than recursive and direct linear forecasts.

With the time series from the M3 and NN5 competitions, we obtained different results when comparing the boost strategy with the recursive and direct strategies. With the M3 time series, the boost strategy had better forecasts than both the recursive and direct strategies. With the NN5 time series, the direct strategy had better forecasts than both the boost and the recursive strategies. The results suggest that the performance of the rectify and boost strategies is closely related to the performance of the recursive base forecasts.

Compared to the averaging strategies composed of recursive and direct nonlinear forecasts, the boost strategy has generally better forecasts, except with the NN5 time series where they benefit from the smaller errors of the direct nonlinear forecasts. When the averaging strategies include recursive linear forecasts, the rectify strategy has either comparable or poorer performance since the averaging strategies benefit from a lower variance, especially for long horizons.

The rectify and boost forecasts are thus good alternatives to recursive and direct forecasts since they avoid the problem of choosing between these two forecasts, and at the same time, often provide comparable or better forecasts than the recursive and direct strategies with both linear and nonlinear models. In particular, because the boost strategy involve weak learners, it can benefit from the resistance to overfitting of the gradient boosting algorithms.

Chapter 6

Multi-horizon forecasting strategies

This chapter is partly based on the following publications

- S Ben Taieb, G Bontempi, A Atiya, et al. (2012). A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *Expert Systems with Applications* **39**(8), 7067–7083.
- G Bontempi and S Ben Taieb (January 2011). Conditionally dependent strategies for multiple-step-ahead prediction in local learning. *International Journal of Forecasting* **27**(3), 689–699.
- S Ben Taieb, A Sorjamaa, and G Bontempi (2010). Multiple-output modeling for multi-step-ahead time series forecasting. *Neurocomputing* **73**(10-12), 1950–1957.

6.1 Introduction

We can generate multi-step forecasts using a different model at each horizon h, where each model is selected by minimizing h-step ahead forecast errors. Depending on whether the forecasts are computed recursively or directly, the strategy is called RECMULTI or DIRECT, as given in expressions (3.3.4) and (3.3.5), respectively.

For each model, we need to select a lag order as well as a set of parameters and hyperparameters depending on the model form, as can be seen in Section 2.2. Having a different model for each horizon allows a large flexibility since we can use a different lag order and a different model form at each horizon. In addition, by minimizing h-step-ahead errors, criteria of model estimation are matched with forecast accuracy, which is not the case for the recursive strategy that minimizes one-step-ahead errors, as can be seen in expression (3.3.2).

However, because the model at horizon h is selected independently from the models for the other horizons, consecutive forecasts can be based on potentially very different models, i.e. different conditioning information and different model forms. This can lead to irregularities in the forecast function. These irregularities are manifest as a contribution to the forecast variance. The problem is exacerbated with short time series and when each of the models is allowed to have different lag orders or to be nonlinear and nonparametrically estimated, as shown in Section 4.4.3. The goal of this chapter is to investigate whether we can improve multi-step forecasts generated with machine learning models that are independently selected at each horizon, by exploiting the information contained in other horizons when learning each model.

To improve multi-step forecasts generated by different machine learning models, we propose to change how each model is selected. More precisely, when selecting the model for horizon h, we also

minimize its forecast errors for other horizons rather than only minimizing the errors at horizon h. In particular, we propose to select the lag order and the hyperparameters of each model by adding the forecast errors of the model for other horizons into the objective function for the current horizon. We do not change the way the parameters are estimated and allow them to be selected independently for each horizon. The main goal is to reduce the forecast variance of independently selected models. Because the objective involves forecast errors from multiple horizons, we will call these strategies *multi-horizon* to differentiate them from the *single-horizon* strategies that consider each horizon in isolation.

Different multi-horizon strategies can be specified, with different formulation of the objective function, and this, consequently, will reflect on the way the lag order and the hyperparameters are optimized. We will first consider the multi-horizon strategies that include the forecast errors from all horizons into the objective, which is equivalent to impose all the models to use the same lag order and the same hyperparameters. This strategy is the opposite of the strategy that selects all the models independently. We will also consider a less extreme strategy where we only include local horizons into the objective. These different strategies together with their respective objective function will be presented in the next section where we will consider both recursive and direct forecasts.

All the proposed multi-horizon strategies will be evaluated and compared with the single-horizon strategies in Section 6.4 where a bias and variance analysis will be applied as in Sections 4.4 and 5.3. In addition, we will also compare the different strategies in Section 6.5 using real-world time series from the M3 and NN5 competitions as in Section 5.4. But before, in the next section, we present some related work.

6.2 Related work

Exploiting the relatedness between a set of learning tasks to improve the performance of each task has been studied in different fields.

In the statistics literature, several methods have been developed to deal with the problem of multivariate (also called *multi-response*) regression, where, instead of applying a univariate regression on each response independently, a regression is applied on all the responses at the same time. The methods include "curds and whey"(Breiman and Friedman, 1997), two-block partial least squares (PLS) (Wold, 1975; Wold, 2001), canonical correlation analysis (CCA) (Hardoon, Szedmak, and Shawe-Taylor, 2004), multivariate reduced rank regression (Izenman, 1975) and adaptive ridge regression (Brown and Zidek, 1980). Nonlinear versions of some of these methods have also been proposed in the literature (Rosipal and Trejo, 2002; Fukumizu, Bach, and Gretton, 2007; Hardoon and Shawe-Taylor, 2010).

In the machine learning literature, the problem of jointly learning multiple tasks is known as *multi-task learning* (Caruana, 1997; Evgeniou and Pontil, 2004), and has been applied for a large number of problems including classification (Xue et al., 2007), regression (Solnon, Arlot, and Bach, 2012), but also for more general dependency estimation problems (Weston et al., 2003). As in the regression setting, the rationale behind this paradigm is that when there are relations between the tasks, the learning performance can be improved by learning the different tasks simultaneously instead of considering each task independently from the other tasks. The performance for the main task can be improved since the learner can exploit the commonality among the tasks to build a better model.

The different methods that have been proposed are often either supervised dimensionality reduction methods (Wold, 2001; Hardoon, Szedmak, and Shawe-Taylor, 2004) or regularization-based methods (Evgeniou and Pontil, 2004; Tresp and Kriegel, 2006).

Among the various machine learning models, neural networks can naturally deal with multi-task learning problems by using multiple-output nodes (Ou and Murphey, 2007). For the other models, considering multiple tasks is not straightforward, and a number of extensions have been proposed for example for support vector machines (Vazquez and Walter, 2003), regression trees (De'Ath, 2002), gradient boosting (Lutz and Buhlmann, 2006) and nonparametric models (Matias, 2005).

In the forecasting literature, the problem of exploiting the relatedness between different forecasting tasks to improve the final forecasts has received little attention.

Weiss (1991) considered the problem of estimating the parameters of a misspecified linear model by minimizing the sum of the forecasts errors for several horizons. However, using Monte Carlo simulations, the author did not find superior performance of the proposed estimator compared to the simple OLS estimator.

Liu (1996) views the problem of multi-step forecasting from an autoregressive linear model as a multivariate linear regression problem and develops a new lag order selection procedure. The author shows that the proposed procedure performs more efficiently than the corresponding univariate procedure.

More recently, Xia and Tong (2011) challenged the conventional fitting methods which assume that the postulated model is the true model and where one-step-ahead errors are minimized to select the parameters. The authors show that these methods fail to capture some of the most basic global features of the data such as cycles, and develop a systematic approach that involves minimizing multi-step recursive forecast errors. However, they focused on the problem of feature matching and did not consider the impact on multi-step forecasting.

Some of the previously described methods have also been applied in the context of multi-step time series forecasting. For example, Franses and Legerstee (2009) considered the PLS method for jointly estimating direct linear models for multi-step forecasting. The authors compared the proposed method with the recursive and direct strategies on the quarterly index of US industrial production, for the period January 1945 to April 2000. They found mixed results with better forecasts for the recursive and the PLS methods compared to the direct strategy.

Applying the multi-output methods in the context of multi-step forecasting of a univariate time series is challenging notably due to the limited amount of data as well as the high level of noise of the real-world time series. Also, it is often hard to see a significant improvement with the multi-output methods on short time series.

In addition, compared to the general setting of a multi-output regression problem, the input and output variables have a specific structure in the context of multi-step forecasting. In fact, because of the temporal correlation in the time series, the different lagged variables will be autocorrelated and the autocorrelation typically decrease with the distance between the variables. Also, at horizon h, we lose h observations for the corresponding regression task, which can become important for short time series and long horizons.

In contrast to the different methods presented in this section, the strategies we proposed do not involve any dimensionality reduction problem. In fact, our approach consists in adding the forecast errors of some other horizons into the objective function of the model at each horizon to select the lag order and hyperparameters. This is in contrast to the traditional approach that considers each horizon in isolation.

A related approach has been considered in Kline (2004) and Jin and Sun (2008) with multi-output neural networks where each output node corresponds to one forecast horizon. With this approach, the lag order, the hyperparameters as well as the parameters are selected jointly. However, in our approach, we allow the parameters to be independently selected for each horizon, and only select the lag order and hyperparameters jointly for some horizons depending on the strategy. By doing so, we allow a larger flexibility for the strategies and the selection procedure is not model dependent, which makes it applicable to any model.

6.3 The multi-horizon strategies

In Chapter 3, we have presented two single-horizon strategies that select a different model for each horizon, depending on how the forecasts are generated.

With recursive forecasts, we have presented the RECMULTI strategy that selects the lag order and hyperparameters as follows

$$(p_h, \hat{\boldsymbol{\psi}}_h) = \underset{p,\boldsymbol{\psi}}{\operatorname{argmin}} \sum_{(\boldsymbol{r}_{t-h}, y_t)\in\mathcal{D}_{\text{val},h}} [y_t - m^{(h)}(\boldsymbol{r}_{t-h}; \boldsymbol{\psi}, \hat{\boldsymbol{\beta}}_h)]^2. \tag{6.3.1}$$

With direct forecasts, we have presented the DIRECT strategy which selects the lag order and hyperparameters as follows

$$(p_h, \hat{\boldsymbol{\psi}}_h) = \underset{p,\boldsymbol{\psi}}{\operatorname{argmin}} \sum_{(\boldsymbol{r}_{t-h}, y_t)\in\mathcal{D}_{\text{val},h}} [y_t - m_h(\boldsymbol{r}_{t-h}; \boldsymbol{\psi}, \hat{\boldsymbol{\beta}}_h)]^2. \tag{6.3.2}$$

In other words, both strategies minimize h-step ahead errors to select the lag order p_h and the hyperparameters $\boldsymbol{\psi}_h$ for each horizon h. In particular, we can see in expressions (6.3.1) and (6.3.2) that the single-horizon strategies select both the lag order and the hyperparameters of the models independently for each horizon h. By doing so, these strategies disregard potential substantial information contained in the other horizons when selecting the model for each horizon h.

In the following, we present multi-horizon strategies that take advantage of that information when selecting the lag order and the hyperparameters by adding the forecast errors of the current model for other horizons into the objective function for the current horizon. The parameters, however, are still independently selected for each horizon. To allow an easier comparison with single-horizon strategies, we will show how the lag order and the hyperparameters are selected by each multi-horizon strategy for each horizon.

The simplest multi-horizon strategies adds the forecast errors from all horizons into the objective function, and the same lag order and hyperparameters are used for all horizons.

With recursive forecasts, the parameters are selected by minimizing the sum of the h'-step ahead recursive errors over all forecast horizons, i.e. from $h' = 1$ to $h' = H$, as follows:

$$(p_h, \hat{\boldsymbol{\psi}}_h) = \underset{p,\boldsymbol{\psi}}{\operatorname{argmin}} \sum_{h'=1}^{H} \sum_{(\boldsymbol{r}_{t-h'}, y_t)\in\mathcal{D}_{\text{val},h'}} \left[y_t - m^{(h')}(\boldsymbol{r}_{t-h'}; \boldsymbol{\psi}, \hat{\boldsymbol{\beta}}_{h'})\right]^2. \tag{6.3.3}$$

where $h = 1, \ldots, H$. Because the objective is the same for each horizon h, we have $p_h = p$ and $\hat{\boldsymbol{\psi}}_h = \hat{\boldsymbol{\psi}}$ for all horizons h. However, the parameters are allowed to be selected independently, that is we can have $\hat{\boldsymbol{\beta}}_{h_1} \neq \hat{\boldsymbol{\beta}}_{h_2}$ for $h_1 \neq h_2$. We will denote this strategy by RECJOINT (RJT).

To see the difference with RECMULTI, we can also write expression (6.3.3) as follows:

$$(p_h, \hat{\boldsymbol{\psi}}_h) = \underset{p,\boldsymbol{\psi}}{\operatorname{argmin}} \underbrace{\sum_{(\boldsymbol{r}_{t-h}, y_t) \in \mathcal{D}_{\text{val},h}} [y_t - m^{(h)}(\boldsymbol{r}_{t-h}; \boldsymbol{\psi}, \hat{\boldsymbol{\beta}}_h)]^2}_{\text{Objective of RECMULTI}} + \underbrace{\sum_{h' \in \{1,\dots,H\} \setminus \{h\}} \sum_{(\boldsymbol{r}_{t-h'}, y_t) \in \mathcal{D}_{\text{val},h'}} \left[y_t - m^{(h')}(\boldsymbol{r}_{t-h'}; \boldsymbol{\psi}, \hat{\boldsymbol{\beta}}_{h'})\right]^2}_{\text{Extra information}},$$

where the first part is the objective of RECMULTI, given in (6.3.1), and the second part contains extra information from other horizons.

To see the difference with the recursive strategy, we can also write expression (6.3.3) as follows:

$$(p_h, \hat{\boldsymbol{\psi}}_h) = \underset{p,\boldsymbol{\psi}}{\operatorname{argmin}} \underbrace{\sum_{(\boldsymbol{r}_{t-h}, y_t) \in \mathcal{D}_{\text{val}}} [y_t - m^{(1)}(\boldsymbol{r}_{t-1}; \boldsymbol{\psi}, \hat{\boldsymbol{\beta}}_1)]^2}_{\text{Objective of REC}} + \underbrace{\sum_{h' \in \{2,\dots,H\}} \sum_{(\boldsymbol{r}_{t-h'}, y_t) \in \mathcal{D}_{\text{val},h'}} \left[y_t - m^{(h')}(\boldsymbol{r}_{t-h'}; \boldsymbol{\psi}, \hat{\boldsymbol{\beta}}_{h'})\right]^2}_{\text{Extra information}}, \tag{6.3.4}$$

where the first part is the commonly used objective of the recursive strategy, given in expression (3.3.2), and the second part contains extra information from other horizons. We can see that the lag order and the hyperparameters are optimized taking into account their whole effect on all future steps, rather than the myopic one-step-ahead view of the recursive strategy.

A variant of the expression (6.3.3) consists in minimizing M-step-ahead errors instead of H-step ahead errors with $M \neq H$, as follows:

$$(p_h, \hat{\boldsymbol{\psi}}_h) = \underset{p,\boldsymbol{\psi}}{\operatorname{argmin}} \sum_{h'=1}^{M} \sum_{(\boldsymbol{r}_{t-h'}, y_t) \in \mathcal{D}_{\text{val},h'}} \left[y_t - m^{(h')}(\boldsymbol{r}_{t-h'}; \boldsymbol{\psi}, \hat{\boldsymbol{\beta}}_{h'})\right]^2. \tag{6.3.5}$$

In other words, the selection of the lag order and the hyperparameters does not depend on the forecast horizon H. This variant has been considered with $M = 2$ in Bontempi, Birattari, and Bersini (1999) and more generally in Xia and Tong (2011). In particular, when $M = 1$ this strategy reduces to the recursive strategy. We will denote this strategy RECJOINTM (RJTM)[1].

Minimizing multi-step recursive errors with linear models is motivated by the fact that the loss of forecasting performance when using an incorrect model with parameters tuned for multi-step forecasts is expected to be smaller than parameters tuned for one-step-ahead forecasts. Traditionally, parameters have been selected by minimizing one-step-ahead errors given that it leads to efficient parameter estimation if the model is well-specified. However, the Box dictum says *"Essentially, all models are wrong, but some are useful"* which means that all the postulated models are misspecified in practice. So, aiming at efficient estimation of the parameters by minimizing one-step-ahead forecast errors will not necessary lead to better forecasts. Xia and Tong (2011) give a small adjustment to the Box dictum that reflects this idea: *"All models are wrong, but some are useful if they are fitted properly"*.

It is worth noting that minimizing multi-step-ahead errors with recursive forecasts is more challenging that minimizing one-step-ahead errors. In fact, the procedure for parameter estimation must take into account the iterative nature of the recursive forecasts. For example, with the MLP model, we need to consider recurrent neural networks with backpropagation through time (Werbos, 1990). In other words, it will depends on the considered model form.

To make it applicable for any model, we considered an alternative where we have replaced $\hat{\boldsymbol{\beta}}_{h'}$ by $\hat{\boldsymbol{\beta}}_1$ in the different objectives. In other words, we select the parameters by minimizing one-step-ahead

[1] Note that RECJOINT1 and REC are equivalent, as well as RECJOINTH and RECJOINT.

errors but the lag order and the hyperparameters are selected by minimizing multi-step-ahead errors. When the model is nonparametric (i.e. $\boldsymbol{\beta} = \emptyset$) as with the KNN model, the problem disappears since we don't have any parameter to estimate.

As with recursive forecasts, we can also consider multi-horizon strategies with direct forecasts. The simplest multi-horizon strategy minimizes the sum of the direct forecast errors over the entire horizon as follows

$$(p_h, \hat{\boldsymbol{\psi}}_h) = \underset{p,\boldsymbol{\psi}}{\operatorname{argmin}} \sum_{h'=1}^{H} \sum_{(\boldsymbol{r}_{t-h'}, y_t) \in \mathcal{D}_{\text{val},h'}} \left[y_t - m_{h'}(\boldsymbol{r}_{t-h'}; \boldsymbol{\psi}, \hat{\boldsymbol{\beta}}_{h'}) \right]^2, \tag{6.3.6}$$

where $h = 1, \ldots, H$. We will denote this strategy DIRJOINT (DJT).

To see the difference with the objective of DIRECT, we can also write expression (6.3.6) as follows

$$(p_h, \hat{\boldsymbol{\psi}}_h) = \underset{p,\boldsymbol{\psi}}{\operatorname{argmin}} \underbrace{\sum_{(\boldsymbol{r}_{t-h}, y_t) \in \mathcal{D}_{\text{val},h}} [y_t - m_h(\boldsymbol{r}_{t-h}; \boldsymbol{\psi}, \hat{\boldsymbol{\beta}}_h)]^2}_{\text{Objective of DIRECT}} + \underbrace{\sum_{h' \in \{1,\ldots,H\} \setminus \{h\}} \sum_{(\boldsymbol{r}_{t-h'}, y_t) \in \mathcal{D}_{\text{val},h'}} \left[y_t - m_{h'}(\boldsymbol{r}_{t-h'}; \boldsymbol{\psi}, \hat{\boldsymbol{\beta}}_{h'}) \right]^2}_{\text{Extra information}},$$

where the first part is the objective of DIRECT, given in (6.3.2), and the second part contains extra information from the other horizons.

The main difference between expression (6.3.3) for RECJOINT and expression (6.3.6) for DIRJOINT is the way the forecasts are generated. In the former, the forecasts are generated recursively while in the latter, the forecasts are generated directly. Finally, although the same lag order and the same set of hyperparameters are used for all horizons, the forecasts at each horizon can be different since they are computed either with different parameters (in the parametric case) or different data (in the nonparametric case).

Let us compare RECJOINT and DIRJOINT (the JOINT strategies) with RECMULTI and DIRECT (the SINGLE strategies). By comparing expressions (6.3.3) and (6.3.6) with (6.3.1) and (6.3.2), we can see that the total error is computed using more residuals in expressions (6.3.3) and (6.3.6). In fact, with the SINGLE strategies the number of residuals at horizon h for a time series with T observations is $N = T - p_h - h + 1$ where p_h is the lag order for horizon h. For the JOINT strategies, it is given by $N = \sum_{h=1}^{H} T - p - h + 1 = H(T-p) - \frac{H(H-1)}{2}$ where p is the lag order for all horizons. For example, if $T = 120$, $p_h = p = 12$ and $H = 12$, the SINGLE strategies will have $N = (109 - h)$ residuals at horizon h while the JOINT strategies will have $N = 1230$ residuals. It is worth noting that although $N = 1230$ is much higher than $N = (109 - h)$, the effective number of residuals is much smaller since these residuals are serially correlated.

In addition, for the data points $(\boldsymbol{r}_t, y_{t+h})$ where $t > T - p + h - 1$, the corresponding output y_{t+h} will be missing for some of the horizons $h = 1, \ldots, H$. In consequence, for some data points, the sum over all horizons (in (6.3.3) and (6.3.6)) cannot be computed and as a result, these data points must be removed from the dataset. For time series with a large number of observations, the gain from a larger number of residuals for the remaining points is expected to be more important than the effect of removing some points. But, for short time series, reducing the dataset can have a detrimental effect even if the remaining points have multiple residuals. Finally, summing over a subset of the horizons is not appropriate since the total error is expected to be smaller than the sum over all the horizons.

The aim of the JOINT strategies is to exploit the information contained in *all* horizons when selecting the lag order and the hyperparameters of the model for horizon h. We have seen that the number of residuals is largely increased compared to SINGLE strategies. However, because

the forecast horizon H is arbitrarily set in real-world applications, the JOINT strategies can have a problem with including residuals that are weakly correlated with the residuals at horizon h into the objective of the model at horizon h. In fact, forecast errors at horizon h are typically less correlated with the errors at horizon h' when the distance between the horizon increases, i.e. when the quantity $|h-h'|$ increases. Including "non-informative" horizons into the objective of the model at horizon h can have a negative effect on the model selection procedure. In particular, since multi-step forecast errors can have different scales with errors at longer horizons being generally larger than errors at short horizons, some horizons can dominate others by implicitly having more weight in the objective function. And if these horizons are beyond the predictability horizon, they will have more weights while bringing no benefit to horizon h.

By comparing expressions (6.3.1) and (6.3.2) with (6.3.3) and (6.3.6), we can see that they are two extremes where in the former case, we only include the horizon of interest into the objective while in the later case, we include all the horizons.

We can define a more general strategy that associates to each horizon h a set $L_h \subset \{1,\dots,H\}$ that contains all the horizons involved in the objective function of that horizon. The strategy is then defined by the set $L = \{L_1,\dots,L_h\}$ which contains the horizons included in the objective function for all horizons h where $h = 1,\dots,H$.

When the forecasts are generated recursively, the general strategy computes the lag order and the hyperparameters as follows

$$(p_h, \hat{\boldsymbol{\psi}}_h) = \underset{p,\boldsymbol{\psi}}{\operatorname{argmin}} \sum_{h' \in L_h} \sum_{(\boldsymbol{r}_{t-h'}, y_t) \in \mathcal{D}_{\text{val},h'}} \left[y_t - m^{(h')}(\boldsymbol{r}_{t-h'}; \boldsymbol{\psi}, \hat{\boldsymbol{\beta}}_{h'}) \right]^2, \tag{6.3.7}$$

where $L_h \subset \{1,\dots,H\}$.

If $L_h = \{h\}$, expression (6.3.7) becomes equivalent to expression (6.3.1), which corresponds to the objective of RECMULTI. Similarly, if $L_h = \{1,\dots,H\}$, then expression (6.3.7) becomes equivalent to expression (6.3.3), which corresponds to the objective of RECJOINT. In addition, if $L_h = \{1\}$, this is equivalent to the objective of the RECURSIVE strategy, given in (3.3.2).

With direct forecasts, the lag order and the hyperparameters are computed as follows

$$(p_h, \hat{\boldsymbol{\psi}}_h) = \underset{p,\boldsymbol{\psi}}{\operatorname{argmin}} \sum_{h' \in L_h} \sum_{(\boldsymbol{r}_{t-h'}, y_t) \in \mathcal{D}_{\text{val},h'}} \left[y_t - m_{h'}(\boldsymbol{r}_{t-h'}; \boldsymbol{\psi}, \hat{\boldsymbol{\beta}}_{h'}) \right]^2. \tag{6.3.8}$$

The objective of DIRECT given in (6.3.2) can be retrieved by using $L_h = \{h\}$ in (6.3.8). Similarly, the objective of DIRJOINT given in (6.3.6) is retrieved with $L_h = \{1,\dots,H\}$.

There are a large number of possible sets $L = \{L_1,\dots,L_H\}$ each of which use a different formulation of the objective function, and this, consequently, reflects on the way the lag orders and the hyperparameters are optimized. Among these strategies, we have the SINGLE and the JOINT strategies where $L_h = \{h\}$ and $L_h = \{1,\dots,H\}$, respectively.

Previously, we have seen that the selection of the lag order and the hyperparameters with the JOINT strategies depends on the forecast errors for all the horizons while with the SINGLE strategies this selection is made independently from other horizons. An alternative to these two strategies consist in considering "local" information, that is horizons that are close to the horizon of interest. Two horizons are considered close if $|h-h'|$ is small. In other words, we can define the set $L_h = \{h-s_1,\dots,h,\dots,h+s_2\}$ where s_1 and s_2 are, respectively, the number of horizons before and after the horizon h, that are considered. This strategy will be denoted RJTLs_1s_2 and DJTLs_1s_2 for recursive and direct forecasts, respectively.

Table 6.1: *Summary of single-horizon and multi-horizon strategies.*

Recursive strategies	Direct strategies	L_h	Weights $w_{h,h'}$
REC (3.3.2)	-	$\{1\}$	$\frac{\mathbb{1}_{\{h' \in L_h\}}}{\lvert L_h \rvert}$
RJTM (6.3.5)	-	$\{1,\dots,M\}$	
RTI (6.3.1)	DIR (6.3.2)	$\{h\}$	
RJT$s_1 s_2$ (6.3.7)	DJT$s_1 s_2$ (6.3.8)	$\{h-s_1,\dots,h,\dots,h+s_2\}$	
RJT (6.3.3)	DJT (6.3.6)	$\{1,\dots,H\}$	

For the SINGLE strategies, we have $s_1 = 0$ and $s_2 = 0$ while for the JOINT strategies, we have $s_1 = h-1$ and $s_2 = H-h$. Intermediate values for s_1 and s_2 allow to have a better trade-off between the SINGLE and the JOINT strategies. In particular, we will consider the configuration where $s_1 = 1$ and $s_2 = 1$, that is $L_h = \{h-1, h, h+1\}$, which means we add the previous and the next horizons into the objective function for horizon h. Of course, when the horizons are not defined ($h \notin \{1,\dots,H\}$) such as in the borders, they are removed from the set L_h. This strategy is denoted RJTL11 and DJTL11 with recursive and direct forecasts, respectively.

With the different strategies we have presented, the same weight is given to all forecast horizons and the errors are limited to H-step ahead forecasts. It is possible to extend these strategies to allow different weights for each horizon and to compute errors beyond the horizon H. In fact, we can write the selection of the lag order and the hyperparameters for the model at horizon h as a general optimization problem as follows

$$(p_h, \hat{\boldsymbol{\psi}}_h) = \underset{p,\boldsymbol{\psi}}{\operatorname{argmin}} \sum_{h'=1}^{H'} \sum_{(\boldsymbol{r}_{t-h'}, y_t) \in \mathcal{D}_{\text{val},h'}} w_{h,h'} \left[y_t - g(\boldsymbol{x}_{t-h'}; \boldsymbol{\psi}, \hat{\boldsymbol{\beta}}_{h'}) \right]^2, \tag{6.3.9}$$

where $H' \in \mathbb{N}$ can take an arbitrarily value (different from H), $w_{h,h'}$ is the weight given to horizon h' when selecting the lag order and hyperparameters for horizon h with $w_{h,h'} \geq 0$ and $\sum_{h'} w_{h,h'} = 1$, $g(\boldsymbol{r}_{t-h}; \boldsymbol{\psi}, \hat{\boldsymbol{\beta}}_{h'})$ is the h-step ahead forecast which can be either generated recursively and written as $m^{(h)}(\boldsymbol{r}_{t-h}; \boldsymbol{\psi}, \hat{\boldsymbol{\beta}}_h)$ or directly and written as $m_h(\boldsymbol{r}_{t-h}; \boldsymbol{\psi}, \hat{\boldsymbol{\beta}}_h)$, and $h = 1\dots, H$.

For all the strategies presented previously, the weights $w_{h,h'}$ take binary value, that is an horizon is either included or removed from the objective. Table 6.1 gives the value of the weights for all horizons to represent the different strategies presented above. Of course, to represent the recursive strategies, $g(\boldsymbol{r}_{t-h}; \boldsymbol{\psi}, \hat{\boldsymbol{\beta}}_{h'})$ is replaced by $m_h(\boldsymbol{r}_{t-h}; \boldsymbol{\psi}$ and by $\hat{\boldsymbol{\beta}}_h)$ for direct strategies.

The general optimization problem in (6.3.9) includes many more strategies than the one we have considered. In particular, we can define strategies that have weights $w_{h,h'} \neq 0$ for $h' > H$, that is strategies that look for information beyond the required forecast horizon H. Also, we can have $w_{h,h'} \in [0,1]$ instead of the binary case we considered where $w_{h,h'} \in \{0,1\}$. In this thesis, we will limit our analysis to the strategies that have been presented above and leave for future work the analysis of the general formulation given in (6.3.9).

6.3.1 Implementation

In expressions (6.3.7) and (6.3.8), we can see that the objective of all strategies involve computing h-step ahead forecast errors for different lag order and hyperparameters.

In order to avoid computing those multiple times for each strategy, we propose to compute the h-step ahead recursive or direct forecast errors for all lag orders, hyperparameters and forecast horizons, and store the results in a large error matrix. Then, for a given forecasting strategy, we use

the corresponding error matrix to select the lag order and hyperparameters for each horizon. By doing so, we can significantly reduce the computational time.

Algorithm 4 gives the different steps to compute the multi-step forecast error matrix. The errors are computed by holdout but the code can easily be extended to time series cross-validation. Also, computing the complete error matrix E is not necessary for the recursive strategy since we only use the information for $h = 1$. Algorithm 4 can be easily modified to limit the computations only for specific horizons.

Algorithm 4 Multi-step forecast error matrix.

$\{y_1, \ldots, y_T\}$: Time series with T observations.
H: Forecast horizon.
$p_{\max}$: Maximum lag order.
$m(\cdot; \boldsymbol{\psi}, \boldsymbol{\beta})$: Learning model with hyperparameters $\boldsymbol{\psi}$ and parameters $\boldsymbol{\beta}$ where $\boldsymbol{\psi} = [\psi_1, \ldots, \psi_S]$ and $\psi_s \in \Psi_s (s = 1, \ldots, S)$.
$\boldsymbol{E}$: Error matrix with dimensionality $p_{\max} \times |\Psi_1| \times \cdots \times |\Psi_S| \times H$.

1: **for** $h \leftarrow 1, \ldots, H$ **do**
2: The dataset for horizon h is $\mathcal{D} = \{(\boldsymbol{x}_{t-h}, y_t)\}$ where $\boldsymbol{x}_{t-h} = [y_{t-h}, \ldots, y_{t-h-p_{\max}}]'$.
3: Divide the dataset $\mathcal{D}$ into a training set $\mathcal{D}_{\text{train},h}$ and a validation set $\mathcal{D}_{\text{valid},h}$.
4: **for** $p \leftarrow 1, \ldots, p_{\max}$ **do**
5: **for** $\psi_1 \in \Psi_1$ **do**
⋮
6: **for** $\psi_S \in \Psi_S$ **do**
7: Using p lagged variables and the hyperparameters $\boldsymbol{\psi} = [\psi_1, \ldots, \psi_S]$, estimate the parameters $\boldsymbol{\beta}$ of the model m using the training set $\mathcal{D}_{\text{train},h}$.
8: Compute the h-step ahead forecasts $g(\boldsymbol{x}_{t-h}; \boldsymbol{\psi}, \hat{\boldsymbol{\beta}}; h)$ for all the validation points $\boldsymbol{x}_{t-h}$ in $\mathcal{D}_{\text{valid},h}$.
9: Compute the average validation errors and store it into the error matrix: $\boldsymbol{E}[p, \psi_1, \ldots, \psi_S, h] \leftarrow \frac{1}{|\mathcal{D}_{\text{valid},h}|} \sum_{(\boldsymbol{x}_{t-h}, y_t) \in \mathcal{D}_{\text{valid},h}} [y_t - g(\boldsymbol{x}_{t-h}; \boldsymbol{\psi}, \hat{\boldsymbol{\beta}}; h)]^2$.
10: **end for**
⋮
11: **end for**
12: **end for**
13: **end for**
14: Return the error matrix $\boldsymbol{E}$.

For the MLP model, the parameters $\boldsymbol{\beta}$ in line 7 are the weights on the edges between the different nodes of the networks. Also, the MLP model has $S = 2$ hyperparameters $\psi_1 = NH$, the number of hidden nodes, and $\psi_2 = \lambda$, the weight decay hyperparameter. For the nonparametric KNN model, there are no parameters, i.e. $\boldsymbol{\beta} = \emptyset$, and $S = 1$ hyperparameter $\psi_1 = K$, the number of neighbors.

Algorithm 5 shows how to compute multi-step forecasts for a given strategy using the error matrix computed with Algorithm 4.

In the next section, we apply a bias and variance analysis to compare the multi-horizon strategies with the single-horizon strategies.

Algorithm 5 Multi-step forecasts using the error matrix computed with Algorithm 4.

$\{y_1,\dots,y_T\}$: Time series with T observations.
H: Forecast horizon.
$L=\{L_1,\dots,L_H\}$: Set of horizons.
$p_{\max}$: Maximum lag order.
$m(\cdot;\boldsymbol{\psi},\boldsymbol{\beta})$: Learning model with hyperparameters $\boldsymbol{\psi}$ and parameters $\boldsymbol{\beta}$ where $\boldsymbol{\psi}=[\psi_1,\dots,\psi_S]$ and $\psi_s\in\Psi_s(s=1,\dots,S)$.
E: Error matrix with dimensionality $p_{\max}\times|\Psi_1|\times\cdots\times|\Psi_S|\times H$

1: Compute the multi-step error matrix $\boldsymbol{E}$ using Algorithm 4.
2: **for** $h\leftarrow 1,\dots,H$ **do**
3: $(p_h,\hat{\boldsymbol{\psi}}_h)=\underset{p,\boldsymbol{\psi}}{\operatorname{argmin}}\sum_{h'\in L_h}\boldsymbol{E}[p,\boldsymbol{\psi},h']$.
4: Using p_h lagged variables and the hyperparameters $\hat{\boldsymbol{\psi}}_h$, estimate the parameters $\boldsymbol{\beta}_h$ from the complete data set $\mathcal{D}=\{(\boldsymbol{x}_{t-h},y_t)\}$ to obtain $g(\boldsymbol{x}_{t-h};\hat{\boldsymbol{\psi}}_h,\hat{\boldsymbol{\beta}}_h;h)$, where $\boldsymbol{x}_{t-h}=[y_{t-h},\dots,y_{t-h-p_h}]'$.
5: Compute $\hat{y}_{T+h}=g(\boldsymbol{x}_T;\hat{\boldsymbol{\psi}}_h,\hat{\boldsymbol{\beta}}_h;h)$, the forecasts for horizon h.
6: **end for**
7: Return the forecasts $\{\hat{y}_{T+1},\dots,\hat{y}_{T+H}\}$.

6.4 Bias and variance analysis

In this section, we consider the strategies that generate multi-step forecasts using a different model at each horizon and where h-step ahead errors are included in the objective function for horizon h, i.e. $h\in L_h$. We analyze these strategies using the methodology described in Section 4.3. We first briefly discuss the role of the set L_h in the bias and variance components. Then, we consider Monte Carlo simulations to effectively compute the bias and variance components for different strategies and different DGPs.

Scenario C and D: Nonlinear model

Because the strategies minimize h-step ahead errors, the expression for the h-step ahead forecasts can be written using the terminology defined in Section 4.3.1, as follows

$$\begin{aligned} &g(\boldsymbol{x}_t;\hat{\boldsymbol{\gamma}};h)\\ &=\underbrace{\mu_{t+h|t}+\delta(\boldsymbol{r}_t;\boldsymbol{\gamma})}_{m_h(\boldsymbol{r}_t;\boldsymbol{\gamma})}+\eta(\boldsymbol{r}_t;\boldsymbol{\gamma})\varepsilon_\eta, \end{aligned} \tag{6.4.1}$$

where $\boldsymbol{r}_t=[y_t,\dots,y_{t-p_h}]$ contains p_h lagged variables and $\boldsymbol{\gamma}=[\boldsymbol{\beta},\boldsymbol{\psi}]$ includes both the parameters $\boldsymbol{\beta}$ and the hyperparameters $\boldsymbol{\psi}$. Recall that the conditional expectation $\mu_{t+h|t}$ appears in the previous expression because of the h-step ahead forecast errors minimization.

The expression for the sum of the bias and variance components at horizon h for these strategies can then be written as

$$\begin{aligned} &B_h(\boldsymbol{x}_t)+V_h(\boldsymbol{x}_t)\\ &=[\mu_{t+h|t}-m_h(\boldsymbol{r}_t;\boldsymbol{\gamma})]^2 &(6.4.2)\\ &\quad+\eta(\boldsymbol{r}_t;\boldsymbol{\gamma})^2. &(6.4.3) \end{aligned}$$

Recall that the set L_h has an impact on how the lag order and the hyperparameters are optimized for the model at horizon h as can be seen in expression (6.3.8). Also, we can see in expression

(6.4.1) that both the offset term $\delta(\boldsymbol{r}_t;\boldsymbol{\gamma})$ and the variability term $\eta(\boldsymbol{r}_t;\boldsymbol{\gamma})\varepsilon_\eta$ depend on the lag order and the model parameters $\boldsymbol{\gamma}$. In consequence, the set L_h will have an impact on the offset and variability terms, that have a direct influence on the bias and variance components in expressions (6.4.2) and (6.4.3).

When $L_h = \{h\}$, which corresponds to the DIRECT strategy, the lag order and the hyperparameters are selected by minimizing the sum of the forecast errors at horizon h, as can be seen in (6.3.8). If additional horizons are included in L_h, then the sum will be computed using additional residuals from other horizons. The increase in the number of residuals is likely to reduce the variability term $\eta(\boldsymbol{r}_t;\boldsymbol{\gamma})$ in (6.4.1) since it will become less dependent on each realization, which will in turn induce a smaller variance component in (6.4.3).

However, because the model parameters for horizon h are selected by also minimizing the forecast errors for $\mu_{t+h'|t}$ with $h' \neq h$, there will be an increase in the bias component that will depend on how close $\mu_{t+h'|t}$ is to $\mu_{t+h|t}$, for all h' included in L_h. In fact, the closer $\mu_{t+h'|t}$ is to $\mu_{t+h|t}$, the better the model $m_h(\boldsymbol{r}_t;\boldsymbol{\gamma})$ will estimate $\mu_{t+h|t}$ and the smaller the bias component in (6.4.2).

In consequence, both the number of horizons in L_h and the distance between these horizons and the horizon h will have an influence on the bias and variance components at horizon h. In particular, a good bias and variance trade-off over the forecast horizon will depend on the set $L = \{L_1, \ldots, L_H\}$.

The best strategy will include horizons where the conditional expectation is not too far from the one at horizon h to limit the increase in bias, and will include enough horizons to decrease the variance. Finding the best trade-off between bias and variance components will ensure the smallest MSE.

Ideally, the set L_h should be selected automatically from the data for each horizon. Achieving that goal with validation methods will require an additional validation set which might not be available with short time series. We leave the automatic selection of the horizons to include in L_h for future work.

We now consider Monte Carlo simulations to effectively compute the bias and variance components, as explained in Section 4.3.2.

We will consider the different strategies in Table 6.1. More precisely, we will consider REC, RJT2, RTI, RJTL11 and RJT for recursive forecasts, and DIR, DJTL11 and DJT for direct forecasts.

The forecasts for the different strategies will be generated with the MLP and the KNN models, as in Algorithm 5, for both the AR and STAR DGPs. We will not present the results of the NAR DGP since they are similar to those of the AR DGP.

Let us first consider the AR DGP. The results with direct forecasts are given in Figures 6.1 and 6.2 for the MLP model and the KNN model, respectively.

In both Figures 6.1 and 6.2, we can see in the third column that DJT has a smaller variance than DIR consistently over the horizon. By comparing Figure 6.2 with Figure 6.1, we can see that DIR-KNN produce forecasts with a smaller variance than DIR-MLP. Consequently, the relative decrease in variance for DJT-MLP compared to DIR-MLP is larger than DJT-KNN compared to DIR-KNN. Also, for $T = 400$, DIR-KNN and DJT-KNN have a comparable variance component while DJT-MLP has a smaller variance component than DIR-MLP.

The decrease in variance for the DJT strategies has also induced an increase in the bias component for short horizons as can be seen in the second column of both Figures 6.1 and 6.2. This confirms the fact that the DJT strategies put implicitly more weights on long horizons when selecting the lag order and the hyperparameters. Also, by comparing the second column of both Figures 6.1 and 6.2, we can see that the MLP model has a smaller bias component than the KNN model for all T. This

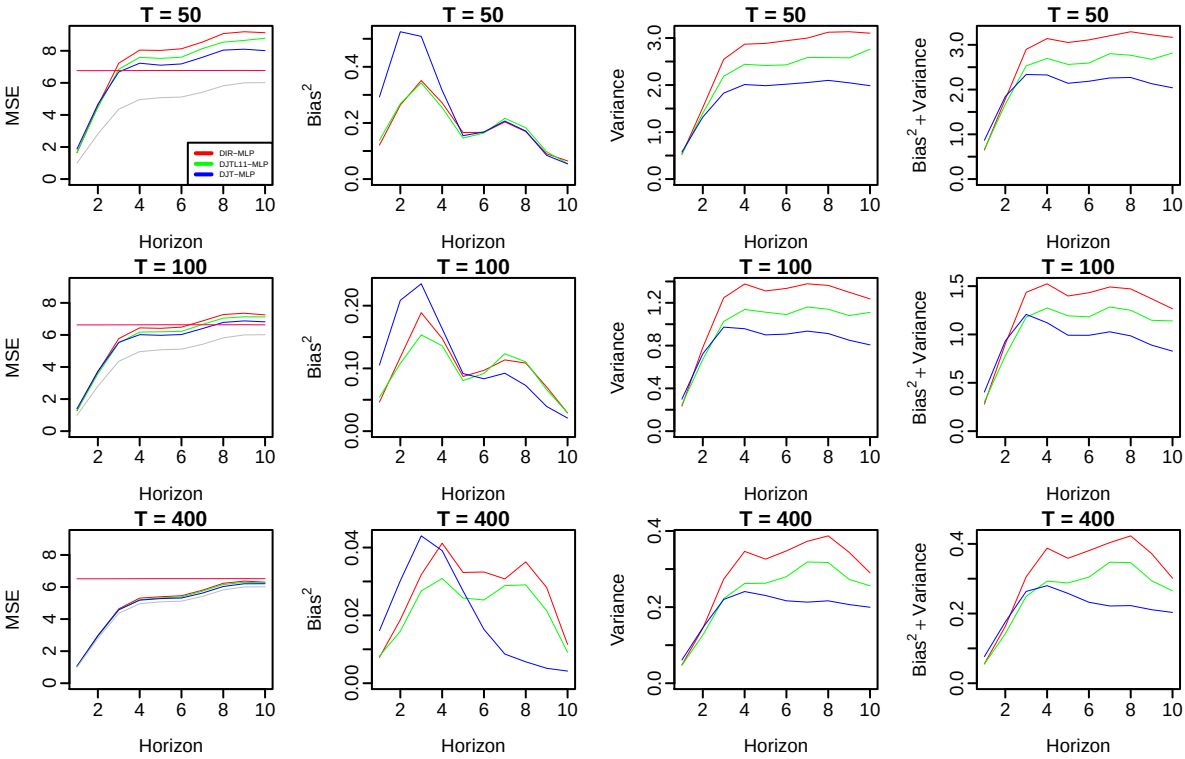

Figure 6.1: *AR DGP. MSE decomposition of direct multi-horizon strategies with the MLP model.*

can be explained by the fact that the MLP model includes the LIN model as a special case which is not the case for the KNN model.

The last column of Figure 6.1 shows that DJT-MLP has a comparable performance to DIR-MLP for the first horizons and smaller errors for longer horizons, due to the reduced variance for these horizons. In Figure 6.2, we can see that the KNN model has larger errors for short horizons (particularly noticeable for $T = 400$) and smaller or similar errors for longer horizons. Compared to the MLP model, the error reduction is smaller with the KNN model.

If we compare DJTL11 and DJT with DIR in the last column of Figures 6.1 and 6.2, we can see that DJTL11 does not suffer from the higher errors for short horizons as with DJT, and at the same time, reduces the errors compared to DIR for longer horizons, but less than DJT. The second columns show that DJTL11 does not suffer from a higher bias component as with DJT, while in the third column, we can see that it has a smaller variance than DIR and a larger variance than DJT.

Because DJTL11 uses the set $L_h = \{h-1, h, h+1\}$, the increase in bias is limited since $\mu_{t+h-1|t}$ and $\mu_{t+h+1|t}$ is not expected to change too much compared to $\mu_{t+h|t}$. In addition, the additional residuals from these horizons can reduce the variance. This suggests that DJTL11 provides a good trade-off between DIR and DJT. Again the error reduction from DIR to DJT is smaller for the KNN model compared to the MLP model which has a larger variance.

The results with recursive forecasts are given in Figures 6.3 and 6.4 for the MLP and the KNN model, respectively.

Recall that with recursive forecasts, if the model is parametric (as with the MLP model), the parameters are estimated by minimizing one-step-ahead errors and do not depend on the set $L = \{L_1, \ldots, L_H\}$. However, the selection of the lag order and the hyperparameters always depends on the set L.

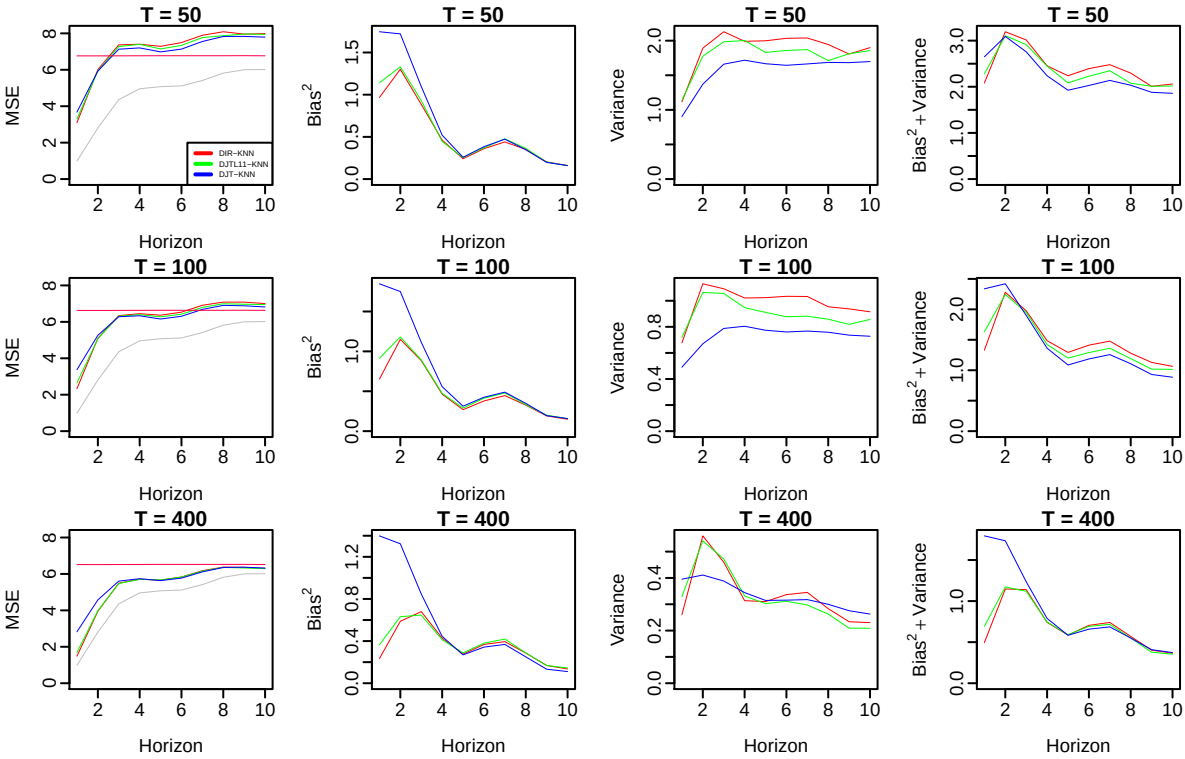

Figure 6.2: *AR DGP. MSE decomposition of direct multi-horizon strategies with the KNN model.*

We can make the same observations about RTI, RJTL11 and RJT in Figures 6.3 and 6.4 (where forecasts are generated recursively) as DIR, DJTL11 and DJT in Figures 6.1 and 6.2 (where forecasts are generated directly).

In fact, in the last column of Figure 6.3, we can see that RJT-MLP has lower errors than RTI-MLP at long horizons and similar error for short horizons. In Figure 6.4, the higher bias of RJT-KNN compared to RTI-KNN is particularly noticeable for $T = 100$ and $T = 400$.

In the last column of Figure 6.3, we can clearly see that RJTL11-MLP is between RTI-MLP and RJT-MLP, but it is less visible for RJTL11-KNN in Figure 6.4.

For the MLP model, we can see in Figure 6.3 that RJT2-MLP has comparable performance to RJT-MLP (or equivalently RJT10-MLP), and both have smaller errors than RTI-MLP. This can be explained by the fact that if the MLP model reduces to the LIN model, then the model is well-specified and consequently, one-step-ahead minimization can provide better parameter estimate than multi-step-ahead or multi-horizon minimization. The good performance of REC-MLP (equivalently RJT1-MLP) can be seen in Figure 4.6.

In contrast to the MLP model, RJT2-KNN has higher errors than the other recursive strategies as can be seen in the last column of Figure 6.4. This can be explained by the fact that recursive forecasts with the KNN model for the AR DGP suffer from a higher variance as can be seen in the third column but also in Figure 4.8 for REC-KNN (equivalently RJT1-KNN). In this case, RTI-KNN reduces the variance component and provides smaller errors.

We now consider the nonlinear STAR DGP. In contrast to the linear AR DGP, the conditional expectation is expected to change more over the horizon since the DGP is nonlinear.

The results with direct forecasts are given in Figures 6.5 and 6.6 for the MLP and the KNN model, respectively.

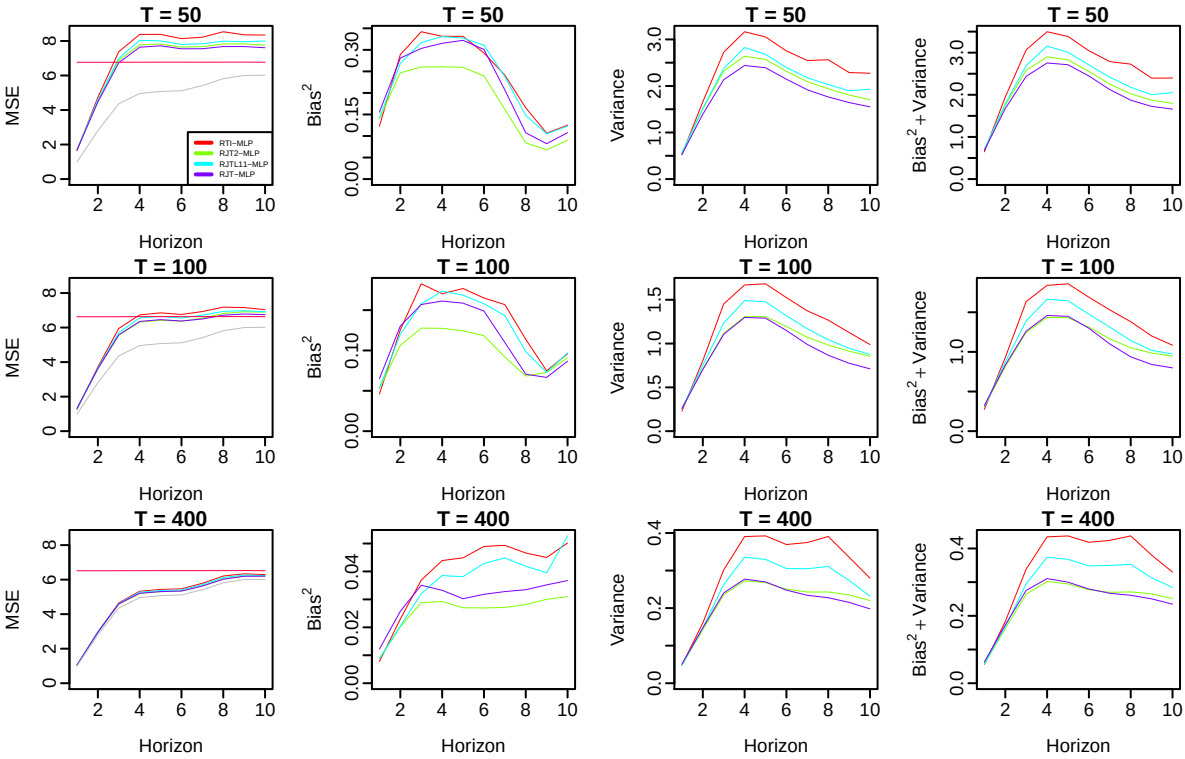

Figure 6.3: *AR DGP. MSE decomposition of recursive multi-horizon strategies with the MLP model.*

As with the AR DGP, we can see in Figures 6.5 and 6.6 the higher bias of the DJT strategies compared to the DIR strategies. However, the relative decrease in variance is less pronounced than with the AR DGP. Finally, the difference between DJTL11 and DIR is small and is hard to see in the last columns of Figures 6.5 and 6.6. The difference between DJTL11 and DIR is more noticeable in the third column where we can see the smaller variance of DJTL11 for long horizons.

The results with recursive forecasts are given in Figures 6.7 and 6.8 for the MLP and the KNN model, respectively.

Because both the model and the DGP are nonlinear, we know that recursive forecasts suffer from the amplification of errors, as can be seen in Figure 4.9 for both REC-KNN and REC-MLP.

In the last column of Figures 6.7 and 6.8, we can see that the errors of RJT2 have a similar behavior to the REC (or equivalently RJT1) strategy in Figure 4.9. This suggests that the parameters selected by minimizing two-step ahead errors are close to the one selected by minimizing one-step-ahead errors.

If we compare RJT2 with RTI and RJT for $T = 400$, we can see that the amplification of errors is reduced since both strategies include the horizon h in their objective function. Also, if we compare RTI with RJT, we can again see that RJT has higher errors at short horizons due to its higher bias component. Finally, as with the AR DGP, it is hard to see the difference between RTI and RJTL11.

We can summarize the results of the bias and variance analysis as follows:

- Compared to the SINGLE strategies, the JOINT strategies suffer from a higher bias at short horizons, but have a lower variance at long horizons.

  The strategies that only include the previous and the next horizons into the objective for the model at horizon h do not have a higher bias at short horizons compared to SINGLE strategies

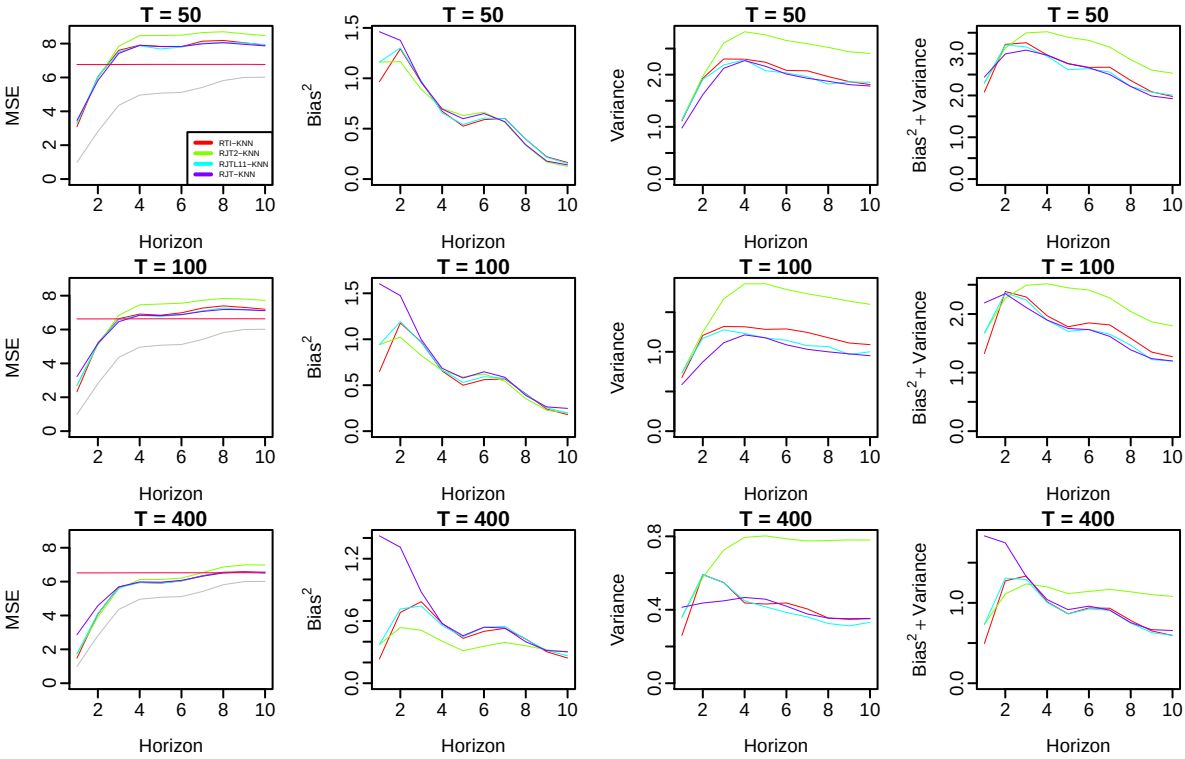

Figure 6.4: *AR DGP. MSE decomposition of recursive multi-horizon strategies with the KNN model.*

as with the JOINT strategies. Also, they reduce the variance for long horizons compared to the SINGLE strategies but less than the JOINT strategies.

- We observe a higher benefit from multi-horizon strategies when the forecasts are generated directly rather than recursively.
- Concerning the time series length T, the benefit from using multi-horizon strategies instead of single-horizon strategies is more pronounced with $T = 100$ rather than $T = 50$ and $T = 400$. Also, multi-horizon strategies tend to have lower errors for long horizons compared to short horizons.
- The difference in performance between multi-horizon and single-horizon strategies is more important with the MLP model than with the KNN model, which can be explained by the larger variance of the MLP model. This difference is also more important with the AR DGP compared the STAR DGP, since the conditional expectation is more stable over the horizon with the AR DGP.

6.5 Real-data experiments

As in Section 5.4, we carried out some experiments with the time series from the M3 and NN5 competitions. Details about the data and the methodology are given in Appendix A.

The tables that follow present the SMAPE and MASE forecast accuracy measures for different forecasting strategies (by row) at different forecast horizons, together with average measures and average rankings (by column), as explained in Section 5.4 . Because we can draw the same conclusions from both the SMAPE and MASE measures, we will focus on the SMAPE measure. If the results between these two measures differ, we will mention it explicitly. Also, the comparison between the strategies is performed on all the columns, unless specified.

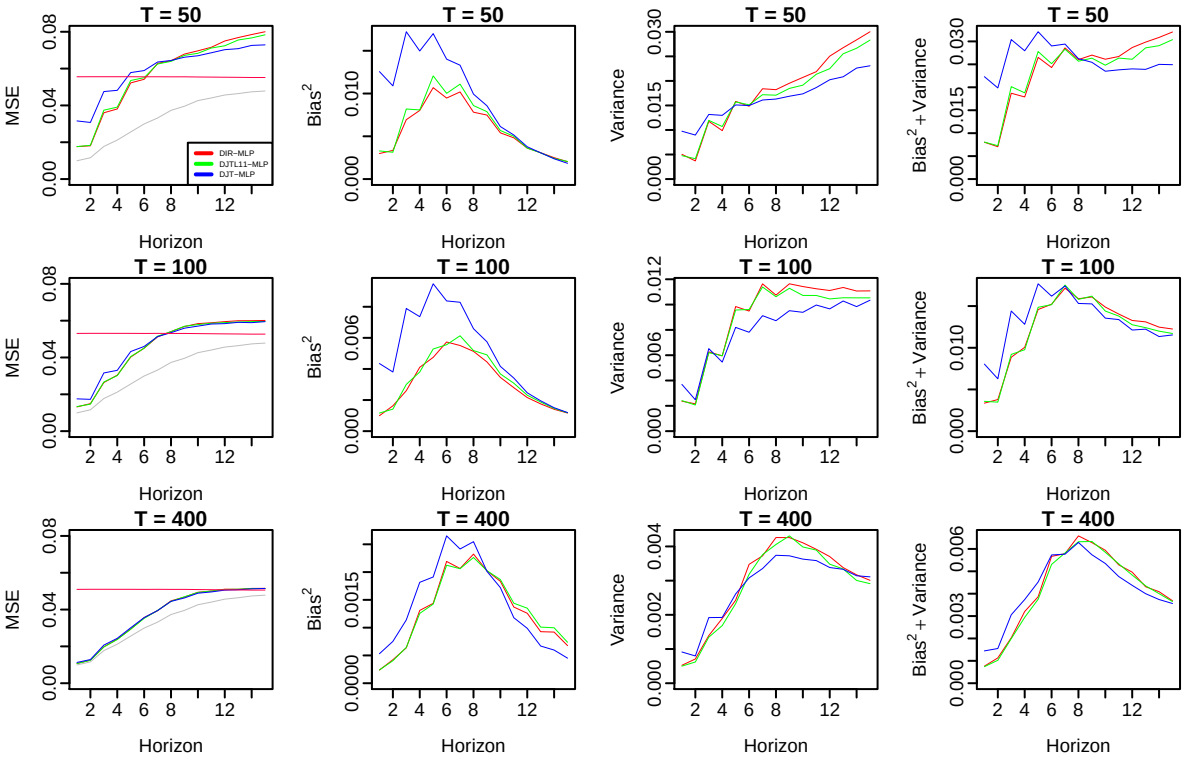

Figure 6.5: *STAR DGP. MSE decomposition of direct multi-horizon strategies with the MLP model.*

We first compare the direct multi-horizon strategies with the MLP model for both the M3 and NN5 competitions. Then, we apply the same comparison with the KNN model. Finally, we consider the recursive multi-horizon strategies for both the MLP and the KNN models, but only for the M3 competition[2]. Note that we will use the acronym DJTL in place of DJTL11 in the following. Also, the NAIVE strategy is defined as $\hat{y}_{T+h} = y_T$ for $h = 1, \ldots, H$.

Let us first consider the results for the MLP model given in Tables 6.2 and 6.3.

The results for the M3 competition are presented in Table 6.2 and can be summarized as follows:

- For the first few horizons (i.e. $h = 1, 2, 3$), DIR-MLP has lower errors than DJT-MLP. This confirms that the DJT strategy has higher errors at short horizons, notably due to a higher bias component (see Figure 6.1).
- For longer horizons (e.g. $h = 12$ and $h = 18$), DJT-MLP has lower errors than DIR-MLP, which confirms the benefit of the DJT strategy for long horizons, notably due to a lower variance (see Figure 6.1).
- When considering the SMAPE measure, DJTL-MLP outperforms both DJT-MLP and DIR-MLP. With the MASE measure, DJTL-MLP has lower errors than DIR-MLP and similar performance than DJT-MLP. This confirms the advantage of considering only local horizons instead of all horizons into the objective function when selecting the lag order and the hyperparameters.

The results for the NN5 competition are presented in Table 6.3 and can be summarized as follows:

- In contrast to the M3 competition, when considering averaged errors, DJT-MLP outperforms DIR-MLP. In other words, restricting the models for all horizons to use the same lag order

[2] We did not consider the NN5 competition because of the heavier computational time of the RTI strategies with long time series.

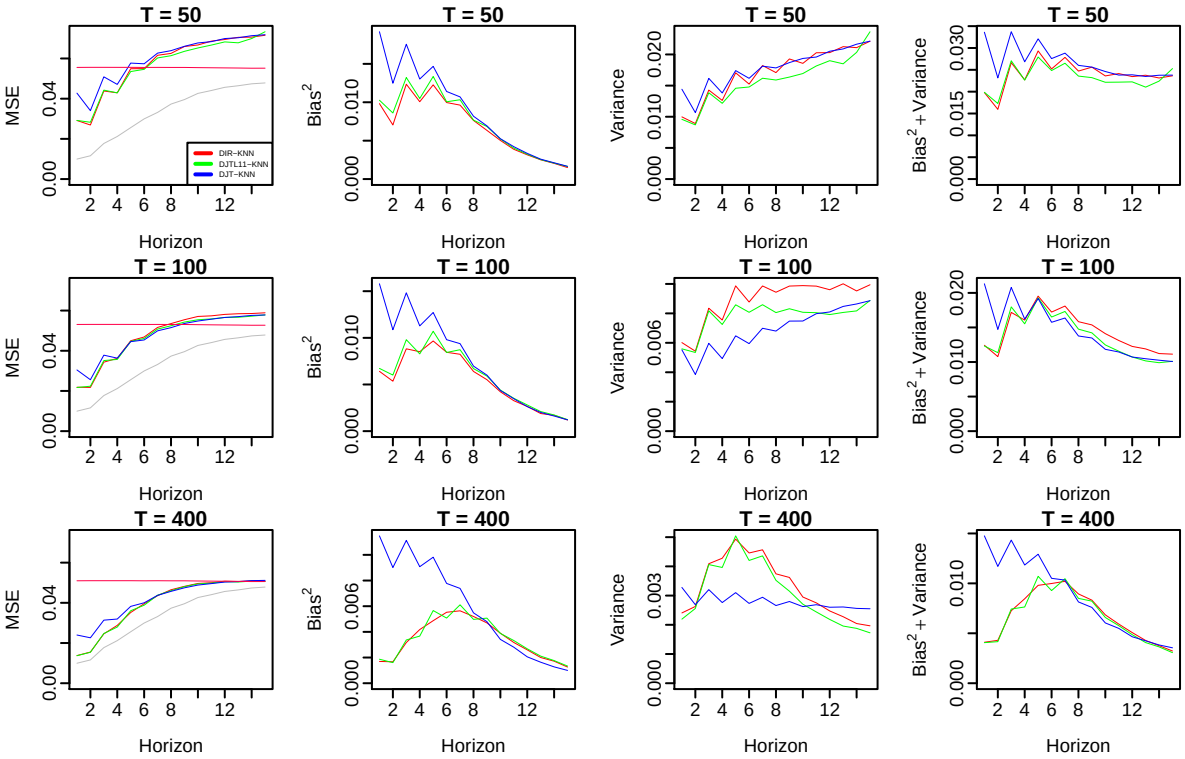

Figure 6.6: *STAR DGP. MSE decomposition of direct multi-horizon strategies with the KNN model.*

Table 6.2: *M3 competition. SMAPE and MASE forecast accuracy measures for direct multi-horizon strategies with the MLP model.*

Strategy	Forecast horizon (h)						Average			Avg. rank
	1	2	3	6	12	18	1-6	1-12	1-18	
SMAPE										
DJTL11-MLP	**12.32**	**11.48**	**13.01**	**13.39**	**14.95**	20.74	**13.27**	**14.01**	**15.87**	**1.56**
DJT-MLP	13.60	12.09	13.96	13.68	15.12	**20.02**	14.01	14.59	16.07	2.28
DIR-MLP	12.43	11.74	13.14	13.65	15.11	21.07	13.43	14.24	16.11	2.44
NAIVE	15.49	14.45	15.86	14.68	15.99	20.86	15.69	16.16	17.37	3.72
MASE										
DJT-MLP	2.79	2.70	3.20	3.66	**4.27**	**6.36**	3.37	3.74	**4.41**	**2.00**
DJTL11-MLP	**2.53**	**2.50**	2.93	**3.42**	4.43	6.72	3.15	**3.61**	4.44	2.06
DIR-MLP	2.54	2.51	**2.87**	3.56	4.44	6.74	**3.14**	3.67	4.50	2.61
NAIVE	3.26	3.08	3.65	3.82	4.54	6.56	3.75	4.12	4.70	3.33

and the same hyperparameters allows to reduce the forecast errors compared to allowing a different value for each horizon. Note however that the parameters of the MLP model are allowed to be different at each horizon.

- DJTL-MLP has lower error than DIR-MLP for long horizons but higher errors for short horizons as with the M3 competition.

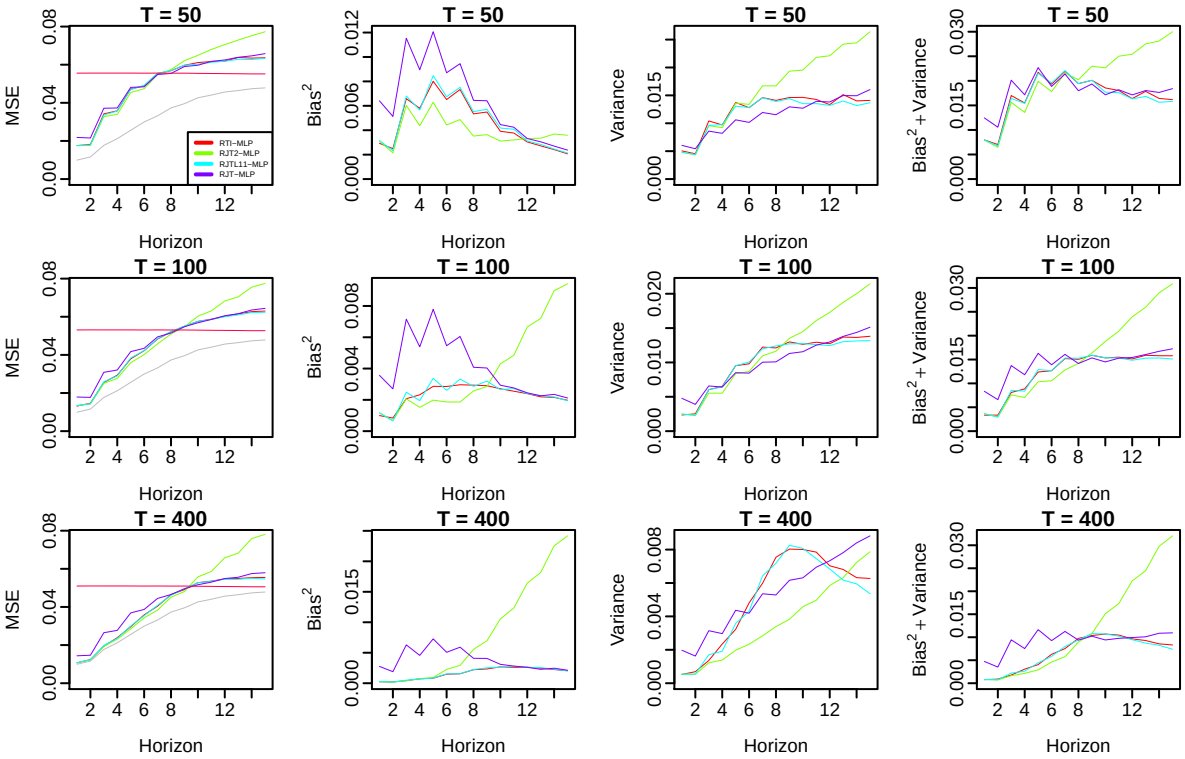

Figure 6.7: *STAR DGP. MSE decomposition of recursive multi-horizon strategies with the MLP model.*

Table 6.3: *NN5 competition. SMAPE and MASE forecast accuracy measures for direct multi-horizon strategies with the MLP model.*

Strategy	Forecast horizon (h)							Average								Avg. rank
	1	2	3	7	14	21	56	1-7	1-14	1-21	1-28	1-35	1-42	1-49	1-56	
SMAPE																
DJT-MLP	**13.63**	**12.50**	12.80	18.65	17.95	**40.39**	17.06	**16.51**	**17.66**	**20.42**	**20.10**	**22.23**	**22.67**	**22.23**	**21.53**	**1.79**
DJTL11-MLP	15.05	13.48	12.50	**18.45**	18.04	40.64	**16.88**	16.99	17.99	20.68	20.28	22.36	22.78	22.33	21.62	2.07
DIR-MLP	14.86	13.36	**12.43**	18.72	18.46	40.76	17.27	16.96	17.96	20.76	20.37	22.43	22.86	22.41	21.73	2.50
MEAN	16.69	14.22	13.64	18.59	**17.83**	41.54	18.03	18.06	18.91	21.42	21.09	23.24	23.80	23.40	22.77	3.64
MASE																
DJT-MLP	**0.26**	**0.30**	0.40	0.44	0.42	**0.80**	0.42	**0.50**	**0.53**	**0.60**	**0.58**	**0.62**	**0.64**	**0.63**	**0.61**	**1.86**
DJTL11-MLP	0.28	0.31	0.39	0.43	0.42	0.81	**0.41**	0.50	0.54	0.61	0.59	0.62	0.64	0.63	0.61	1.98
DIR-MLP	0.28	0.31	**0.39**	0.44	0.43	0.81	0.42	0.50	0.53	0.61	0.59	0.62	0.65	0.63	0.62	2.48
MEAN	0.33	0.34	0.44	**0.43**	**0.41**	0.82	0.44	0.54	0.57	0.64	0.62	0.65	0.68	0.67	0.65	3.68

Let us consider the results for the KNN model which are given in Tables 6.4 and 6.5. In contrast to the MLP model, the KNN model does not have any parameter but only one hyperparameter (the number of neighbors).

The results for the M3 competition are presented in Table 6.4 and can be summarized as follows:

- DIR-KNN outperforms DJT-KNN at short horizons, but the difference in performance between the two strategies reduces with the horizons, and for $h = 18$, DJT-KNN has smaller errors than DIR-KNN. In the simulation study, we observed a similar behavior due to a large increase in bias for DJT-KNN at short horizons (see Figure 6.2).

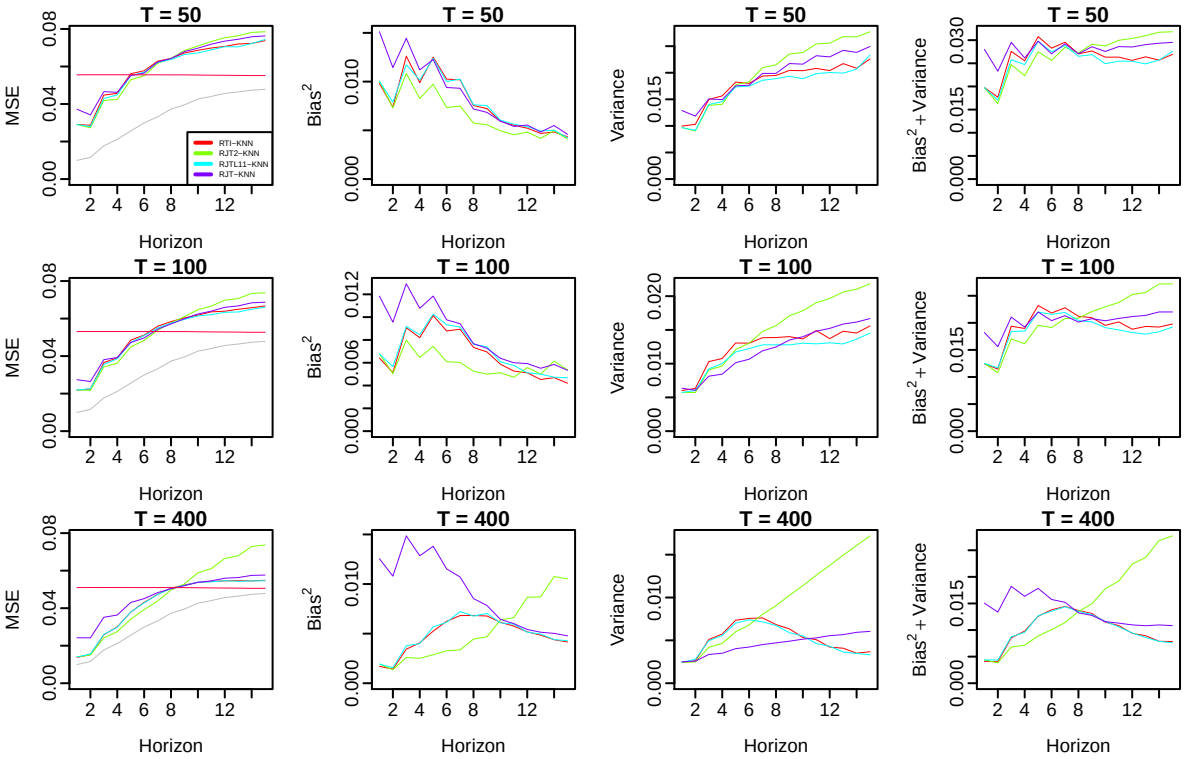

Figure 6.8: *STAR DGP. MSE decomposition of recursive multi-horizon strategies with the KNN model.*

Table 6.4: *M3 competition. SMAPE and MASE forecast accuracy measures for direct multi-horizon strategies with the KNN model.*

Strategy	Forecast horizon (h)						Average			Avg. rank
	1	2	3	6	12	18	1-6	1-12	1-18	
SMAPE										
DJTL11-KNN	13.12	**11.95**	**13.55**	**14.20**	16.35	20.54	**13.77**	14.87	**16.38**	**1.67**
DIR-KNN	**13.10**	12.21	13.65	14.26	16.58	20.72	13.83	**14.86**	16.42	1.83
NAIVE	15.49	14.45	15.86	14.68	**15.99**	20.86	15.69	16.16	17.37	2.94
DJT-KNN	17.83	15.09	17.31	16.70	17.39	**20.18**	17.09	17.31	18.07	3.56
MASE										
DIR-KNN	**2.77**	2.60	**3.21**	**3.76**	5.00	6.61	3.38	**3.89**	**4.57**	**1.89**
DJTL11-KNN	2.80	**2.59**	3.22	3.78	4.96	6.53	**3.37**	3.91	4.58	2.22
NAIVE	3.26	3.08	3.65	3.82	**4.54**	6.56	3.75	4.12	4.70	2.39
DJT-KNN	4.35	3.97	4.54	4.65	5.32	**6.42**	4.57	4.80	5.19	3.50

Compared to DJT-MLP in Table 6.2, the increase in bias of DJT-KNN is larger which can be explained by the fact that the MLP model has parameters that are allowed to change with the horizon. The reduced flexibility of DJT-KNN makes it even worse than NAIVE. Finally, the smaller error at $h = 18$ for DJT-KNN can be explained by a smaller variance for long horizons compared to DIR-KNN and DJTL-KNN.

Table 6.5: *NN5 competition. SMAPE and MASE forecast accuracy measures for direct multi-horizon strategies with the KNN model.*

Strategy	Forecast horizon (h)							Average								Avg. rank
	1	2	3	7	14	21	56	1-7	1-14	1-21	1-28	1-35	1-42	1-49	1-56	
SMAPE																
DJTL11-KNN	14.63	**11.67**	**11.63**	**17.47**	18.34	**41.18**	**17.05**	**16.12**	**17.40**	**20.29**	**20.01**	**22.22**	**22.71**	**22.30**	**21.64**	**1.79**
DJT-KNN	**14.32**	12.19	12.23	17.95	**17.69**	41.38	17.10	16.54	17.60	20.40	20.12	22.32	22.82	22.40	21.73	2.14
DIR-KNN	14.66	12.12	11.64	17.78	18.38	41.48	17.27	16.28	17.47	20.35	20.06	22.25	22.75	22.34	21.69	2.27
MEAN	16.69	14.22	13.64	18.59	17.83	41.54	18.03	18.06	18.91	21.42	21.09	23.24	23.80	23.40	22.77	3.80
MASE																
DJTL11-KNN	0.28	**0.28**	0.37	**0.41**	0.43	0.82	**0.41**	**0.48**	**0.53**	**0.60**	**0.58**	**0.62**	**0.65**	**0.63**	**0.62**	**1.89**
DJT-KNN	**0.28**	0.29	0.39	0.42	**0.41**	0.82	0.42	0.50	0.53	0.61	0.58	0.62	0.65	0.64	0.62	2.14
DIR-KNN	0.28	0.29	**0.37**	0.42	0.43	0.83	0.42	0.49	0.53	0.60	0.58	0.62	0.65	0.63	0.62	2.23
MEAN	0.33	0.34	0.44	0.43	0.41	**0.82**	0.44	0.54	0.57	0.64	0.62	0.65	0.68	0.67	0.65	3.73

- DJTL-KNN and DIR-KNN have comparable performances as can be seen by considering both SMAPE and MASE accuracy measures. In particular, DJTL-KNN is a better alternative than DJT-KNN since it does not increase the errors at short horizons as with DIR-KNN.

The results for the NN5 competition are presented in Table 6.5 and can be summarized as follows:

- In contrast to the M3 competition, DJT-KNN has comparable performance to DIR-KNN which means that we can achieve the same performance by using the same lag order and the same number of neighbors for all the $H = 56$ horizons instead of using a different one for each horizon.
- When considering averaged errors, DJTL-KNN outperforms both DJT-KNN and DIR-KNN.

We now consider the multi-horizon recursive strategies for the M3 competition. Recall that REC (equivalently RJT1) and RJT (equivalently RJT18 since $H = 18$) are two extremes and RJT2 is in between. RTI and RJT are also two extremes with RJTL11 in between.

The results for the MLP and the KNN models are presented in Tables 6.6 and 6.7, respectively, and can be summarized as follows:

- REC has smaller errors than RJT at short horizons ($h = 1, 2$), and vice versa for long horizons ($h = 12, 18$). These results are consistent with the findings of the simulation study in Section 6.4, where we have seen the smaller variance of RJT compared to REC for long horizons.

  Also, RJT2 outperforms both REC and RJT with the MLP model, while it has better performance than REC and similar performance to RJT with the KNN model. Because RJT2 minimizes two-step ahead errors, it improves the performance for long horizons compared to REC and does not penalize short horizons as RJT.

- RTI has smaller errors than RJT for short horizons ($h = 1, 2$ and 3), and vice versa for long horizons ($h = 12$ and $h = 18$). Compared to REC, the results are mixed but REC has a better ranking. The lack of nonlinearity and the short time series that characterize the M3 competition data can favor REC over RTI. Finally, RJTL has larger errors than RJT for long horizons (but smaller than RTI) and vice versa for short horizons.

Table 6.6: *M3 competition. SMAPE and MASE forecast accuracy measures for recursive multi-horizon strategies with the MLP model.*

Strategy	Forecast horizon (h)						Average			Avg. rank
	1	2	3	6	12	18	1-6	1-12	1-18	
SMAPE										
RJT2-MLP	**12.28**	**11.40**	13.22	**12.75**	**15.07**	**19.55**	**12.81**	**13.76**	**15.15**	**1.17**
REC-MLP	12.55	11.65	13.59	12.97	15.23	19.98	13.05	14.01	15.42	2.94
RJT-MLP	12.81	11.78	13.50	12.93	15.09	19.74	13.29	14.06	15.44	3.00
RJTL11-MLP	12.30	11.54	**13.19**	12.87	15.44	20.58	12.99	14.05	15.60	3.39
RTI-MLP	12.39	11.64	13.34	13.21	15.94	20.91	13.11	14.22	15.88	4.56
NAIVE	15.49	14.45	15.86	14.68	15.99	20.86	15.69	16.16	17.37	5.94
MASE										
RJT2-MLP	2.58	2.54	3.01	**3.32**	**4.33**	**5.88**	**3.12**	**3.56**	**4.14**	**1.61**
REC-MLP	2.56	2.53	3.09	3.44	4.36	6.01	3.13	3.60	4.20	2.44
RJT-MLP	2.72	2.70	3.16	3.38	4.35	6.00	3.29	3.68	4.27	3.28
RJTL11-MLP	2.58	2.48	**2.96**	3.34	4.49	6.06	3.15	3.66	4.29	3.50
RTI-MLP	**2.51**	**2.47**	3.03	3.44	4.51	6.15	3.17	3.69	4.34	4.28
NAIVE	3.26	3.08	3.65	3.82	4.54	6.56	3.75	4.12	4.70	5.89

Table 6.7: *M3 competition. SMAPE and MASE forecast accuracy measures for recursive multi-horizon strategies with the KNN model.*

Strategy	Forecast horizon (h)						Average			Avg. rank
	1	2	3	6	12	18	1-6	1-12	1-18	
SMAPE										
RJT2-KNN	**13.04**	11.94	13.48	13.80	15.63	**19.75**	13.72	**14.48**	**15.87**	**2.36**
RJT-KNN	13.23	**11.86**	13.58	14.10	**15.40**	19.80	13.85	14.56	15.88	2.56
RJTL11-KNN	**13.04**	11.86	**13.45**	**13.73**	15.73	20.02	**13.70**	14.52	15.91	2.58
REC-KNN	13.10	12.20	13.54	14.05	15.94	20.07	13.82	14.66	16.05	3.67
RTI-KNN	13.12	12.11	13.50	13.93	15.81	20.11	13.90	14.71	16.07	3.83
NAIVE	15.49	14.45	15.86	14.68	15.99	20.86	15.69	16.16	17.37	6.00
MASE										
RJT-KNN	**2.69**	**2.60**	3.26	3.72	**4.51**	6.17	3.38	3.79	4.37	**2.17**
RJTL11-KNN	2.75	2.61	**3.20**	**3.54**	4.61	6.21	**3.31**	**3.77**	**4.37**	2.36
RJT2-KNN	2.75	2.68	3.26	3.61	4.67	**6.13**	3.35	3.80	4.40	3.14
RTI-KNN	2.77	2.65	3.24	3.61	4.61	6.21	3.36	3.82	4.41	3.67
REC-KNN	2.77	2.68	3.27	3.67	4.70	6.20	3.36	3.83	4.42	3.89
NAIVE	3.26	3.08	3.65	3.82	4.54	6.56	3.75	4.12	4.70	5.78

Overall, we have seen that many factors can affect the performance of multi-horizon strategies compared to single-horizon strategies.

With the MLP model, multi-horizon strategies have provided better forecasts than single-horizon strategies for both the M3 and NN5 competitions. With the KNN model, multi-horizon strategies have better forecasts with the NN5 competition but poor forecasts with the M3 competition.

In addition to the different class of models considered in the KNN and the MLP models, the MLP model has a set of parameters that are independently selected for each forecast horizon. Also, the M3 time series are short and noisy while the NN5 time series are longer and less noisy.

The results of the comparison between multi-horizon and single-horizon strategies are similar for both recursive and direct forecasts. In particular, the DJT strategy has always obtain better performance for long horizons compared to the DIR strategy. However, the DJT strategy has often higher errors than the DIR strategy for short horizons.

The DJTL strategy is a better alternative to DJT since it does not suffer from higher errors at short horizons, and at the same time improves the forecasts at long horizons compared to the DIR strategy. The DJTL strategy has often comparable performance to the best between the DIR and DJT strategies. However, the error reduction of DJT compared to DIR for long horizons is typically larger than the DJTL strategy.

6.6 Concluding remarks

Using a different model for each forecast horizon provides a large flexibility to generate multi-step forecasts. However, if each model is selected independently from the other models, potential irregularities in consecutive forecasts can arise if very different models are used at each horizon. These irregularities are manifest as a contribution to the forecast variance, especially with nonlinear machine learning models and short time series.

We address this issue by proposing multi-horizon strategies that select the lag order and the hyperparameters of each model by minimizing forecast errors over multiple horizons rather than just the horizon of interest. We still allow a certain flexibility since the parameters can be independently selected for each horizon.

We considered different multi-horizon strategies depending on which horizons are used for each model, and this, consequently, reflects on the way the lag order and the hyperparameters are selected. In particular, we considered the extreme case where all the horizons are used by each model, which is the opposite of the case where the models are independently selected. We also considered an intermediate configuration where only local horizons are used for each model.

Exploiting the relatedness between different forecasting tasks to improve multi-step forecasts has received little attention in the forecasting literature. There have been some attempts to apply some existing multi-response and multi-task methods for multi-step forecasting for example in Liu (1996); Kline (2004); Franses and Legerstee (2009). However, the success has been limited, notably due to the limited amount of data with univariate time series. In contrast, the multi-horizon strategies we proposed can be easily applied with short time series and with any machine learning model.

We compared the multi-horizon strategies with the single-horizon strategies in a bias and variance study as well as with the time series from the M3 and NN5 competitions. The conclusions for recursive and direct forecasts are often similar and can be summarized as follows.

Constraining all the models to use the same lag order and hyperparameters for all horizons induces a large increase in bias for short horizons, but allows a large decrease in variance for long horizons, compared to using a different model at each horizon.

When the lag order and the hyperparameters are allowed to change for each horizon but are selected using forecast errors from local horizons, we did not observe an increase in bias for short horizons while the variance is still lower for long horizons compared to selecting each model independently. However, when constraining all the models to have the same lag order and the same hyperparameters effectively decrease the errors compared to selecting all the models independently, the error decrease for long horizons is lower when only local horizons are considered. Also, we observed a more important variance decrease with the AR and NAR DGPs compared to the STAR DGP; in other words when the DGP is linear or weakly nonlinear.

With the M3 time series, we have effectively observe larger errors at short horizons and smaller errors for long horizons when all the models have the same lag order and hyperparameters, but not with the NN5 time series. The advantage of only using local horizons has been confirmed with both the M3 and NN5 time series. Finally, we observed a better performance of multi-horizon strategies with the MLP model compared to the KNN model, which can be explained by the larger variance of the forecasts generated with the MLP model.

Overall, the results suggest that better forecasts are generated with multi-horizon strategies when the conditional mean does not change too much with the horizon, the machine learning model is highly flexible and the time series is short. Also, the strategy that only considers local horizons provides a better bias and variance trade-off than the strategy which constrains all the models to use the same lag order and hyperparameters.

Chapter 7

The Global Energy Forecasting Competition 2012

This chapter is heavily based on the following publication:

- S Ben Taieb and RJ Hyndman (August 2013). A gradient boosting approach to the Kaggle load forecasting competition. *International Journal of Forecasting*, 1–19.

We participated in the Load Forecasting track of the Global Energy Forecasting Competition 2012. The competition involved a hierarchical load forecasting problem. We were required to backcast and forecast hourly loads (in kW) for a US utility with 20 geographical zones. This chapter describes and analyses the approach used by our team TinTin (Souhaib Ben Taieb and Rob J Hyndman), which ranked fifth out of 105 participating teams.

7.1 Introduction

The Global Energy Forecasting Competition 2012 (GEFCom2012) has been organized by the IEEE Working Group on Energy Forecasting (WGEF)[1] which focuses on the practical needs of the utilities. The goal of the GEFCom2012 competition was to (Hong, Pinson, and Fan, 2013)

- "improve the forecasting practices of the utility industry",
- "bring together state-of-the-art techniques for energy forecasting",
- "bridge the gap between academic research and industry practice",
- "promote analytics in power and energy education", and
- "prepare the industry to overcome the forecasting challenges posed by the smart grid world".

The competition was active for two months, starting the 1st September 2012 and ending on 31st October 2012[2] and included two different tracks: *load forecasting* and *wind forecasting*.

The Kaggle platform[3] has been used to host the GEFCom2012 competition. Kaggle is an online platform that allows companies to organize predictive modeling competitions, and data scientists from all over the world compete to build the best models and produce the best predictions.

[1] www.drhongtao.com/ieee-wgef

[2] Note that this is also the birthday of our son Harone.

[3] www.kaggle.com

Figure 7.1: *Photo of the eight winning teams members for the Global Energy Forecasting Competition 2012 (GEFCOM 2012) taken at the IEEE PES GM meeting in Vancouver, Canada.*

The standard Kaggle rules include the following features:

- The data and a description of the problem is available on the competition webpage[4]
- The initial dataset is split into a learning dataset and a testing dataset. All the teams are provided with the learning set, with which they will build and learn their models.
- Each team can submit its predictions for a subset of the complete testing data (e.g. 20% of testing data) and these submissions are scored immediately.
- There is a public leaderboard where the actual best score of each time is displayed.
- In order to prevent teams to retrieve the testing data, each team is limited to two submissions per day.
- A discussion forum is also available where both the administrators and participants can ask questions, share ideas and findings with each other.

This approach is different from the traditional centralized communication scheme where the competitors were only able to communicate with the administrators but not with other competitors.

[4] www.kaggle.com/c/global-energy-forecasting-competition-2012-load-forecasting

In addition, the results on the complete testing data and ranks were only available after the end of the competition.

We have participated to the load forecasting track of GEFCom2012. Our team has ranked fifth out of 105 participating teams. The winning teams of GEFCom2012 have been announced in a press release of the IEEE Power & Energy Society[5] and have been broadcasted by some media channels including Yahoo Finance[6]. Also, our team has been awarded the IEEE Power & Energy Society award for ranking fourth in the Load forecasting track of the GEFCom2012 competition.

We give more details about the load forecasting track in the next section. More information about the wind forecasting track can be found in Hong, Pinson, and Fan (2013).

7.2 The load forecasting track

Load forecasting is critical for utilities notably in order to better plan electricity demand, and energy market activities. Hong (2010) provides a literature review of the different techniques used for load forecasting.

The goal of the load forecasting track was to encourage new ideas on the following aspects (Hong, Pinson, and Fan, 2013): *data cleansing*, *hierarchical forecasting*, *special days forecasting*, *temperature forecasting*, *ensemble forecasting* and *integration*.

The load forecasting track involved a hierarchical forecasting problem. We were required to backcast and forecast hourly loads (in kW) for a US utility with 20 geographical zones. Thus, 21 separate time series needed to be backcast and forecast: the 20 zonal level series, and the aggregate series.

The electricity demand is subject to a range of factors, including weather conditions, calendar effects, economic activity, population growth and electricity prices. However, we were required to use only temperatures and calendar information.

The available data consisted of the hourly loads for the 20 zones, and hourly temperatures from 11 weather stations, from the first hour of 1 January 2004 to the sixth hour of 30 June 2008 (4.5 years of hourly data). We are not aware of the locations of the 20 zones and the 11 weather stations, and in particular, we do not know which stations are located in or near which zones.

Demand data were missing for the following eight non-consecutive weeks (although temperature data were available for these periods): 2005/3/6 – 2005/3/12; 2005/6/20 – 2005/6/26; 2005/9/10 – 2005/9/16; 2005/12/25 – 2005/12/31; 2006/2/13 – 2006/2/19; 2006/5/25 – 2006/5/31; 2006/8/2 – 2006/8/8; 2006/11/22 – 2006/11/28.

Our task was to forecast the hourly electricity load during these eight weeks, as well as for the week following the end of the available data, namely 2008/7/1 – 2008/7/7. No actual temperatures were given for this additional week.

Consequently, our task was to develop models which take temperature and calendar information as inputs, and predicts electricity load as the output. Our models will not explicitly take account of economic changes, population changes or price changes, as we do not have any available data on these variables. In any case, for short-term forecasting (up to one week ahead), these variables are unlikely to be useful predictors.

[5] http://ieee-pes.org/ieee-pes-announces-the-eight-winning-teams-for-gefcom2012
[6] http://finance.yahoo.com/news/ieee-power-energy-society-announces-120700338.html

The forecasts were evaluated using a Weighted Root Mean Squared Error (WRMSE) given by

$$\sqrt{\frac{\sum_h w_h(\hat{y}_{t+h} - y_{t+h})^2}{\sum_h w_h}}, \tag{7.2.1}$$

where $\hat{y}$ and y are the forecasted and actual values, with w_h being the weight for horizon h.

Higher weights have been assigned for the aggregated series and the forecasted weeks, compared to the disaggregated series and the backcasted weeks. More precisely, the forecasted and the backcasted weeks of the aggregated series were assigned the weights 160 and 20, respectively. Similarly, the disaggregated series were assigned the weights 8 and 1.

A benchmark model has been included in the competition in order to evaluate the improvement of the newly proposed methods. The benchmark model is a simple but yet effective linear model (in parameters) that includes main effects and interaction effects of load, temperature and calendar variables. More details about the benchmark model can be found in Hong (2010).

A wide range of models and methodologies have been used by the different teams of the load forecasting track. Among the top five teams, different models have been used including gaussian process regression, gradient boosting, multiple linear regression and semi-parametric regression. More details can be found in Hong, Pinson, and Fan (2013).

Our Team, called *TinTin*, ranked fifth out of 105 participating teams. Team TinTin consisted of Souhaib Ben Taieb (PhD student at Université Libre de Bruxelles, Belgium) and Rob J Hyndman (Professor of Statistics at Monash University, Australia, and PhD co-supervisor for Souhaib). Souhaib wrote all of the code and made all of the submissions. Rob's involvement was solely to provide advice and suggestions when Souhaib asked for, and to help in writing the report (Ben Taieb and Hyndman, 2013).

We used separate models for each hourly period, with component-wise gradient boosting to estimate each model using univariate penalised regression splines as base learners (see Section 2.2.4). Our models allow for the electricity demand to change with time-of-year, day-of-week, time-of-day, and on public holidays, with the main predictors being current and past temperatures as well as past demand. The next section will provide more details about the methodology and models we used.

7.3 Methodology of the TinTin team

7.3.1 Data analysis and preprocessing

Figure 7.2 shows the average demand for all zones during the period of the data. The average demand varied greatly across zones with Zone 18 having the highest demand levels and Zone 4 the least. By exploring the data, we noticed that Zones 3 and 7 contain identical data, and Zone 2 contains values that are exactly 92.68% of the demand values in Zones 3 and 7. Finally, Zone 9 contained very erratic demand patterns which did not seem to be related to the temperature values (see Figure 7.3). It appeared later that Zone 9 was an industrial customer load, which is largely not weather sensitive (Hong, Pinson, and Fan, 2013).

Figure 7.4 shows the hourly electricity demand for Zone 18. The months are marked with vertical gold lines. The periods shaded gray correspond to the weeks in which predictions of electricity demand were required. No demand data were available for these periods.

To see the intra-day patterns, we plot in Figure 7.5 the demand for Zone 18 for one month during 2005. There is a clear time of day effect, but no obvious day-of-week effect.

Figure 7.6 gives the hourly temperature data for all 11 weather stations. Again, the months are marked with vertical gold lines and the gray shaded regions correspond to the weeks in which predictions of electricity demand were required. For the first eight of these weeks, temperature data were available, but for the last week (beginning 1 July 2008), neither temperature nor demand data were available.

Electricity demand is subject to a wide variety of exogenous variables including calendar effects. Figure 7.7 shows that there is a clear time-of-year effect in the demand data with peaks in mean demand around February and July, and troughs in April/May and October. In others words, winter and summer are showing high demand while fall and spring have lower demand.

The average demand for each month and each day of the week is plotted in Figure 7.8. Because the day-of-week pattern is similar for all months, it is unlikely to have a strong interaction between day-of-week and time-of-year.

Boxplots of the demand by day of week are shown in Figure 7.9. While the day-of-week effect is relatively small, there is a drop in demand for the weekends.

We look at the way demand changes with the time of day in Figures 7.10 and 7.11. Here, hour 0 corresponds to 12am–1am, hour 1 corresponds to 1am–2am, and so on. The night-time pattern is similar for the two plots, but there is a difference during the working hours (8am–5pm).

Figure 7.12 shows for each time of day the demand in Zone 18 plotted against current temperature from station 9. There is a clear non-linear relationship, indicating current temperature is an important predictor of demand. For temperatures above 20°C, air conditioning usage drives demand, whereas for temperatures below 15°C, heating drives demand. Similar but weaker

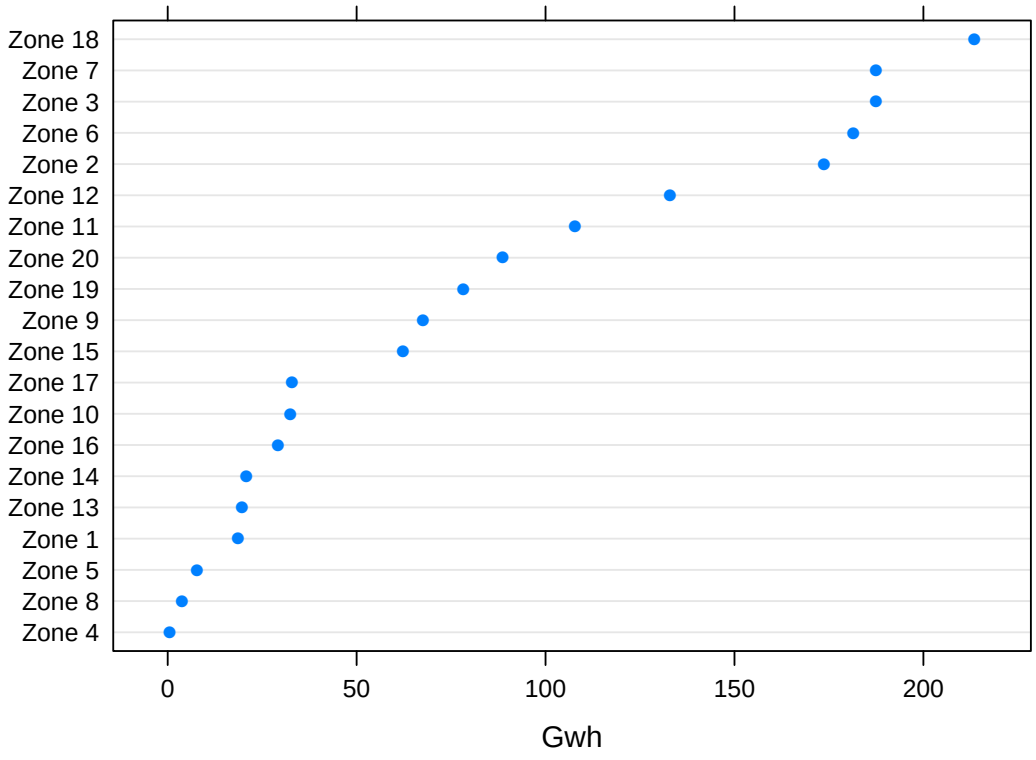

Figure 7.2: *Average demand for each zone.*

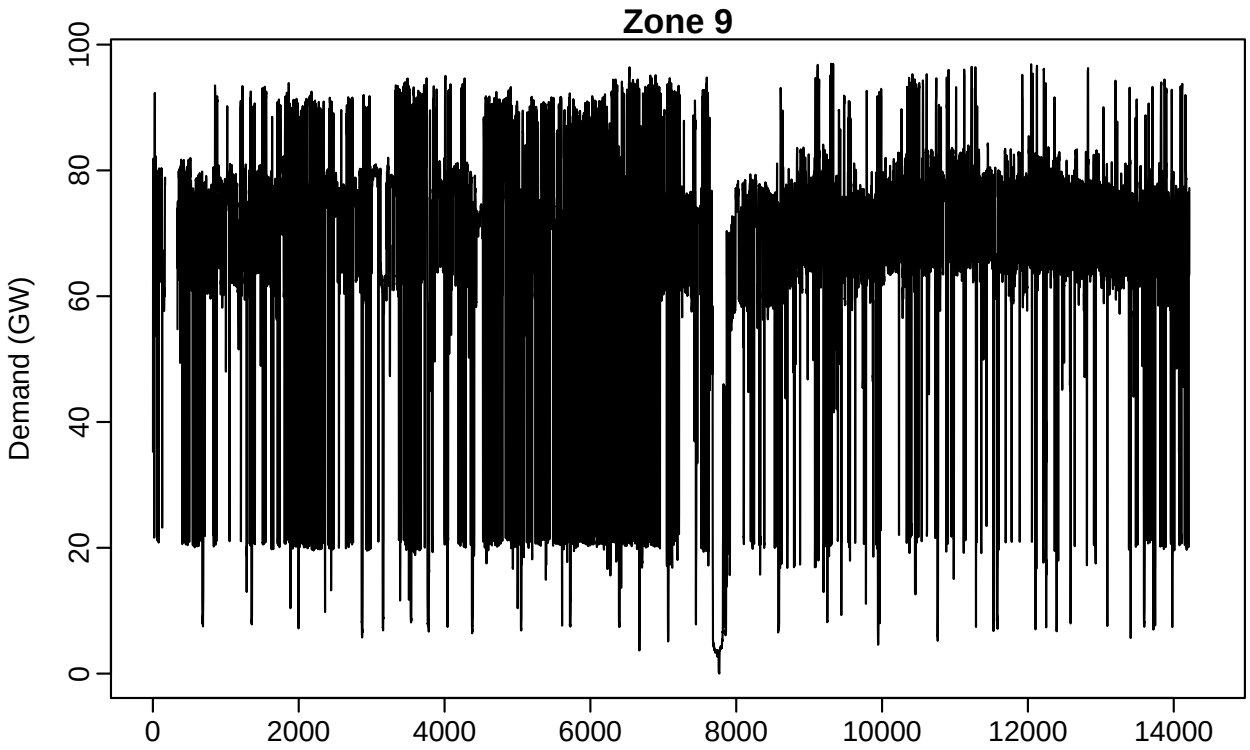

Figure 7.3: *Hourly demand (GW) for Zone 9, an industrial customer load.*

relationships are seen in plots against lagged temperatures. Due to thermal inertia in buildings, it is important to consider lagged temperatures as well as current temperatures in any demand forecasting model.

Figure 7.13 shows current demand (at hour p) plotted against lagged demand for different lags (i.e. at hour $p-i$ with $i = 1,\ldots,48$). We can see the relationship between these variables due to the serial dependence within the demand time series. In particular, we see a stronger dependence with lags corresponding to the same hour of the day. For example, in Figure 7.13, we can see that there is a stronger dependence between demand at hour p and both $p-24$ (the day before) and $p-48$ (two days before).

Before developing any forecasting models, we pre-processed the data to avoid some potential problems. First, we removed leap days to have 365 days for each year. This avoided problems with uneven seasonal periods, and resulted in only a small loss of information. Second, we took a log transformation for the demand. This is to stabilize the variance of the time series across time. There were also some outliers and unusual features in the data which we corrected before proceeding.

We identified some outliers in site 8 for the temperature data, and replaced them with the data of the same period in the closest site in terms of Euclidean distance.

Zone 4 had some outliers in demand. We used Loess for fitting and then classified a point y_t as an outlier if $y_t - \hat{y}_t > \text{median}(y_t - \hat{y}_t) + k * \text{MAD}$ where $\hat{y}_t$ is the Loess fit of y_t, MAD is the mean absolute deviation and k is chosen so that the probability of an outlier is 0.002 under a normal distribution. Then, the outliers have been replaced by the mean of the data.

For Zone 10, there was a big jump in demand in year 2008 (see Figure 7.15). We computed the mean before the jump and the mean after the jump on the log-transformed data. Then we took the

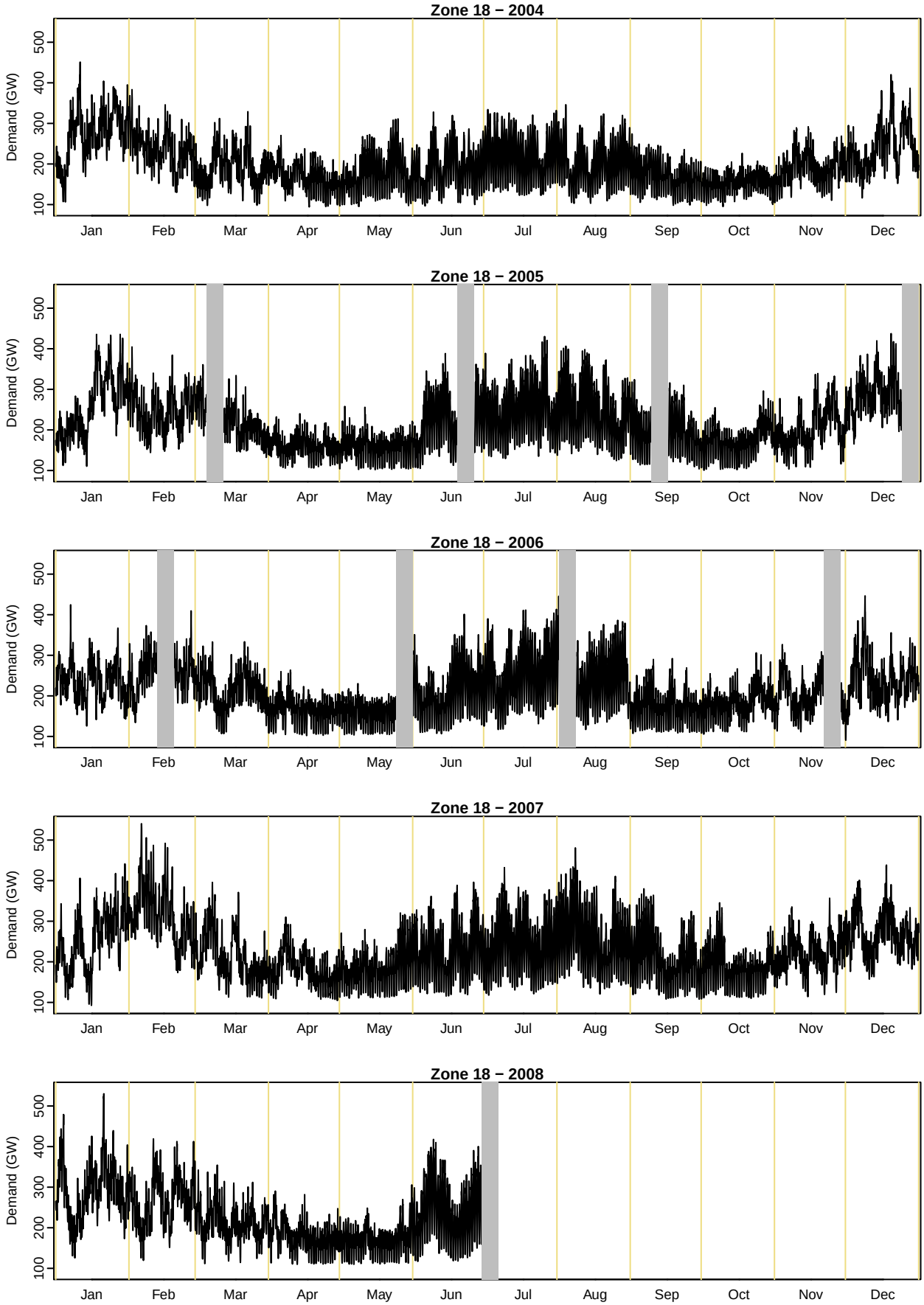

Figure 7.4: *Hourly demand (GW) for Zone 18.*

difference and removed the jump in 2008. For the final forecasts, we restored the jump into the forecasts.

7.3.2 Forecasting methodology

The competition involved a hierarchical load forecasting problem with 20 zonal level series, and the aggregate series. We used a bottom-up approach, that is we forecast each zone independently and then we took the sum of the 20 zonal forecasts to obtain forecasts for the aggregate (Fliedner, 2001).

For each of the twenty zones, we were required to backcast the demand for eight in-sample weeks for which the temperature at various sites was provided. Because we do not know which temperature site corresponds to which zone, the temperatures from all sites are potential predictors in each model. We used a testing week (the last week of the available data) to determine which sites to use for each zone. Figure 7.16 gives the root mean squared error (RMSE) obtained on the testing week using the real temperature (in blue) and the forecasted temperature (in red) from the eleven sites for ten zones. We can see that the difference in forecasts obtained using the real temperature (in blue) at different sites can be huge. Consequently, when forecasting the eight in-sample weeks, we use, for each zone, the site which minimizes the error over the testing week.

For the eight in-sample weeks, data was available before and after the week for demand and during the week for the temperature. The data after each of the in-sample weeks is also useful for predicting the demand values during the in-sample weeks (although this is not possible in real forecasting operations). In order to use this data, we fitted two forecasting models. The first model was estimated in the usual way using data available up to the start of the in-sample week. The

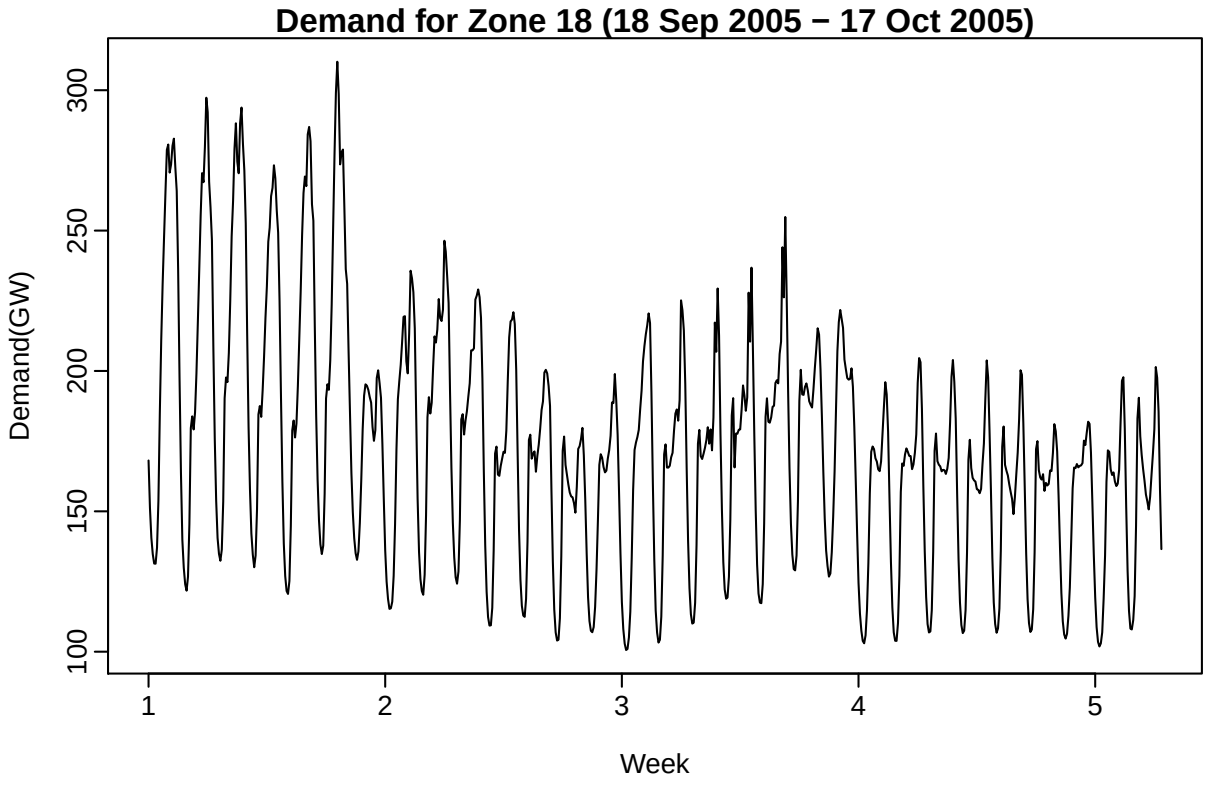

Figure 7.5: *Hourly demand (GW) for one month for Zone 18 from Sunday 18 September 2005 to Monday 17 October 2005.*

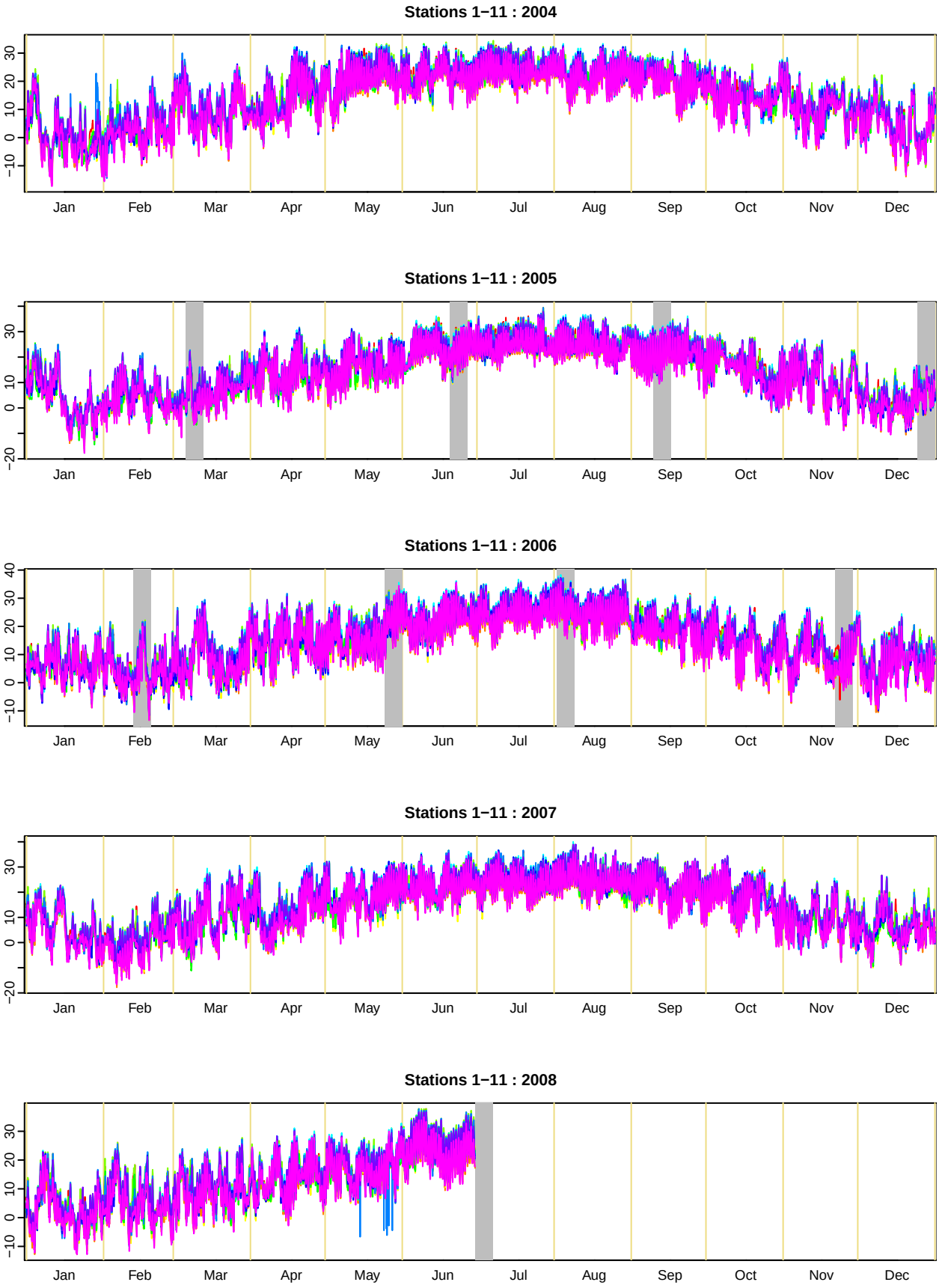

Figure 7.6: *Hourly temperatures for 11 weather stations from the first hour of 1 January 2004 to the sixth hour of 30 June 2008.*

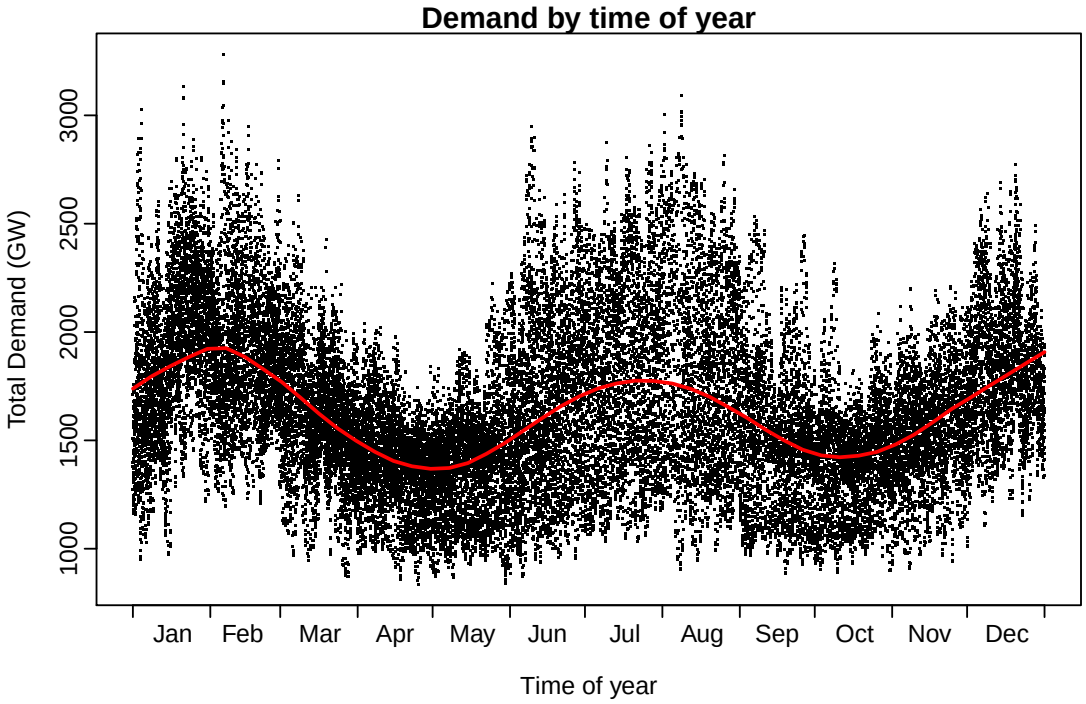

Figure 7.7: *Total demand plotted against the time of year. The smoothed mean demand is shown as a red line.*

second model reversed the time ordering of the data so we were back-casting using data available after the end of the in-sample week. We expect the forecasts to do best at the beginning of the week, and the backcasts to do best at the end of the week, because they involve data closer to the days being predicted. So, after estimating the two sets of models, we took a weighted combination of both sets of forecasts to produce the final forecasts. More precisely, if we denote $\hat{y}_{t+h}^{(\mathrm{F})}$, the forward forecasts and $\hat{y}_{t+h}^{(\mathrm{B})}$, the backward forecasts with $h = 1, \ldots, 168$, then the final forecasts are given by

$$\hat{y}_{t+h} = \alpha_h \hat{y}_{t+h}^{(\mathrm{F})} + (1 - \alpha_h)\hat{y}_{t+h}^{(\mathrm{B})}$$

where $\alpha_h = \mathrm{sigmoid}(-7 + \frac{14*h}{168})$ and $\mathrm{sigmoid}(x) = 1/(1 + e^{-x})$ is the sigmoid function.

Forecasts were also required for one out-of-sample week without temperature data. in contrast to the in-sample weeks, temperatures were not provided for the out-of-sample week. So we had to forecast the temperatures for this week and used these when forecasting demand during that week. We used the average temperature at the same period across the years as forecasts, shown in Figure 7.17. Differences between out-of-sample forecasts obtained using different temperature sites are not as large as for the in-sample weeks as can be seen in Figure 7.16 (see red bars). This is because we do not have actual temperatures in that week, and our forecast temperatures have a smaller range than the actual temperatures. So, for out-of-sample forecasts, we average the forecasts obtained from the three best temperature sites when forecasting demand in order to reduce the variance of the forecasts. These sites are also shown in Figure 7.16.

Since the demand patterns vary greatly during the year, and for computational convenience, we did not use the whole demand data to estimate our models. Instead, for each of the available years, we used part of the data around the week to be forecasted. More precisely, we computed the average temperature for the week to be forecasted and, for each year, we select the 12 consecutive weeks around this week which have the closest average temperature. Then we filtered the demand data

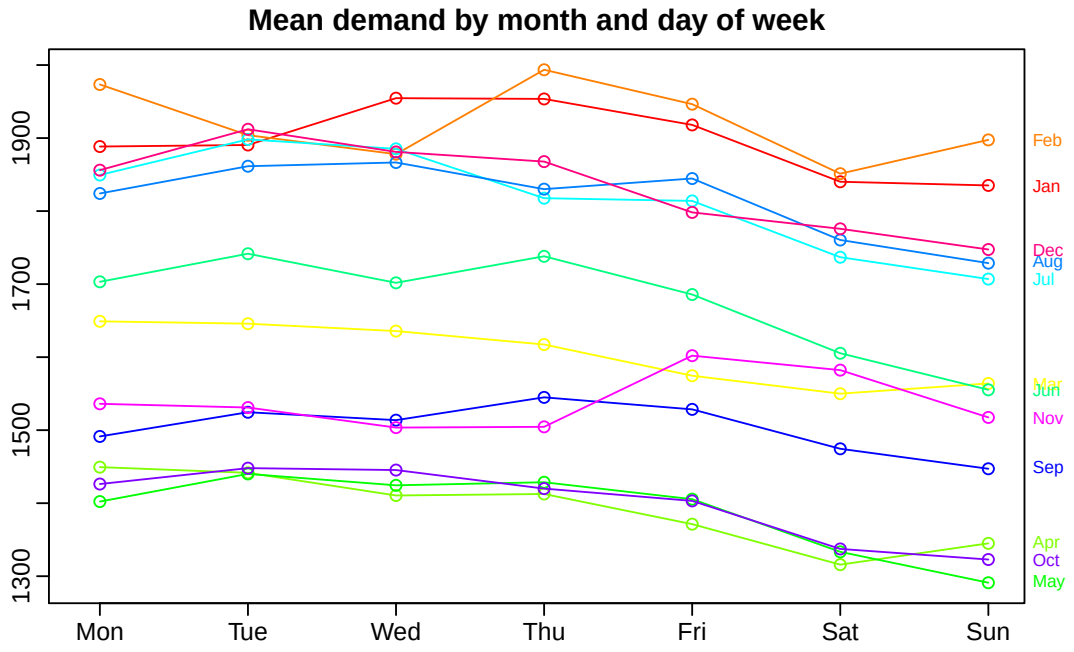

Figure 7.8: *Average total demand (GW) by month and day of week.*

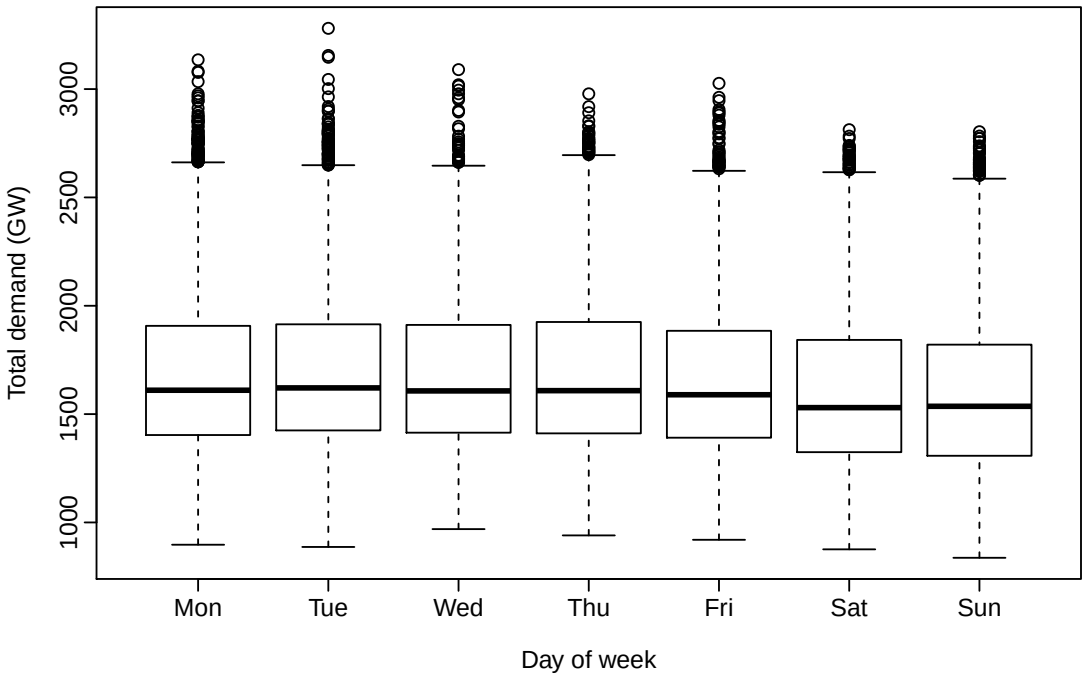

Figure 7.9: *Boxplots of total demand by day of week.*

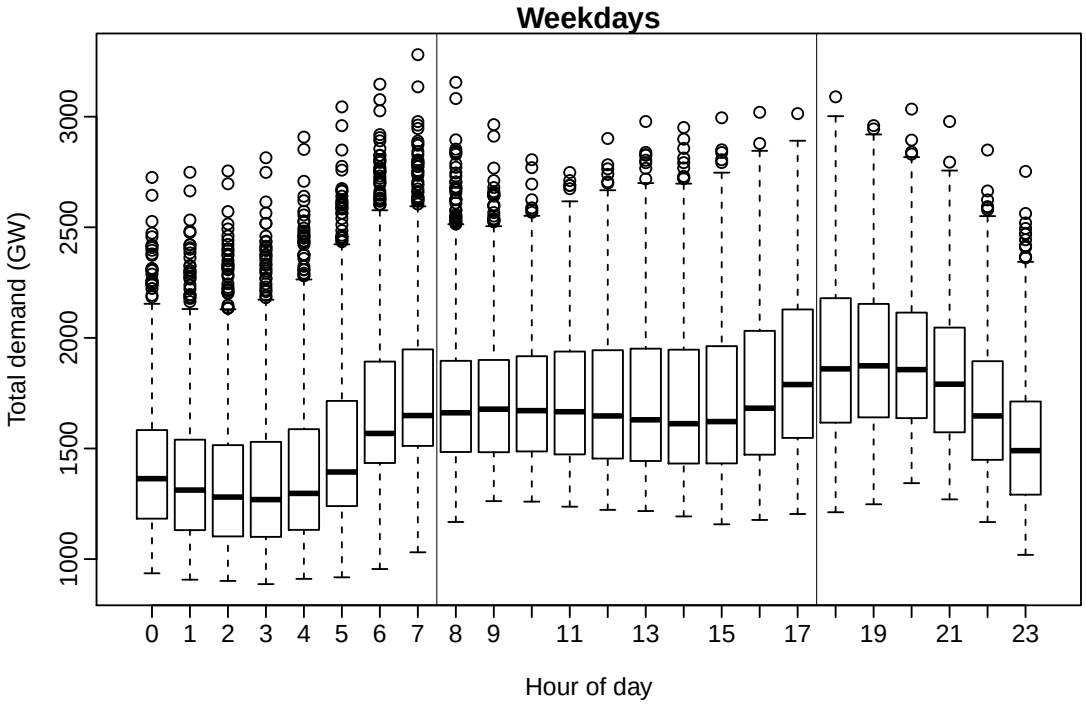

Figure 7.10: *Boxplots of demand by time of day for Monday–Friday.*

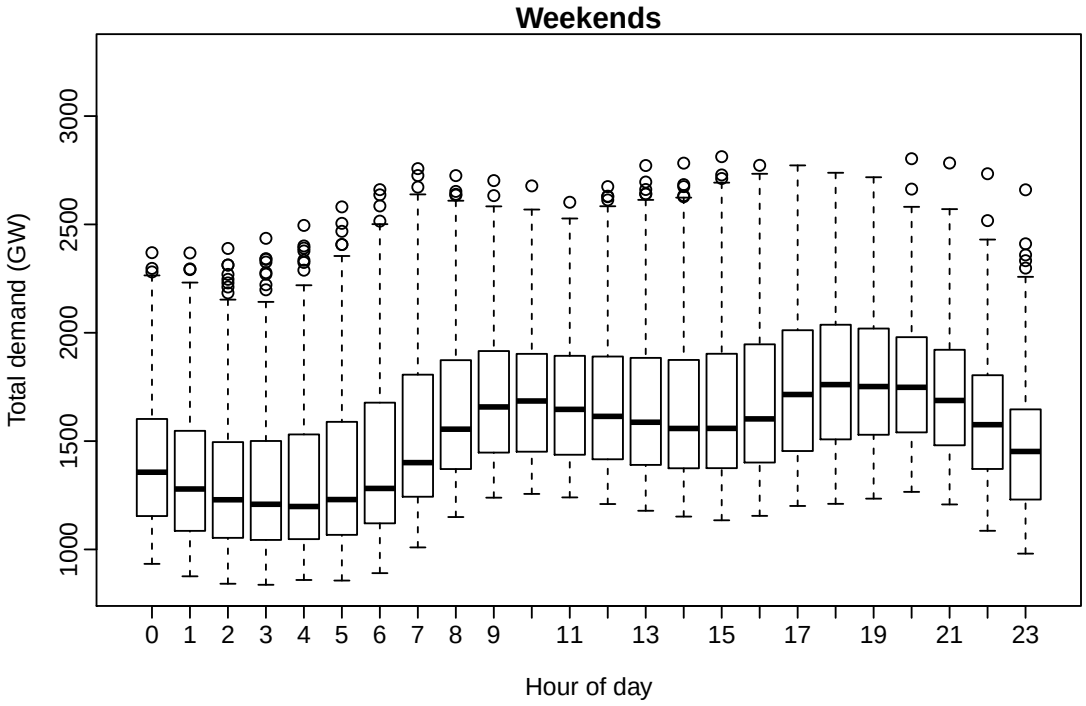

Figure 7.11: *Boxplots of demand by time of day for Saturday–Sunday.*

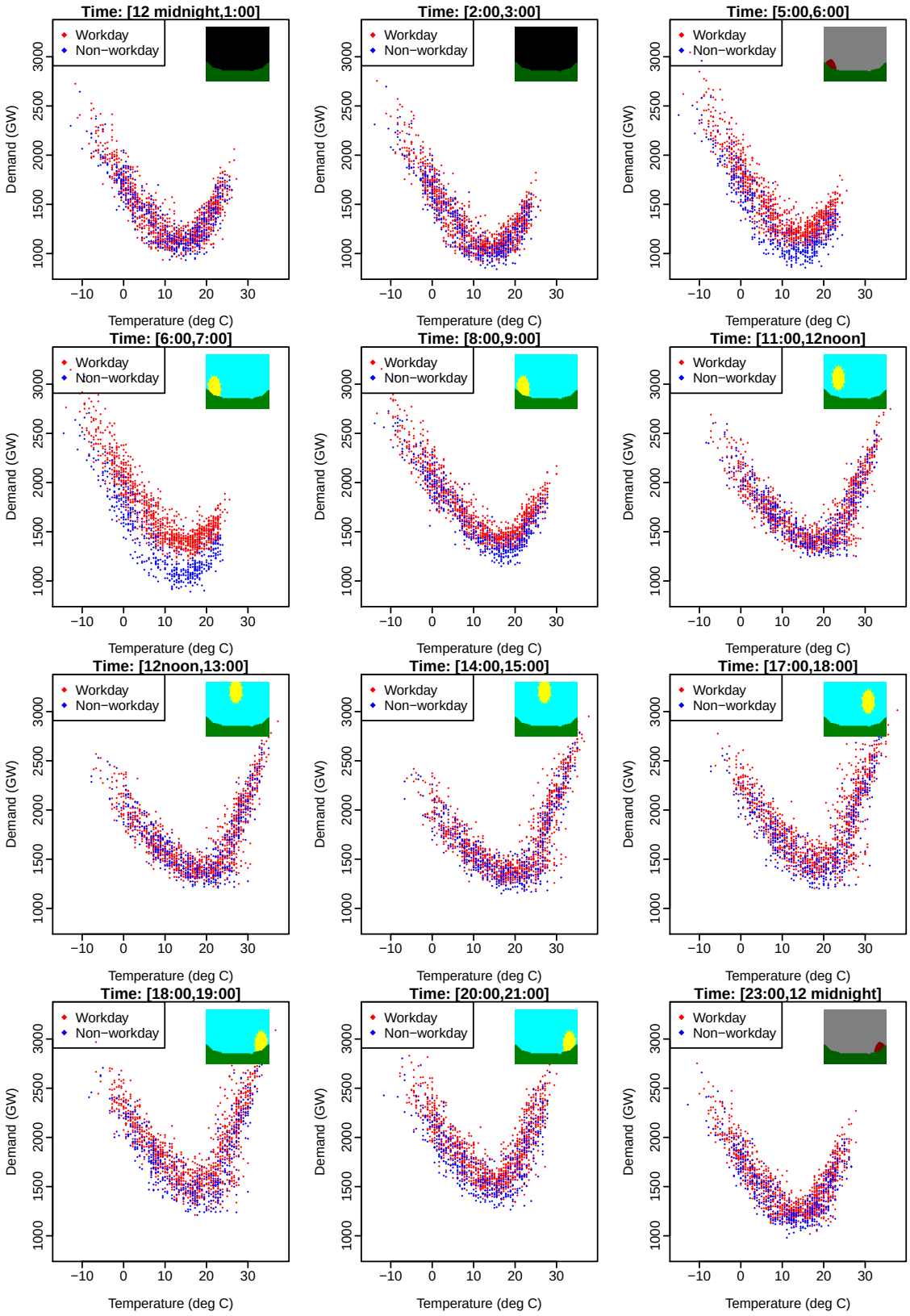

Figure 7.12: *Hourly demand (GW) plotted against temperature (degrees Celsius) for Zone 18 and station 9.*

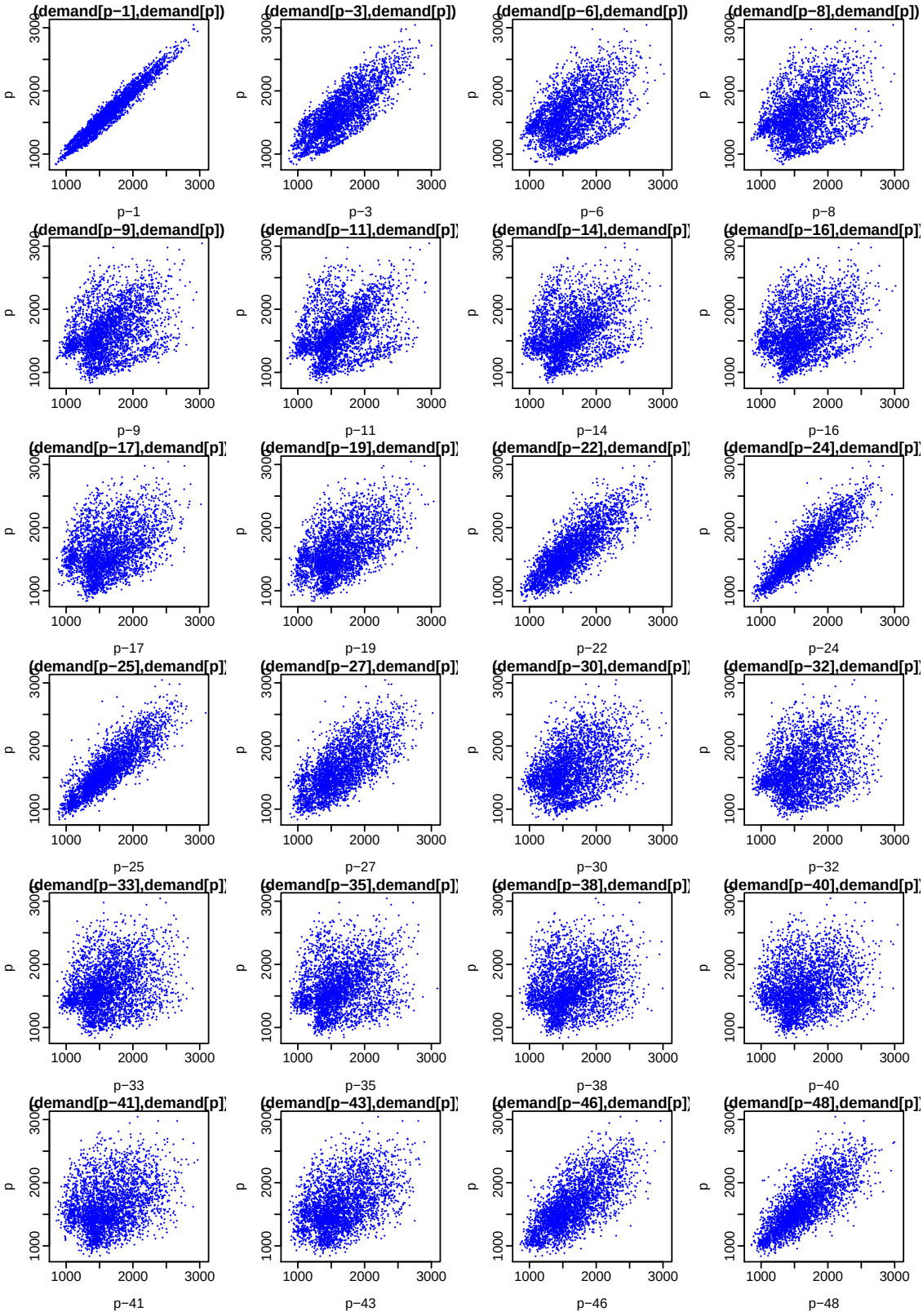

Figure 7.13: *Current demand plotted against lagged demand for different lags for Zone 18.*

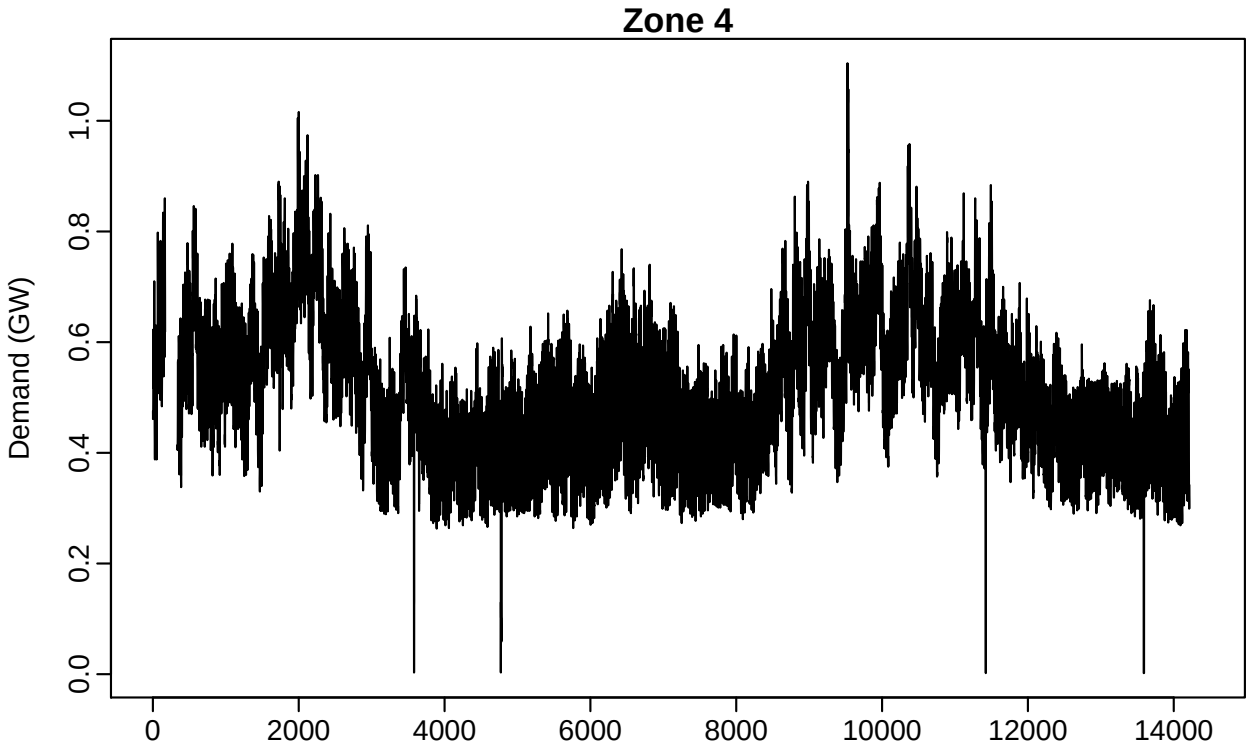

Figure 7.14: *Hourly demand (GW) for Zone 4 with outliers.*

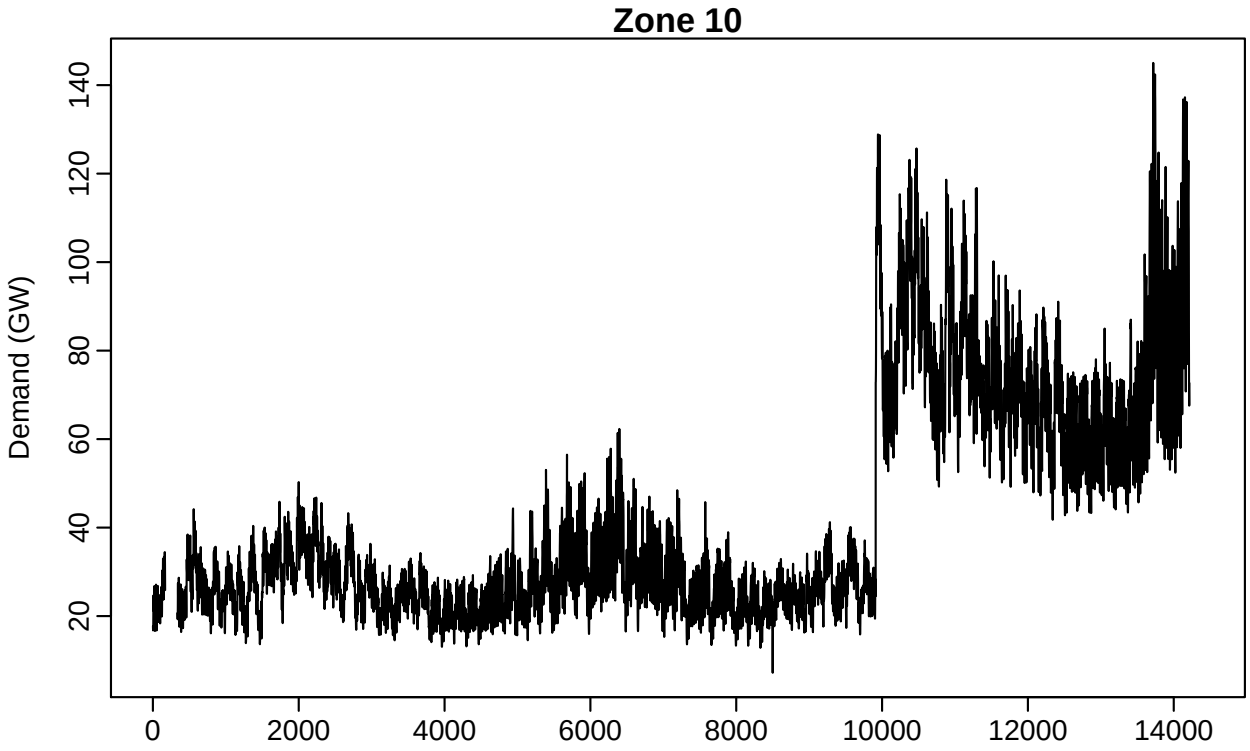

Figure 7.15: *Hourly demand (GW) for Zone 10 with a big jump in demand.*

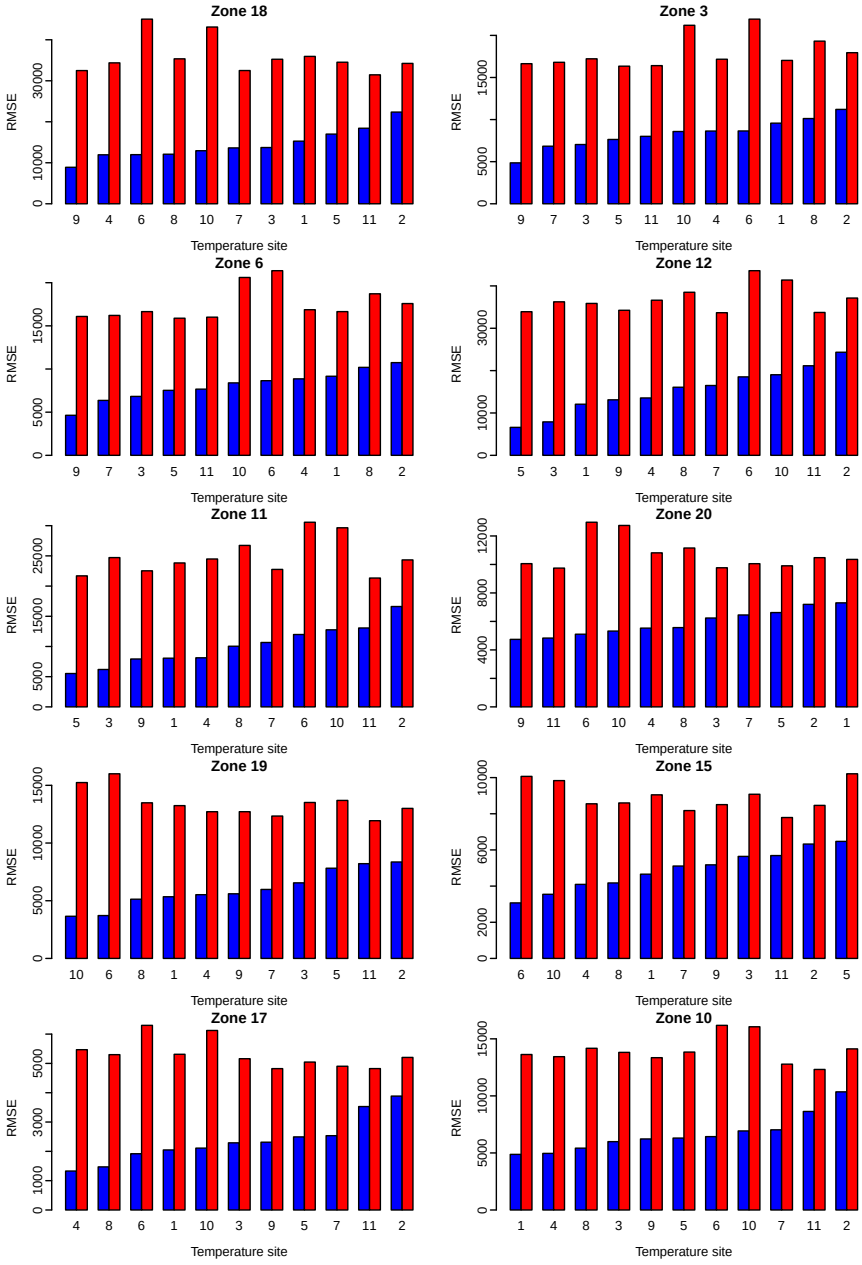

Figure 7.16: *Root mean square error (RMSE) over the testing week using real temperature (in blue) and forecasted temperature (in red). The sites are ranked according to the RMSE when using real temperature.*

by keeping these 12 consecutive weeks of each year. For the out-of-sample week, for which we do not have the average temperature for the week to be forecasted, we used the average temperature of the previous week to perform the same calculations.

The above procedures was followed for all zones but zones 2, 7 and 9. Zones 3 and 7 contain identical data, and Zone 2 contains values that are exactly 92.68% of the demand values in Zones 3 and 7. Consequently, we do not fit separate models for Zones 2 and 7, instead we use the forecasts from Zone 3 to compute forecasts for Zones 2 and 7. And for Zone 9, we used the average of the same hour for every day of the week over the entire data set to obtain forecasts for each period as we didn't find any temperature-related patterns.

7.3.3 Model specification

One of the earliest electricity forecasting competitions was won by Ramanathan et al. (1997) using a separate regression model for each hour of the day. This idea has subsequently been used by Fay et al. (2003); Taylor (2003); McSharry, Bouwman, and Bloemhof (2005); Fan and Chen (2006) and Fan and Hyndman (2012). In other words, the forecasts have been generated with the direct strategy, which is expected to provide better forecasts than the recursive strategy because of the strong nonlinearity of electricity load data and the large size of the data sets.

We follow the same approach and have a separate model for each hour of the day. However, to generate the forecasts for the 7 days, we used the MSVR strategy described in Section 3.5.3. In other words, these models are used recursively to produce the 168 required forecasts (24 hours multiplied by 7 days). That is, we first produce the forecasts of the next day and then, use them as inputs to make forecasts for the day after. We also added the observations from the previous hour and the next hour in order to have more data. That is, when estimating the model for hour p, we used data from hours $p-1$, p and $p+1$.

We fit a model of this form for each hour of the day, for each zone, and for each of the nine weeks to be forecast. We ended up with $24 \times 17 \times 9 = 3672$[7] models to estimate with datasets having approximately 1000 observations and 43 input variables. The main predictors of the models are current and past temperatures (up to a week earlier) and past demand (up to a week earlier). In addition, the models allow for the demand to change with time-of-year, day-of-week, time-of-day, and on public holidays.

We use nonparametric additive models for forecasting electricity demand. These models are in the regression framework but with some non-linear relationships. In particular, the proposed models allow nonlinear and nonparametric terms using the framework of additive models (Hastie and Tibshirani, 1990).

The demand in hour p on day t is modelled with

$$y_{t,p} = c_p(t) + f_p(\boldsymbol{y}_{t,p}) + g_p(\boldsymbol{z}_{t,p}) + \varepsilon_t, \tag{7.3.1}$$

where

- $y_{t,p}$ denotes the demand on day t in hour p;
- $c_p(t)$ models all calendar effects (including time-of-year, day-of-week, holidays, etc.);
- $f_p(\boldsymbol{y}_{t,p})$ models the effect of recent demand variables where $\boldsymbol{y}_{t,p}$ is a vector of past demands, prior to hour p on day t;
- $g_p(\boldsymbol{z}_{t,p})$ models the temperature effects where $\boldsymbol{z}_{t,p}$ is a vector of recent temperatures variables at one of the temperature sites, prior to and including hour p on day t; and
- ε_t denotes the model error at time t.

We fit separate models of this form for each zone and all in-sample and out-of-sample weeks.

Calendar effects

The calendar effects term, $c_p(t)$, includes annual, weekly and daily seasonal patterns as well as public holidays.

- The day-of-week effect was modelled with a factor variable, taking a different value for each day of the week.

[7] There are only 17 zones to be modelled because of the relationship between Zones 2, 3 and 7, and because we use a different approach for Zone 9.

- The holiday effect was modelled with a factor variable, taking value zero on a non-work day[8], some non-zero value on the day before a non-work day and a different value on the day after a non-work day.
- The time-of-year effect was estimated using a simple first-order Fourier approximation (one sine and one cosine variables). A more complicated time-of-year effect was not necessary as each model only included data within a 12 weeks period.

Temperature effects

Due to thermal inertia in buildings, it is important to consider lagged temperatures as well as current temperatures in any demand forecasting model. The function $g_p(\boldsymbol{z}_{t,p})$ models the effects of recent temperatures on the aggregate demand where $\boldsymbol{z}_{t,p}$ includes

- The current temperature and temperatures from the preceding 12 hours, and for the equivalent hour on each of the previous two days;
- The minimum and maximum temperature in both the last 24 hours and the previous day;
- The average temperature for the previous day, the day preceding the previous day and the last seven days.

Recall that when forecasting the eight in-sample weeks, the temperatures from the site which gave the best forecasts (on the testing week) for each zone were used. When forecasting the out-of-sample week, the demand forecasts obtained using the best three temperature sites were averaged.

Lagged demand effects

We incorporate recent demand values into the model via the function $f_p(\boldsymbol{y}_{t,p})$ where $\boldsymbol{y}_{t,p}$ includes

- Lagged demand for each of the preceding 12 hours, and for the equivalent hour in each of the previous two days. For example, when predicting demand at 4pm, we use lagged demand from 4am to 3pm, as well as lagged demand at 4pm on the preceding two days.
- The minimum and maximum demand in the last 24 hours.
- The average demand for the last seven days.

By doing this, the serial correlations within the demand time series can be captured within the model, and the variations of demand level throughout the time can be embedded into the model as well.

Finally, we did not necessarily use all the previous predictors in each model, but these were all candidate variables in our models. The process of estimating the model given in expression (7.3.1) including the variable selection is described in the next section.

7.3.4 Model estimation

Notice that expression (7.3.1) can be rewritten as

$$y_{t,p} = F_p(\boldsymbol{x}_t) + \varepsilon_{t,p}. \tag{7.3.2}$$

where $\boldsymbol{x}_t = [t, \boldsymbol{y}_{t,p}, \boldsymbol{z}_{t,p}]$ contains all *potential* predictors to be considered in the model.

Since, the forecasting accuracy for the competition was evaluated by *weighted root mean square error*, as given in expression (7.2.1), we used squared errors to estimate the function $F_p : \mathbb{R}^d \to \mathbb{R}$.

[8] We used the US federal holidays as listed in US Office of Personnel Management website: http://www.opm.gov/Operating_Status_Schedules/fedhol/2008.asp

We considered *component-wise gradient boosting* (see Section 2.2.4) with univariate P-splines (See Section 2.2.1) to learn all the models of the form given in (7.3.2). By doing so, we take advantage of the good performance and the automatic variable selection of the boosting algorithm. In addition, P-splines allow us to have smooth estimation of the demand.

In our implementation, we used P-splines with 20 equally spaced knots and four degrees of freedom for the weak learner terms. For the hyperparameters values, we set the value of ν to 0.15 and the maximum number of components (or iterations) J to 500. Our implementation of the model depended on the **mboost** package for R (Hothorn et al., 2010). We used the `gamboost` function with the following values for the `mboost_control` parameters: `nu=0.15` and `mstop=500`. For the base learners, we used the `bbs` function for numerical variables and the `bols` function for factor variables. Finally, we select the best number of stages J($J \in \{1,\ldots,$`mstop`$\}$) using the `cvrisk` function with 5-fold cross-validation.

7.3.5 Model analysis

In this section, we analyse and interpret the results of the proposed forecasting model to shed light about its attractive features.

Figure 7.18 gives the root mean squared error (RMSE) obtained for the different zones on the testing week. Zones that have higher errors are also zones with high average demand (see Figure 7.2). This suggests that zones with high average demand will carry a higher weight in the final error.

Figure 7.19 gives the true hourly demand together with the fitted values for an increasing number of boosting iterations. Recall that gradient boosting is a stagewise fitting procedure which depends on an hyperparameters J (the number of boosting iterations), as shown in formula (2.2.14). We can see that the first iterations significantly reduce the error of the model while the final iterations are less and less contributing to the model. This confirms the theoretical analysis performed in Bühlmann and Yu (2003) where the authors prove that the bias decreases exponentially fast and the variance increases with exponentially diminishing terms as the number of boosting iteration J increases.

One of the attractive features of the component-wise boosting algorithm is the automatic variable selection induced by the procedure at each iteration. That is, among all the potential predictors which are given in Table 7.1, few will be selected and contribute to each hourly model. See Section 7.3.3 for a more detailed description of the different predictors.

Id	Variable	Id	Variable	Id	Variable	Id	Variable
1	day of the week	12	demand[t-8]	23	temp.[t+h-2]	34	temp.[t+h-13]
2	holiday	13	demand[t-9]	24	temp.[t+h-3]	35	temp.[t+h-25]
3	time of year (sin)	14	demand[t-10]	25	temp.[t+h-4]	36	temp.[t+h-49]
4	time of year (cos)	15	demand[t-11]	26	temp.[t+h-5]	37	min. temp. (prev 1)
5	demand[t-1]	16	demand[t-12]	27	temp.[t+h-6]	38	min. temp. (prev 2)
6	demand[t-2]	17	demand[t+h-25]	28	temp.[t+h-7]	39	max. temp. (prev 1)
7	demand[t-3]	18	demand[t+h-49]	29	temp.[t+h-8]	40	max. temp. (prev 2)
8	demand[t-4]	19	min. demand (prev 1)	30	temp.[t+h-9]	41	avg. temp. (prev 1-7)
9	demand[t-5]	20	max. demand (prev 1)	31	temp.[t+h-10]	42	avg. temp. (prev 1)
10	demand[t-6]	21	avg. demand (prev 1-7)	32	temp.[t+h-11]	43	avg. temp. (prev 3)
11	demand[t-7]	22	temp.[t+h-1]	33	temp.[t+h-12]	–	–

Table 7.1: *Description of all potential predictors. Forecasts are from demand[t-1] to demand[t+h-1] where* $h \in \{1,\ldots,24\}$.

Note that the final solution of the boosting procedure, given in expression (2.2.14), can be rewritten as

$$\hat{F}_p(\boldsymbol{x}_t) = \sum_{j=0}^{J} \nu \hat{l}^{[j]}(x_{k_j t}) = \sum_{k\in\{1,\ldots,d\}} \underbrace{\sum_{\{j:k_j=k\}} \nu \hat{l}^{[j]}(x_{k_j t})}_{\hat{M}(x_{kt})} = \sum_{k\in\{1,\ldots,d\}} \hat{M}(x_{kt}),$$

where $\hat{M}(x_{k_j t})$ is the relative contribution of the variable k to the final model and d is the number of initial predictors (the dimensionality of $\boldsymbol{x}_t$).

Let us define the model without the effect of predictor a as

$$\hat{F}_p^{(-a)}(\boldsymbol{x}_t) = \sum_{k\in\{1,\ldots,d\}\setminus\{a\}} \hat{M}(x_{kt}),$$

and the corresponding squared error as

$$E^{(-a)} = \sum_{t=1}^{T} (y_{t,p} - F_p^{(-a)}(\boldsymbol{x}_t))^2.$$

We then define the relative importance of a predictor a as

$$I_a = \frac{E^{(-a)} - E}{E}$$

where E is the squared error of the final model with all the selected variables. In other words, the predictors which most increase the error after removing their relative effect are the most influential variables *given* the other predictors.

Figure 7.20 shows the ten most influential variables (according to I_a) on the demand at different hours of the day. We assign the importance value $I_a* = 100$ to the most influential variable and the values of the others are scaled accordingly as in Friedman and Hastie (2000). To help visualization, variables belonging to the same effect are plotted with the same color. That is, demand variables are colored in green, temperature variables in blue and calendar variables in red.

We can see that variables from the different effects are selected: calendar effect (in red), temperature effect (in blue) and lagged demand effect (in green). The importance of these different effects has been shown in Section 7.3.1. We have seen the different calendar effects in Figures 7.7–7.11. The clear dependence between demand and temperature was illustrated in Figure 7.12. Finally, Figure 7.13 has shown that there is a clear dependence between the actual demand and previous lagged demand variables.

For the first horizon ($h = 1$), we see that the most important variable is the current demand (variable 5). This is not surprising since the demand at hour $p + 1$ is highly dependent on the demand at hour p. However, as one moves away from the starting point (i.e. h=2–3), other variables, such as the demand at the equivalent hour for the previous day (variable 17), become important while the current demand loses importance.

During working hours (10:00–19:00 or h=4–13), temperature variables (in blue) dominate demand variables (in green). In fact, among the most important variables we find the current temperature with some of the corresponding lagged temperatures (variables 22−−25). During that period of the day, we can also see that the variable 17 is gaining importance with the horizon, with the highest importance at 19:00.

For the last hours of the day (20:00–24:00 or h=14–18), temperature variables become the most influential variables. For 20:00 and 21:00, both variable 17 and 21 have a high importance while for the remaining hours of the day, only variable 17 remains as a most influential variable.

For the first hours of the next day (1:00–05:00), temperature variables are gaining more importance with a new variables appearing from 03:00 to 05:00, the average temperature of the previous day (variable 42).

Finally, at 06 : 00, we see that variable 17 is again the most influential variable together with the day of the week.

The analysis of Figure 7.20 has shown that the relative importance of the different variables and effects is changing with the time of the day. This shows that the dependence between the demand and the different considered variables is changing with the forecasting horizon, making electricity load forecasting a challenging statistical problem.

For illustration purposes, Figure 7.21 gives the forecasts of the aggregate series (i.e. the sum of the forecasts over all zones) for the eight in-sample weeks and the out-of-sample week.

7.4 Concluding remarks

Our entry ranked fifth out of 105 participating teams. This suggests that our modelling strategy is competitive with the other models used to forecast the electricity demand. We have identified several aspects that makes our modelling strategy successful.

First, as in any prediction task, data analysis allowed us to identify and clean the data from any corrupted information for better model performance. The data analysis step was also important for identifying useful variables to use in the model.

Second, we used different models including different effects for each hour of the day to model the demand patterns which are changing throughout the day.

Third, each hourly model allowed nonlinear and nonparametric terms to be included in the final model. This provided a great flexibility to the model and avoided making too many assumptions about the data generating process.

Finally, gradient boosting has proven many times to be an effective prediction algorithm for both classification and regression tasks. By selecting the number of components included in the model, we can easily control the so-called bias-variance trade-off in the estimation. In addition, component-wise gradient boosting increases the attractiveness of boosting by adding automatic variable selection during the fitting process.

The Load forecasting track of the Global Energy Forecasting competition included challenging prediction tasks which required solving several statistical problems such as data cleaning, variable selection, regression and multi-step time series forecasting.

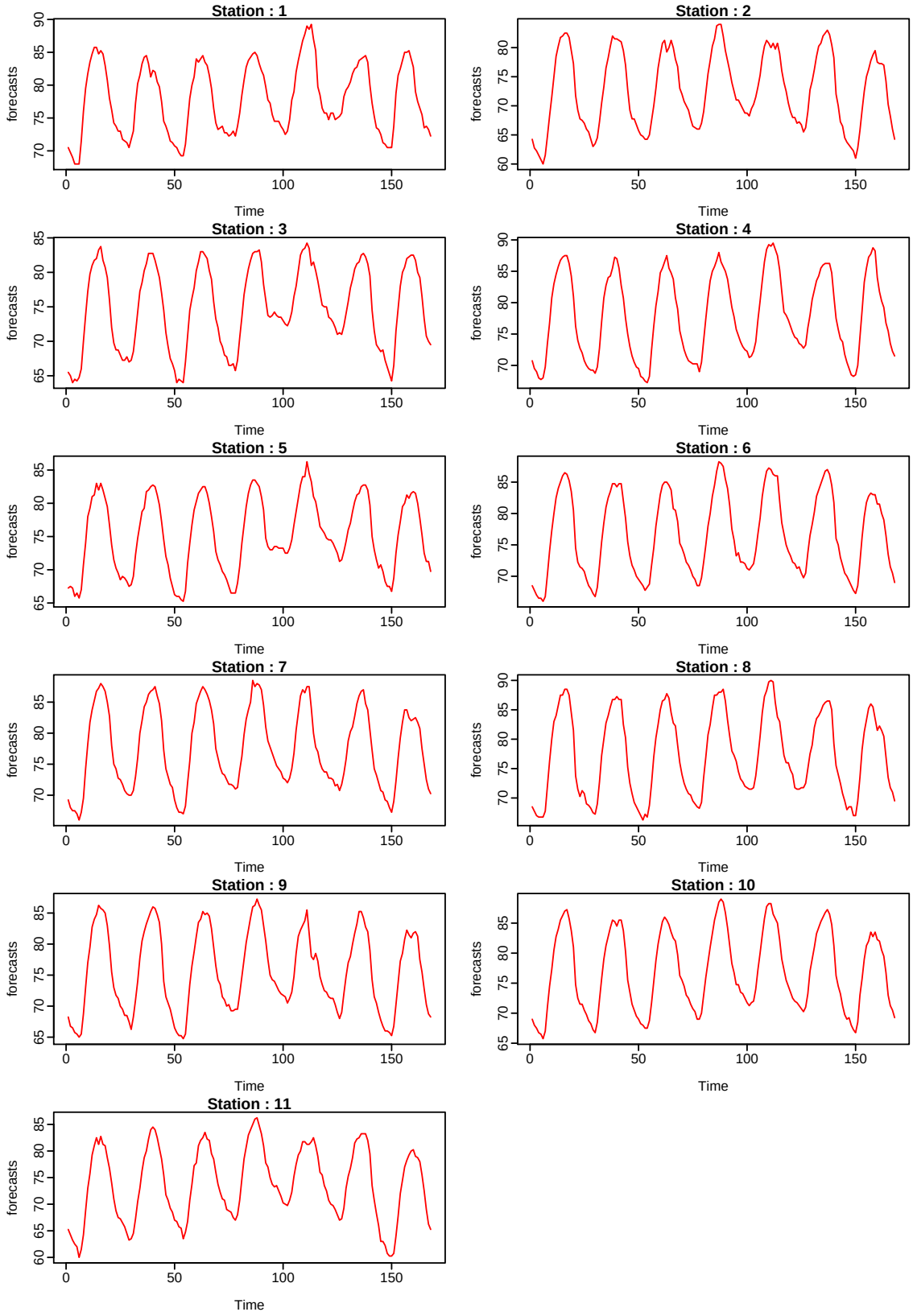

Figure 7.17: *Forecasts of temperature for the eleven stations.*

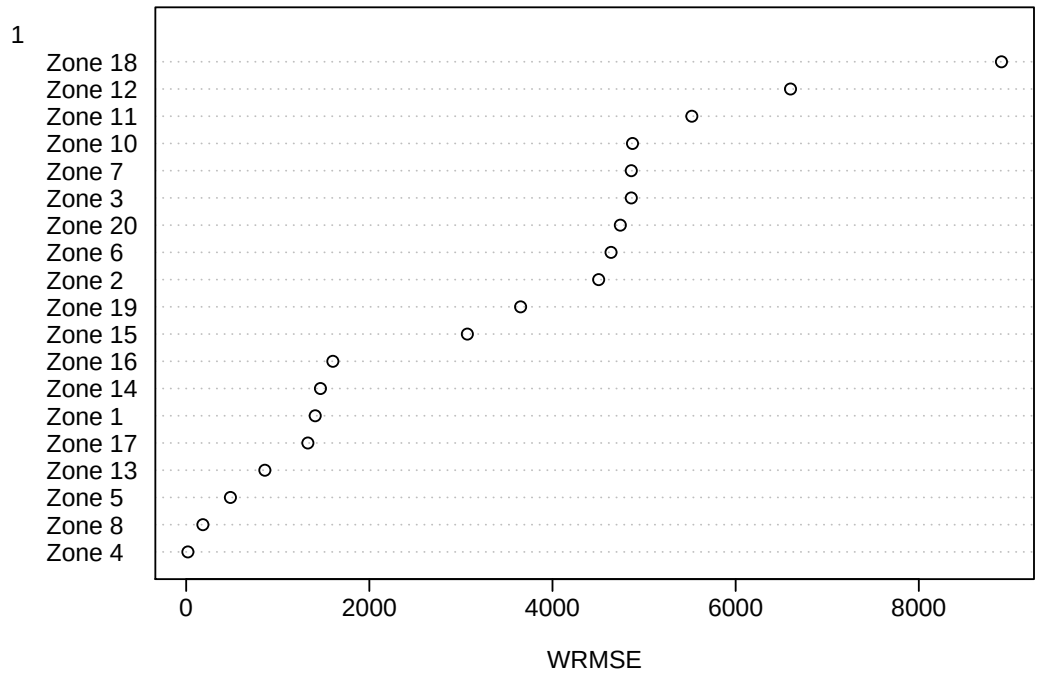

Figure 7.18: *Root mean squared error (RMSE) obtained for each zone on the testing week.*

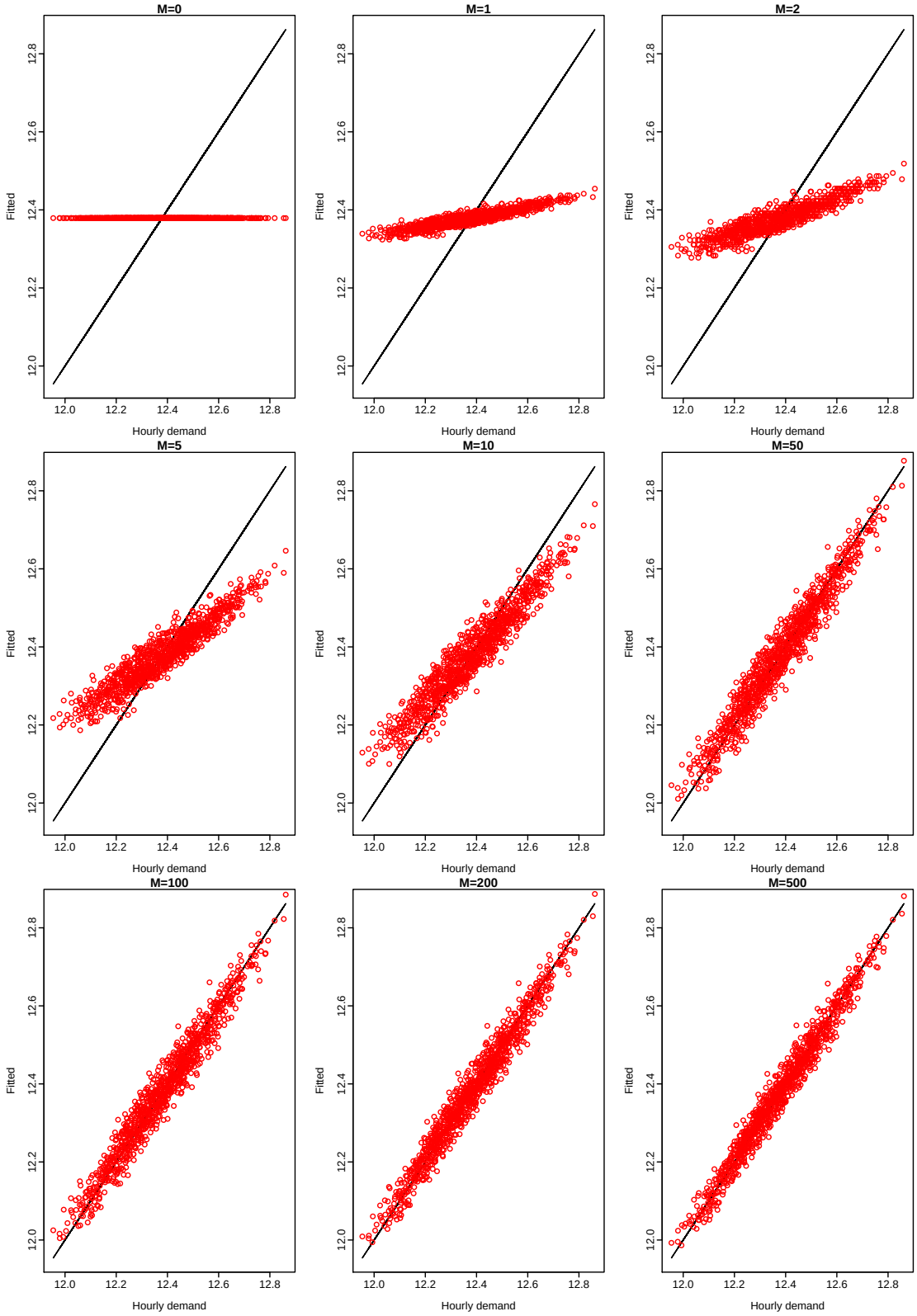

Figure 7.19: *$M = 500$ boosting iterations is the best in terms of cross-validation.*

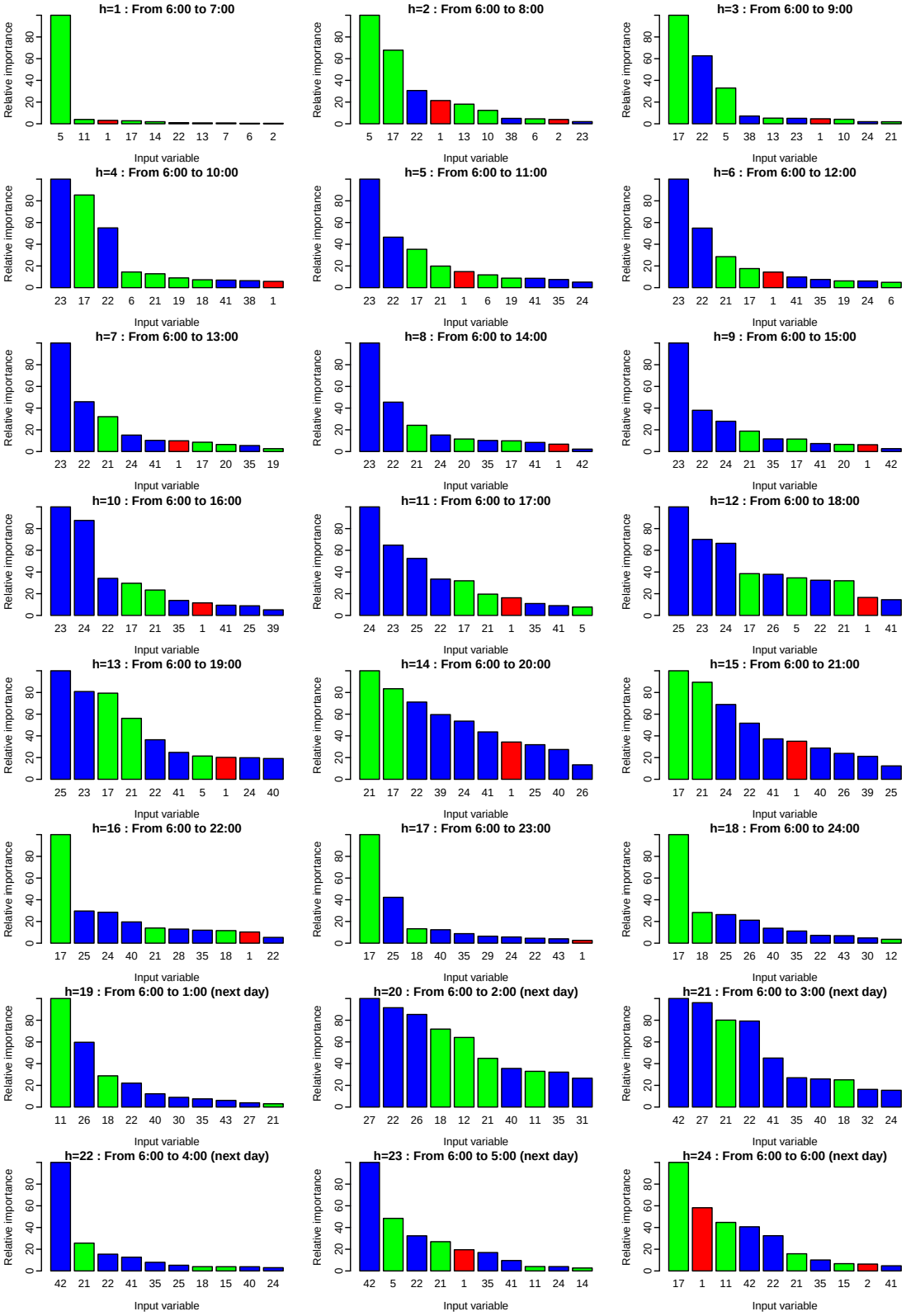

Figure 7.20: *Relative importance of the five first variables on the demand for different times of the day. Demand variables are colored in green, temperature variables in blue and calendar variables in red.*

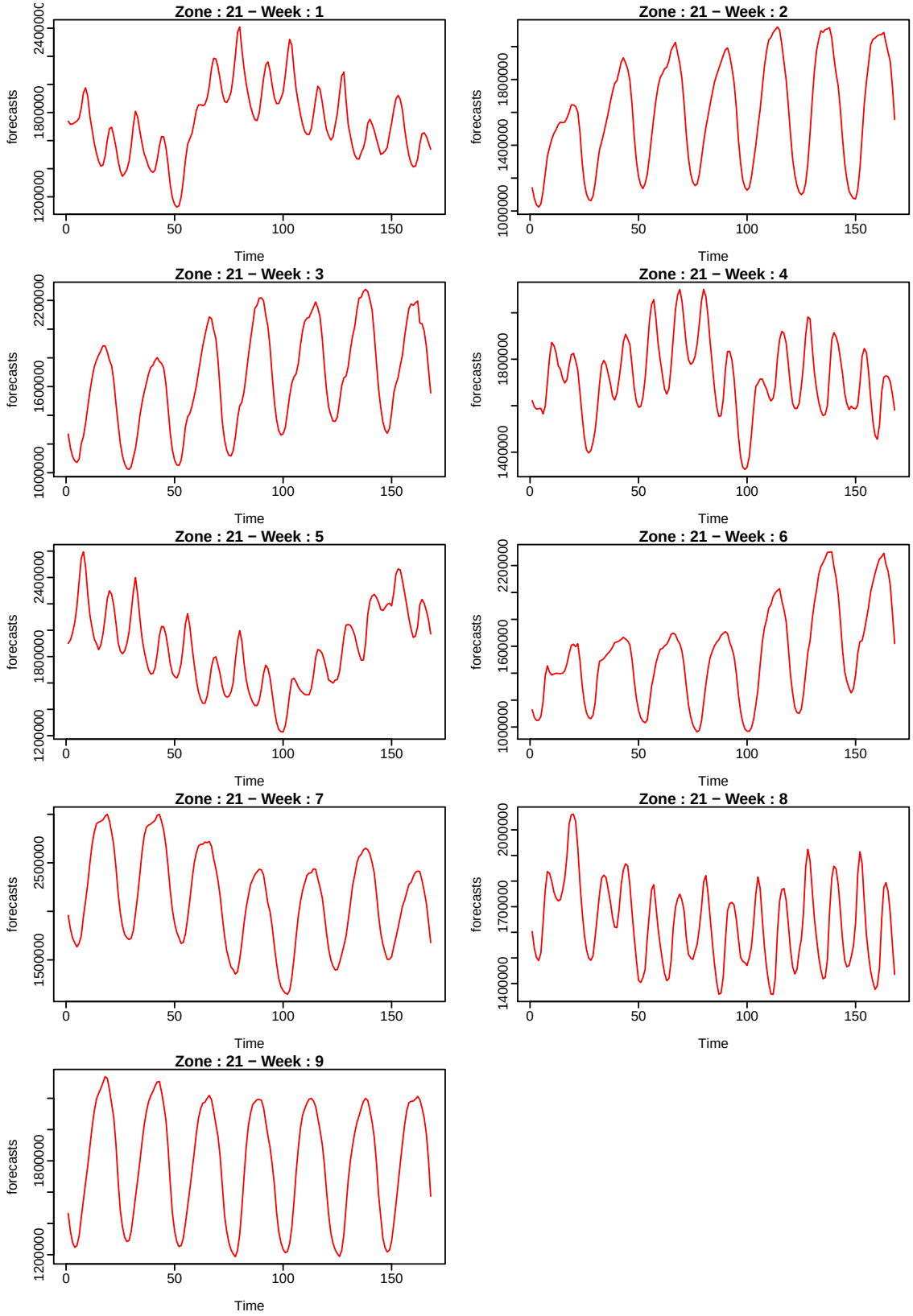

Figure 7.21: *Forecasts for zone* 21 *for the eight in-sample weeks and the out-of-sample week.*

Chapter 8

Conclusions and directions for future works

With the massive amount of data collected every day, learning from data has become a key tool for knowledge discovery, and as a result, machine learning is a growing field of interest in many fields of science (Abu-Mostafa, 2012). However, machine learning algorithms and methodologies have received little attention in the econometrics and forecasting literature, especially the multi-step-ahead forecasting problem. This thesis has contributed to the study and development of multi-step-ahead forecasting strategies based on algorithms and methodologies from machine learning.

Forecasting several future observations of a given univariate time series can be either applied recursively by iterating a one-step-ahead model forward for the desired number of steps, or directly using a specific model for each forecast horizon. A number of studies have focused on comparing these two strategies and investigating under which conditions one strategy is better than the other.

However, these studies have appeared in different fields, and the lack of communication between these fields has prevented a broad overview of the different developments in this area. To fill this gap, Chapter 3 provided an overview of the literature for both linear and nonlinear models, including some works involving machine learning models.

With linear models, the theoretical literature tends to conclude that model misspecification plays an important role in the relative performance between the recursive and direct strategies (Chevillon, 2007). In particular, when the model is correctly specified, recursive forecasts benefit from more efficient parameter estimation, but if the model is misspecified, direct forecasts are more robust. However, empirical studies often found superior performance of recursive linear forecasts compared to direct linear forecasts, especially for long horizons (Marcellino, Stock, and Watson, 2006; Pesaran, Pick, and Timmermann, 2011).

With nonlinear models, recursive forecasts are known to be asymptotically biased since they do not consider the innovation terms (Brown and Mariano, 1984; Lin and Granger, 1994). The direct strategy is often preferred over the recursive strategy since it avoids the accumulation of errors. In particular, empirical studies often found superior performance of the direct strategy compared to the recursive strategy (Atiya et al., 1999; Kline, 2004; Sorjamaa, Hao, Reyhani, et al., 2007).

When comparing linear with nonlinear models, it has been found that nonlinear models have often larger forecast errors than linear models, especially with economic time series. Arguments put forward include the weak nonlinearity of the time series, the rare occurrence of nonlinear features and the overfitting with short time series (Stock and Watson, 1999; Teräsvirta, 2006; Teräsvirta, Tjostheim, and Granger, 2010).

The review performed in Chapter 3 has pointed out a number of limitations of the current studies that compare recursive forecasts with direct forecasts. It was found that the majority of studies have focused on forecasts generated with linear models. In particular, less work has been done with nonlinear models, even less with machine learning models. Also, among the few in-depth studies that have considered machine learning algorithms, many have either focused on the neural network model or have been limited to one-step-ahead forecasting (Berardi and Zhang, 2003; Ahmed et al., 2010).

We have addressed these different issues in Chapter 4 where we performed an in-depth bias and variance study to compare recursive and direct forecasts using different machine learning algorithms for both linear and nonlinear DGPs. Our study has also investigated the role of the time series length and the forecast horizon in the performance of the recursive and direct strategies.

Overall, the results have shown that the accuracy of the recursive and direct forecasts depend on different factors, including the learning algorithm, the DGP, the time series length and the forecast horizon. This has emphasized the difficulty of choosing between recursive and direct forecasts in real-world applications.

Among the main results, we have found that recursive linear forecasts have good performance in many different scenarios, even when the DGP is nonlinear, notably due to its reduced variance, especially for short time series and long horizons. These results have also emphasized the fact that weakly nonlinear time series and/or short time series will favor linear forecasts over nonlinear forecasts, notably due to the overfitting problem (Teräsvirta, 2006).

Our results have also indicated that machine learning models provide better performance with direct forecasts and long time series. However, direct forecasts suffer from a large variance at long horizons with short time series. Better performance has been found with machine learning models that have the ability to switch to the linear model, such as neural networks.

Motivated by the results of the bias and variance analysis of Chapter 4, we proposed new forecasting strategies in Chapters 5 and 6 to overcome some of the current limitations. All the proposed strategies have been evaluated with both simulated time series as well as with real-world time series from the M3 and NN5 forecasting competitions.

In Chapter 5, we introduced multi-stage forecasting strategies to address the difficult task of choosing between recursive and direct forecasts. Rather than treating recursive and direct strategies as competitors, multi-stage strategies combine the best properties of both strategies.

The main idea is to produce multi-step recursive linear forecasts, and then adjust these forecasts using direct nonlinear models, which model the multi-step linear forecast errors at each horizon. By doing so, the multi-stage strategies allow nonlinearity and interactions between lagged variables at each forecast horizon, and can benefit from the reduced variance of recursive linear forecasts, provided the direct rectification models do not increase the variance too much. We proposed a first multi-horizon strategy, called the rectify strategy, where the rectification models are estimated using nearest neighbors.

Because recursive linear forecasts often need few adjustments at each horizon with real-world time series, we considered a second strategy, called the boost strategy, that estimates the rectification models using gradient boosting algorithms that involve the so-called weak learners. The boost strategy is particularly useful since boosting algorithms have received little attention in the forecasting literature, and it provides a procedure for applying boosting algorithms to multi-step forecasting problems. When comparing the rectify and boost strategies, we found that the boost strategy outperforms the rectify strategy, notably due to the resistance to overfitting of the gradient boosting algorithms.

Compared to the recursive and direct strategies, the boost strategy provides either close or better performance than the best between recursive and direct strategies. In particular, we have observed a large decrease in errors especially for short horizons with the boost strategy compared to the recursive and direct forecasts. This has shown the advantage of modeling part of the signal with the linear model which allows simpler direct rectification models compared to the models of the direct strategy.

The results that we obtained with real-world time series suggested that the performance of the boost strategy is either better or closely related to the performance of the recursive linear forecasts. In other words, provided that the recursive linear forecasts provide a first good approximations, the boost strategy will improve the forecasts. Similar results have been observed with the rectify strategy.

Overall, multi-stage strategies are good alternatives to the recursive or direct strategy since they avoid the problem of choosing between one of these two strategies, and at the same time, often provide comparable or better forecasts than the recursive and direct strategies with both linear and nonlinear models. In particular, the boost strategy can benefit from the resistance to overfitting of the gradient boosting algorithms.

In Chapter 6, we have investigated whether we can improve direct forecasts generated by machine learning models that have been selected independently at each horizon, which typically suffer from a high variance especially with short time series at long horizons.

We proposed the multi-horizon forecasting strategies that exploit the information contained in other horizons when learning the model for each horizon. In particular, we proposed to select the lag order and the hyperparameters of each model by minimizing forecast errors over multiple horizons rather than just the horizon of interest, but we still allow the parameters to be independently selected for each horizon.

Depending on which horizons are considered for each model, a different multi-horizon strategy is defined and different lag order and hyperparameters will be selected. We have considered the extreme case where all the horizons are used for each model, which is the opposite of the case where the models are independently selected. We also considered an intermediate configuration where only nearby horizons are used for each model. The two strategies were implemented with both nearest neighbors and neural networks.

We have found that the multi-horizon strategies are very effective in reducing the forecast variance compared to the single-horizon strategies where each model is selected independently at each horizon. The largest decrease in variance has been observed with the strategy that constrains all the models to use the same lag order and hyperparameters. However, this strategy suffers from a larger bias at short horizons which can increase the forecast errors when the bias becomes more important than the variance. We did not observe a larger bias at short horizons with the strategy that only consider local horizons, and the variance was still lower for long horizons compared to selecting each model independently. However, the decrease in variance is lower compared to the strategy that considers all horizons for each model.

Finally, the advantage of multi-horizon strategies has been more noticeable with linear or weakly nonlinear DGPs. Also, we observed a better performance of multi-horizon strategies with neural networks compared to nearest neighbors.

In summary, the results have suggested that the multi-horizon strategies will provide better forecasts when the conditional mean does not change too much with the horizon, the machine learning model is highly flexible and the time series is short.

Last but not least, a key contribution of this thesis has been the participation in the Load Forecasting track of the *Kaggle Global Energy Forecasting Competition 2012*, an international forecasting competition that involved a hierarchical load forecasting problem for a US utility with twenty geographical zones. The data available consisted of the hourly loads for the twenty zones and hourly temperatures from eleven weather stations, for four and a half years. For each zone, the hourly electricity loads for nine different weeks needed to be predicted without having the locations of either the zones or stations. We used separate models for each hourly period, with component-wise gradient boosting for estimating each model using univariate P-splines as base learners. The models allow for the electricity demand changing with the time-of-year, day-of-week, time-of-day, and on public holidays, with the main predictors being current and past temperatures, and past demand.

Our entry ranked fifth out of 105 participating teams, and is described and analyzed in Chapter 7. The good ranking of our approach suggests that our modelling strategy is competitive with the other models used to forecast the electricity demand. We have identified several aspects that makes our modelling strategy successful.

First, data analysis is a key step that allows to identify useful variables to consider in the model. The data analysis step was also important for identifying and cleaning the data from any corrupted information for better model performance.

Second, we used different models including different effects for each hour of the day to model the demand patterns which are changing throughout the day.

Third, each hourly model allowed nonlinear and nonparametric terms to be included in the final model. This provided a great flexibility to the model and avoided making too many assumptions about the data generating process.

Finally, gradient boosting has proven many times to be an effective prediction algorithm for both classification and regression tasks. By selecting the number of components included in the model, we can easily control the so-called bias-variance trade-off in the estimation. In addition, component-wise gradient boosting increases the attractiveness of boosting by adding automatic variable selection during the fitting process.

The Kaggle load forecasting competition was a challenging prediction task which required solving several statistical problems such as data cleaning, variable selection, regression and multi-step time series forecasting.

The main message of this thesis can be summarized as follows: algorithms and methodologies from machine learning have been neglected in the forecasting literature, especially for the multi-step-ahead forecasting problem. However, we have shown that the performance of multi-step-ahead forecasts generated with machine learning algorithms can be significantly improved if an appropriate strategy is used to generate the forecasts and to select the lag order and the hyperparameters of the models. Our findings and experiences raised several interesting questions that invite future research at the intersection of time series forecasting and machine learning, as summarized in the following section.

8.1 Limitations and future work

This thesis is a first step towards a larger research agenda that aims at developing new machine learning algorithms for time series forecasting. In this section, we describe some limitations of our work and provide potential directions for future research.

We have exclusively focused on univariate time series while in practice we often encounter multivariate forecasting problems where we are required to forecast a set of possibly dependent time series. Although a multivariate forecasting problem can be reduced to a set of independent univariate forecasting problems, taking into account the interdependence between the time series may improve the forecast accuracy. An important future direction is to extend the strategies developed in this thesis to the multivariate setting.

Although many machine learning algorithms have been successful in applications such as in Bioinformatics, where the data is typically high-dimensional (Larrañaga et al., 2006), there is a fundamental difference between multivariate time series and the type of data that have been considered in the machine learning literature (Caruana, Karampatziakis, and Yessenalina, 2008). In fact, multivariate time series are typically highly inter-related and correlated over time (Stock and Watson, 2006). Therefore, the development of new machine learning algorithms that can deal with high-dimensional multivariate time series is a key future challenge.

We have considered several machine learning models in this work, including neural networks, nearest neighbors and gradient boosting. However, a number of other models have been proposed in the machine learning literature (e.g. support vector machines and random forests) (Hastie, Tibshirani, and Friedman, 2009), and it would be interesting to investigate their performance on forecasting problems. Furthermore, except for neural networks, machine learning algorithms have rarely been compared with nonlinear time series models such as threshold and smooth transition autoregressive models (Teräsvirta, Tjostheim, and Granger, 2010). A comparison between these different types of models could lead to important insights about their differences.

Although real-world time series are expected to behave nonlinearly (Kantz and Schreiber, 2004), we have found that nonlinear forecasts are more prone to overfitting compared to linear forecasts, especially with short time series, as confirmed by Teräsvirta (2006). In consequence, the development of new nonlinear forecasting methods should focus on how to reduce the overfitting phenomenon, and a deeper investigation of the role and benefits of nonlinear models for real-world time series forecasting would be very useful.

In particular, the combination of forecasts generated by different machine learning models has been proven to be very effective against overfitting (Andrawis, Atiya, and El-Shishiny, 2011). Therefore, a deeper investigation of the benefits of the so-called ensemble methods (Dietterich, 2000) for multi-step forecasting would be very useful. For example, bagging has received little attention in the context of time series forecasting (Inoue and Kilian, 2008; Adeodato et al., 2009). One reason is the difficulty of generating bootstrap replicates for time series data (Kreiss and Paparoditis, 2011), which in itself can be an important future contribution.

Cross-validation is the most popular method to select the hyperparameters of machine learning models. However, cross-validation (as usually applied) is known to be inappropriate with time series data due to the time dependence between observations (Opsomer, Wang, and Yang, 2001). Except for some works in kernel regression (Brabanter and Brabanter, 2011) and some experimental comparisons of cross-validation methods (Bergmeir and Benítez, 2012), there has been little work that has extended cross-validation methods for multi-step time series forcasting. We believe the study of cross-validation methods for time series data is an important area of future research. In particular, an in-depth study of cross-validation methods for time series forecasting is still missing.

The VC theory that we briefly discussed in Section 2.1.3 is a non-asymptotic theory (Boucheron, Lugosi, and Massart, 2013) that has been very useful in studying learning algorithms. However, the VC theory has been initially developed for independent and identically distributed data. There have been some recent research works on the learning theory for autoregressive models (Mcdonald, Shalizi, and Schervish, 2011) and on model selection for time series forecasting (Alquier and

Wintenberger, 2012). We believe the extension of the learning theory for time series data is a promising future direction.

Although we have performed an in-depth bias and variance study in Chapter 4 and we have considered real-world time series from both the M3 and NN5 competitions in Chapter 5, a large-scale comparison between recursive and direct forecasts for time series with very different characteristics (e.g. the 10,000 time series from the M4 competition[1]) seems not yet to be available. Such comparison would be very useful for better understanding the differences between the two strategies. Also, as far as the selection of recursive and direct forecasts is concerned, a meta-learning approach (Lemke and Gabrys, 2010; Smith-Miles, 2008) has not yet been proposed. This approach can be very useful for selecting between the two strategies in real-world applications (Wang, Smith-Miles, and Hyndman, 2009).

For the rectify strategy that we proposed in Chapter 5, an important future work is a deeper exploration of the different degrees of freedom of the strategy. For example, it would be interesting to investigate the impact of a weaker base model than the best AR model as well as alternative weak learners for the rectification models. Furthermore, a study on the effect of the number of boosting iterations on the bias and variance components would be very useful to deeply understand the strategy. Finally, more investigation on the benefit of boosting algorithms for time series forecasting, especially for multivariate time series, is an important future direction (Bai and Ng, 2009; Buchen and Wohlrabe, 2011).

For the multi-horizon strategies we developed in Chapter 6, a natural extension is to automatically select the horizons to consider in the objective of each model. Also, we can consider alternative forms of regularization, for example by imposing to all the hyperparameters to be close to the average value of the hyperparameters for all or a subset of the horizons (Evgeniou and Pontil, 2004). Finally, although methods such as PLS had little success with univariate time series (Franses and Legerstee, 2009), these methods may provide better results with multivariate time series.

We have mainly considered the mean squared error (MSE) as a forecast accuracy measure, and as a result, multi-step forecasting reduced to estimation of the conditional mean at each forecast horizon. Although using MSE allowed us to decompose the errors into bias and variance components, it is important to note that MSE is not always an appropriate forecast accuracy measure (Hyndman and Koehler, 2006). Furthermore, Gneiting (2011) pointed out the importance of matching the fitting criteria with the forecast accuracy measure. It would be interesting to investigate the impact of using different fitting criteria and forecast accuracy measures in real-world forecasting tasks.

Because we have only estimated the conditional mean, the forecasts we have generated are called *point forecasts*. One weakness of point forecasts is that they provide no description of the uncertainty associated to the forecasts. A better alternative is the density forecasts which provide a complete description of the associated uncertainty (Tay and Wallis, 2000; Corradi and Swanson, 2006). Intermediate forecasts between these two forecasts are called *prediction intervals* (Chatfield, 2000), and an interval within which the true future values will fall with specified probability. Except for neural networks (Khosravi et al., 2011a; Khosravi et al., 2011b), generating interval or density forecasts with machine learning models has received little attention and is a key challenge for future works (Shrestha and Solomatine, 2006). We refer to Gneiting and Katzfuss (2014) for a recent overview of probabilistic forecasting.

We are entering the era of *Big data* where massive amounts of data are continuously produced and stored every day. Big data not only brings opportunities but also many computational and statistical challenges (Hand, 2013; Fan, Han, and Liu, 2014). In the time series forecasting context, Big data is expected to play an important role in dealing with a large amount of time series rather

[1] http://m4competition.com/

than "Bigger" time series. In fact, a large number of time series that are inter-related and correlated over time are collected in many fields of science, and making sense of all these time series is a key future challenge. In fact, in the forecasting literature, many studies have considered a limited number of time series, but nowadays millions of time series are available.

Forecasting is of course one important application with all these time series, especially the development of large-scale forecasting models that can handle a large number of time series (Agarwal et al., 2010). Another important application is the study of the dependence between a set of time series over time in order to find key variables that govern certain phenomenon (Liu et al., 2010).

The problem of *hierarchical forecasting* is one example that can greatly benefit from large-scale forecasting models. Time series can often be represented in a hierarchical structure, which includes a bottom level, several intermediate levels and a top level (Fliedner, 2001). In contrast to the conventional forecasting framework, hierarchical forecasting requires an additional constraint: that the sum of the forecast at one level must be equal to the forecast at the superior level across the whole hierarchy.

Forecasting hierarchical time series is considered a very challenging problem in the forecasting literature (Hyndman, Ahmed, et al., 2011). The various time series of the hierarchy can interact in varying and complex ways. The complex dependencies between the time series can make a change in one series at one level have an important impact on other series at the same level as well as series at higher and lower levels. Furthermore, time series at different levels of the hierarchy can be of very different type. For example, time series at the bottom level are typically very noisy while series at higher levels are much smoother. Finally, in practice the hierarchy can be very large with millions of time series; e.g., time series representing a quantity for a whole country disaggregated by states, cities and homes.

Large-scale forecasting models will be useful in many applications including energy and business applications (Hong, 2014; Varian, 2014). For example, as a result of the modernization process of the electrical grid (also called the smart grid), utilities are facing a "data deluge" with all the communication devices such as "smart meters", which record and transmit electric consumption information at 15 minute or hourly intervals (Depuru, Wang, and Devabhaktuni, 2011). In fact, although the smart grid offers great potential for a more reliable grid, it also brings new challenges such as demand-response forecasting (Balijepalli et al., 2011) and renewable energies integration, such as wind and solar energies (Yin et al., 2013; Pinson, 2013).

Finally, when we talk about time series, we often think about a sequence of numbers. However, time series can include a large variety of data types (Lam, 2014), such as a set of evolving images (Verbesselt et al., 2010), texts (Blei and Lafferty, 2006) or networks (Kim and Leskovec, 2013). An important future challenge is to develop methods that can handle different types of data (Wasserman, 2014), and we may benefit from all the methods that have already been developed for traditional time series.

The recent book "Past, Present, and Future of Statistical Science" (Lin, Genest, et al., 2014) is an excellent reference for future work not only for time series forecasting but for statistics and machine learning in general.

Bibliography

Abu-Mostafa, YS (June 2012). Machines that Think for Themselves. *Scientific American* **307**(1), 78–81.

Abu-Mostafa, YS, M Magdon-Ismail, and HT Lin (2012). *Learning From Data*. AMLBook.

Adeodato, PJ, AL Arnaud, GC Vasconcelos, RC Cunha, and DS Monteiro (August 2009). MLP ensembles improve long term prediction accuracy over single networks. *International Journal of Forecasting* **27**(3), 1–11.

Agarwal, D, D Chen, Lj Lin, J Shanmugasundaram, and E Vee (2010). Forecasting high-dimensional data. In: *Proceedings of the 2010 international conference on Management of data - SIGMOD10*. New York, USA: ACM Press, pp.1003–1012.

Ahmed, N, AF Atiya, NE Gayar, and H El-Shishiny (September 2010). An Empirical Comparison of Machine Learning Models for Time Series Forecasting. *Econometric Reviews* **29**(5), 594–621.

Akaike, H (1969). Fitting autoregressive models for prediction. *Annals of the institute of statistical mathematics* **21**(1), 243–247.

Allen, D (1974). The Relationship Between Variable Selection and Data Augmentation and a Method for Prediction. *Technometrics* **16**(1), 125–127.

Alquier, P and O Wintenberger (August 2012). Model selection for weakly dependent time series forecasting. *Bernoulli* **18**(3), 883–913.

Altman, N (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician* **46**(3), 175–185.

Andrawis, R, A Atiya, and H El-Shishiny (2011). Forecast combinations of computational intelligence and linear models for the NN5 time series forecasting competition. *International Journal of Forecasting* **27**(3), 1–29.

Arlot, S and A Celisse (2010). A survey of cross-validation procedures for model selection. *Statistics Surveys* **4**, 40–79.

Assaad, M, R Boné, and H Cardot (January 2008). A new boosting algorithm for improved time-series forecasting with recurrent neural networks. *Information Fusion* **9**(1), 41–55.

Athanasopoulos, G and R Hyndman (September 2011). The tourism forecasting competition. *International Journal of Forecasting* **27**(3), 822–844.

Atiya, AF, SM El-shoura, SI Shaheen, and MS El-sherif (1999). A comparison between neural-network forecasting techniques–case study: river flow forecasting. *IEEE Transactions on Neural Networks* **10**(2), 402–409.

Atkeson, C, A Moore, and S Schaal (1997). Locally weighted learning. *Artificial intelligence review* **11**(1), 11–73.

Audrino, F and P Bühlmann (June 2009). Splines for financial volatility. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **71**(3), 655–670.

Bai, J and S Ng (2009). Boosting diffusion indices. *Journal of Applied Econometrics* **24**(4), 607–629.

Balijepalli, V, V Pradhan, SA Khaparde, and RM Shereef (2011). Review of demand response under smart grid paradigm. In: *Innovative Smart Grid Technologies - India (ISGT India), 2011 IEEE PES*, pp.236–243.

Baltagi, B (2001). *A companion to theoretical econometrics*. Ed. by BH Baltagi. Blackwell companions to contemporary economics [1]. Oxford [u.a.]: Blackwell.

Barber, D, AT Cemgil, and S Chiappa (2011). *Bayesian Time Series Models*. Bayesian Time Series Models. Cambridge University Press.

Bellman, R and RE Bellman (1961). *Adaptive Control Processes: A Guided Tour*. 'Rand Corporation. Research studies. Princeton University Press.

Ben Taieb, S and AF Atiya (2014). "A bias and variance analysis for multi-step time series forecasting." Submitted to IEEE Transactions on Neural Networks and Learning Systems (under revision).

Ben Taieb, S and G Bontempi (December 2011). Recursive Multi-step Time Series Forecasting by Perturbing Data. In: *Proceedings of the 11th IEEE International Conference on Data Mining (ICDM)*. IEEE, pp.695–704.

Ben Taieb, S, G Bontempi, A Atiya, and A Sorjamaa (2012). A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *Expert Systems with Applications* **39**(8), 7067–7083.

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 (IJCNN)*. Atlanta, U.S.A., pp.3054–3061.

Ben Taieb, S and RJ Hyndman (August 2013). A gradient boosting approach to the Kaggle load forecasting competition. *International Journal of Forecasting*, 1–19.

Ben Taieb, S and RJ Hyndman (2014a). Boosting multi-step autoregressive forecasts. In: *Proceedings of the 31th International Conference on Machine Learning (ICML)*, pp.109–117.

Ben Taieb, S and RJ Hyndman (2014b). "Recursive and direct multi-step forecasting: the best of both worlds." Submitted to International Journal of Forecasting (under revision).

Ben Taieb, S, A Sorjamaa, and G Bontempi (2010). Multiple-output modeling for multi-step-ahead time series forecasting. *Neurocomputing* **73**(10-12), 1950–1957.

Berardi, VL and GP Zhang (January 2003). An empirical investigation of bias and variance in time series forecasting: modeling considerations and error evaluation. *Neural Networks, IEEE Transactions on* **14**(3), 668–79.

Bergmeir, C and JM Benítez (May 2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences* **191**, 192–213.

Bhansali, RJ (1999). Parameter estimation and model selection for multistep prediction of time series : a review. In: *Asymptotics, Nonparametrics and Time Series*. Ed. by S. Gosh. New York: Marcel Dekker.

Bhansali, RJ (2002). Multi-Step Forecasting. *A Companion to Economic Forecasting* (1), 206–221.

Bhansali, R (1996). Asymptotically efficient autoregressive model selection for multistep prediction. *Annals of the Institute of Statistical Mathematics* **48**(3), 577–602.

Bhansali, R (1997). Direct autoregressive predictors for multistep prediction: Order selection and performance relative to the plug in predictors. *Statistica Sinica* **7**, 425–449.

Bhansali, R and P Kokoszka (April 2002). Computation of the forecast coefficients for multistep prediction of long-range dependent time series. *International Journal of Forecasting* **18**(2), 181–206.

Blei, DM and JD Lafferty (2006). Dynamic topic models. In: *Proceedings of the 23rd international conference on Machine learning (ICML 2006)*, pp.113–120. arXiv: arXiv:0712.1486v1.

Bontempi, G and S Ben Taieb (January 2011). Conditionally dependent strategies for multiple-step-ahead prediction in local learning. *International Journal of Forecasting* **27**(3), 689–699.

Bontempi, G, S Ben Taieb, and YA Le Borgne (2013). Machine Learning Strategies for Time Series Forecasting. In: *Business Intelligence*. Ed. by MA Aufaure and E Zimányi. Vol. 138. Lecture Notes in Business Information Processing. Springer, pp.62–77.

Bontempi, G, M Birattari, and H Bersini (1999). Local learning for iterated time series prediction. In: *International Conference on Machine Learning*. In, pp.32–38.

Boor, C de (2001). *A Practical Guide to Splines*. Applied Mathematical Sciences. Springer New York.

Boucheron, S, G Lugosi, and P Massart (2013). *Concentration Inequalities: A Nonasymptotic Theory of Independence*. OUP Oxford.

Box, GEP and GM Jenkins (1976). *Time series analysis: forecasting and control*. Holden-Day series in time series analysis and digital processing. Holden-Day.

Brabanter, KD and JD Brabanter (2011). Kernel regression in the presence of correlated errors. *The Journal of Machine Learning Research* **12**, 1955–1976.

Brahim-Belhouari, S and A Bermak (November 2004). Gaussian process for nonstationary time series prediction. *Computational Statistics & Data Analysis* **47**(4), 705–712.

Breiman, L (1999). Prediction games & arcing algorithms. *Neural computation* **11**(7), 1493–1517.

Breiman, L (2001). Statistical modeling: The two cultures. *Statistical Science* **16**(3), 199–215.

Breiman, L and J Friedman (1997). Predicting multivariate responses in multiple linear regression. eng. *Journal of the Royal Statistical Society - Series B (Statistical Methodology)* **59**(1), 3–54.

Brockwell, PJ and RA Davis (2002). *Introduction to time series and forecasting*. 2nd. Taylor & Francis.

Brodsky, J and C Hurvich (1999). Multi-step forecasting for long-memory processes. *Journal of Forecasting* **75**(September 1996).

Brown, B and R Mariano (1984). Residual-Based Procedures for Prediction and Estimation in a Nonlinear Simultaneous System. *Econometrica: Journal of the Econometric Society* **52**(2), 321–344.

Brown, P and J Zidek (1980). Adaptive multivariate ridge regression. *The Annals of Statistics* **8**(1), 64–74.

Buchen, T and K Wohlrabe (October 2011). Forecasting with many predictors: Is boosting a viable alternative? *Economics Letters* **113**(1), 16–18.

Bühlmann, P (April 2006). Boosting for high-dimensional linear models. *The Annals of Statistics* **34**(2), 559–583.

Bühlmann, P and T Hothorn (November 2007). Boosting Algorithms: Regularization, Prediction and Model Fitting. *Statistical Science* **22**(4), 477–505.

Bühlmann, P and B Yu (2003). Boosting With the L2 Loss: Regression and Classification. *Journal of the American Statistical Association* **98**(462), 324–339.

Burman, P, E Chow, and D Nolan (1994). A cross-validatory method for dependent data. *Biometrika* **81**(2), 351–358.

Burman, P and D Nolan (1992). Data dependent estimation of prediction functions. *Journal of Time Series Analysis* **13**(3), 189–207.

Carmack, PS, WR Schucany, JS Spence, RF Gunst, Q Lin, and RW Haley (January 2009). Far Casting Cross-Validation. *Journal of Computational and Graphical Statistics* **18**(4), 879–893.

Caruana, R (1997). Multitask Learning. *Machine learning* **28**(1), 41–75.

Caruana, R, N Karampatziakis, and A Yessenalina (2008). An empirical evaluation of supervised learning in high dimensions. In: *International Conference on Machine*, pp.96–103.

Casdagli, M (1989). Nonlinear prediction of chaotic time series. **35**(3), 335–356.

Chatfield, C (December 2000). *Time-Series Forecasting*. CRC Press. Chap. 3, p. 267.

Chen, R, L Yang, and C Hafner (August 2004). Nonparametric multistep-ahead prediction in time series analysis. eng. *Journal of the Royal Statistical Society, Series B* **66**(3), 669–686.

Chevillon, G (2007). Direct multi-step estimation and forecasting. *Journal of Economic Surveys* **21**(4), 746–785.

Chevillon, G (2008). Multi-step forecasting in the presence of weak trends, 1–40.

Chevillon, G and D Hendry (April 2005). Non-parametric direct multi-step estimation for forecasting economic processes. *International Journal of Forecasting* **21**(2), 201–218.

Chu, C and J Marron (1991). Comparison of two bandwith selectors with dependent errors. *The Annals of Statistics* **19**(4), 1906–1918.

Claeskens, G and NL Hjort (2008). *Model selection and model averaging*. Vol. 330. Cambridge University Press Cambridge.

Clements, M and D Hendry (1998). *Forecasting Economic Time Series*. Forecasting economic time series. Cambridge University Press.

Clements, MP and DF Hendry (2006). Chapter 12 Forecasting with Breaks. *Handbook of Economic Forecasting* **1**, 605–657.

Clements, M and D Hendry (May 1996). Multi-step estimation for forecasting. *Oxford Bulletin of Economics and Statistics* **48**(1-2), 135–149.

Cleveland, RB, WS Cleveland, JE McRae, and I Terpenning (1990). STL: A Seasonal-Trend Decomposition Procedure Based on Loess (with Discussion). *Journal of Official Statistics* **6**(1), 3–73.

Cordeiro, C (2009). Forecasting time series with BOOT.EXPOS procedure. *REVSTAT - Statistical Journal* **7**(2), 135–149.

Corradi, V and NR Swanson (2006). "Predictive Density Evaluation." In: ed. by G Elliott, C Granger, and A Timmermann. Vol. 1. Handbook of Economic Forecasting. Elsevier. Chap. 5, pp. 197–284.

Cover, TM and JA Thomas (2012). *Elements of information theory*. John Wiley & Sons.

Cox, DR (1961). Prediction by exponentially weighted moving averages and related methods. *Journal of the Royal Statistical Society, Series B* **23**(2), 414–422.

Crone, SF (2006). Forecasting with Computational Intelligence - An Evaluation of Support Vector Regression and Artificial Neural Networks for Time Series Prediction. *International Joint Conference on Neural Networks*, 3159–3166.

Crone, SF (2009a). Mining the past to determine the future: Comments. *International Journal of Forecasting* **25**(3), 456–460.

Crone, SF (2009b). *NN5 forecasting competition*. http://www.neural-forecasting-competition.com/NN5/datasets.htm (visited on 06/04/2014).

Crone, SF, M Hibon, and K Nikolopoulos (July 2011). Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction. *International Journal of Forecasting* **27**(3), 635–660.

De'Ath, G (2002). Multivariate Regression Trees : A New Technique for Modeling Species-Environment Relationships. *Ecology* **83**(4), 1105–1117.

Depuru, SSSR, L Wang, and V Devabhaktuni (August 2011). Smart meters for power grid: Challenges, issues, advantages and status. *Renewable and Sustainable Energy Reviews* **15**(6), 2736–2742.

Dietterich, TG (2000). Ensemble Methods in Machine Learning. *Lecture Notes in Computer Science*. Lecture Notes in Computer Science **1857**. Ed. by J Kittler and F Roli, 1–15.

Durbin, J and SJ Koopman (2001). *Time series analysis by state space methods*. Vol. 24. Oxford University Press, p. 253.

Efron, B (1979). Bootstrap methods: another look at the jackknife. *The annals of Statistics* **7**, 1–26.

Efron, B and RJ Tibshirani (1993). *An Introduction to the Bootstrap*. Vol. 57. CRC Press, p. 436.

Eilers, PH and BD Marx (June 2003). Multivariate calibration with temperature interaction using two-dimensional penalized signal regression. *Chemometrics and Intelligent Laboratory Systems* **66**(2), 159–174.

Eilers, P and B Marx (May 1996). Flexible smoothing with B-splines and penalties. *Statistical science* **11**(2), 89–121.

Elman, J (June 1990). Finding structure in time. *Cognitive Science* **14**(2), 179–211.

Evgeniou, T and M Pontil (2004). Regularized multi–task learning. In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp.109–117.

Fan, J, F Han, and H Liu (2014). Challenges of Big Data analysis. *National Science Review*, 1–38. arXiv: arXiv:1308.1479v1.

Fan, J and Q Yao (2003). *Nonlinear time series: nonparametric and parametric methods*. New York: Springer, p. 576.

Fan, S and L Chen (2006). Short-Term Load Forecasting Based on an Adaptive Hybrid Method. *IEEE Transactions on Power Systems* **21**(1), 392–401.

Fan, S and RJ Hyndman (2012). Short-term load forecasting based on a semi-parametric additive model. *IEEE Transactions on Power Systems* **27**(1), 134–141.

Fay, D, JV Ringwood, M Condon, and M Kelly (2003). 24h electrical load data - a sequential or partitioned time series? *Neurocomputing* **55**(3-4), 469–498.

Fiebig, D (2001). Seemingly unrelated regression. *A companion to theoretical econometrics*, 101–121.

Findley, DF (1983). On the use of multiple models for multi-period forecasting. In: *Proceedings of Business and Economic Statistics, American Statistical Association*, pp.528–531.

Findley, D, B Pötscher, and C Wei (2004). Modeling of time series arrays by multistep prediction or likelihood methods. *Journal of econometrics* **118**(1-2), 151–187.

Fliedner, G (2001). Hierarchical forecasting: issues and use guidelines. *Industrial Management and Data Systems* **101**(1), 5–12.

Franses, PH and R Legerstee (July 2009). A unifying view on multi-step forecasting using an autoregression. *Journal of Economic Surveys* **24**(3), 389–401.

Freund, Y and RRE Schapire (1996). Experiments with a New Boosting Algorithm. In: *International Conference on Machine Learning*, pp.148–156.

Freund, Y and RE Schapire (August 1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences* **55**(1), 119–139.

Friedman, J and T Hastie (2000). Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors). *The annals of statistics* **28**(2), 337–407.

Friedman, JH (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics* **29**(5), 1189–1232.

Friedman, J (March 1998). Data mining and statistics: what's the connection? de. *Computing Science and Statistics* **29**(1), 3–9.

Fukumizu, K, F Bach, and A Gretton (2007). Statistical consistency of kernel canonical correlation analysis. *The Journal of Machine Learning Research* **8**, 361–383.

Gardnerjr, E (2006). Exponential smoothing: The state of the art - Part II. *International Journal of Forecasting* **22**(4), 637–666.

Geman, S, E Bienenstock, and R Doursat (1992). Neural Networks and the Bias/Variance Dilemma. *Neural Computation* **4**(1), 1–58.

Girard, A, C Rasmussen, and J Candela (2003). Gaussian process priors with uncertain inputs-application to multiple-step ahead time series forecasting. *Neural Information Processing Systems*, 545–552.

Gneiting, T (2011). Making and evaluating point forecasts. *Journal of the American Statistical Association* **106**(494), 746–762. arXiv: arXiv:0912.0902v2.

Gneiting, T and M Katzfuss (January 2014). Probabilistic Forecasting. *Annual Review of Statistics and Its Application* **1**(1), 125–151.

Gooijer, JGD and RJ Hyndman (2006). 25 years of time series forecasting. *International Journal of Forecasting* **22**(3), 443–473.

Greene, WH (2012). *Econometric analysis*. 7th ed. Prentice Hall.

Grigorievskiy, A, Y Miche, AM Ventelä, E Séverin, and A Lendasse (March 2014). Long-term time series prediction using OP-ELM. *Neural networks : the official journal of the International Neural Network Society* **51**, 50–56.

Haggan, V and T Ozaki (1981). Modelling nonlinear random vibrations using an amplitude-dependent autoregressive time series model. *Biometrika* **68**(1), 189–196.

Hamzaçebi, C, D Akay, and F Kutay (2009). Comparison of direct and iterative artificial neural network forecast approaches in multi-periodic time series forecasting. *Expert Systems with Applications* **36**(2,), 3839–3844.

Hand, DJ (May 1998). Data Mining: Statistics and More? *The American Statistician* **52**(2), 112.

Hand, DJ (July 2009a). Mining the past to determine the future: Problems and possibilities. *International Journal of Forecasting* **25**(3), 441–451.

Hand, DJ (July 2009b). Mining the past to determine the future: Rejoinder. *International Journal of Forecasting* **25**(3), 461–462.

Hand, D (2013). Data, Not Dogma: Big Data, Open Data, and the Opportunities Ahead. *Advances in Intelligent Data Analysis XII*, 1–12.

Hardoon, DR and J Shawe-Taylor (November 2010). Sparse canonical correlation analysis. *Machine Learning* **83**(3), 331–353.

Hardoon, DR, S Szedmak, and J Shawe-Taylor (December 2004). Canonical correlation analysis: an overview with application to learning methods. *Neural computation* **16**(12), 2639–64.

Hart, D and TE Wehrly (1986). Using Repeated Estimation Regression Kernel Data Measurements. *Journal of the American Statistical Association* **81**(396), 1080–1088.

Hart, J (1994). Automated kernel smoothing of dependent data by using time series cross-validation. *Journal of the Royal Statistical Society. Series B (Methodological)* **56**(3), 529–542.

Hastie, TJ and RJ Tibshirani (1990). *Generalized additive models*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Chapman & Hall/CRC.

Hastie, T, R Tibshirani, and J Friedman (2009). *The elements of statistical learning: data mining, inference and prediction*. 2nd ed. Springer.

Haywood, J and GT Wilson (1997). Fitting Time Series Models by Minimizing Multistep-ahead Errors: a Frequency Domain Approach. *Journal of the Royal Statistical Society B* **59**(1), 237–254.

Hechenbichler, KSK̃ (2014). *kknn: Weighted k-Nearest Neighbors*. http://cran.r-project.org/package=kknn.

Hendry, DF and GE Mizon (2013). "Open-Model Forecast-Error Taxonomies." In: *Recent Advances and Future Directions in Causality, Prediction, and Specification Analysis*, pp.219–240.

Hendry, D (2000). A general forecast-error taxonomy. In: *Econometric Society World Congress*, pp.1–17.

Hoerl, AE and RW Kennard (1970). Ridge Regression: Applications to Nonorthogonal Problems. *Technometrics* **12**, 69–82.

Holan, SH, R Lund, and G Davis (2010). The ARMA alphabet soup: A tour of ARMA model variants. *Statistics Surveys* **4**, 232–274.

Hong, T (2010). "Short Term Electric Load Forecasting." Doctoral dissertation.

Hong, T (2014). Energy forecasting: Past, present, and future. *Foresight: The International Journal of Applied Forecasting*, 43–49.

Hong, T, P Pinson, and S Fan (August 2013). Global Energy Forecasting Competition 2012. *International Journal of Forecasting* **38**(2), 357–363.

Hothorn, T, P Bühlmann, T Kneib, M Schmid, and B Hofner (2010). Model-based boosting 2.0. *The Journal of Machine Learning Research* **11**, 2109–2113.

Huang, GB, QY Zhu, and CK Siew (December 2006). Extreme learning machine: Theory and applications. *Neurocomputing* **70**(1), 489–501.

Hyndman, R and Y Khandakar (2008). Automatic time series for forecasting: the forecast package for R. *Journal Of Statistical Software* **27**(3), 1–22.

Hyndman, RJ, RA Ahmed, G Athanasopoulos, and HL Shang (September 2011). Optimal combination forecasts for hierarchical time series. *Computational Statistics and Data Analysis* **55**(9), 2579–2589.

Hyndman, RJ and G Athanasopoulos (2014). *Forecasting: principles and practice*. OTexts.

Hyndman, RJ and A Koehler (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting* **22**(4), 679–688.

Hyndman, RJ, AB Koehler, JK Ord, and RD Snyder (2008). *Forecasting with exponential smoothing: the state space approach*. Berlin: Springer-Verlag.

Ing, CK (2003). Multistep prediction in autoregressive processes. *Econometric Theory* **19**(2), 254–279.

Ing, CK (April 2004). Selecting optimal multistep predictors for autoregressive processes of unknown order. *The Annals of Statistics* **32**(2), 693–722. arXiv: 0406433v1 [arXiv:math].

Inoue, A and L Kilian (June 2008). How Useful Is Bagging in Forecasting Economic Time Series? A Case Study of U.S. Consumer Price Inflation. *Journal of the American Statistical Association* **103**(482), 511–522.

Izenman, AJ (1975). Reduced-rank regression for the multivariate linear model. *Journal of Multivariate Analysis* **5**(2), 248–264.

James, G, T Hastie, D Witten, and R Tibshirani (2013). *An Introduction to Statistical Learning: With Applications in R*. Springer Texts in Statistics. Springer London, Limited.

Jin, F and S Sun (June 2008). Neural network multitask learning for traffic flow forecasting. *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 1897–1901.

Judd, K and M Small (2000). Towards long-term prediction. *Physica D* **136**(1), 31–44.

Kang, IB (July 2003). Multi-period forecasting using different models for different horizons: an application to U.S. economic time series data. *International Journal of Forecasting* **19**(3), 387–400.

Kantz, H and T Schreiber (2004). *Nonlinear time series analysis*. New York, NY, USA: Cambridge University Press.

Khosravi, A, S Nahavandi, D Creighton, and AF Atiya (July 2011a). Comprehensive Review of Neural Network-Based Prediction Intervals and New Advances. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council* **22**(9), 1341–1356.

Khosravi, A, S Nahavandi, D Creighton, and AF Atiya (March 2011b). Lower upper bound estimation method for construction of neural network-based prediction intervals. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council* **22**(3), 337–46.

Kim, K (September 2003). Financial time series forecasting using support vector machines. *Neurocomputing* **55**(1), 307–319.

Kim, M and J Leskovec (2013). Nonparametric multi-group membership model for dynamic networks. *Advances in neural information processing systems*, 1–10.

Kline, DM (2004). "Methods for multi-step time series forecasting with neural networks." In: *Neural Networks in Business Forecasting*. Ed. by GP Zhang. Information Science Publishing, pp.226–250.

Kneib, T, T Hothorn, and G Tutz (June 2009). Variable selection and model choice in geoadditive regression models. *Biometrics* **65**(2), 626–34.

Kock, A and T Teräsvirta (2011). Forecasting with nonlinear time series models. *Oxford Handbook of Economic Forecasting*.

Kohavi, R (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, pp.1137–1143.

Kohonen, T, MR Schroeder, and TS Huang, eds. (2001). *Self-Organizing Maps*. 3rd. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Kreiss, JP and E Paparoditis (December 2011). Bootstrap methods for dependent data: A review. *Journal of the Korean Statistical Society* **40**(4), 357–378.

Ladiray, D and B Quenneville (2001). *Seasonal Adjustment With the X-11 Method*. Lecture Notes in Statistics. Springer.

Lam, BC (2014). "Challenges to Time Series Analysis in the Computer Age." In: *Statistics - Discovering your future power*. China Statistics Press, pp.1–15.

Larrañaga, P, B Calvo, R Santana, C Bielza, J Galdiano, I Inza, JA Lozano, R Armañanzas, G Santafé, A Pérez, et al. (2006). Machine learning in bioinformatics. *Briefings in bioinformatics* **7**(1), 86–112.

Law, AM and WD Kelton (2000). *Simulation Modelling and Analysis*. 3rd. McGraw-Hill.

Lee, KL and Sa Billings (January 2003). A new direct approach of computing multi-step ahead predictions for non-linear models. *International Journal of Control* **76**(8), 810–822.

Lehmann, EL and JP Romano (2005). *Testing Statistical Hypotheses*. Springer Texts in Statistics. Springer.

Lemke, C and B Gabrys (2010). Meta-learning for time series forecasting and forecast combination. *Neurocomputing* **73**(10), 2006–2016.

Lendasse, A, T Honkela, and O Simula (2010). European Symposium on Times Series Prediction. *Neurocomputing* **73**(10-12), 1919–1922.

Lin, J and C Granger (1994). Forecasting from Non-linear models in practice. *Journal of Forecasting* **13**(1), 1–9.

Lin, J and R Tsay (1996). Co-integration constraint and forecasting: An empirical examination. *Journal of Applied Econometrics* **11**(5), 519–538.

Lin, X, C Genest, DL Banks, G Molenberghs, DW Scott, and JL Wang (2014). *Past, Present, and Future of Statistical Science*. Ed. by X Lin, C Genest, DL Banks, G Molenberghs, DW Scott, and JL Wang. Chapman and Hall/CRC.

Liu, S (1996). Model selection for multiperiod forecasts. *Biometrika* **83**(4), 861–873.

Liu, Y, A Niculescu-Mizil, AC Lozano, and Y Lu (2010). Learning temporal causal graphs for relational time-series analysis. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp.687–694.

Lutz, R and P Buhlmann (2006). Boosting for high-multivariate responses in high-dimensional linear regression. *Statistica Sinica* **16**(2), 471–494.

Makridakis, SG and M Hibon (2000). The M3-Competition: results, conclusions and implications. *International Journal of Forecasting* **16**(4), 451–476.

Mallows, C (2000). Some comments on Cp. *Technometrics* **15**(4), 661–675.

Marcellino, M, J Stock, and M Watson (November 2006). A comparison of direct and iterated multistep AR methods for forecasting macroeconomic time series. *Journal of Econometrics* **135**(1), 499–526.

Marx, BD and PH Eilers (February 2005). Multidimensional Penalized Signal Regression. *Technometrics* **47**(1), 13–22.

Matias, J (2005). Multi-output Nonparametric Regression. In: *Progress in artificial intelligence: 12th Portuguese Conference on Artificial Intelligence, EPIA 2005, Covilhã, Portugal, December 5-8, 2005: proceedings*. Vol. 3808. Springer-Verlag New York Inc, pp.288.

Mcdonald, DJ, CR Shalizi, and M Schervish (2011). "Generalization error bounds for stationary autoregressive models."

McElroy, T and M Wildi (July 2013). Multi-step-ahead estimation of time series models. *International Journal of Forecasting* **29**(3), 378–394.

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.

McNames, J, JAK Suykens, and J Vandewalle (1999). Winning Entry of the {K. U.} Leuven Time Series Prediction Competition. *International Journal of Bifurcation and Chaos* **9**(8), 1485–1500.

McSharry, PE, S Bouwman, and G Bloemhof (2005). Probabilistic forecasts of the magnitude and timing of peak electricity demand. *IEEE Transactions on Power Systems* **20**(2), 1166–1172.

Mease, D (2008). Evidence Contrary to the Statistical View of Boosting : A Rejoinder to Responses. *Journal of Machine Learning Research* **9**, 195–201.

Mease, D and A Wyner (2008). Evidence contrary to the statistical view of boosting. *The Journal of Machine Learning Research* **9**, 131–156.

Medeiros, MC, T Teräsvirta, and G Rech (January 2006). Building neural network models for time series: a statistical approach. *Journal of Forecasting* **25**(1), 49–75.

Mitchell, TM (1997). *Machine Learning*. Ed. by A Mellouk and A Chebira. Vol. 4. An Artificial Intelligence Approach 1. McGraw-Hill. Chap. Classifier, pp. 417–433.

Mohri, M, A Rostamizadeh, and A Talwalkar (2012). *Foundations of Machine Learning*. The MIT Press.

Murphy, KP (2012). *Machine Learning: A Probabilistic Perspective*. Adaptive computation and machine learning series. MiT Press.

Opsomer, J, Y Wang, and Y Yang (2001). Nonparametric Regression with Correlated Errors. *Statistical Science* **16**(2), 134–153.

Ou, G and YL Murphey (January 2007). Multi-class pattern classification using neural networks. *Pattern Recognition* **40**(1), 4–18.

Palit, AK and D Popovic (2005). *Computational Intelligence in Time Series Forecasting: Theory and Engineering Applications (Advances in Industrial Control)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Pesaran, MHM, A Pick, and A Timmermann (March 2011). Variable selection, estimation and inference for multi-period forecasting problems. *Journal of Econometrics* **164**(250), 173–187.

Pinson, P (November 2013). Wind Energy: Forecasting Challenges for Its Operational Management. *Statistical Science* **28**(4), 564–585.

Poskitt, DS and AR Tremayne (1986). The selection and use of linear and bilinear time series models. *International Journal of Forecasting* **2**, 101–114.

Price, S (2009). Mining the past to determine the future: Comments. *International Journal of Forecasting* **25**(3), 452–455.

Proietti, T (April 2011). Direct and iterated multistep AR methods for difference stationary processes. *International Journal of Forecasting* **27**(2), 266–280.

Racine, J (2000). Consistent cross-validatory model-selection for dependent data: hv-block cross-validation. *Journal of econometrics* **99**, 39–61.

Ramanathan, R, R Engle, CW Granger, F Vahid-Araghi, and C Brace (June 1997). Shorte-run forecasts of electricity loads and peaks. *International Journal of Forecasting* **13**(2), 161–174.

Ridgeway, G (1999). The state of boosting. *Computing Science and Statistics* **31**, 172–181.

Rissanen, J (1986). Order estimation by accmulated prediction error. *Journal of Applied Probability* **23**(A), 55–61.

Robinzonov, N, G Tutz, and T Hothorn (2012). Boosting techniques for nonlinear time series models. *AStA Advances in Statistical Analysis* **96**, 99–122.

Rosipal, R and LJ Trejo (March 2002). Kernel partial least squares regression in reproducing kernel hilbert space. *The Journal of Machine Learning Research* **2**, 97–123.

Rubio, G, H Pomares, I Rojas, and LJ Herrera (May 2010). A heuristic method for parameter selection in LS-SVM: Application to time series prediction. *International Journal of Forecasting*.

Rudin, C and K Wagstaff (November 2014). Machine learning for science and society. *Machine Learning* **95**(1), 1–9.

Rumelhart, DE, GE Hinton, and RJ Williams (1986). Learning representations by back-propagating errors. *Nature* **323**, 533–536.

Ruppert, D (2002). Selecting the Number of Knots for Penalized Splines. *Journal Of Computational And Graphical Statistics* **11**, 735–757.

Sapankevych, N and R Sankar (2009). Time Series Prediction Using Support Vector Machines: A Survey. *IEEE Computational Intelligence Magazine* **4**(2), 24–38.

Schapire, R (1990). The strength of weak learnability. *Machine learning* **5**(2), 197–227.

Schapire, RE and Y Freund (2012). *Boosting: Foundations and Algorithms*. The MIT Press.

Schapire, RE, Y Freund, P Bartlett, and WS Lee (1998). Boosting the margin: a new explanation for the effectiveness of voting methods. *Annals of Statistics* **26**(5), 1651–1686.

Schmid, M and T Hothorn (December 2008). Boosting additive models using component-wise P-Splines. *Computational Statistics & Data Analysis* **53**(2), 298–311.

Schwarz, G (1978). Estimating the dimension of a model. *The annals of statistics* **6**(2), 461–464.

Shafik, N and G Tutz (May 2009). Boosting nonlinear additive autoregressive time series. *Computational Statistics & Data Analysis* **53**(7), 2453–2464.

Shao, J (1993). Linear model selection by cross-validation. *Journal of the American statistical Association* **88**(422), 486–494.

Shao, J (1997). An asymptotic theory for linear model selection. *Statistica Sinica* **7**, 221–264.

Shmueli, G (August 2010). To Explain or to Predict? *Statistical Science* **25**(3), 289–310. arXiv: arXiv:1101.0891v1.

Shrestha, DL and DP Solomatine (March 2006). Machine learning approaches for estimation of prediction interval for the model output. *Neural networks : the official journal of the International Neural Network Society* **19**(2), 225–35.

Simon, G, a Lendasse, M Cottrell, JC Fort, and M Verleysen (September 2005). Time series forecasting: Obtaining long term trends with self-organizing maps. *Pattern Recognition Letters* **26**(12), 1795–1808.

Simon, G, Ja Lee, M Cottrell, and M Verleysen (August 2007). Forecasting the CATS benchmark with the Double Vector Quantization method. *Neurocomputing* **70**(13), 2400–2409.

Simon, G, A Lendasse, M Cottrell, JC Fort, and M Verleysen (2004). Double quantization of the regressor space for long-term time series prediction: method and proof of stability. *Neural Networks* **17**(8), 1169–1181.

Smith-Miles, Ka (December 2008). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys* **41**(1), 1–25.

Solnon, M, S Arlot, and F Bach (2012). Multi-task regression using minimal penalties. *Journal of Machine Learning Research* **13**, 2773–2812.

Sorjamaa, A, J Hao, and A Lendasse (2005). Mutual Information and k-Nearest Neighbors Approximator for Time Series Prediction. In: pp.553–558.

Sorjamaa, A, J Hao, N Reyhani, Y Ji, and A Lendasse (October 2007). Methodology for long-term prediction of time series. *Neurocomputing* **70**(16), 2861–2869.

Sorjamaa, A and A Lendasse (April 2006). Time series prediction using dirrec strategy. In: *ESANN, European Symposium on Artificial Neural Networks, European Symposium on Artificial Neural Networks*. Ed. by M Verleysen. European Symposium on Artificial Neural Networks. Bruges, Belgium: Citeseer, pp.143–148.

Spinoza, M and T Falck (2008). Time Series Prediction using LS-SVMs. In: *ESTSP 2008: Proceedings*. Ed. by A Lendasse, pp.294.

Stock, JH and MW Watson (2006). Chapter 10 Forecasting with Many Predictors. *Handbook of Economic Forecasting* **1**, 515–554.

Stock, J and M Watson (1999). "A Comparison of Linear and Nonlinear Univariate Models for Forecasting Macroeconomic Time Series." In: *Cointegration, Causality, and Forecasting A Festschrift in Honour of Clive W.J. Granger R.F. Engle and H. White*. Oxford Uni.

Stoica, P and A Nehorai (1989). On Multistep Prediction Error Methods for time series models. *Journal of Forecasting* **8**(4), 357–368.

Stone, CJ (1985). Additive Regression and Other Nonparametric Models. *The annals of Statistics* **13**(2), 689–705.

Stone, M (1977). An asymptotic equivalence of choice of model by cross-validation and Akaike's information. *Journal of the Royal Statistical Society. Series B* **39**(1), 44–47.

Suykens, J, T Van Gestel, KD Brabanter, B De Moor, and J Vandewalle (2002). *Least Squares Support Vector Machines*. World Scientific, Singapore.

Suykens, J and J Vandewalle (1999). Least Squares Support Vector Machine Classifiers. *Neural processing letters* **9**, 293–298.

Suykens, JAK and J Vandewalle (2000). The K.U.Leuven competition data: a challenge for advanced neural network techniques. In: *ESANN*, pp.299–304.

Swanson, NR (1995). Model-Selection Approach to Information in Assessing Using Networks. *Journal of Business & Economic Statistis* **13**(3), 265–275.

Tashman, LJ (2000). Out-of-sample tests of forecasting accuracy: an analysis and review. *International Journal of Forecasting* **16**(4), 437–450.

Tay, AS and KF Wallis (July 2000). Density forecasting: a survey. *Journal of Forecasting* **19**(4), 235–254.

Taylor, JW (2003). Short-term electricity demand forecasting using double seasonal exponential smoothing. *Journal of the Operational Research Society* **54**, 799–805.

Teräsvirta, T (1994). Specification, estimation, and evaluation of Smooth Autoregressive Models. *Journal of the american Statistical association* **89**(425), 208–218.

Teräsvirta, T (2006). Forecasting economic variables with nonlinear models. *Handbook of economic forecasting* **1**, 413–457.

Teräsvirta, T and HM Anderson (1992). Characterizing nonlinearties in business cycles using smooth transition autoregressive models. *Journal of Applied Econometrics* **7**, 119–136.

Teräsvirta, T, D van Dijk, and MC Medeiros (October 2005). Linear models, smooth transition autoregressions, and neural networks for forecasting macroeconomic time series: A re-examination. *International Journal of Forecasting* **21**(4), 755–774.

Teräsvirta, T, D Tjostheim, and CW Granger (2010). *Modelling Nonlinear Economic Time Series*. Advanced Texts in Econometrics. OUP Oxford.

Theodosiou, M (February 2011). Forecasting monthly and quarterly time series using STL decomposition. *International Journal of Forecasting* **27**(4), 1178–1195.

Thissen, U (November 2003). Using support vector machines for time series prediction. *Chemometrics and Intelligent Laboratory Systems* **69**(1), 35–49.

Tiao, G and D Xu (1993). Robustness of maximum likelihood estimates for multi-step predictions: the exponential smoothing case. *Biometrika* **80**(3), 623–641.

Tiao, GC and RS Tsay (1994). Some advances in non-linear and adaptive modelling in time-series. *Journal of forecasting* **13**(2), 109–131.

Tong, H (1990). *Non-linear Time Series: A Dynamical System Approach*. Oxford University Press.

Tong, H and KS Lim (1980). Threshold Autoregression, Limit Cycles and Cyclical Data. *Journal of the Royal Statistical Society. Series B (Methodological)* **42**(3), pages.

Tong, H (1995). A personal overview of non-linear time series analysis from a chaos perspective. *Scandinavian Journal of Statistics* **22**, 399–445.

Tong, H (2011). Threshold models in time series analysis – 30 years on. *Statistics and Its Interface* **4**(2), 107–118.

Tran, VT, BS Yang, and ACC Tan (July 2009). Multi-step ahead direct prediction for the machine condition prognosis using regression trees and neuro-fuzzy systems. *Expert Systems with Applications* **36**(5), 9378–9387.

Tresp, V and HP Kriegel (December 2006). Multi-Output Regularized Feature Projection. *IEEE Transactions on Knowledge and Data Engineering* **18**(12), 1600–1613.

Ueda, N and R Nakano (1996). Generalization error of ensemble estimators. *Neural Networks, 1996., IEEE International Conference on* **1**, 90–95.

Vaccaro, A, G Bontempi, SB Taieb, and D Villacci (November 2012). Adaptive local learning techniques for multiple-step-ahead wind speed forecasting. *Electric Power Systems Research* **83**(1), 129–135.

Van Gestel, T, J Suykens, D Baestaens, A Lambrechts, G Lanckriet, B Vandaele, B De Moor, and J Vandewalle (2001). Financial time series prediction using least squares support vector machines within the evidence framework. *Neural Networks, IEEE Transactions on* **12**(4), 809–821.

Vapnik, VN (1998). *Statistical Learning Theory*. Wiley-Interscience.

Varian, HR (2014). Big Data: New Tricks for Econometrics. *Journal of Economic Perspectives* **28**(2), 3–28.

Vazquez, E and E Walter (2003). Multi-output support vector regression. In: *13th IFAC Symposium on System Identification*. Citeseer, pp.1820–1825.

Venables, WN and BD Ripley (2002). *Modern Applied Statistics with S*. Fourth. New York: Springer.

Verbesselt, J, R Hyndman, G Newnham, and D Culvenor (January 2010). Detecting trend and seasonal changes in satellite image time series. *Remote Sensing of Environment* **114**(1), 106–115.

Wang, X, K Smith-Miles, and RJ Hyndman (2009). Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series. *Neurocomputing* **72**(10-12), 2581–2594.

Wasserman, L (2004). *All of Statistics: A Concise Course in Statistical Inference*. Springer Texts in Statistics. Springer.

Wasserman, L (2014). "Rise of the Machines." In: *Past, Present and Future of Statistical Science*. Ed. by X Lin, C Genest, DL Banks, G Molenberghs, DW Scott, and JL Wang, pp.646.

Weigend, AS and NA Gershenfeld (1994). *Time Series Prediction: Forecasting the Future and Understanding the Past*. Ed. by AS Weigend and NA Gershenfeld. Santa Fe Institute. Addison-Wesley.

Weiss, AA (1991). Multi-step estimation and forecasting in dynamic models. *Journal of Econometrics* **48**, 135–149.

Werbos, P (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks* **1**, 339–356.

Werbos, P (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE* **78**(10), 1550–1560.

Weston, J, O Chapelle, A Elisseeff, B Schölkopf, and V Vapnik (2003). "Kernel Dependency Estimation." In: *Advances in Neural Information Processing Systems 15*. Ed. by ST S Becker and K Obermayer. MIT Press, pp.873–880.

Wichard, JD (May 2010). Forecasting the NN5 time series with hybrid models. *International Journal of Forecasting* **27**(3), 700–707.

Wold, H (1975). "Soft modelling by latent variables: The nonlinear iterative partial least squares (NIPALS) approach." In: *Perspectives in Probability and Statistics*, pp.117–142.

Wold, H (2006). "Partial Least Squares." In: *Encyclopedia of Statistical Sciences*, pp.9.

Wold, S (2001). PLS-regression: a basic tool of chemometrics. *Chemometrics and Intelligent Laboratory Systems* **58**(2), 109–130.

Wood, SN (December 2006). Low-rank scale-invariant tensor product smooths for generalized additive mixed models. *Biometrics* **62**(4), 1025–36.

Wooldridge, J (2012). *Introductory Econometrics: A Modern Approach*. Upper Level Economics Titles Series. Cengage Learning.

Xia, Y and H Tong (February 2011). Feature Matching in Time Series Modeling. *Statistical Science* **26**(1), 21–46. arXiv: arXiv:1104.3073v2.

Xue, Y, X Liao, L Carin, and B Krishnapuram (2007). Multi-task learning for classification with Dirichlet process priors. *Journal of Machine Learning Research* **8**, 35–63.

Yin, J, P Sharma, I Gorton, and B Akyoli (2013). Large-Scale Data Challenges in Future Power Grids. *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, 324–328.

Zadeh, LA (1975). Fuzzy logic and approximate reasoning. *Synthese* **30**(3-4), 407–428.

Zhang, GP (2012). "Neural Networks for Time-Series Forecasting." In: *Handbook of Natural Computing*. Ed. by R Grzegorz, B Thomas, and K JoostN., pp.461–477.

Zhang, GP and M Qi (2005). Neural network forecasting for seasonal and trend time series. *European Journal of Operational Research* **160**(2), 501–514.

Zhang, G and D Kline (2007). Quarterly time-series forecasting with neural networks. *IEEE Transactions on Neural Networks* **18**(6), 1800–1814.

Zhang, GP, EB Patuwo, and H Michael Y. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting* **14**(1), 35–62.

Zhang, L, WD Zhou, PC Chang, JW Yang, and FZ Li (January 2013). Iterated time series prediction with multiple support vector regression models. *Neurocomputing* **99**, 411–422.

Zhang, X and J Hutchinson (1994). Simple architectures on fast machines: practical issues in nonlinear time series prediction. In: *Time Series Prediction Forecasting the Future and Understanding the Past*. Ed. by AS Weigend and NA Gershenfeld. Santa Fe Institute. Addison-Wesley, pp.219–241.

Appendix A

Real-world experiments

We describe the time series data and the methodology used in the real-world experiments of both sections 5.4 and 6.5.

A.1 Time series data

A.1.1 The M3 competition data

The M3 competition dataset consists of 3003 monthly, quarterly, and annual time series. The competition was organized by the *International Journal of Forecasting* (Makridakis and Hibon, 2000), and has attracted a lot of attention. The time series of the M3 competition have a variety of features. Some have a seasonal component, some possess a trend, and some are just fluctuating around some level.

We have considered all the monthly time series in the M3 data. The number of time series considered was $M = 1428$ with a range of lengths between $T = 48$ and $T = 126$. For these monthly time series, the competition required forecasts for the next $H = 18$ months, using the given historical points. Figure A.1 shows four time series from the set of 1428 time series.

A.1.2 The NN5 competition data

The NN5 competition dataset (Crone, 2009b) comprises $M = 111$ daily time series each containing $T = 735$ observations. Each of these time series represents roughly two years of daily cash money withdrawal amounts at ATM machines at one of several cities in the UK. The competition was organized in order to compare and evaluate the performance of computational intelligence methods. For all these time series, the competition required forecasts of the next $H = 56$ days, using the given historical points. Figure A.2 shows four time series from the NN5 data set.

A.2 Methodology

The NN5 dataset includes some zero values that indicate no money withdrawal occurred and missing observations for which no value was recorded. We replaced these two types of gaps using the method proposed in Wichard (2010): the missing or zero observation y_t is replaced by the median of the set $\{y_{t-365}, y_{t-7}, y_{t+7}, y_{t+365}\}$ using only non-zero and non-missing values.

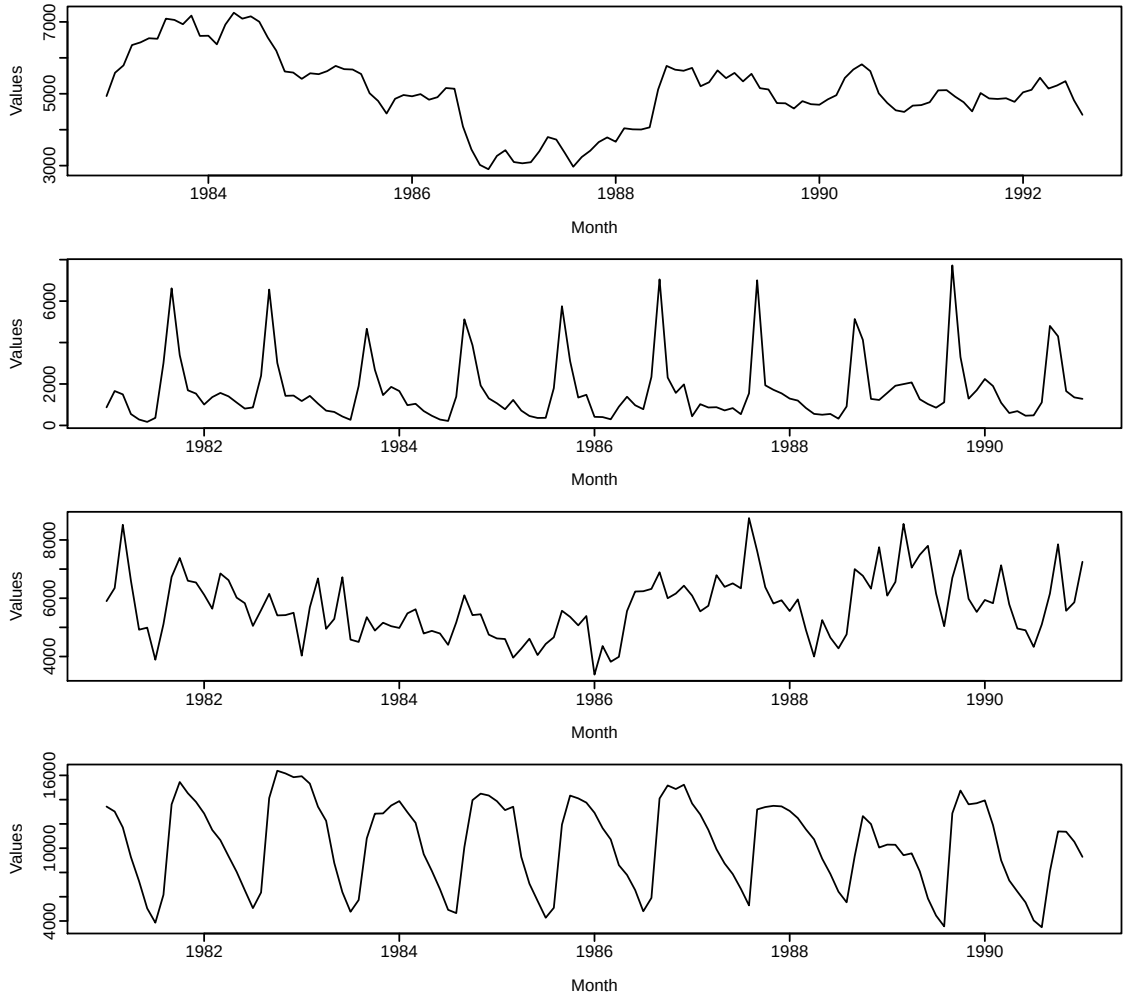

Figure A.1: *Four time series from the M3 forecasting competition.*

We adopted the *forecasting with decomposition* approach, that is instead of directly forecasting the initial time series, we forecast each components separately and then combine the forecasts of each component to obtain the final forecasts, as explained in Section 2.3.2.

For both competitions, we deseasonalized the time series using the STL (Seasonal-Trend decomposition based on Loess smoothing). Of course, the seasonality has been restored after forecasting. For the parameter controlling the loess window for seasonal extraction, we used the value $s = 50$ for the M3 competition to allow a small change in the seasonality over the time series and $s =$ periodic for the NN5 competition.

After deseasonalization, we applied the *Kwiatkowski-Phillips-Schmidt-Shin* (KPSS) test for which the null-hypothesis is that the time series is stationary. If the test is rejected, we applied a first difference. Of course, to produce the final forecasts, we first undifferenced the forecasts from the differenced time series and then we restored back the seasonality.

For the model selection procedure, we used the same approach as for the Monte Carlo simulations which is explained in Section 4.3.2. One difference is that we allowed the value of the lag orders to vary in the set $\{1 : 5\}$ for the M3 competition and in the set $\{1 : 10\}$ for the NN5 competition.

Since the time series have different scales, we did not use MSE. Instead, we considered two other forecast accuracy measures. The first is the symmetric mean absolute percentage error (sMAPE)

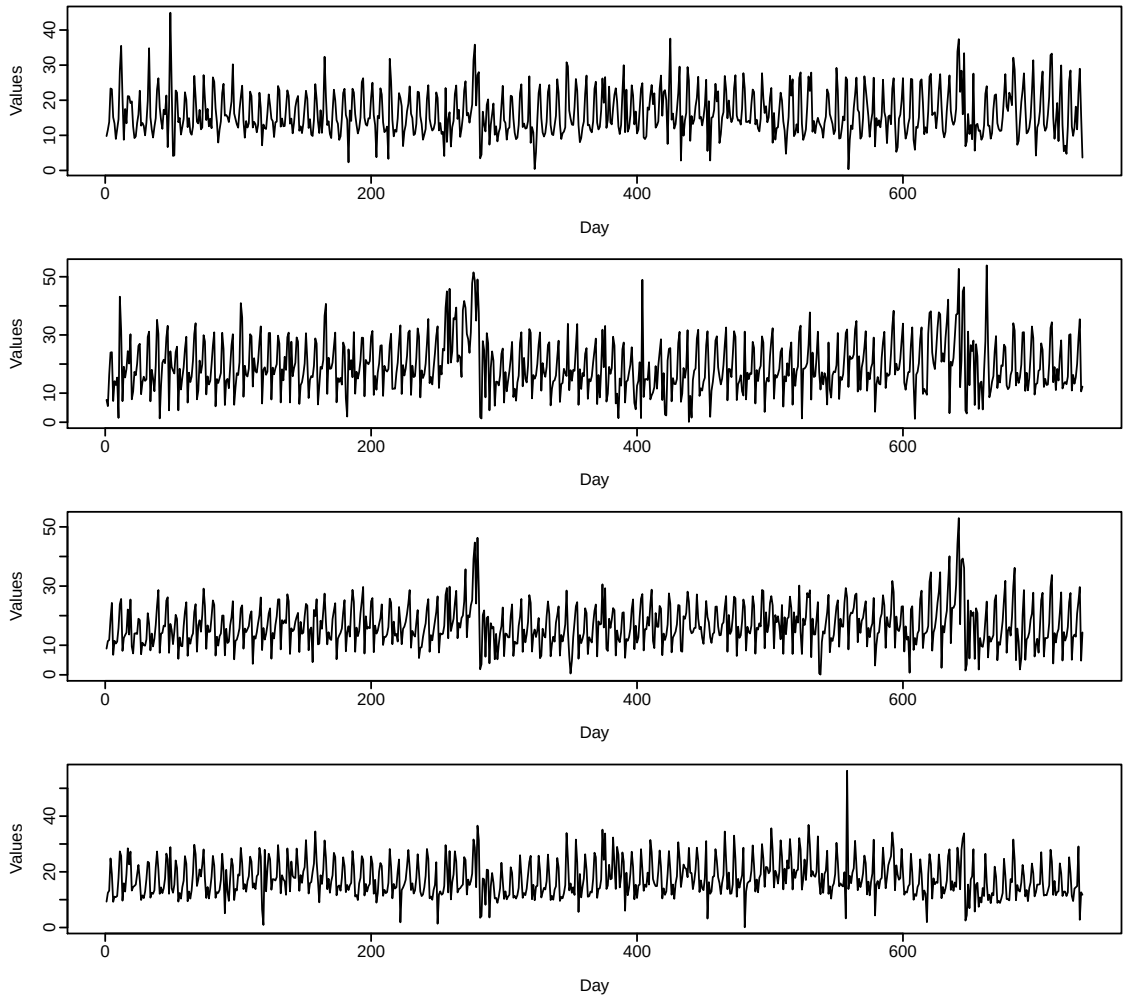

Figure A.2: *Four time series from the NN5 forecasting competition.*

as defined in expression (2.3.3) of Section 2.3.6. Hyndman and Koehler (2006) discussed some problems with this measure, but as it was used by Makridakis and Hibon (2000), we use it to enable comparisons with the M3 competition. The second accuracy measure is the mean absolute scaled error (MASE) introduced by Hyndman and Koehler (2006) and defined in expression (2.3.4) of Section 2.3.6.

Appendix B

Simulated time series

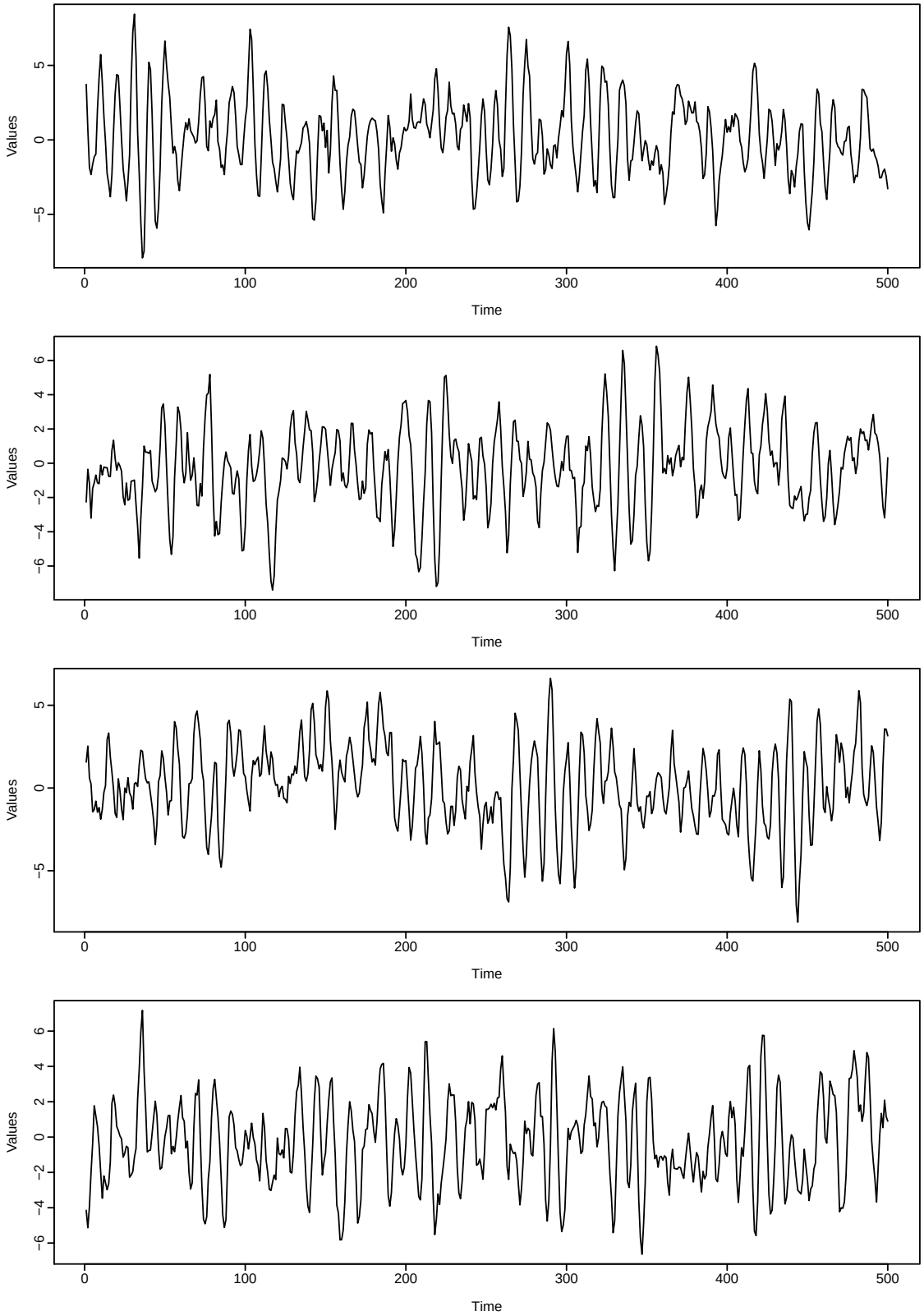

Figure B.1: *Four simulated time series from the AR DGP.*

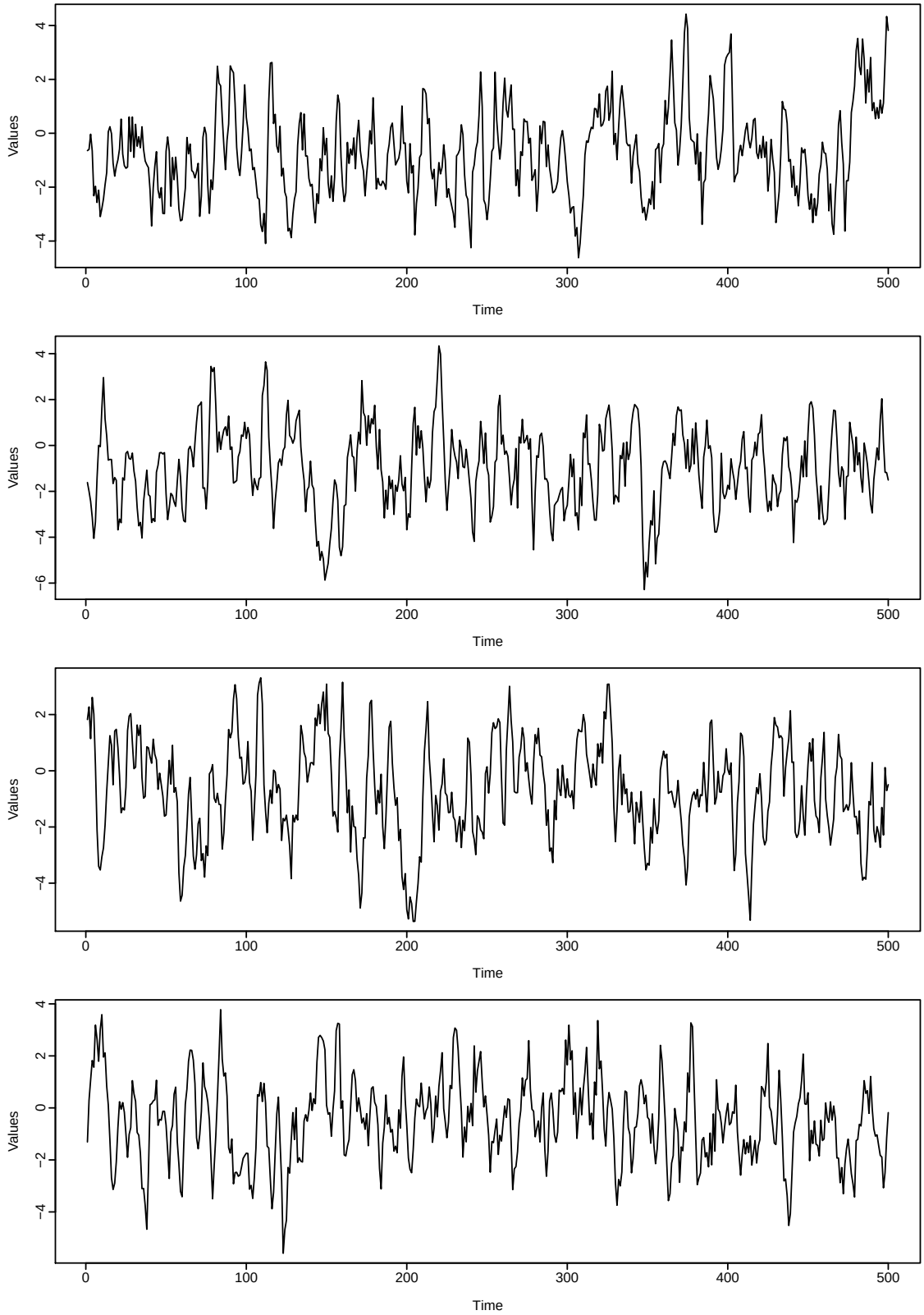

Figure B.2: *Four simulated time series from the NAR DGP.*

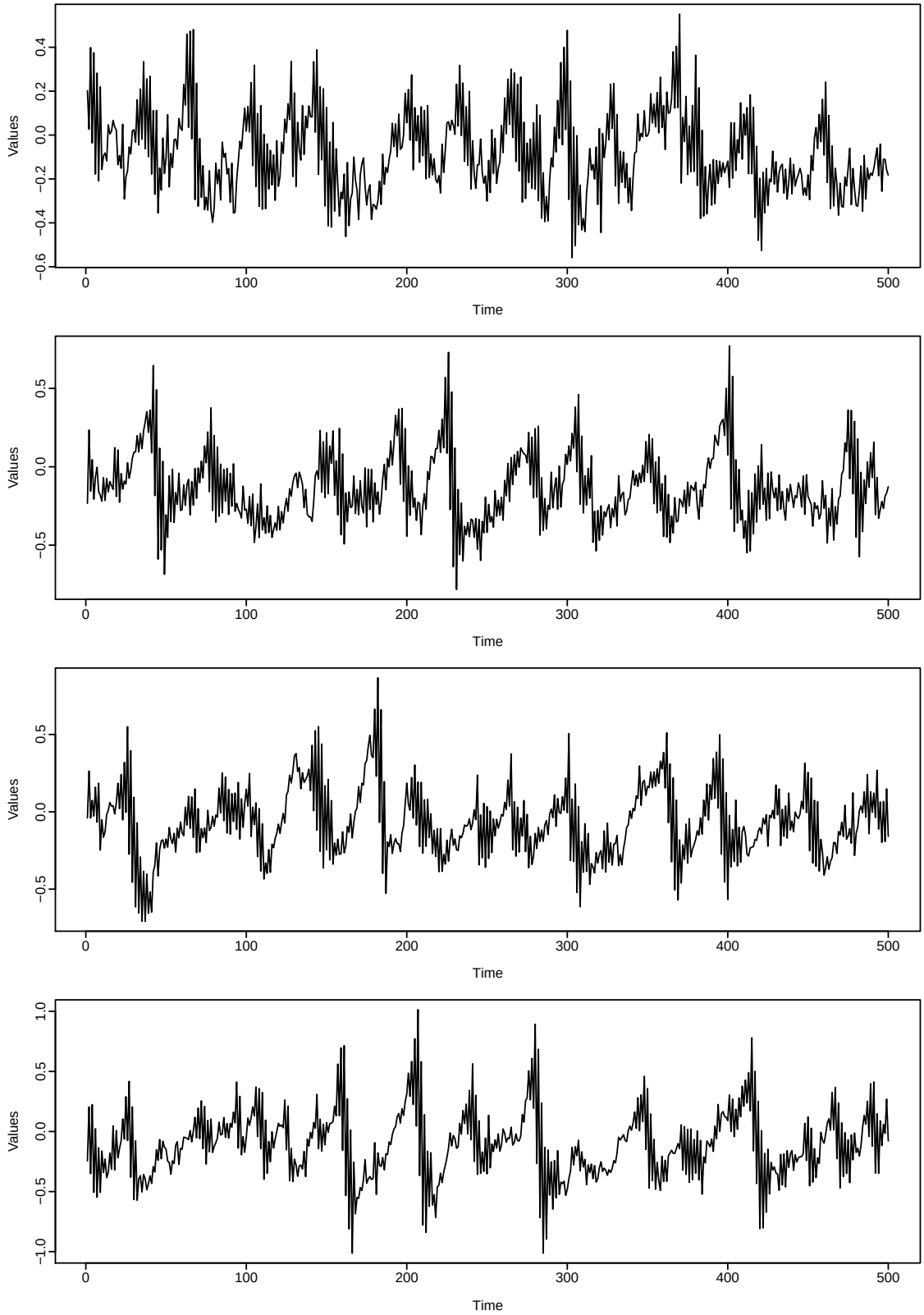

Figure B.3: *Four simulated time series from the STAR DGP.*

List of Notations

$Y_T = [y_1, \ldots, y_T]$ Time series with T observations

H Number of forecast horizons required

$\boldsymbol{x}_t = [y_t, \ldots, y_{t-d}]'$ Input vector at time t with d lagged variables

h Forecast horizon

p Estimated lag order

$\boldsymbol{z}_t = [y_t, \ldots, y_{t-p}]'$ Input vector at time t with p lagged variables

$\boldsymbol{r}_t = [y_t, \ldots, y_{t-p_h}]'$ Input vector at time t with p_h lagged variables

$\boldsymbol{\beta}$ Vector of parameters

$\boldsymbol{\psi}$ Vector of hyperparameters

$\boldsymbol{\theta} = [\boldsymbol{\beta}, \boldsymbol{\psi}]$ Vector of parameters and hyperparameters

$m^{(h)}$ Model m used h times recursively

m_h Direct model for horizon h

$\boldsymbol{\phi}$ Vector of parameters for the one-step-ahead model

$\boldsymbol{\gamma}$ Vector of parameters for direct models

ν Shrinkage coefficient

α Regularization parameter

REC Strategy defined in expression (3.3.2)

RTI Strategy defined in expression (3.3.4)

RJT Strategy defined in expression (6.3.3)

RJT2 Strategy defined in expression (6.3.4)

RJTL Strategy defined in expression (6.3.7)

DIR Strategy defined in expression (3.3.6)

DJT Strategy defined in expression (6.3.6)

DJTL Strategy defined in expression (6.3.8)

LIN Linear model

MLP Neural networks

KNN Nearest neighbors

BST1 Gradient boosting model with univariate weak learner (additive model)

BST2 Gradient boosting model with bivariate weak learner

AR Linear autoregressive process

NAR Nonlinear autoregressive process

STAR Smooth transition autoregressive process

DGP Data Generating Process

OLS Ordinary Least Squares

MS(F)E Mean Squared (Forecast) Error

List of Figures

List of Tables