Master Thesis
in Statistics
at Ludwig-Maximilians-Universität München
Faculty for Mathematics, Informatics and Statistics
Department Statistics

Benchmarking of Classical and Machine-Learning Algorithms (with special emphasis on Bagging and Boosting Approaches) for Time Series Forecasting

presented by
Uwe Pritzsche

Advisor:	Prof. Dr. Christian Heumann
Examiner:	Prof. Dr. Christian Heumann
Timeframe:	27.11.2014 - 27.05.2015

Abstract

The goal of this Master thesis is to evaluate the time series forecast capability of several Machine Learning approaches, in detail Neural Nets, Random Forests, Kernel Machines (Support Vector Machines and Gaussian Processes), tree-based and component-wise linear and spline-based Boosting, by a comparison with classical ARIMA and ETS models. For the classical models also time-series specific bagging approaches, Moving Block Bootstrap and Maximum Entropy Bootstrap, are tested. For this purpose, extensive benchmarks are conducted, utilizing the well-known official *Tourism*, *M3* and *NN5* competition data with the latter comprising also several exogenous covariate effects. In order to uncover specific problems the Machine Learning approaches reveal for the typical time series components of trend and seasonality, a simulation is executed helping in understanding some benchmark results as well as suggesting combinations of the Machine Learning algorithms with classical deseasoning and detrending steps (Box-Cox transformation, STL decomposition, seasonal Differencing). Furthermore different multi-step-ahead forecasting strategies are applied to the NN5 time series.

It can be shown that ARIMA based models are competitive to Machine Learning models for the investigated classical (without any exogenous covariates) time series forecasting situation. On the other hand, ETS approaches are less promising. And the classical models can be enhanced by the tested bagging approaches with the easy-to-use Maximum Entropy Bootstrap showing some advantages over the more known Moving Block Bootstrap. One simple but very important result from the conducted simulation using phenotypic time series is represented by the fact that tree-based models as well as splines with a locality property are incapable of modeling a future trend. In such situations these approaches must be combined with a detrending step resulting in inferior results also for the tree-based boosting model (gbm) as one of the most popular Machine Learning algorithm. Actually the Support Vector Machine is the most promising candidate mostly outperforming all other methods including classical approaches, especially in conjunction with exogenous covariates. Even though Gaussian Processes can be founded in the same theoretical context of Kernel machines, this approach is demystified by its results. Further enhancing the Machine Learning models by direct and hybrid (combination of recursive and direct) forecasting strategies does not reveal any substantial improvements for the tested NN5 series. Interestingly, the benchmark on the M3 data conducted in this thesis seem to be the first one revealing a more than competitive prediction of special naïve methods for most series making performance conclusions of other studies based on this data highly questionable.

Contents

1 Introduction

1.1 Motivation

Time Series Forecasting has a long history not only in scientific field but in practice as well. Beside the most known usage for stock market forecasting numerous additional applications in business exist, like Budget Forecasting, Inventory Planning and Energy Consumption Prediction, to mention just a few. And in the ages of big data and the internet of things more and more fine grained information is recorded not only with respect to the timely development of interested target leading to new applications in many fields. Additionally to the chronological target variable more exogenous information is available increasing the set of possible covariates. This evolution brings into play relatively new (compared to the classical methods) algorithms from the Machine Learning field which have proven a remarkable performance in classical prediction problems (without time related information) often outperforming classical statistical models like linear regression for instance. But it is also of interest whether these modern approaches can also compete in a classical time series setting using just the consecutive target variable information to predict the future. In fact this ability seems to be a prerequisite to at least improve forecasting results settings with high exogenous covariate influence.

The most used Machine Learning model in the time series context so far are Neural Nets (referred to as *nnet* in this thesis). But especially tree based algorithms in conjunction with bagging, i.e. Random Forests (rf), and boosting (*gbm*) are very popular in the Machine Learning community due to some properties highly advantageous for prediction like automatic interaction detection or robustness against irrelevant predictors. The latter holds also for boosting approaches in conjunction with linear (*glmboost*) or semi-parametric (*gamboost*) regression models which further have much better interpretability capabilities as well and are therefore favored in the statistical community. All these models account for the correlation of the target variable values by using lagged target information. An alternative that recently got very popular in the Machine Learning community are Gaussian Processes (*gp*) which are based in the theory of so-called Kernel machines as the covariance matrix of the target can be interpreted as a Kernel matrix. The ancestor of a such Kernel machines are the well-known Support Vector Machines (*svm*) which originated in classification task but can be adapted for regression problems as well.

Especially the success of Random Forests creates attention for bagging (bootstrap aggregation) as a general tool for improving forecasts. If, as common for the Machine Learning models, the lagged target variable information is just added to the standard covariate set, the bootstrapping can be conducted as usual, i.e. random sampling with replacement. But it the modeling approach like for the classical Arima (or for Gaussian Processes as well!) needs the original data order for estimating the model the usual approach is not applicable as it destroys this order. Therefore special sampling schemes must be applied. One part of this thesis work tests the improvements which can be gained for classical models by bagging with the well-known Moving Block Bootstrap as well as a more modern and easy-to-use approach, the Maximum Entropy Bootstrap.

All in all the aim of this thesis is to shed some light onto the performance of all above mentioned machine learning approaches compared to classical models and bagging approaches for the latter. This is done by benchmarking the methods utilizing several data sets from known official competitions, i.e. the well-known classical *M3* (Makridakis & Hibon (2000)), the *Tourism* competitions (Athanasopoulos et al. (2011)) and the *NN5* contest (Crone (2009b)) comprising additional exogenous covariate information.

Time series forecasting exhibit some special characteristics not common in the usual prediction setting like a whole bunch of official metrics for measuring performance that are highly sensitive to the forecasts. Moreover one is usually interested in multi-horizon forecasts, i.e. predictions not only for the next future time point but e.g. for the next few seasons raising the question whether to reuse forecasts as inputs in so-called recursive forecasts or better use direct predictions employing just known information. This also complicates the assessment of forecast accuracy as the performance can differ from horizon to horizon further depending on the evaluation metric used. Consequently some attention is put on the careful handling of performance evaluation in this thesis.

Also somewhat uncommon for the Machine Learning models are the usual time series components of trend and seasonality. At first sight the trend component does not seem to represent a problem for prediction models but one has to keep in mind that the trend has to be put forth into the future and therefore outside the range of the time input variable, resulting in critical modeling problems, especially for tree based approaches for instance. Related problems might grow when dealing with increasing seasonality. In order to uncover such general problems Machine Learning models might have with these components, a simulation study is conducted that reveals some interesting insights that not only help in understanding benchmark results but also suggest different variants of each Machine Learning model.

Actually the typical approach with Machine Learning models for handling the seasonality is using seasonal dummy variables. Alternatively it is possible to combine the Machine Learning algorithm with classical means for stripping of the seasonality like seasonal differencing, a modeling approach invented in classical ARIMA modeling, or decompositions like STL (seasonal-trend decomposition based on Loess). Also initial Box-Cox transformation of the time series might help in case of increased seasonality for instance. It is of interest whether the Machine Learning approaches can profit from such combined approaches which is therefore extensively tested by applying several variants of each algorithm.

In order to assess the performance of these combination approaches they must be compared to the corresponding classical models, at least with the most popular representative, i.e. ARIMA models. Usually this model class is embedded in a whole modeling operating manual, the Box-Jenkins approach. On the other hand, ETS (exponential smoothing) models are mostly neglected by the academic canon even though successfully applied by practitioners for a long time. A reason for this disregard might be a lack of statistical founding of this initially heuristic approach just recently patched by Hyndman et al. (2008) utilizing state space modeling. This thesis also tries to avoid the common lack of dedication for this (in non-academic community well-known) modeling approach.

Last but not least the extensive evaluations conducted in this theses are motivated by several other studies (listed in the following Chapter 1.2) that lack of several before mentioned aspects. For instance, usually only a few of before mentioned Machine Learning approaches are tested. Or no comparison with classical methods is conducted. More severe in this conjunction is the typical practice of neglecting to conduct at least one naïve forecast or several naïve forecast variants as benchmarks in order to assess the forecastability of the series which is crucial especially for the M3 competition data as will be shown later. Sometimes the investigated time series are too short or too few in number which prevents from getting a valid result which is also the problem when just 1-step-ahead forecasts are executed. In case of multi-horizon forecasts nearly always only the recursive strategy is applied. Some studies are more theoretically motivated and therefore lack of real-life datasets comprising trend and seasonality. Last but not least just a few authors try different deseasonalizing strategies.

This thesis is structured as follows: Already in this introductory chapter the related work is discussed. Then Chapter 2 first explains the benchmark datasets with a more detailed discussion of possible covariate effects for the NN5 time series. Further prerequisites for the investigations like forecasting strategies as well as the applied evaluation approach follow, already producing important insights. The

classical models are explained in detail in Chapter 3 together with the possible general preprocessing steps, i.e. Box-Cox transformations and STL decomposition. All explanations are accompanied by some initial benchmark results. This holds also for the passage dealing with the utilized bagging approaches for the classical models before presenting a comparison of just the classical models for all three competition benchmark series at the end of this Chapter. The theory of the Machine Learning algorithms is explained in Chapter 4. In order to get a direct impression of the performance of these models some benchmark insights are added even though this kind of anticipates some results of the simulation study presented at the end of Chapter 4 using some phenotypic time series to uncover general problems of the Machine Learning approaches with time series (more precise the trend and seasonal component). Chapter 5 is dedicated to the overall benchmark results comparing all classical and Machine Learning approaches for all competition benchmark data. The thesis closes with a summary giving also a short outlook.

1.2 Related Work

Literature for applying Machine Learning or Data Mining approaches for time series forecasting is relatively sparse. Reasons for, but also opportunities resulting from this gap, are shortly discussed in Crone (2009a), listing what forecasters can learn from Data Mining and vice versa. Also Gooijer & Hyndman (2006) in their overview article “25 years of time series forecasting” spend their attention regarding machine learning techniques just on neural nets in the chapter about nonlinear methods. Nonetheless they state that “with the ability of very large datasets and high powered computers, we expect this [bagging and boosting approaches] to be an important area of research in the coming years”.

The following subsections first list related investigations for machine learning approaches in general and then dedicate two subsections explicitly for bagging techniques, comprising classical bootstrapping of time series, and boosting approaches.

Machine Learning approaches

The most used algorithms from Machine Learning field for forecasting so far are neural nets (*nnet*). Already in the famous M3 competition the *nnet* is the only non-classical method used (Makridakis & Hibon (2000)). Furthermore Krollner et al. (2010) nicely show in their overview article for financial time series forecasting that ANN is the dominant technique in this area (mostly used for 1-day ahead forecasting).

Apart from ANNs the majority of investigations using Machine Learning models are restricted to k-nearest neighbor (kNN), support vector machines (*svm*), gaussian processes (*gp*) and boosting techniques with trees (gbm), component-wise linear (*glmboost*) or spline (*gamboost*) models.

For example, the relatively extended study of Ahmed et al. (2010) compares *nnet*, kNN, *svm*, *gp* and CARTs (classification and regression trees) utilizing different preprocessing steps regarding detrending techniques (lagged target values, differencing and moving average filtering). Furthermore they deseasonalize in advance leaving just the remainder for the ML methods for forecasting and incorporate a tuning of the number of lagged target values. Unfortunately they apply these methods on the whole M3 competition putting all categories of time series into one pot (which is problematic as discussed in Chapter 2.3.2). Also only 1-step ahead predictions are done and no comparison with a naïve method or classical approaches is conducted. With these restrictions *nnet* and *gp* end up as the overall winners.

Much less models are used in Lora et al. (2004) who let kNN compete with dynamic regression models for a 24h energy load forecasting problem and show superior performance for the kNN approach.

For a similar problem in the GEFCom2012 forecasting competition (Hong (2012)), Lloyd (2014) applies *gp* using “periodic” kernel in a decorrelated ensemble with *gbm*. Actually the *gp* performance as standalone prediction was far behind the *gbm*. Interestingly no lag information was utilized apart from an implicit usage by the *gp* correlation.

For this competition Hong et al. (2014) summarize the methodology of the top competitors, wondering why no classical ARIMA model is used by anyone. Basically most of the winning teams apply linear regression with splines, sometimes enhanced by boosting.

The second placed team of the NN5 competition (Crone (2009b)), Andrawis et al. (2011), utilize an exhausting ensemble of several *nnet* and *gp* models with classical ARIMA and ETS approaches, claiming a big advantage by their somewhat difficult desesonalizing preprocessing.

Again the *nnet* is chosen for a study of Matteo et al. (2013) for temperature forecasts in buildings resulting in better prediction than using a classical standard regression models with and without autoregressive components.

Similarly Kandananond (2012) lets *nnet* and *svm* compete against classical ARIMA models but just for 6 really short datasets leaving *svm* as the winner.

Also the *svm* takes part in a comparison with *nnet* and ForecastPro, a commercial forecasting package basically using an ARIMA-ETS combination, applied on simulated time series in Crone et al. (2009). Actually the simulation study of Chapter 4.5 in this thesis is influenced by their simulated combinations of no, additive or multiplicative seasonality with no, linear, progressive or degressive trend each combined with different noise levels. Also some lag selection tuning was conducted by them utilizing the PACF (partial autocorrelation function). Here *svm*, followed by *nnet*, wins the nonlinear, i.e. multiplicative seasonality and/or progressive or degressive trend case whereas for the linear series the classical ARIMA-ETS combination outperforms *svm* and *nnet*.

A different intention has Bontempi et al. (2013), explaining different Machine Learning forecasting strategies, including recursive and direct forecasts, basically using a kNN model.

Related is the study of Taieb & Hyndman (2012a) using a combination of initial recursive autoregressive prediction and a direct adjustment for the forecast error by a kNN model. They apply their method to simulated series as well as to the M3 and NN3 benchmark datasets with mixed results regarding forecast performance. A more detailed presentation of the NN5 results together with additional averaging strategies can be found in Taieb et al. (2012c). For preprocessing they use the before mentioned deseasonalizing steps from Andrawis et al. (2011) for this data.

Taieb & Hyndman (2014) further apply a modification of their own approach described above by using a direct *gamboost* (with bivariate interaction of covariates) based forecast on the residuals of a STL decomposition followed by a recursively applied ARIMA model for the M3 and NN5 competition datasets. This builds the bridge (apart from above mentioned Lloyd (2014) and Hong et al. (2014) mentioned above) to the next subsection looking at boosting techniques applied to time series forecasting.

Boosting

For the GEFCom2012 competition Taieb & Hyndman (2012b) compete with a combination of recursive and direct forecasts of *gamboost* models with intensive lag value usage finishing fifth out of 105 teams.

A very promising investigation can be found in Robinzonov et al. (2010) showing the results of a simulation study, aiming to assess the capability of *glmboost* and *gamboost* to model the simulated

autoregressive function of nonlinear time series. Actually *glmboost* shows impressive performance in identifying the data generating lags. But the study also states that “strong serial dependence might mislead the fitting procedure [i.e. *gamboost*] to produce erroneous transformations”. A final comparison of forecast performance for some macroeconomic time series favors the linear boosting approach.

These nonlinear simulations together with some simulated linear time series are already discussed in Shafik & Tutz (2007).

Furthermore Buchen & Wohlrabe (2011) try to apply *glmboost* based direct forecasts to some macroeconomic time series ending up with mixed results regarding the performance.

Bootstrap

A good overview of classical bootstrap methods for time series, i.e. mainly Block, Sieve and Stationary bootstrap, is given in the more technical paper from Härdle et al. (2003) concentrating on the accuracy of the methods for parameter estimation of a sample. The more readable summary from Kreiss & Lahiri (2012) on the other hand focusses more on advantages of the different methods in different data situations. Even though the Stationary bootstrap (Politis & Romano 1994) has the appealing capability of preserving the stationarity property, its success seem to be limited as “with respect to higher order properties the moving block bootstrap outperforms the version with non-overlapping blocks and both achieve a higher order accuracy as the stationary bootstrap” (Mammen & Nandi (2012)).

The more recent approach of Maximum Entropy Bootstrap (MEboot) is invented in Vinod & Lopez-de-Lacalle (2009) and concentrating on showing the main advantage of this approach, nicely summarized by the author with “the algorithm's practical appeal is that it avoids all structural change and unit root type testing involving complicated asymptotics and all shape-destroying transformations like detrending or differencing to achieve stationarity”. Furthermore the authors state that “the constructed ensemble elements retain the basic shape and time dependence structure of the autocorrelation function (ACF) and the partial autocorrelation function (PACF) of the original time series.”

Above resources concentrate more on the statistical properties of the different drawing techniques. Applications of bootstrap methods for time series forecasting by bagging the predictions of the different samples can be found e.g in Fan & Hyndman (2010) who use block bootstrap samples to get a complete distribution of prediction intervals when applying generalized additive models.

Bergmeir et al. (2014) improve the accuracy of forecasts by a moving block bootstrap applied on the remainder of a STL decomposition of M3 competition data and forecast with an ETS model. This is one of the rare studies also assessing the lift of initial Box-Cox transformations of the target variable.

Cordeiro & Neves (2009) utilize the ETS initially followed by an ARIMA on the residuals before ending up in bootstrapping the residuals left over from this process. This parametric bootstrap approach, similar to the so-called SIEVE bootstrap, shows mixed behavior when applied on the M3 competition data.

Contrary to the typical bagging approach for Machine Learning methods (i. e. bootstrap as usual with lagged target information as standard covariates) Kourentzes et al. (2014) use a combination of classic bootstrap methods and ANN models by bagging with the moving block bootstrap to improve forecast accuracy, suggesting a median instead of a standard mean aggregation in the bagging step.

2 Prerequisites

2.1 Benchmark Datasets

For the forthcoming benchmarks the mostly used time series datasets utilized in other academic investigations are chosen. In detail these are the M3 competition datasets (Makridakis (2000)), the Tourism data from the Tourism2 competition hosted on Kaggle (Athanasopoulos et al. (2011)), and the series used in the NN5 competition (Crone (2009b)). Only the latter allows deriving additional exogenous effects like holiday, Christmas or Easter whereas the Tourism and M3 datasets are classical time series just comprising the consecutive target variable information. The latter holds as well for 300 simulated ARIMA series (see Chapter 3.1 for an introduction to Arima time series), named *Arimasim* in the following. Regarding Tourism and M3 it was decided to use only the most fine-grained data, i.e. the monthly data. Furthermore only time series comprising more than 120 time points are used, ending up in removing one Tourism and several M3 series (see below).

Tourism

The monthly Tourism data comprises 365 time series (after removing 1 series having less or equal 120 time points). Figure 2.1 shows the distribution of the number of time points. The maximum horizon to forecast is 24. Unfortunately the data source from Kaggle website does not comprise the testing data, therefore the last 24 time points for each series are used instead for the test fold which is also the reason why a comparison with the official contest results can only be given on a qualitative level. Most of the series show a trend and seasonality (sometimes increasing); some typical examples are plotted in the Appendix (see “A Supporting Plots and Tables”). In order to allow Box-Cox transformations (cf. Chapter 3.3) a constant c=1 is added to have all series consisting of only strictly positive values.

The winning methods for this competition conducted in 2010 are presented in the International Journal of Forecasting (Brierly (2011b), Baker & Howard (2011)) but also described in short on two Kaggle blog posts (Brierly (2011a), Baker (2010)). Basically these solutions used a heuristic approach and an ensemble of classical (ETS and ARIMA, cf. Chapter 3) models respectively. Furthermore an extensive comparison of classical approaches applied on these data is available in Athanasopoulos et al. (2011), the accompanying paper for this competition. Main result for the monthly series is that ARIMA methodology is most accurate when considering MASE performance metric (see Chapter 2.3.1) and exhibits also clearly better predictions than a seasonal naïve forecast (cf. Chapter 2.3.2).

M3

The used monthly M3 data consists of 1010 series (after removing the shortest ones, see following explanations) from 5 different areas named *DEMOGRAPHIC*, *FINANCE*, *INDUSTRY*, *MACRO*, *MICRO*. The category *OTHER* is removed as all time series of this area has less or equal 120 time points which is the case for 418 time series in total. Figure 2.2 shows the distribution of time points for the remaining data sets. Some examples series are also plotted in the Appendix.

Results from this competition, with a maximum horizon to forecast of 18, are described in Makridakis & Hibon (2000) and several accompanying papers published in the same edition of the International Journal of Forecasting. The main result, which is somewhat surprising at first sight, is that “simple methods developed by practicing forecasters (…) [i.e. ETS] do as well, or in many cases better, than statistically sophisticated ones like ARIMA (…)”. Actually Chapter 3.2 of this thesis explains that ETS

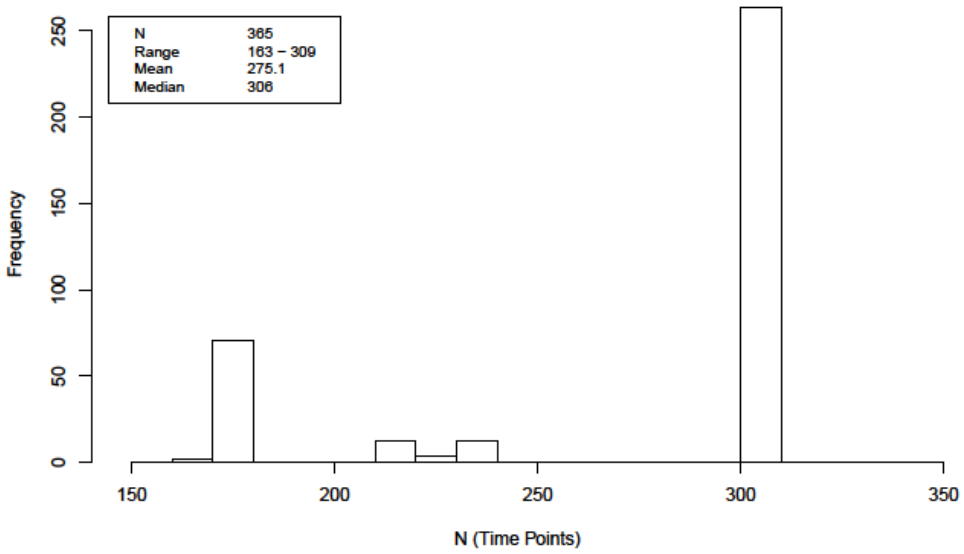

Figure 2.1: *Distribution of number of time points (i.e. months) for Tourism data.*

models can be put into a strict statistical frame and that many ETS model are also equal to ARIMA models.

It should already be mentioned that the usage of the M3 data for algorithm benchmarking is highly questionable as explained in Chapter 2.3.2. Despite this fact and the age of the competition datasets, these time series are still used in several investigations, e.g.: Cordeiro & Neves (2009), Ahmed et al. (2010), Taieb & Hyndman (2012a), Taieb et al. (2012c), Bergmeir et al. (2014), Taieb & Hyndman (2014).

NN5

This more recent competition from 2008 comprises 111 time series consisting of daily ATM withdrawal amounts in UK, each comprising 791 time points from 18/03/1996 – 17/05/1998 (i.e. > 2 years) with a maximum horizon of 56 (8 weeks) to be forecasted. For the training fold comprising the first 735 observations in each series, missing and zero values are imputed by the mean of the seasonal (with period=7) neighbors. Missings that are still left over are filled by the *na.interp* function from *forecast* R-package (Hyndman (2015a)). Missing values in the test fold (last 56 time points) are set to zero whereas original zero values are left unchanged. This imputation approach treats withdrawals amounts of zero as missing only in the training period and therefore assures to be comparable to the original competition benchmark results.

Public holiday information is taken from http://www.work-day.co.uk/.

It was required in this competition to train a model on the whole training fold even though it might be advantageous to concentrate just on the period to be forecasted! This approach is overtaken in this thesis to be comparable. The first and second placed entries of the original competition are described in Wildi (2008) and Andrawis et al. (2011) respectively. The latter utilizes an ensemble of 9 classical and machine learning approaches selected from a total of 140 models. The authors also apply a special deseasonalizing process comprising several stages. Both lead to a somewhat sophisticated

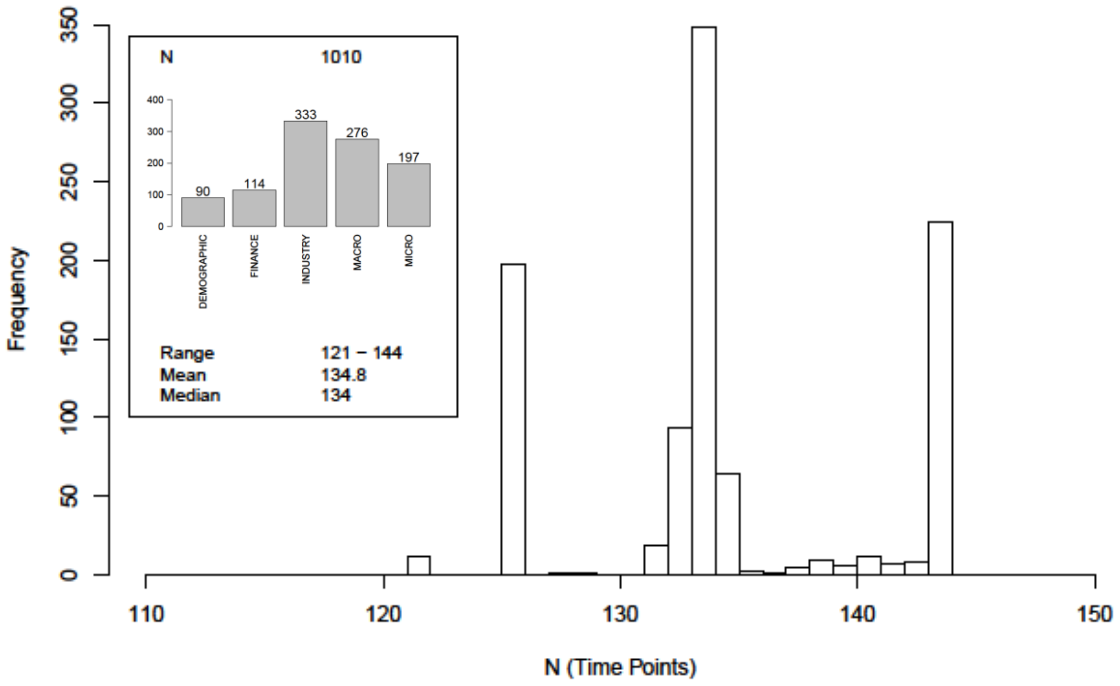

Figure 2.2: *Distribution of number of time points count (i.e. months) for M3 data. Inlay in legend additionally shows number of time series by category.*

strategy which also holds for the solution from Wildi (2008) whose approach is totally different as it is framed by the frequency domain of time series forecasting which is not covered at all in this thesis (see e.g. Shumway & Stoffer (2011) for this topic). With regard to the complexity of these winning solutions, the result of the benchmark conducted in this thesis for the NN5 data presented in Chapter 5.3 is remarkable.

Other benchmark investigations using the datasets are e.g. Taieb & Hyndman (2012a) and Taieb & Hyndman (2014). As usual, some typical representatives of the series can be found in the Appendix.

Figure 2.3 shows the averaged (!) series including the 56-horizon test data. A clear Christmas effect can already be identified. The number of public holidays is sparse in the test period (just one, apart from Easter), therefore its effect would at most be important for correct modeling of other covariate effects (e.g the Christmas effect). On the other hand, especially effects around Easter are important to be recognized by the model as the test data period comprises them. With this regard it is important that the first Easter period in the training fold is not cut off due to creation of lags of the target variable used as autoregressive covariate effects (see left reference line in Figure 2.3).

When comparing the inlay of Figure 2.3 with the top right plot of Figure 2.4**,** it can be seen that the two weeks around Easter show a different seasonality than the average, i.e. a boosting of Easter Monday withdrawals and as well for Wednesdays before Easter. One additional week in advance, the seasonality is normal which is important as this week is cut in the first Easter period of the time series but is part of the test data. The clear *weekday* seasonality can also be identified in the STL decomposition shown on the left in Figure 2.4 (see Chapter 3.3 for an explanation of STL). Furthermore the remaining trend in this plot indicate not only an additional New Year effect but also more striking an additional *monthday* seasonality due to higher withdrawals at the end of each month obviously resulting from usual salary payout conventions. The latter is confirmed by the second plot on

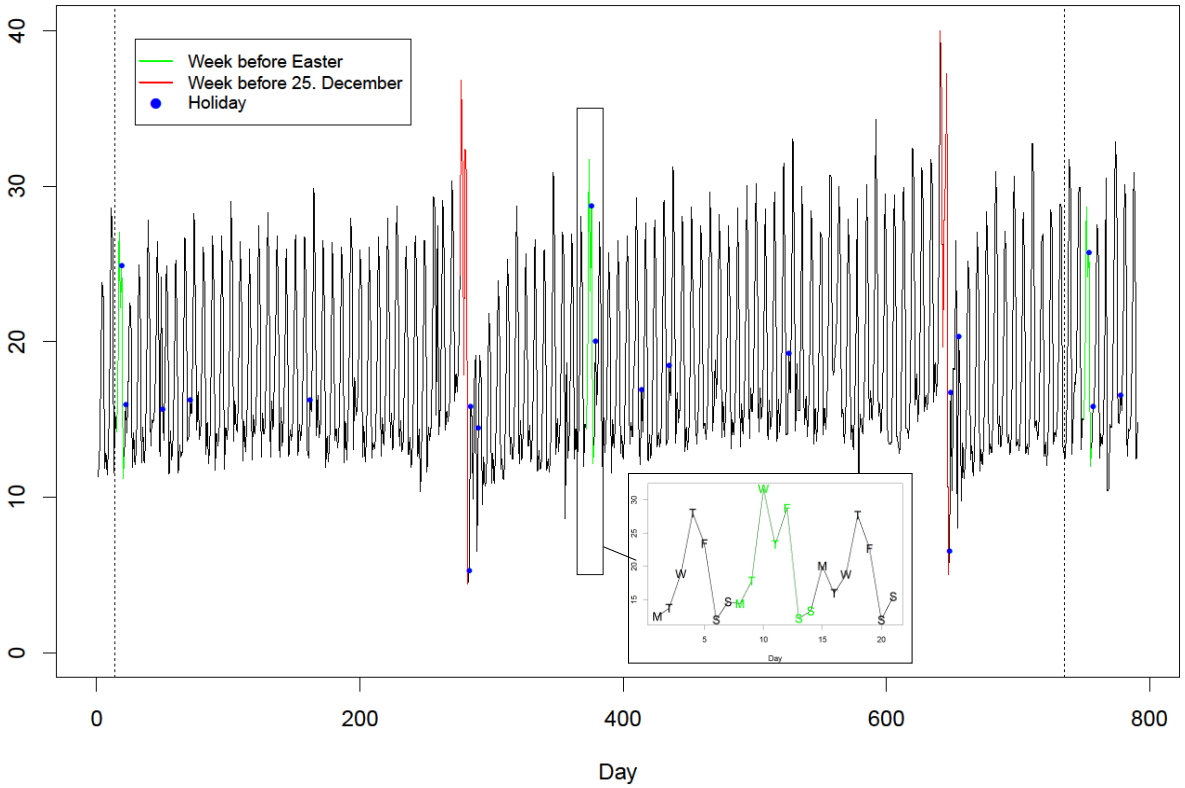

Figure 2.3: *Averaged (over all series) NN5 time series. The right reference line splits train from test data. The left reference line denotes which time points are removed for lag creation. The inlay shows 3 weeks around Easter.*

the right in Figure 2.4 showing a clear effect even though much smaller than the *weekday* seasonality (beware of different scales in the plots!). Recognize that it is important to visualize this effect after *weekday* related deseasoning in order to rectify both effects. Outliers around the 25th due to Christmas are cut for effect display purity, but can be indirectly identified by the skewness of the distribution around this day of month seen through a gap between mean and median values. Due to the sparse number of public holidays, the holiday effect, without Easter and Christmas period, has high variability but shows only very small averaged effects. Nonetheless this might be an important covariate for special time series corresponding to ATMs standing in industrial or commercial areas. The final partial correlation plot (see Chapter 3.1.1 for more information to a PACF plot) reveals some lagged weekly effects, see peaks around the 7th lag, still left over after deseasonalizing. Additionally a high lag1 effect can be identified. But the *monthday* seasonality, i.e. a peak around 30, does not clearly pop up which might be because of a more smeared seasonality due to flipping month lengths. Also the concentration of the PACF plot on solely linear correlation effects might hide this effect.

Above analysis proposes to add the dummy coded variables *weekBefEaster*, *weekAftEaster* and holiday related (*holiday, dayBefHoliday, dayAftHoliday*) to the covariate set of seasonal dummies for the weekday and 1 to at most 14 lags (i.e. two seasons) catching any autoregressive dependencies. But it must be kept in mind that adding these variables just as main effects would prevent models which cannot automatically identify interactions from modeling the Easter effect for instance. Furthermore a *monthday* covariate is needed for the second seasonality and, together with a covariate for the *month*, can also catch the Christmas and New Year effects as interaction effect. For algorithms capable of modeling nonlinear effects these covariates can be added on a metric scale (numeric

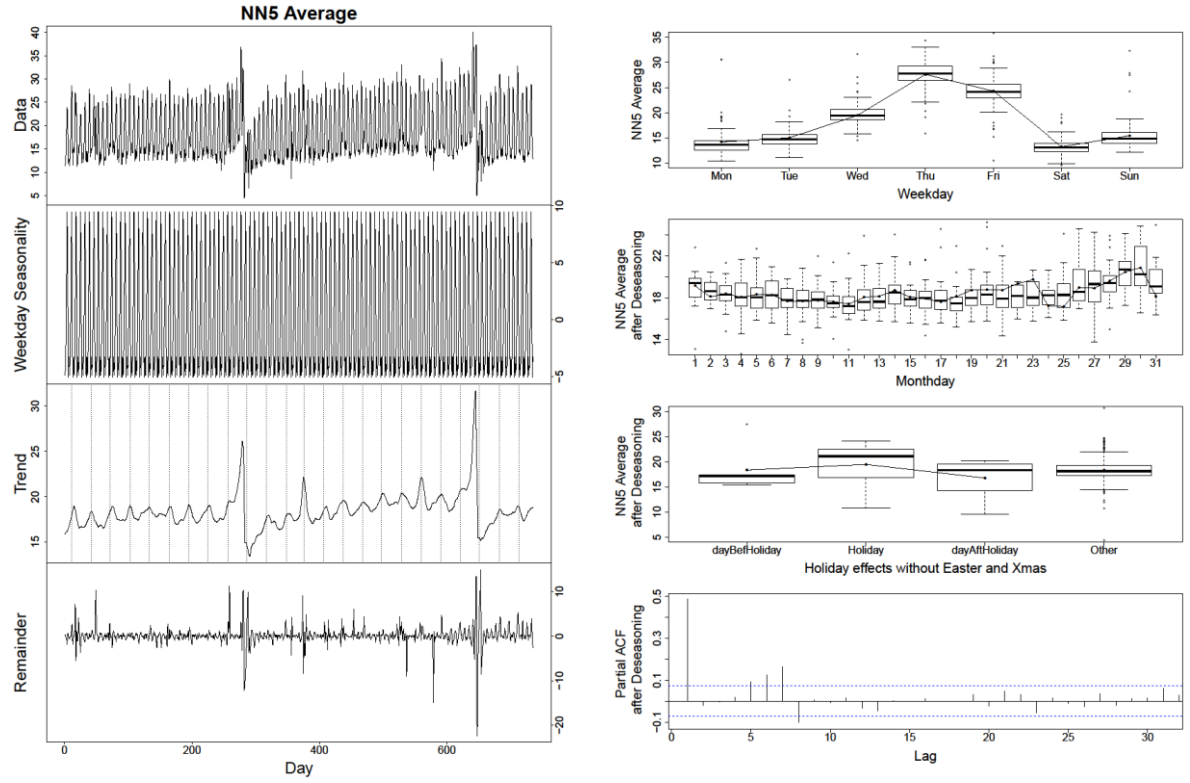

Figure 2.4: *Seasonality and covariate analysis of averaged NN5 time series train fold. The plots on the left represent a STL decomposition (cf. Chapter 3.3) using weekday seasonality with reference lines in the third plot denoting the 28th of each month. The first two plots on the right show the weekday and monthday seasonality (without some outliers lying out of chosen scale range). The 2nd plot also refers to the "Trend" plot on the left side, i.e. showing the data after weekday deseasoning. The 3rd boxplot shows holiday effects and the last plot the partial autocorrelation left over after weekday deseasoning. See text for more explanations.*

covariate) to provide a parsimonious covariate set. Furthermore the slight trend can for instance be modeled by adding a numeric season variable counting up the weeks.

Arimasim

The capability of machine learning methods for forecasting also strictly classical correlated time series can be tested with 100 simulated time series each comprising 300 time points created by the *arima.sim* function of the standard *stats* R-package named *Arimasim* in the following. Actually all series contain one autoregressive and one moving average coefficient randomly chosen from a uniform distribution $U[0.1,0.9]$. Further a stochastic trend is assumed, resulting in an ARIMA(1,1,1) (cf. Chapter 3.1.1) with a constant added to make all series strictly positive. For the maximum horizon a value of 24 is chosen for the 100 simulations. Some examples are shown in the Appendix.

2.2 Forecasting Strategies (Recursive, Direct and Combinations)

For multi-step forecasting, i. e. forecasting for a horizon greater one, several approaches exist. The 2 main ones are called *Recursive* (or *Iterative*) and *Direct*.

In a Recursive forecast the 1-step ahead forecast model f is just repeatedly applied up to the desired horizon. So using a rolling lagged values window of size d, i.e. target values y_t from time point $t = T - d + 1$ till time point $t = T$, together with other external covariates $\boldsymbol{x}$, the forecasts are given by:

$$\hat{y}_{T+h} = \begin{cases} f(y_T, \dots, y_{T-d+1}, \boldsymbol{x}) & if\ h = 1 \\ f(\hat{y}_{T+h-1}, \dots, \hat{y}_{T+1}, y_T, \dots, y_{T+h-d}) & if\ 2 \le h \le d \\ f(\hat{y}_{T+h-1}, \dots, \hat{y}_{T+h-d}, \boldsymbol{x}) & otherwise \end{cases} \tag{2.1}$$

Throughout this thesis the covariate vector $\boldsymbol{x}$ is assumed to be known also for future time points which results in *ex-post* forecasts for the target values and stands in contrast to the usage of forecasted covariate values which represents *ex-ante* forecasts.

For a Direct forecast different models f_h are calculated, one for each horizon h, using only known lagged target values till time point T:

$$\hat{y}_{T+h} = f_h(y_T, \dots, y_{T-d+1}, \boldsymbol{x}) \tag{2.2}$$

As the Recursive strategy is reusing forecasted values in its lag-set, possible errors, due to misspecification, cumulate over horizons. Furthermore the Recursive forecast is per se biased in case of a nonlinear data generating process. On the other hand the Direct forecast is more robust to such misspecifications but has higher variance due to the manifold models created and is therefore also computational expensive and possible interpretation gets quite complicated. Moreover the Direct forecast ignores the dependencies of target values. Only in case of a strict linear data generating process and a linear model, both forecast strategies produce equal results. In all other cases, the question whether the lower bias of the Direct or the lower variance of the Recursive strategy carries more impact is an empirical one.

On the other hand, several variations and hybrid strategies exist that try to tackle one or more of these problems. For instance, one can use a Recursive approach but optimize the h-step-ahead forecast error instead of 1-step-aheads. Or the forecast errors can be used as additional input for a Recursive forecast which is actually related to a moving average term in ARIMA models (see Chapter 3.1.1). Somewhat similar, the forecasts themselves can be used as input for direct forecast models. A variation which is also more related to the Direct strategy is the multi-output forecast where all horizons are forecasted at once using a multivariate target variable. Further combinations are possible especially when also taking into account different strategies for different parts of the horizon-range.

One attractive approach in this context that keeps computation time more manageable, is represented by a Direct strategy that is used for the first nth part of the whole horizon-range and recursively applied to the other n-1 blocks by using the “older” forecasts as input for a recursive strategy. For seasonal time series the period s offers a natural partition for the horizon-range blocks and the strategy can be described as follows:

$$\hat{y}_{T+h} = \begin{cases} f_h(y_T, \dots, y_{T-d+1}, \boldsymbol{x}) & if\ h \le s \\ f_{h \bmod s}(\hat{y}_{T+h-1}, \dots, \hat{y}_{T+h-s}, y_{T+h-s-1}, \dots, y_{T+h-d}, \boldsymbol{x}) & if\ s < h \le d \\ f_{h \bmod s}(\hat{y}_{T+h-1}, \dots, \hat{y}_{T+h-d}, \boldsymbol{x}) & otherwise \end{cases} \quad (2.3)$$

This approach (without the strict seasonal block adaption) is used by Zhang et al. (2013) and named *MSVR* as they apply multiple support vector regression models. But obviously the strategy can be applied to any prediction model and is therefore named *sRecDir* (for seasonal recursive application of direct forecasts) in this thesis, better reflecting the nature of the strategy.

It is important to notice that the classical approaches ARIMA and ETS (see Chapter 3.1 and 3.2) are using a Recursive strategy by design even though it is possible to estimate their parameters by minimizing h-step ahead loss (with $h > 1$). Even though there is no such limitation for the Machine Learning approaches, it was decided to use the Recursive strategy for these models throughout this thesis but apply the sRecDir strategy again to the winner models of the NN5 benchmark to check whether these models can still be improved, see final comparison in Chapter 5.3. This approach not only keeps processing time in a reasonable frame (as h=1-56 is relatively wide) but also allows investigating whether a pure Direct strategy helps in the first season (h=1-7).

2.3 Performance Analysis for Time Series Forecasting

This chapter explains the manifold options to measure forecast performance which is inevitable to assess the results of the following chapters. For illustrating purposes some results for models first explained in later chapters, are presented. Actually the explicit models used are not important here as they should only illustrate some general facts. But already starting from now on a special coloring scheme is used if results from several variants of different models are presented. This scheme is kept throughout this thesis (apart from some line plots in Chapter 3) and helps in distinguishing the approaches used in a constant manner.

2.3.1 Performance Metrics

A vast bunch of metrics for measuring continuous target forecast performances exist and often make the results of forecast benchmarks confusing due to different rankings for different metrics, additionally depending on the forecast horizon (cf. Makridakis & Hibon (2000)). Due to this fact, it is absolutely important to understand the advantages, disadvantages and pitfalls of the commonly used metrics in order to avoid misleading benchmark results influencing the decision for an algorithm as well as false assessment of the benefit a forecasting method can provide. The best reference for this often overlooked fact is Hyndman & Koehler (2006). The following explanations are highly influenced by this resource adding some important points not mentioned there.

See also Table 2.1 summarizing the main performance metrics together with their advantages and disadvantages.

The first group discussed is about scale-dependent measures. The most popular metric form this group due to some optimality properties in classical regression is the Mean Squared Error $MSE = mean(y_t - \hat{y}_t)^2$. Notice that the mean operator here usually stands for the mean of squared forecast errors over the horizons, e.g. $mean(y_t - \hat{y}_t)^2 = \frac{1}{h}\sum_{t=T+1}^{T+h}(y_t - \hat{y}_t)$, but can as well be used for the mean over different benchmark time series for exactly one (!) specified horizon $mean(y_i - \hat{y}_i)^2 =$

Name	Formula	Disadvantages	Advantages
Mean/Median Squared Error	$MSE = mean(y_t - \hat{y}_t)^2$ $MdSE = median(y_t - \hat{y}_t)^2$	- unscaled - sensitive to outliers - on different scale than observations	+ known in other fields
Root Mean Squared Error	$RMSE = \sqrt{MSE}$	- unscaled - sensitive to outliers	+ same scale as observations
Mean/Median Absolute Error	$MAE = mean(\lvert y_t - \hat{y}_t \rvert)$ $MdAE = median(\lvert y_t - \hat{y}_t \rvert)$	- unscaled	+ less sensitive to outliers + simple to explain
Mean/Absolute Percentage Error	$MAPE = \text{mean}\left(100 * \frac{\lvert y_t - \hat{y}_t \rvert}{y_t}\right)$ $MdAPE = \text{mean}\left(100 * \frac{\lvert y_t - \hat{y}_t \rvert}{y_t}\right)$	- need $y_t > 1$ - higher penalty on positive errors - need similar ranges for compared time series	+ scaled + simple
Symmetric Mean/Absolute Percentage Error	$sMAPE = mean\left(200 * \frac{\lvert y_t - \hat{y}_t \rvert}{(y_t + \hat{y}_t)}\right)$ or $mean\left(200 * \frac{\lvert y_t - \hat{y}_t \rvert}{(\lvert y_t \rvert + \lvert \hat{y}_t \rvert)}\right)$ $sMdAPE = median\left(200 * \frac{\lvert y_t - \hat{y}_t \rvert}{(y_t + \hat{y}_t)}\right)$ or $median\left(200 * \frac{\lvert y_t - \hat{y}_t \rvert}{(\lvert y_t \rvert + \lvert \hat{y}_t \rvert)}\right)$	- higher penalty on lower forecasts errors - need similar ranges for compared time series - can be negative (for the first definition)	+ scaled
Mean/Median Absolute Relative Error	$MRAE = mean\left(\frac{\lvert y_t - \hat{y}_t \rvert}{\lvert y_t - \hat{y}_t^* \rvert}\right)$ $\text{MdRAE} = median\left(\frac{\lvert y_t - \hat{y}_t \rvert}{\lvert y_t - \hat{y}_t^* \rvert}\right)$	- can be undefined/infinite	+ scaled + benefit over naïve method directly assessable
Mean/Median Absolute Scaled Error	$MASE = MAE/MAE_{in}^*$ $MdASE = MdAE/MAE_{in}^*$	- higher penalty on time series for which naïve benchmark has good in-sample performance	+ scaled + no restrictions regarding time series values

Table 2.1: *Main performance metrics in time series forecasting with advantages and disadvantages (* marks benchmark values; see text for details).*

$\frac{1}{S}\sum_{i=1}^{S}(y_i - \hat{y}_i)^2$. Even though the first version is mostly meant, this double usage can lead to some confusion as further explained below. In order to have the metric on the same scale as the observations, also the Root Mean Squared Error $RMSE = \sqrt{MSE}$ is used. A more outlier insensitive measure represents the Mean Absolute Error $MAE = mean(|y_t - \hat{y}_t|)$ or an even more robust version, the Median Absolute Error $MAE = median(|y_t - \hat{y}_t|)$.

All these measures are not appropriate for comparing or aggregating results for time series with different scales (e.g. time series ranging between 1-2 and 1-1000). And it is very uncommon for time series to be initially scaled in order to circumvent this problem. Furthermore this would not solve the problem if the train and test fold are on a different scales already for a single time series due to a trend or increasing seasonality for instance. Though, for same scaled series like the data sets of the NN5 competition representing daily ATM withdrawals, these measures make sense.

The usual approach to tackle different scaled data is the usage of measures based on percentage errors. The most popular representative from this group is the Mean Absolute Percentage Error $MAPE = mean(100 * |y_t - \hat{y}_t|/y_t)$ with the more robust version taking the median, resulting in MdAPE (Median Absolute Percentage Error). To use this metric, one must assure that the time series are strictly positive or even assure it to only have responses greater than 1 in order to avoid excessively

high metric values. Furthermore this type of metric have the disadvantage to put a higher penalty on positive errors $y_t - \hat{y}_t$ than on negative errors (assuming the average of y_t and $\hat{y}_t$ is the same); e.g. think of a forecast value of $\hat{y}_t = 2$ for a real value of $y_t = 1$ and vice versa resulting in percentage errors of 100 and 50 respectively. To tackle the latter problem the Symmetric Mean Absolute Percentage Error $sMAPE = mean(200 * |y_t - \hat{y}_t|/(y_t + \hat{y}_t))$ was invented together with its robust counterpart sMdAPE using the median. Now the measure is symmetric regarding positive and negative errors but not regarding high and low forecasts. E.g. a forecast of $\hat{y}_t = 3$ for a value $y_t = 2$ results in $sMAPE = 40$ whereas $\hat{y}_t = 1$ leads to $sMAPE = 66.6$, putting a higher penalty on lower forecasts. This problem also resists the adaption of using absolute values in the denominator ($mean(200 * |y_t - \hat{y}_t|/(|y_t| + |\hat{y}_t|))$) to avoid negative sMAPE values due to negative forecasts.

One further problem occurs when the range (maximum-minimum) of different series in the test fold is similar but the minima highly differ. Actually this pitfall is equal to the one regarding values near zero already mentioned above. For example, a time series ranging between 1001-1002 in the test fold (not uncommon for time series with a trend) produces a sMAPE of $o(10^{-3})$ whereas a time series with a range of 1-2 results in $o(1)$. This situation is also happening for the Tourism and the M3 competition data and even occurs less severe for the simulated Arimasim data. Again the NN5 series do not suffer from this problem as all series are on the same scale and have similar ranges.

All above problems are circumvented by a different type of scaling utilized by metrics based on relative errors. Here each forecast error is scaled by its counterpart from a benchmark method usually the *naive* (taking last value) or *snaive* (seasonal naïve: taking last seasonal value) forecast. Unfortunately the Mean Relative Absolute Error $MRAE = mean(|y_t - \hat{y}_t|/|y_t - \hat{y}_t^*|)$ (with $\hat{y}_t^*$ denoting the forecast from the benchmark method) can be undefined in case of a perfect fit of the benchmark method leading to a denominator of zero. If this happens in the analysis of this thesis, the NA (not applicable) value is used as a replacement. Even though assuming an infinite metric value in this case and applying the robust version using the median MdRAE does only help when multi-horizon forecast are aggregated by this measure (and the naïve forecast does not perfectly fit the majority of horizons) or when summarizing over time series instead of forecast horizons (and the naïve forecast does not perfectly fit the majority of series). On the other hand, a nice feature of this metric is that one can get the information that the chosen method is x% better on average (using mean or median for averaging) than the naïve method which is a very helpful information for evaluating the benefit against the effort of the non-naïve method.

A third alternative for scaling performance metrics avoiding all described disadvantages so far, was invented by Hyndman & Koehler (2006). The Mean Absolute Scaled Error $MASE = mean\left(|y_t - \hat{y}_t|/\left(\frac{1}{T-1}\sum_{i=2}^{T}|y_i - y_{i-1}|\right)\right)$ scales every error by the in-sample MAE (which is the mean of the 1-step-ahead in-sample errors) of the naïve forecast: $\frac{1}{T-m}\sum_{i=m+1}^{T}|y_i - y_{i-1}|$. The formula for the MASE can also be written as $MASE = MAE/MAE_{in}^*$ with MAE_{in}^* denoting the in-sample MAE of the naïve benchmark. It is important to notice that the counterpart using the median instead of the mean is restricted to the nominator, i.e. $MdASE = MdAE/MAE_{in}^*$. The only disadvantage also popping up in the Tourism forecasting competition and commented on Hyndman (2015b), is that the "MASE can be very sensitive to a few series [i.e. the ones with a small MAE_{in}^*], and to optimize MASE [in a competition] it is worth concentrating on these".

Somewhat between the scaled metrics based on relative errors and metrics based on scaled errors, which also preserves the nice comparison feature related to the naïve benchmark, is the Relative Mean Absolute Error $RelMAE = MAE/MAE^*$ (with MAE^* denoting the already aggregated MAE of the benchmark method). Obviously one can use arbitrary performance metrics in the ratio in order to compare the desired measure with the benchmark value, resulting in RelRMSE, RelMAPE, etc.

Similar to this approach, as also comparing just the final metrics, are rankings. Here several (including naïve methods) algorithms are assigned ranks due to their performance (measured by any of the metrics shown in Table 2.1). The latter is highly used in the benchmark tests of this thesis as explained below.

It is generally important to use different naïve benchmarks when conducting such comparisons. For the analyses of this thesis not only the seasonal naïve (*snaive*) is used but also two approaches to account for a possible trend by *tsnaiveD* and *tsnaiveSTL*. The first one is applying a seasonal differencing and assumes a random walk with drift for the differenced series; see Chapter 3.1.1 for an explanation of differencing and the random walk with drift model. The second one is doing the same but accounting for seasonality by a STL decomposition (cf. Chapter 3.3). In fact both models assume that the time series can be best forecasted by a linear trend + seasonality + random variation. The additional alternatives *tsnaiveD_bc* and *tsnaiveSTL_bc* apply an initial Box-Cox transformation (cf. Chapter 3.1.1 for explanations of Box-Cox transformations) to account also for increasing seasonality.

In fact this last group of metrics (relative measures and rankings) is different from the former ones because of computing the comparison on already calculated performance metrics. But it makes the topic of time series performance metrics even more confusing as one always not only has to make clear for a benchmark study when and how (using mean or median) the aggregation over horizons is conducted in conjunction with the aggregation over different time series but also denote the point when different forecast algorithms are compared, e.g. by ranking, during this process.

Actually it is just the additional aggregation over different horizons, together with the known phenomenon that performances can differ from horizon to horizon, which makes interpreting time series benchmarks more complicated than standard benchmarks!

2.3.2 Performance Plots and Tables

For the benchmark studies of this thesis only the robust versions of sAPE, RAE and ASE, i. e. sMdAPE, MdRAE and MdASE, are used as outliers can occur just by chance due to the sheer amount of time series. Just for the NN5 data, a final comparison with the results of other authors is done on the basis of sMAPE as this was the target metric in the competition. Already note that a direct quantitative comparison with the results of the Tourism competition (with MASE as the evaluation metric) is not possible as the test data was not available (see also Chapter 2.1). Also for M3 only a qualitative comparison with the competition results can be given partly due to the reduction to series comprising more than 120 time points in this thesis (see below).

As an exception, the MAE is suitable for the simulation study in Chapter 4.5 as the predictions from different models are all comparable and do not contain outliers.

The main shortage which prohibits a comparison with quantitative M3 results is that these are presented in Makridakis & Hibon (2000) just on an overall aggregated level (over all time series) even though forecastability according the different categories highly differs! This can be nicely illustrated by a lineplot showing for example how the MdRAE, aggregated over all series of a category, evolves by horizon. This is done in Figure 2.5 for a bunch of classical (explained in Chapter 3) and naïve models. Remarkably *DEMOGRAPHIC* and *FINANCE* tagged time series show a similar behavior regarding the forecast performance of the methods. Especially for the *MACRO* time series, the “ets” tagged models show an inferior performance. For these three categories a clear improvement to a seasonal naïve model (grey line, constant due to plotting the relative to *snaive* forecast error) results. But when comparing to the *tsnaiveSTL* based models the dominance disappears at least after half of a season (6 time points) indicating that the time series from these categories just consist of a trend + seasonality

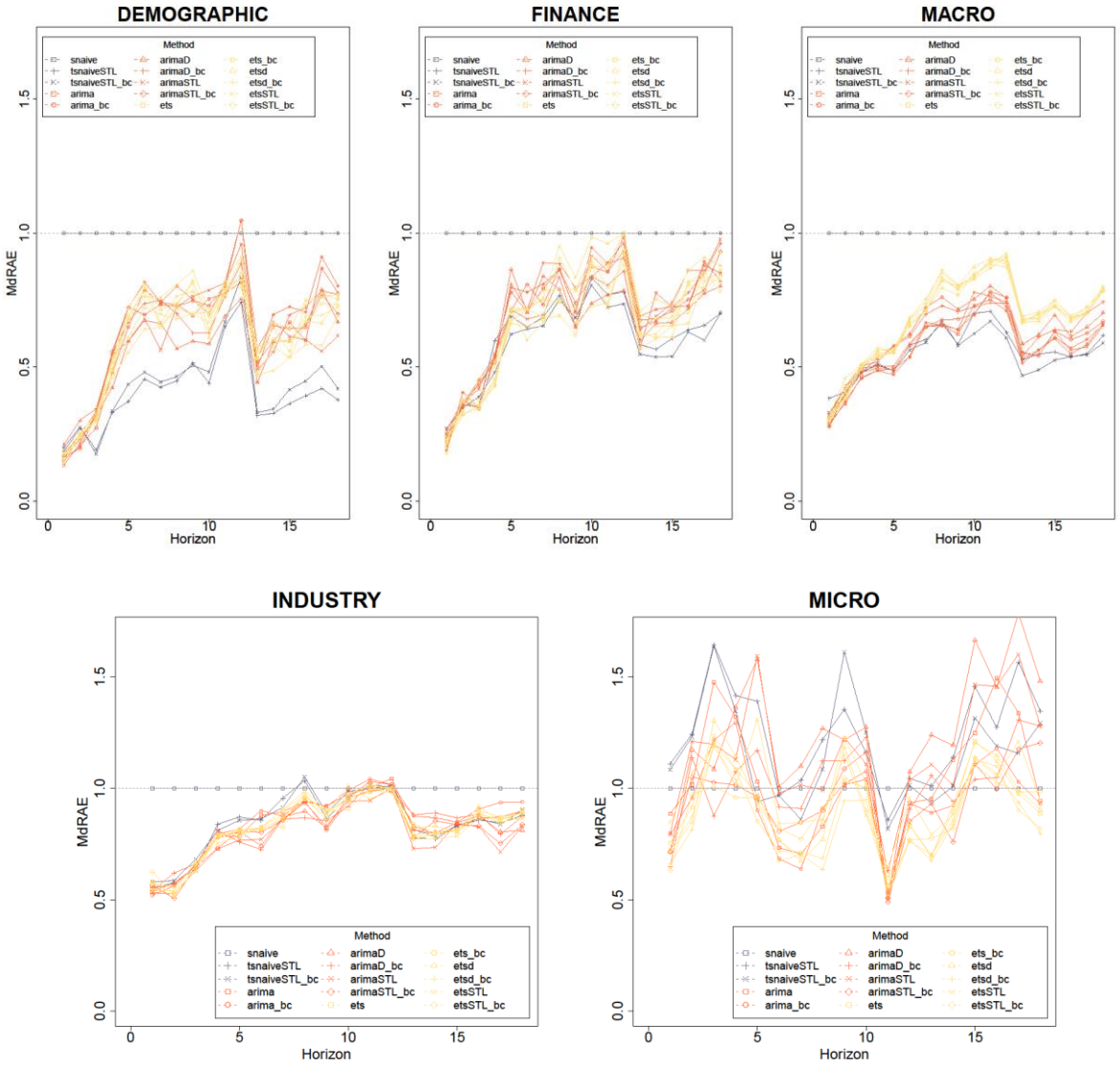

Figure 2.5: *MdRAE by horizon for M3 competition time series grouped by category for some classical forecast models.*

+ random variation and can be best forecasted by a deseasonalized random walk with drift which is the forecast technique behind *tsnaiveSTL*. This fact makes any benchmarking of sophisticated models for these time series highly questionable!

For the two remaining categories *INDUSTRY* and *MICRO* the situation is different, showing a better forecast performance than the naïve methods even though the classical models for the *MICRO* time series exhibit a very special shape, kind of alternating between better and worse than seasonal naïve forecasts emphasizing again the importance of distinguishing also these two categories. This alternating behavior can also be found for the Tourism competition data (not shown here), leaving the *INDUSTRY* data sets as the only interesting time series from the M3 competition that are eligible to reveal additional insights.

Due to these findings any further comparison for M3 data is restricted to *INDUSTRY* tagged time series which is by chance also the category comprising most of the competition data sets. For reasons of completeness the results for *DEMOGRAPHIC*, *FINANCE*, *MACRO* and *MICRO* are compactly put into the Appendix.

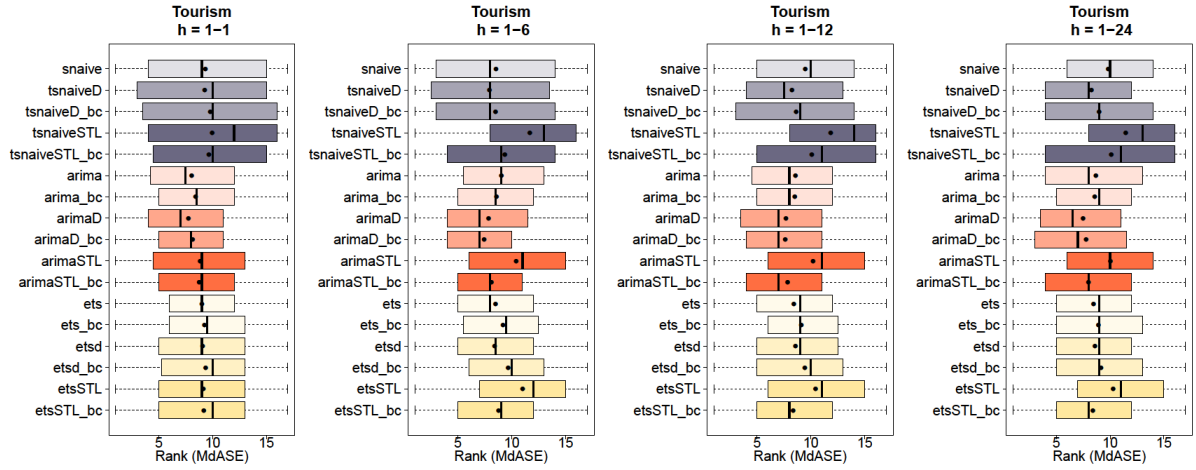

Figure 2.6: *Ranked MdASE by horizon range of some naïve and classical forecast models for Tourism competition data.*

Method	Tourism h=1-1	Tourism h=1-6	Tourism h=1-12	Tourism h=1-24	Method	Tourism h=1-1	Tourism h=1-6	Tourism h=1-12	Tourism h=1-24
snaive	9.35	8.54	9.48	9.82	arimaSTL	8.84	10.41	10.18	10.04
tsnaiveD	9.27	7.94	8.24	8.27	arimaSTL_bc	8.76	8.11	7.84	8.00
tsnaiveD_bc	9.79	8.49	8.64	8.98	ets	9.01	8.50	8.42	8.46
tsnaiveSTL	9.96	11.68	11.85	11.46	ets_bc	9.25	9.20	9.10	8.92
tsnaiveSTL_bc	9.66	9.36	10.09	10.10	etsd	9.07	8.42	8.58	8.58
arima	8.05	9.03	8.56	8.68	etsd_bc	9.35	9.66	9.45	9.16
arima_bc	8.42	8.57	8.49	8.58	etsSTL	9.14	11.01	10.44	10.29
arimaD	**7.76**	7.85	7.68	**7.48**	etsSTL_bc	9.19	8.79	8.35	8.41
arimaD_bc	8.14	**7.44**	**7.62**	7.77					

Table 2.2: *Average Ranked MdASE corresponding to dots in Figure 2.6. Best model metric per horizon range (with vertical split just for visual convenience) is bold & underlined and shading denote Top20% accordingly.*

The dominant method for evaluating the overall performance between algorithms in this thesis is a Ranking based on the MdASE metric. The metric is calculated for every model according a special horizon range and subsequently a ranking of the models per time series is done. This ranking is then aggregated over time series and displayed by a boxplot which allows also a visual impression of the variability of the performance. The horizon range typically comprises h=1 (1-step-ahead forecast), at least the first season and the largest range which is usually the competition objective, nicely summarizing the most informative time areas, even though alternatively this analysis can be conducted for every season (or half season) individually for instance. For example, for the NN5 data the latter would exhibit the performance boost in the 3rd week due to Easter effects for the models using covariates.

Figure 2.6 shows an example, plotting the MdASE for some naïve methods together with some classical approaches for the Tourism data. As the top performing models are often close, an additional tabulation of the mean ranks is usually given, underlining the best metric per horizon. Additionally shading the TopX% helps in identifying the runner-ups (see Table 2.2).

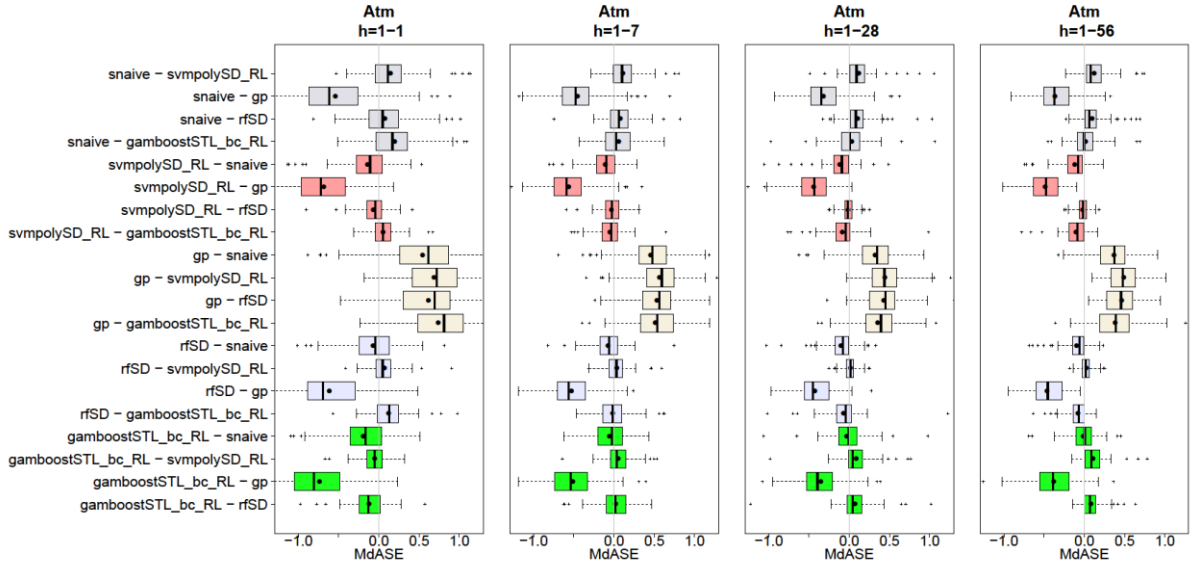

Figure 2.7: *Differences of MdASE ("MdASE of left model – MdASE of right model") by horizon range for some selected models applied on NN5 competition data.*

When using a different robust metric, the results can change, but usually not very extreme on a qualitative scale. This also holds when switching to unrobust measures as the ranking per time series kind of robustifies the performance metrics again. Obviously the situation changes when just reporting e.g. the unrobust MASE averaged over all time series without the intermediate ranking! Something similar happens for the NN5 benchmark using the 4th best method (of 226) according ranked MdASE to outperform all competition competitors when using the competition objective metric, i.e. sMAPE (see Chapter 5.3).

The boxplots in Figure 2.6 exhibit a special shape, i.e. a left skewed form for methods better than the average and right skewed for the other half. This is due to the fact that also the best overall method is also under the worst for some benchmark time series and vice versa which can already be seen by the whiskers comprising the whole range of ranks. This circumstance further indicates that also a paired comparison of performance metrics will not reveal more information which theoretically can happen as one deals with paired measures. Actually it might happen that even the boxplots of two methods highly overlap their paired metric difference shows a more distinct behavior. To evaluate this possibility one can plot the paired metric differences like in Figure 2.7 showing results for some selected models (for explanations of these models see the forthcoming chapters). More precise, these models are the seasonal naïve (*snaive*); the best model according MdASE and the whole test horizon h=1-56 named *svmpolySD_RL*; the according worst model *gp* and two medium performant approaches *rfSD* and *gamboostSTL_bc_RL*. Actually this plot exhibits some typical behavior that often occurs in time series forecasting benchmarks. First of all, it can be seen that the *snaive* beats any of the other models (including the winner *svmpolySD_RL*) at all horizon ranges for some time series. More striking is the fact that this holds mostly also for the worst model *gp*. Furthermore it can be seen that for the 1-step ahead forecast the *gamboostSTL_bc_RL* is better than the overall winner *svmpolySD_RL* but drastically loosing when compared on the whole horizon range h=1-56. This gets more remarkable when comparing the *gamboostSTL_bc_RL* with the *rfSD* model that improves its forecast performance the wider the horizon range is and therefore showing an opposite behavior regarding horizon related change.

It must be added that above demonstrated “performance overlap” is even worse for the Tourism and M3 competition due to the models being less differentiating regarding the forecast performance and as well because of dealing with more time series compared to NN5!

Together with above findings regarding differing rankings for different metrics, the no-free-lunch theorem stating that there is no best predictive model for all data must be enlarged when dealing with time series, indicating a dependency of the question for the best algorithm not only on the used metric but also on the horizon!

This has severe implications on the practical application of forecast models as the user must carefully define the objective regarding the desired horizon range. Also the performance metric should be sensibly chosen taking the results from Chapter 2.3.1 into account and simultaneously decide whether a robust or outlier-sensitive performance metric better suits the objective.

3 Classical Time Series Models

3.1 ARIMA and Friends

In the following the classical ARIMA model is explained together with extensions. A very handy introduction can be found in Ruppert (2010). Shumway & Stoffer (2011), on the other hand, is a more technical text book. As always, the book from Hyndman & Athanasopoulos (2014) has a nice introduction for the practitioner. Furthermore the blog entry “The ARIMAX model muddle” from Hyndman (2015b) is very helpful.

3.1.1 ARIMA

The classical ARIMA (autoregressive integrated moving average) approach tries to model a time series by the autocorrelation due to the dependence of y_t on former values. For example, an autoregressive process of order p, named AR(p), models the current mean-centered target value by (auto-)regressing it on its own past values $y_t = \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + w_t$ with w_t denoting a white noise process, i.e. a family of uncorrelated, not necessarily independent (!), random variables with constant covariance and expected value equal zero. Mostly a Gaussian distribution is assumed which in turn results in i.i.d. white noise. Using the autoregressive operator $\phi(B) = 1 - \phi_1 B - \cdots - \phi_p B^p$, with B denoting the backshift or lag operator ($B^p y_t = B^{p-1} y_{t-1} = \cdots = B y_{t-p+1} = y_{t-p}$), above model equation can be written more compact by $\phi(B) y_t = w_t$.

An important concept for ARIMA modeling is weakly stationarity, which means that $E(y_t) = \mu_t = const.$ and $\gamma(t, t') = cov(y_t, y_{t'}) = \gamma(|t - t'|)$, i.e. the autocovariance is only dependent on the time difference $|t - t'|$, including constant variance assumption ($t == t'$). This requirement not only leads to some constraints for the possible coefficients $\phi_{1..p}$, e.g. $|\phi_1| < 1$ in case of an AR(1) process, but more importantly, is at least needed to estimate the parameters $\phi_{1..p}$ from just one time series (as otherwise not enough data is left for estimation).

ACF and PACF

An important diagnostic tool for ARIMA modeling is the sample PACF (partial autocorrelation function) plot showing the direct influence, or correlation adjusted on the influence of intermediate points, of y_{t-h} on y_t. Figure 3.1 shows an example of this plot together with the sample ACF (autocorrelation function) plot denoting the unadjusted influence for a simulated AR(2) process. One can see that the PACF needles cut off after p=2 lags (apart from non-significant values) hinting to the correct number of coefficients, whereas in the ACF the coefficients die out slowly.

Apart from applying these plots to stationary data, they can also be used determining the relationship with lagged values in other modeling approaches, e.g. detecting seasonality by periodic ACF spikes or a significant PACF spike at the first seasonal lag. But one always has to keep in mind that these plots just visualize the linear correlation of time points and can therefore hide nonlinear relationships which would pop up when plotting the time point values against their lagged counterparts.

Sometimes data shows the opposite behavior compared to Figure 3.1: An ACF plot with just one or two significant lags and a PACF plot with out-dying lags. This cannot be modeled by an AR process as such a process always results in having some correlation to all past values. In order to model this parsimoniously, i.e. without estimating lots of (or infinite) AR-coefficients, a MA(q) (moving average) process is assumed where the current mean-centered value of the time series is modeled as a moving average of q+1 past white noise realizations: $y_t = w_t + \theta_1 w_{t-1} + \cdots + \theta_q w_{t-q}$. One can also say that

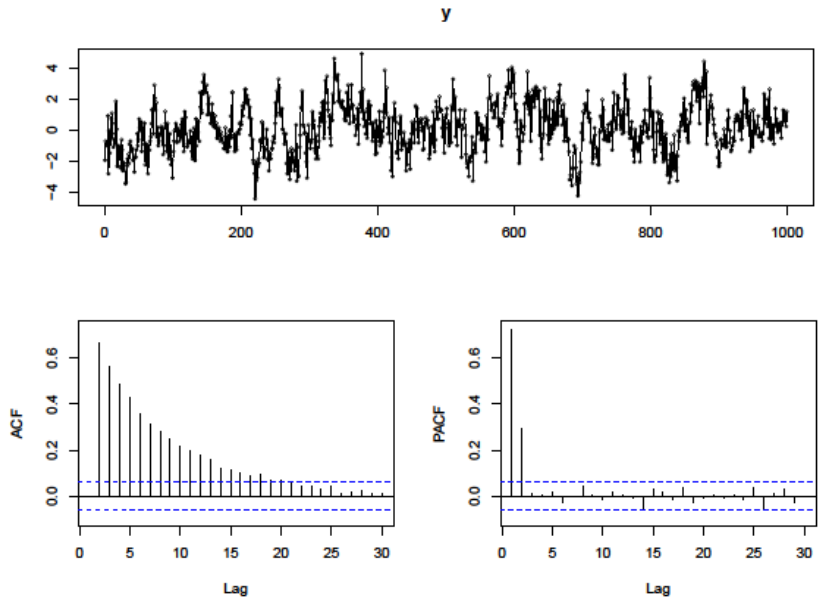

Figure 3.1: *AR(2) ($\phi_1 = 0.5, \phi_2 = 0.3$) simulated time series together with sample ACF and PACF plot. The method-of-moments estimation of the "last" coefficient (here $\hat{\phi}_2$) can always be directly read from the PACF plot.*

the current value is a weighted average of past forecast errors as the forecast of white noise is always zero. Similar to the AR case this can be written in a more compact fourmulation now using a (moving average) operator $\theta(B) = 1 + \theta_1 B + \cdots + \theta_q B^q$ resulting in: $y_t = \theta(B) w_t$.

Assuming a MA process for the AR white noise part, a combined ARMA(p,q) process results in the compact notation:

$$\phi(B) y_t = \theta(B) w_t \tag{3.1}$$

Which model, i.e. how many p- and q-coefficient, fit an empirical time series can in theory (!) be determined by the ACF and PACF plots as the relations shown in Table 3.1 hold, which also hint to the interesting and theoretical important property that an AR(p) process can always be written as a MA(∞) process and vice versa, i.e. MA(q) as AR(∞).

Trend and Seasonality

Usually a time series exhibits trend and seasonality which invalidates the stationarity assumption, in fact the $\mu_t = const$ requirement. With classical ARIMA modeling this is tackled by differencing (or "integrating") the target value. For example, first differencing $\Delta y_t = (1 - B) y_t = y_t - y_{t-1}$ removes a linear trend and second differencing $\Delta^2 y_t = (1 - B)^2 y_t = y_t - 2y_{t-1} + y_{t-2}$ a quadratic trend. Applying a seasonal differencing $\Delta_m y_t = (1 - B^m) y_t = y_t - y_{t-m}$ removes the seasonal variation. Theoretically it does not make a difference which differencing is applied first on a time series with trend and season but Hyndman & Athanasopoulos (2014) suggest to do the seasonal differencing first as this often makes additional non-seasonal differencing unnecessary in practice. Furthermore it is common experience that it is almost never required to apply differencing of order greater than 2.

	AR(p)	MA(q)	ARMA(p,q)
ACF	dies out	cuts off after q lags	dies out
PACF	cuts off after p lags	dies out	dies out

Table 3.1: *Theoretical behavior of significant coefficients in ACF and PACF plots for ARMA processes.*

Alternatively the time series can be detrended and deseasonalized, e.g. by standard STL decomposition (see Chapter 3.2). Which approach is more suitable depends on the time series. But, e.g. a strict random walk $y_t = y_{t-1} + w_t$, which is non-stationary due to the increasing variance by time, can only be transformed to a stationary time series (i.e. white noise in this case) by first differencing; the same holds for a so-called random walk with drift: $y_t = c + y_{t-1} + w_t$. The latter relationship is sometimes referred to as "stochastic" trend in contrast to a "deterministic" trend ($y_t = c + bt$) and is the reason for the so-called spurious regression phenomenon that occurs when regressing a random walk on another random walk, which can result in significant regression coefficients (see Cowpertwait & Metcalfe (2009) for an example).

If one now additionally assumes a seasonal correlation similar to the one for single time points, an integrated seasonal ARMA or ARIMA(p,d,q)(P,D,Q)$_m$ results, with d and D denoting the differencing grade for non-seasonal and seasonal differencing respectively. Further p&q and P&Q represent the non-seasonal and seasonal autoregressive and moving average coefficients with index m defining the period of the season (e.g. m=12 for monthly series). With the corresponding seasonal autoregressive and moving average operators $\Phi(B_s)$ and $\Theta(B_s)$ this model can be written in the compact form:

$$\phi(B)\Phi(B^m)\Delta^d\Delta_m^D y_t = \theta(B)\Theta(B^m)w_t \tag{3.2}$$

Box-Cox Transformations

Sometimes a time series exhibit some nonlinear shape, e.g. an increasing seasonality. Such a behavior can be accounted for by an initial Box-Cox transformation where λ can be determined by maximizing the profile log likelihood for a linear model $\tilde{y}_t = \beta_0 + \beta_1 t$ (enlarged by seasonal dummies for seasonal data), treating (β_0, β_1) as nuisance parameter and assuming the following transformation:

$$\tilde{y}_t = \begin{cases} \left({y_t}^{\lambda} - 1\right)/\lambda & \lambda \neq 0 \\ \log y_t & \lambda = 0 \end{cases} \tag{3.3}$$

This transformation is has also some general variance stabilizing capabilities and therefore helps to enforce stationarity. Furthermore it supports linearity in the final model. See Ruppert (2010) for a good reference regarding this transformation class.

In fact, the default Box-Cox detection method of the forecast package which is also used in the benchmark experiments of this thesis, utilizes the alternative method of Guerrero which additionally suggests restricting λ to the interval $[-1;\ 2]$ (Guerrero (1993)).

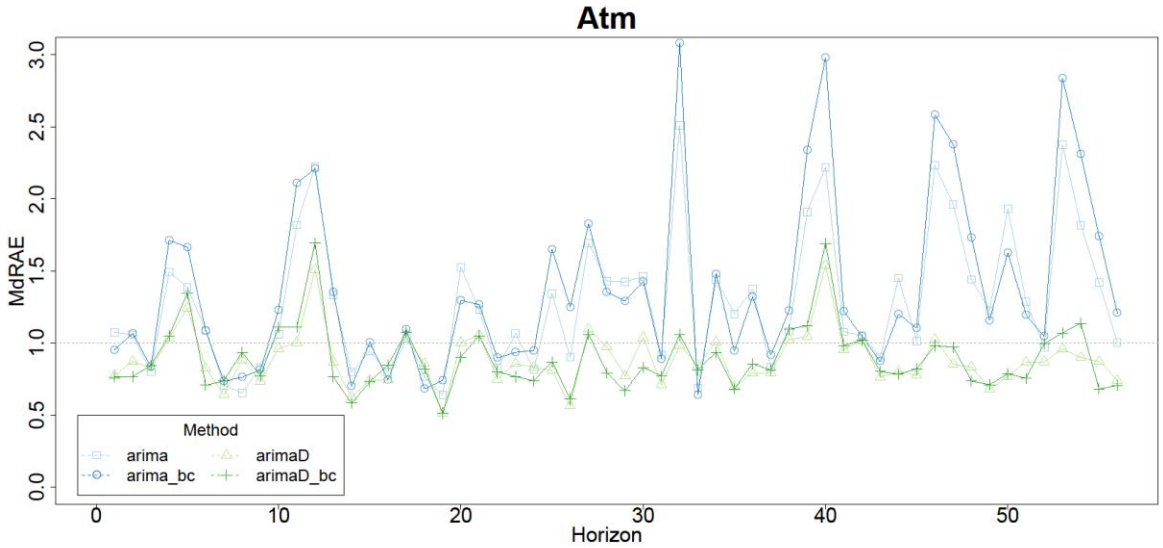

Figure 3.2: *Performance per horizon of standard and seasonal hard-differenced ARIMA models with and without initial Box-Cox transformation for NN5 time series. Notice that in this line plots the coloring scheme used for the ranking boxplots is not utilized as this would make the lines more indistinct. This practice is kept in similar plots that follow.*

Notice that apart from benchmark situations where lots of time series should be model automatically, this transformation should be applied also under interpretability considerations, i.e. choosing a λ parameter which is not exactly equal the optimum might make more sense, e.g. using a log transformation if λ is very small or a square root transformation for λ near 0.5.

Box-Jenkins-Approach

ARIMA modeling for forecasting consists of the following 4 main steps ("Box-Jenkins-Approach"): 1. Identify the orders (p,d,q) and (P,D,Q). 2. Estimate parameters. 3. Conduct diagnostic steps. In the first step the differencing must be evaluated at the beginning by utilizing ACF and PACF plots on original and differenced series and applying formal, so called unit root tests, for stationarity testing. These tests are usually the *Augmented-Dickey-Fuller* and the *KPSS* test, that differ mainly in the null hypothesis which claims stationarity for the KPSS and non-stationarity for the Augmented-Dickey-Fuller test. For seasonal differencing the default test (Osborn-Chui-Smith-Birchenhall test) is assuming non-stationary as the Null hypothesis, see also the *forecast* R-package documentation (Hyndman (2015a)). The range of possible autoregressive and moving average orders can also be read from the diagnostic plots and are typically final-determined by AIC based (AIC or AIC_C) comparison of fitted models. It is important to notice that AIC based values can only be compared for models with same differencing order. For the resulting stationary process the estimation of parameters in the second step can be done by a method-of-moments approach, called "Yule-Walker" in the time series context, or by ML, which is equal to least squares estimation (assuming Gaussian white noise). The modeling is finished if the residuals result in uncorrelated white noise which can be visually checked with ACF plots again (and the corresponding confidence intervals) or more formally tested with a Portmanteau (also named Lijung-Box) test. All these steps apart from the residual checks are automatically conducted when using the *auto.arima* function of the *forecast* R-package (Hyndman (2015a)), which is

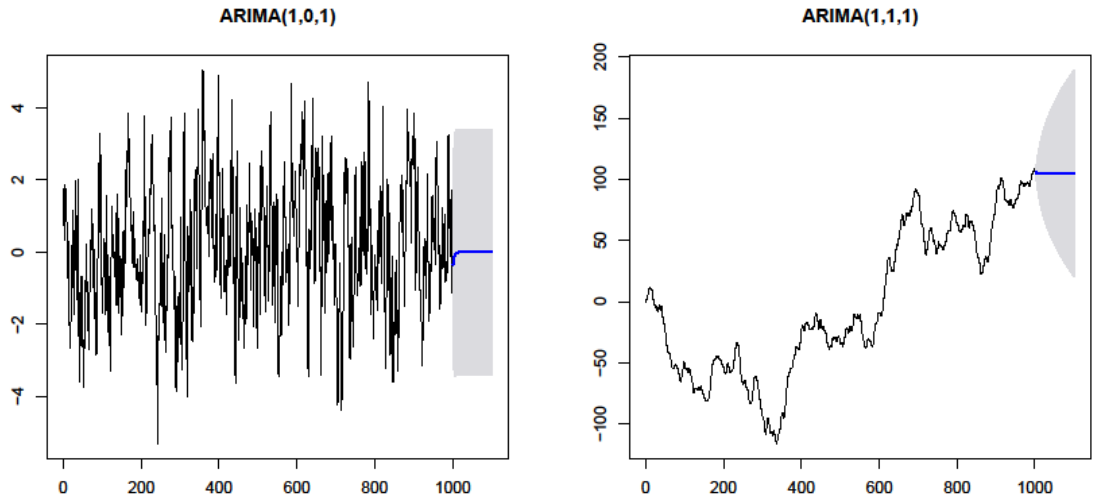

Figure 3.3: *Forecast of simulated ARIMA time series with 0.95 prediction intervals.*

used in all benchmarks presented in this thesis. Possible slight modifications like alternative stationarity tests can be found in the documentation of the R-package. Furthermore whether an initial Box-Cox transformation is required must be derived in advance.

Figure 3.2 shows the results of applying the *auto.arima* function with and without (automatic) Box-Cox transformation, as described in above subsection, coded as *arima_bc* and *arima* respectively. Additionally the seasonal differencing is forced for the corresponding *arimaD_bc* and *arimaD* models. One can see that the Box-Cox transformation does not have a big influence on the prediction performance in the long run which might be expected as the time series do not exhibit high nonlinear behavior. But the strongly reduced performance (which is actually mostly worse than a seasonal naïve benchmark) when letting *auto.arima* determine the seasonal differencing, already occurring for the 1-step-ahead forecast, is striking and indicates that the default test (with H0 denoting seasonal non-stationarity) is too conservative for the data situation which might be due to the high number of observations per time series! Even though less severe, this behavior also occurs for the Tourism and M3-INDUSTRY data suggesting to keep hard-coded seasonal differencing as possible advantageous model alternative in mind!

Forecasting

The final forecasting with ARMA models is solely recursive and minimizes mean squared error (MSE). Basically it results in a straight forward handling by resolving equation (3.1) for y_t, insert known former values, replace future observations by their forecast and use zero for future errors but successive residuals for past errors w_t. When assuming Gaussian white noise for w_t also the prediction intervals can be directly calculated from (3.1). Both lead to the typical and somewhat at first sight surprising ARMA forecast which reverts to the mean value surrounded by intervals converging to a constant width, equal to the marginal variance of the process $\gamma(0)$ (e.g. $\sigma^2/(1-\phi_1{}^2)$ for AR(1)), see left plot of Figure 3.3. This scheme changes to a width-increasing interval when additional non-seasonal and seasonal differencing has been applied (see right plot in Figure 3.3). Obviously also a seasonal component changes the forecast shape from a constant mean to one looking “less naïve”.

It is important to notice that in contrast to classical regression the uncertainty resulting from estimation of the model parameters, i.e. the ARIMA coefficients, is not recognized as it is assumed that their

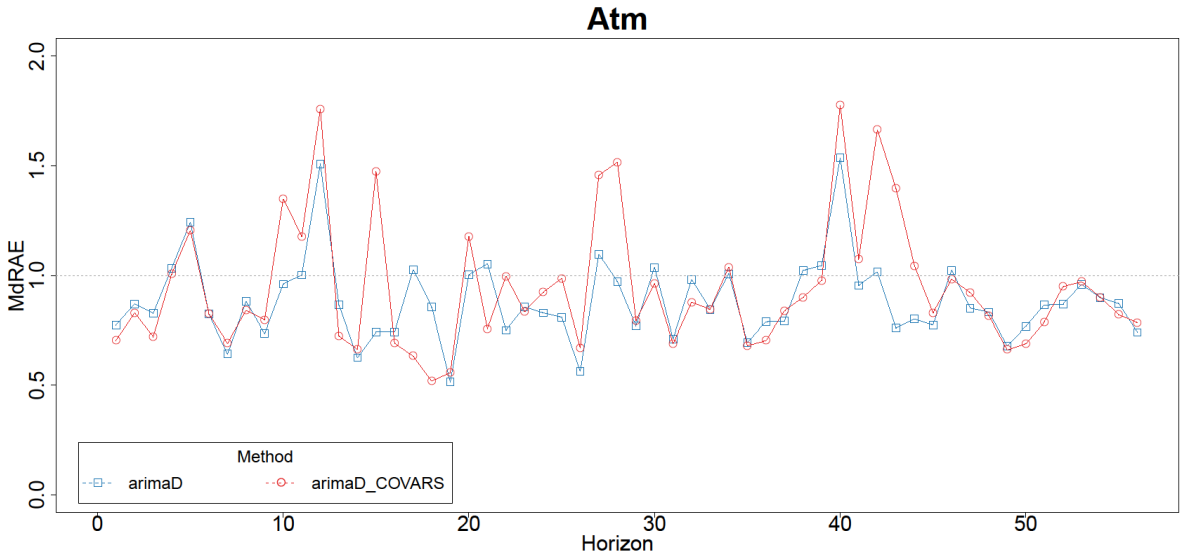

Figure 3.4: *Performance per horizon of seasonal hard-differenced ARIMA models with and without exogenous covariate effects for NN5 time series.*

influence is negligible compared to the error model generating process, resulting in prediction intervals which are usually too liberal due to this neglect.

3.1.2 ARIMAX tagged models

If one also wants to account for exogenous covariate effects several model extensions exist, which are explained in the following.

Regression with ARIMA errors

As the name already says, the i.i.d. error in the standard linear model is replaced by a (possibly differenced) ARMA process, which, by using (3.1), results in the following model equation:

$$z_t = \boldsymbol{x_t'\beta} + y_t = \boldsymbol{x_t'\beta} + \frac{\theta(B)\Theta(B_s)}{\phi(B)\Phi(B_s)}w_t \tag{3.4}$$

It is important to keep in mind that if an initial differencing is applied on the error process y_t, this must also be done for $\boldsymbol{x_t}$ and z_t. More precise, all predictors must be stationary in such a model.

The division by an operator in equation (3.4) is a short notation of applying the inverse operator, which can be derived by treating the backshift operator B as a complex number, see Shumway & Stoffer (2011). For example, the inverse of the autoregressive operator of order 1: $\phi(B) = 1 - \phi_1 B$ can be written as $\phi(B)^{-1} = 1 + \phi_1 B + {\phi_1}^2 B^2 + \cdots$ which in fact is a moving average operator of infinite order and reflects the already stated fact that an AR(p) can always be converted to a MA(∞) process and vice versa.

It is important to recognize that prediction intervals generated by a forecast of an *auto.arima* R object also do not account for the uncertainty of estimating the β coefficients! Another special issue of this

modeling function is related to estimation which is not fully ML (or REML, cf. with below explanations) based but iterating between standard linear regression model estimation and ARMA coefficient estimation.

This model is used for the NN5 benchmark when accounting for additional covariates. Figure 3.4 illustrates that the model comprising covariate effects can be even worse than without. Actually this not very surprising in this case as only a linear predictor with just main effects of the covariates listed in Chapter 2.1 (of course without the seasonal dummies and lag covariates whose influence is modeled directly by the ARIMA model) is used and, as noted in Chapter 2.1, main influences are actually interaction effects. But it is somewhat unexpected that this results in a few but extreme performance reduction peaks even not following any pattern.

ARIMAX

In an ARIMAX model a standard ARIMA model for y_t is enlarged by exogenous covariate effects through adding the linear predictor $\boldsymbol{x_t'\beta}$ on the right side of the ARMA equation (3.1), resulting in (after resolving for y_t):

$$y_t = \boldsymbol{\beta}' \frac{1}{\phi(B)\Phi(B_s)} \boldsymbol{x_t} + \frac{\theta(B)\Theta(B_s)}{\phi(B)\Phi(B_s)} w_t \tag{3.5}$$

For an AR(1) process this leads to mixing up the β-coefficients with exponentiations of the AR coefficients (due to the inverse operators) which makes the covariate effects hard to interpret in this approach. Even though this is not the primary task in forecasting, this model is not used in the benchmarks of this thesis, also due to the practical limitation that the R package for fitting ARIMAX models named *vars* (mostly intended to fit much more complicated multivariate time series) does not provide the same handy order selection as the auto.arima function from the *forecast* package.

Transfer function modeling

If one wants to use lagged covariate effects a *transfer function model* can be used. Here this lagging exogenous influence is modeled as an ARIMA covariate process and combined with an ordinary ARIMA error process to model the response y_t:

$$y_t = \frac{\theta_x(B)\Theta_x(B_s)}{\phi_x(B)\Phi_x(B_s)} \boldsymbol{x_t} + \frac{\theta(B)\Theta(B_s)}{\phi(B)\Phi(B_s)} w_t \tag{3.6}$$

Actually the regression with ARIMA errors (3.4) and the ARIMAX model (3.5) are special cases of this flexible approach which also allows for modelling decaying effects of covariates by applying the moving average operator on the covariate vector. Transfer functions models can be fit with the (confusingly named) *arimax* function of the *TSA* R-package. Again the non-existing automate order selection stopped it from being used in the benchmarks.

Dynamic linear models

These models are just standard multiple regression approaches using lagged target values as additional predictors. Notice that the term “dynamic linear model” is sometimes used for other models

by some authors. This model type is utilized indirectly by employing lagged target values in the *glmboost* model (see chapter 4.4) which therefore represents a boosted dynamic linear model.

Further related approaches

As the situation of correlated response in regression is not uncommon, there exist several other model and estimation approaches invented in different fields. For instance, when the parametrization of the error structure is assumed to be known, like e.g. for AR(1) errors, also a standard weighted regression can be applied (see e.g. Fahrmeir et al. (2013)).

In the field of longitudinal analysis a REML (restricted maximum likelihood) estimation approach is used for this data situation, see e.g. Verbeke & Molenberghs (2008). Usually here only an AR(1) structure is assumed for the residual covariance not trapped by the random covariate effects (which represent the main modeling instrument for correlated responses in this area). For these classical regression approaches the correlation of the data is transferred to the error and usually seen as nuisance which has to be accounted for in order to conduct correct inferential analysis. This stands in contrast to ARIMA modeling where the correlation is the main feature to explain the behavior of the data points!

Strongly related to these classical statistical approaches are Gaussian Process models, popular in the Machine Learning field (cf. Chapter 4.1).

Apart from the latter, none these approaches is regarded further in this thesis as well as additional models popular in the financial sector eligible to care for time-based correlations of the process variance modeled with ARCH/GARCH (generalized autoregressive conditional heteroscedasticity) models. Here also volatility cluster can be realized by assuming an AR process for the conditional (on the past) process variance.

Also multivariate autoregressive models (VAR models) are omitted as these treat target and “covariate time series” symmetrically which does not make sense as all covariates used in the investigated data sets are known in advance (which leads to so-called “ex post” forecasts) and usually exhibit a clear causal direction to the target.

3.2 ETS (Exponential Smoothing)

Even though based on a more heuristic approach, ETS models have a long history in time series forecasting due to a good performance shown in many classical forecasting situations. In contrast to the correlation based ARIMA modeling, ETS approaches concentrate more on modeling the trend and seasonality of the time series. Recently they are based into a strict statistical framework utilizing state space models (Hyndman et al. (2008)) increasing their acceptance and allow the application of a deeper statistical toolset, at least the creation of prediction intervals.

In the simplest form, for a time series with no trend or seasonality, a future value is model as a weighted aggregation of past values with exponentially decaying, and therefore smoothing, weights for older observations: $\hat{y}_{t+1|t} = \alpha y_t + \alpha(1-\alpha)y_{t-1} + \alpha(1-\alpha)^2 y_{t-2} + \alpha(1-\alpha)^3 y_{t-3} + \cdots$ with $0 \leq \alpha \leq 1$. This equation can also be derived by recursively calculating $\hat{y}_{t+1|t} = \alpha y_t + (1-\alpha)\hat{y}_{t|t-1}$, showing the heuristic recipe of creating a weighted average of the most recent value and its forecast. Alternatively, by inventing the level term l_t, the so-called *component form* results, comprising the forecast (3.7) and the smoothing (3.8) equation:

$$\hat{y}_{t+1|t} = l_t \tag{3.7}$$

$$l_t = \alpha y_t + (1-\alpha) l_{t-1} \tag{3.8}$$

Further using the forecast error $e_t = y_t - l_{t-1} = y_t - \hat{y}_{t|t-1}$ and interpreting it as model noise $\varepsilon_t \sim N(0, \sigma^2)$, the following state space model can be assumed:

$$y_t = l_{t-1} + \varepsilon_t \tag{3.9}$$

$$l_t = l_{t-1} + \alpha \varepsilon_t \tag{3.10}$$

Notice the switch $t+1 \rightarrow t$ which is helpful for getting a more convenient model equation for y_t. Here equation (3.10) represents the so-called state equation of the latent variable l_t that cannot be observed but which is characterizing the process behind. Equation (3.9) is named *observation (or measurement) equation* and represents the model equation for the (usually linear transformed) observable y_t. Apart from the simple form, the occurrence of the same error in both equations makes this state space model exceptional.

This simple model can be enhanced by adding a trend resulting in *Holt's linear trend* method. In order to derive the component form it helps to informally (!) interpret equation (3.8) as a weighted average of the estimation of the level $\hat{l}_{t|t}$ at time t which is y_t and at time t-1 $\hat{l}_{t|t-1}$ which is l_{t-1} as the level does not change over time: $l_t = \alpha \hat{l}_{t|t} + (1-\alpha)\hat{l}_{t|t-1}$. If one now adds a trend to the forecast equation (3.7) (resulting in (3.11), see below), both the following level equation (3.12) and the additional trend equation (3.13) can be derived as such a heuristically justified weighted average of the recent value and its forecast:

$$\hat{y}_{t+h|t} = l_t + h b_t \tag{3.11}$$

$$l_t = \alpha \hat{l}_{t|t} + (1-\alpha)\hat{l}_{t|t-1} = \alpha y_t + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{3.12}$$

$$b_t = \beta \hat{b}_{t|t} + (1-\beta)\hat{b}_{t|t-1} = \beta(l_t - l_{t-1}) + (1-\beta) b_{t-1} \tag{3.13}$$

Again the state space representation, now comprising two state equations, can be derived from this formulation easily by resolving:

$$y_t = l_{t-1} + b_{t-1} + \varepsilon_t \tag{3.14}$$

$$l_t = l_{t-1} + b_{t-1} + \alpha \varepsilon_t \tag{3.15}$$

$$b_t = b_{t-1} + \tilde{\beta} \varepsilon_t \tag{3.16}$$

with $\tilde{\beta} = \alpha\beta$.

Similarly a seasonal component can be invented to create a Holt-Winters seasonal model:

$$\hat{y}_{t+h|t} = l_t + h b_t + s_{t-m+h} \tag{3.17}$$

$$l_t = \alpha \hat{l}_{t|t} + (1-\alpha)\hat{l}_{t|t-1} = \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{3.18}$$

$$b_t = \beta \hat{b}_{t|t} + (1-\beta)\hat{b}_{t|t-1} = \beta(l_t - l_{t-1}) + (1-\beta) b_{t-1} \tag{3.19}$$

$$s_t = \gamma \hat{s}_{t|t} + (1-\gamma)\hat{s}_{t|t-1} = \gamma(y_t - l_t) + (1-\gamma) s_{t-m} \tag{3.20}$$

With the corresponding state space model:

$$y_t = l_{t-1} + b_{t-1} + s_{t-m} + \varepsilon_t \tag{3.21}$$

$$l_t = l_{t-1} + b_{t-1} + \alpha\varepsilon_t \tag{3.22}$$

$$b_t = b_{t-1} + \tilde{\beta}\varepsilon_t \tag{3.23}$$

$$s_t = s_{t-m} + \tilde{\gamma}\varepsilon_t \tag{3.24}$$

and $\tilde{\gamma} = \gamma/(1-\alpha)$.

Further models result when assuming a multiplicative trend $\hat{y}_{t+h|t} = l_t b_t^h$, a damped trend $h \to \rho_h = \rho + \rho^2 + \cdots + \rho^h$ or treating the seasonality as multiplicative component, e.g. together with a linear trend: $\hat{y}_{t+h|t} = (l_t + hb_t)s_{t-m+h}$. All combinations can be compactly written in the notation (T,S) with T and S denoting the trend and the season component respectively and taking the values N (none), A (additive), A_d (additive damped), M (multiplicative). For instance, (A,M) stands for a multiplicative Holt-Winters method. The state space approach also allow for a multiplicative error resulting in $y_t = l_{t-1}(1+\varepsilon_t)$ for the observation equation and $l_t = l_{t-1}(1+\alpha\varepsilon_t)$ for the state equation. This enlarges above notation with an additional option of A (additive) and M (multiplicative) error term E in (E,T,S) and allows the notation of e.g. an Holt's linear model with additive error by ETS(A,A,N). Notice the handy double usage of the abbreviation of ETS for "exponential smoothing" as well as for "error-trend-season".

The state space model formulation now allows standard ML based estimation of the model parameters. Apart from some constraints for the parameters to allow for interpretation of weighted averages in the component model, also some specific ETS model can result in numerical difficulties and are therefore excluded from candidate models in the AIC based tuning process of the *ets* function in the *forecast* R-package, cf. Hyndman & Athanasopoulos (2014). Furthermore this function conducts all other modeling steps explained above in an automatic way pretty much like the before mentioned *auto.arima* for ARIMA models. For details see again the documentation in the *forecast* R-package (Hyndman (2015a)).

One big disadvantage of ETS models represents the disability to include exogenous covariates! Although initially used by Hyndman et al. (2008), Hyndman clarified in his blog (Hyndman (2015b)) that "… they are never forecastable in the sense explained in Section 10.2 my 2008 book (forecastability is the ETS equivalent of invertibility in ARIMA models)."

Forecasting

Forecasting ETS models is straight forward by assuming an expectation of 0 for the error term and applying the state space formulas recursively and specify initial values for e.g. l_0, b_0 and s_0. For some models also prediction intervals can be exactly computed. Due to the strict statistical model assumed also parametric and non-parametric bootstrap approaches can always be generated.

Relationships to ARIMA

There is a strong relationship between ETS and ARIMA models. E.g. all linear ETS models, i.e. with additive components, can be equivalently written as ARIMA processes. Though, this does not hold for the multiplicative ETS models. Also many ARIMA models do not have an ETS counterpart. See Hyndman & Athanasopoulos (2014) for some corresponding examples.

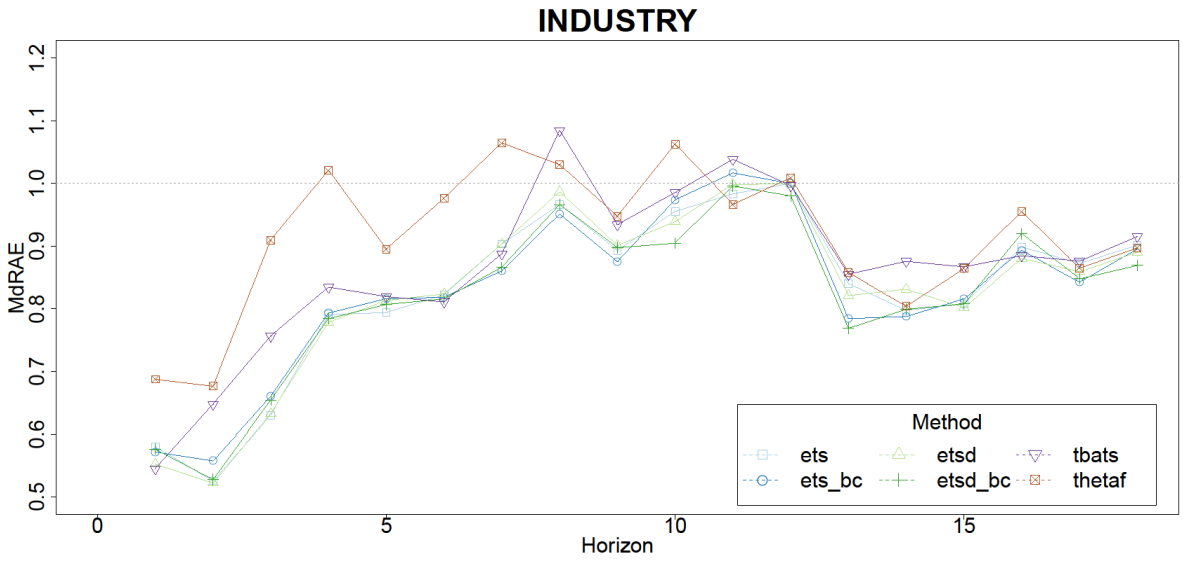

Figure 3.5: *Performance per horizon of ETS models for M3-INDUSTRY time series.*

Related approaches

There are two more interesting models closely related to the ETS approach. The *TBATS* model (De Livera et al. (2011)), implemented under the same name in the *forecast* R-package, is an ETS method handling complex seasonal patterns and allows automatic selection of Box-Cox transformation, seasonal and trend components together with possible ARMA errors.

The other one is *Theta* which showed superior behavior in the M3 competition. Actually Hyndman & Billah (2003) proved that Theta it is just simple exponential smoothing with an additive trend having special slope value.

A first sight on the performance of these methods compared to a standard automatically determined ETS model with and without Box-Cox transformation (identifiable by the suffix *bc*) is given in Figure 3.5 for the INDUSTRY data of the M3 benchmarks for which ETS models are under the best in an overall comparison (cf. Chapter 5.1). Additionally a damped ETS model (*etsd*) is added to the ETS variants used in all forthcoming benchmarks as “Methods that include a damped trend have proven to be very successful and are arguably the most popular individual methods when forecasts are required automatically for many series“ (Hyndman & Athanasopoulos (2014)), especially for longer forecast horizons. It can be seen that there is no big difference regarding the “unrestricted” ETS models *ets* and *ets_bc* and the damped variant *etsd* and *etsd_bc*. But even though more flexible, *tbats* shows inferior performance. Even more worse are the results for *thetaf* exhibiting a bad performance beginning already at the 1-step-ahead prediction. As will be shown in Chapter 3.5 the latter situation also occurs for the other competition time series! Especially for the M3-INDUSTRY data this result is somewhat surprising as the Theta method was one of the best methods in the M3 competition (Makridakis & Hibon (2000)) and might indicate that the *thetaf* implementation of the forecast R-package differs from the one used in the official competition.

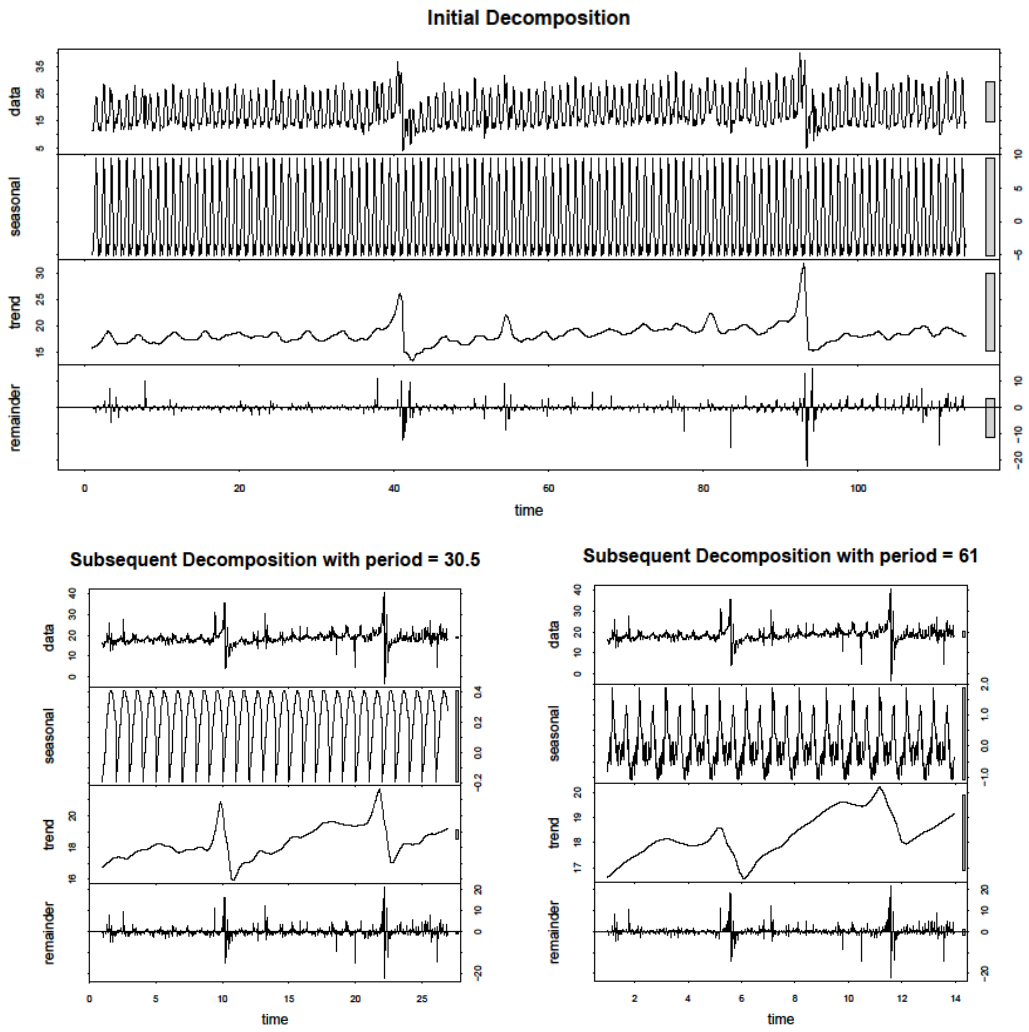

Figure 3.6: *Repeated STL decomposition of averaged (over all series) NN5 time series.*

3.3 Combinations with STL Decomposition

Both ARIMA and ETS can be combined with a deseasonalizing step, which in turn means that the seasonality is not accounted for by differencing in an ARIMA model or a seasonal ETS component. In such a case one hopes to get a better seasonality fit than by these embedded approaches. Usually the deseasonalizing is part of a time series decomposition in trend or trend-cycle component (with cycles denoting patterns not occurring repeatedly by a fixed time frame like seasonal components and which are therefore stationary!), seasonal component and remainder.

Easy-to-use but also somewhat imprecise approaches use combinations of moving-average smoothers for the decomposition. In the econometrics field the sophisticated X-12-ARIMA method is popular (Ladiray (2001)).

A very versatile method is represented by STL (Seasonal and Trend decomposition using Loess, Cleveland et al. (1990)) even though a multiplicative decomposition must be tackled with an initial Box-Cox transformation. This method loops several times through the data for repeated detrending and deseasonalizing utilizing eigenvalue and frequency response analyses and of course, as the name states, *loess* (locally weighted regression, see Fahrmeir et al. (2013) for instance) based

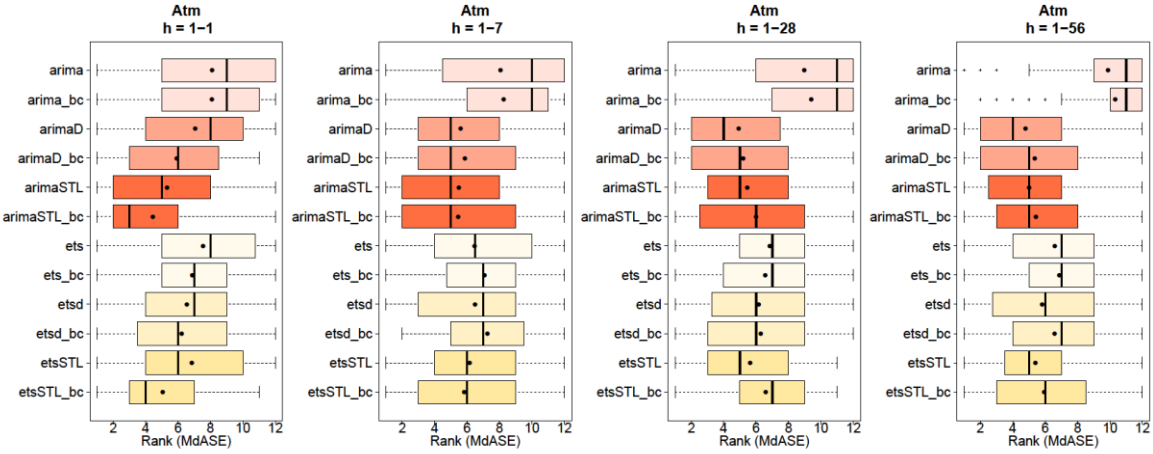

Figure 3.7: *Ranked MdASE by horizon range of some classical forecast models for NN5 competition data.*

Method	Atm h=1-1	Atm h=1-7	Atm h=1-28	Atm h=1-56	Method	Atm h=1-1	Atm h=1-7	Atm h=1-28	Atm h=1-56
arima	8.10	8.06	8.97	9.87	ets	7.54	6.47	6.84	6.59
arima_bc	8.08	8.26	9.40	10.32	ets_bc	6.87	7.07	6.56	6.88
arimaD	7.06	5.60	**4.93**	**4.79**	etsd	6.54	6.49	6.15	5.82
arimaD_bc	5.91	5.86	5.20	5.36	etsd_bc	6.23	7.25	6.29	6.58
arimaSTL	5.32	5.50	5.45	5.01	etsSTL	6.85	6.16	5.63	5.40
arimaSTL_bc	**4.44**	**5.45**	5.99	5.43	etsSTL_bc	5.05	5.82	6.59	5.94

Table 3.2: *Average Ranked MdASE corresponding to dots in Figure 3.7. Best model metric per horizon range (with vertical split just for visual convenience) is bold & underlined and shading denote Top20% accordingly.*

smoothing. It further provides robustification features and allows also changing seasonality (cf. Cleveland et al. (1990)).

Figure 3.6 shows the decomposition for a time series build from the average of all NN5 data sets. An initial decomposition using a period of 7 (plot combination at the top of Figure 3.6) nicely shows the weekly seasonality and the trend comprising yearly Christmas effect (2 big peaks). Furthermore the second monthly (or *monthday*) seasonality due to increased withdrawals at the end of each month can be guessed due to the tiny regularly occurring peaks in the trend component. Therefore a subsequent STL decomposition can be applied. One might use a period of 30.5 to catch this, allowing a slight imprecision for February and for August, and feels satisfied with the nice shaped seasonal component shown on the lower left side in Figure 3.6. But it is very important to recognize the size of the seasonal effect compared to the original signal. With a range of approximately 0.6 this definitely does not catch the *monthday* effect which can be roughly estimated by 3 regarding the little peaks in the trend plot of the initial decomposition. The lower right plot shows that this seasonality can be better modeled using a period of 61.

Basically a combination of STL decomposition and forecasting model let the STL deseasonalize the data and transfers just the trend plus remainder to the forecasting method. The deseasoning has to be rewind after the forecasting step by adding the most recent seasonal pattern of the training fold which represents a naïve approach for this component.

Figure 3.7 and Table 3.2 compare the performance of an initial deseasoning step by STL decomposition for ARIMA and ETS models (with and without Box-Cox transformation) applied on the NN5 time series with some "standard" variants invented in previous chapters.

It seems that some improvement can be achieved by STL up to half-season horizon range especially when combined with a Box-Cox transformation. Actually for the 1-step-ahead prediction the *arimaSTL_bc* is clearly the best method! But the STL advantage mostly disappears in the long run and the Box-Cox transformation is then contra productive especially for the *etsSTL* model.

On the other hand, for the Tourism data (cf. Figure 3.13 in Chapter 3) the Box-Cox transformation seems to be inevitable to get a satisfying performance when using STL already for a half-season timeframe up to the whole competition horizon range, which might be due to some Tourism time series with increased seasonality! But for the ARIMA model class a hard-coded differencing to account for seasonality represents a better approach for this data as already discussed in Chapter 3.1.1 (and can be seen in Figure 3.13 as well).

Furthermore, for the M3-INDUSTRY data the *arimaSTL* is ending up as the best classical model (apart from ensembling approaches introduced in Chapter 3.4) in the long run (cf. Chapter 3.5).

Finally two different applications for this easy-to-use decomposition should already be mentioned: As the tree-based machine learning algorithms (cf. Chapter 4.5) cannot model a (future) trend, an initial STL decomposition is used to extract the trend first and leave the rest, i.e. seasonal and remainder component, for the algorithm to be forecasted. The trend is then separately fitted with an ETS model and added in the end again to rewind the detrending. Notice that here the decomposition is used for detrending with the trend being forecasted "externally" in contrast to above application that needs the decomposition for deseasoning and predicting the seasonal influence by a naïve forecast.

Furthermore, the STL is inevitable to remove a trend or increased seasonality before creating bootstrap samples with the moving block bootstrap approach as explained in the following chapter.

3.4 Bagging Approaches for Classical Time Series Models

Bagging (Bootstrap aggregation) is a common approach to reduce the variance and therefore MSE (mean squared error) of a prediction model even though the bias in increased due to fewer observations used for estimation. As for time series the target values are correlated, it is not possible to randomly select the bootstrap sample which would destroy the data dependency structure. Actually this fact is only crucial for forecast models that need the "original" order of data points like it is for ARIMA and ETS model. But for the Machine Learning models (apart from Recurrent Neural Nets, cf. Chapter 4.1) explained in Chapter 4 that just need and use lagged target values as standard covariates, bootstrapping can be applied as usual (random sampling with replacement)! Therefore the following explanations are just important for the classical forecasting approaches!

The main idea behind so-called block bootstrap methods is to preserve the dependence structure of the target (random) variable at short distances. E.g., with the moving block bootstrap (MBB) for a time series of length T, $N = T - l + 1$ overlapping blocks of length l are defined $\mathcal{B}_{k=1..N} = (y_k, \dots, y_{k+l-1})$ from which T/l (assuming l divides T for simplicity) blocks are drawn with replacement and appended to form one bootstrap sample. Typical sizes for l are $l \approx T^{1/3}$ (cf. Kreiss & Lahiri (2012)).

In order to prevent from the possible structural breaks resulting from the overlap, a non-overlapping block bootstrap (NBB) can be used which creates $N = T/l$ blocks $\mathcal{B}_{k=1..N} = \left(y_{(k-1)l+1}, \dots, y_{kl}\right)$. But

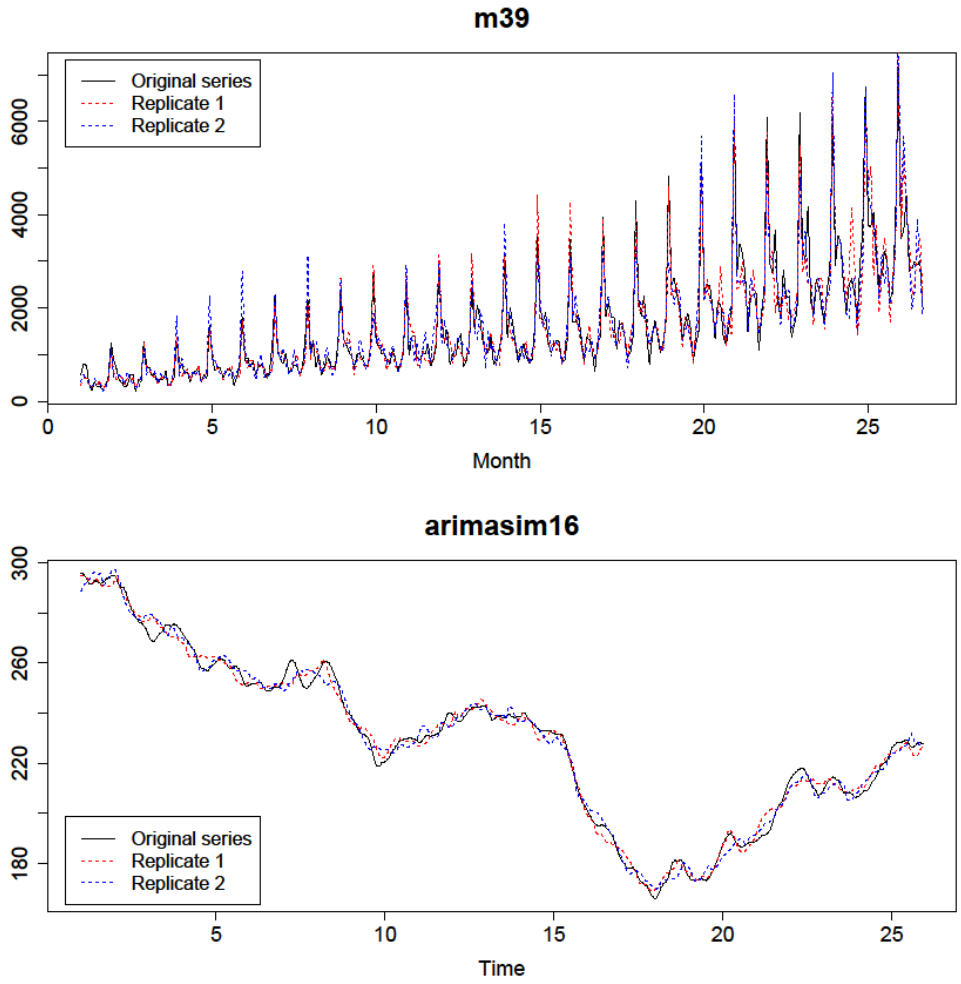

Figure 3.8: *Two MBB samples for one Tourism time series (top row) and one Arimasim times series (bottom row).*

Kreiss & Lahiri (2012) state that "the NBB estimators typically have higher MSEs at any block size l, compared to their MBB counterparts".

For seasonal time series it makes sense to use multiples of the period as block length, in line with the amount of lagged influence. This is done for the benchmarks using MBB with block lengths comprising 2 seasons with a slight adjustment: When putting the blocks together for one of the b bootstrap samples, $N = \lfloor T/l \rfloor + 2$ blocks are appended and a random number, less than the seasonal period, of observations are discarded from the beginning. The resulting time series is further trimmed to the length of the original series. This strategy, taken over from Bergmeir et al. (2014), assures that the bootstrap samples do not necessarily begin or end on a seasonal boundary. Furthermore, due to a possible trend or increased seasonality this whole approach must be applied on the remainder of a STL decomposition after applying a BoxCox-transformation! Clearly both preprocessing steps are rewound after sampling in order to get a bootstrapped version of the original time series. The need for this additional processing exhibits a main disadvantage of this approach compared to the maximum entropy bootstrap explained below! Figure 3.8 shows examples for time series from the Tourism data with increasing seasonality and from the simulated ARIMA(1,1,1) data.

Several additional variations for the block bootstrap exist like the stationary bootstrap (Politis & Roman (1994)) using a random block length drawn from a Geometric distribution with expected value $\mu = l$ remarkably leading to the desirable property of stationarity for the whole boostrapped time series.

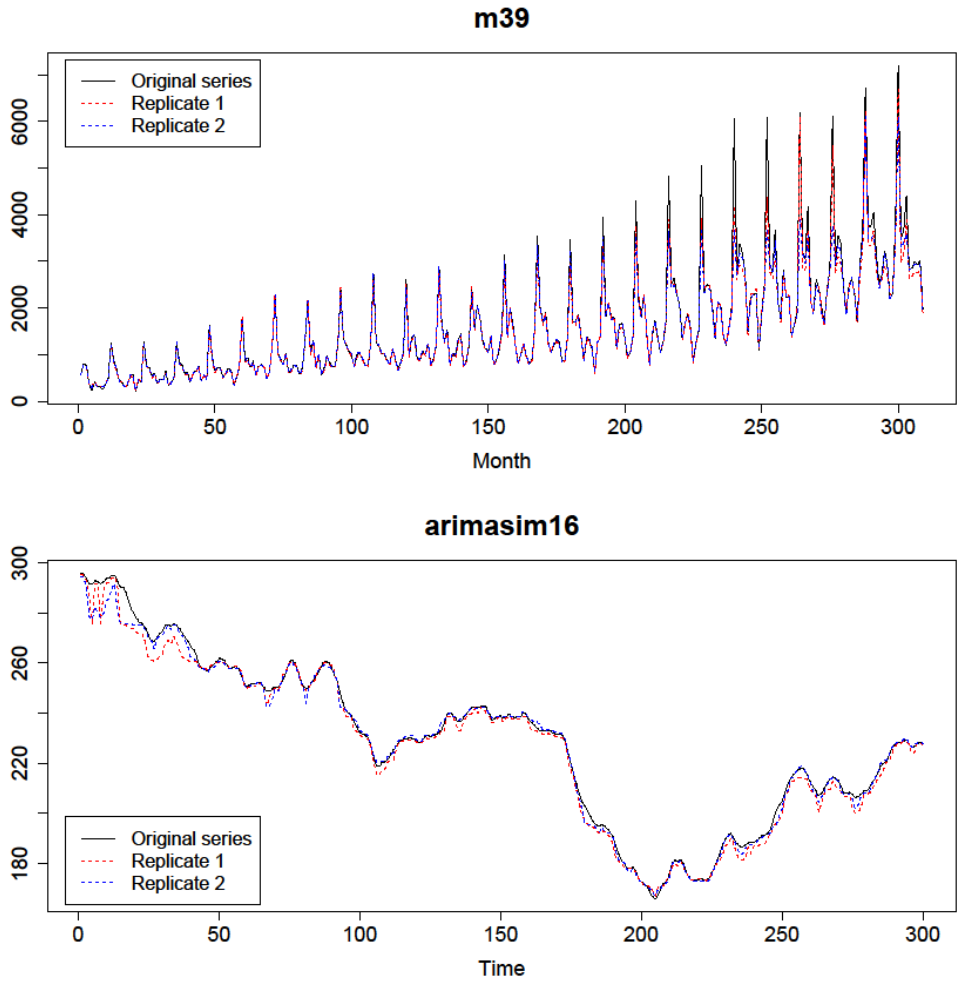

Figure 3.9: *First two of B=30 maximum entropy bootstrap samples for one Tourism time series (top row) and one Arimasim times series (bottom row).*

Though, this approach is disregarded for the benchmarks due to the reasons already mentioned in Chapter 1.2. Further variants, shortly described in Kreiss & Lahiri (2012), are also ignored following Bergmeir et al. (2014) who did not receive any substantial advantages by these other procedures when bagging ETS methods.

A parametric bagging approach is represented by the *sieve* bootstrap which samples from the residuals of an AR model. Actually this strategy presumes that the data generating process is in fact following the assumed autoregressive process which seems too restrictive. In fact, Cordeiro & Neves (2009) are using this approach with just mixed results when applying on the M3 data. Also some tests conducted by Bergmeir et al. (2014) get "results [that] were consistently worse than with the procedure presented here [i.e. the MBB method]".

A totally different approach constitutes the maximum entropy (ME) bootstrap invented by Vinod (2006). The central idea behind this algorithm is in guaranteeing to hold the ergodic property regarding the ensemble which means that the grand mean of the bootstrap ensemble equals the mean over the original series. Even though that ergodicity does not automatically result in stationarity, the inventor Vinod states that the proposed algorithm assures to "retain the basic shape and dependence structure of autocorrelation function (ACF) and partial autocorrelation function (PACF) of the original time

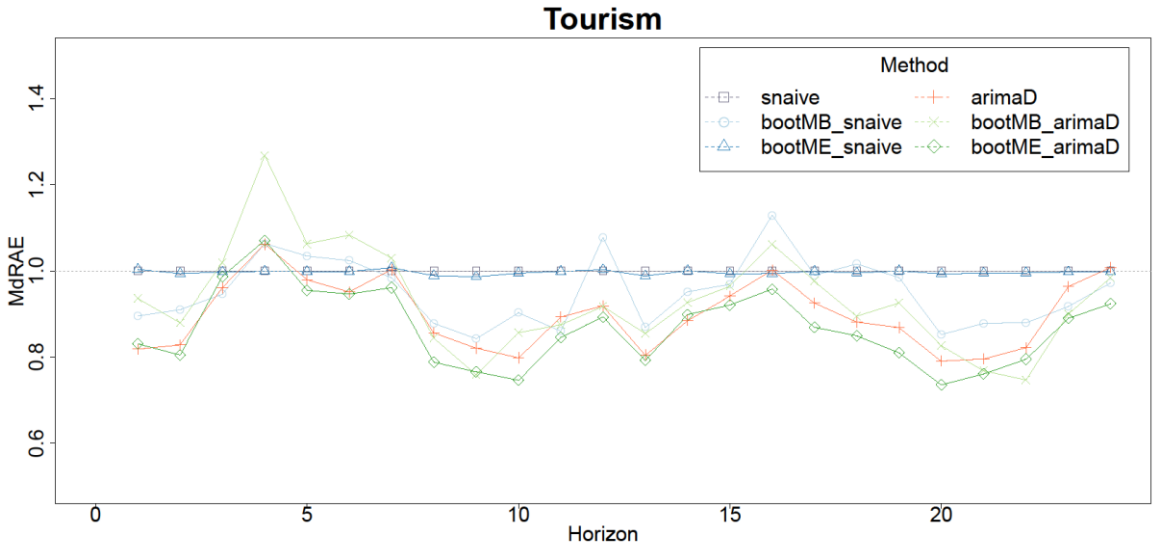

Figure 3.10: *Performance of bagging approaches for snaive and arimaD models applied on Tourism data.*

series". An example is plotted in Figure 3.9 (for the same time series used in Figure 3.8**)** assuming a total bootstrap sample size of 30 which is also used for the benchmarks in this thesis.

The fact that no data transformation like block building, initial differencing, AR-fitting or STL decomposition has to be done in advance, makes this approach very appealing. Nonetheless its usability is highly dependent on its asymptotic overlay properties for distributional parameters (not investigated in this thesis) or the general performance resulting in prediction tasks when using it for bagging (see below).

Figure 3.10 shows a comparison of the used bagging approaches for Tourism benchmark time series. A big difference for MBB based bagging in conjunction with a simple seasonal naïve forecast result due to the structural breaks occurring at every second season and therefore highly influence the seasonal naive forecast. This is obviously not the case for ME bootstrap based bagging. But for the MBB case also the described STL preprocessing has an influence. Nonetheless the overall performance improves for this version. More insight gives the bagging of the *arimaD* model. Here the ME approach shows a superior performance over the MBB and clearly improves the *arimaD* forecast in a horizon constant manner! This is a very special behavior as usually the performance lines intersect for any two methods. Such a slight but constant improvement over horizons gets masked by the typical rank analysis used in the benchmark result assessment of this thesis.

Figure 3.11 shows the same analysis for the NN5 time series. The situation for bagging *snaive* is nearly the same as for the Tourism data. But for *arimaD* both approaches ME and MBB highly overlap (and now typically intersect) regarding forecast performance with the pure *arimaD* model and does not seem to highly improve the performance of *arimaD*.

3.5 Benchmark Results for Classical Methods

In the following the results for the classical methods together with bagging approaches are given. An overall comparison with Machine learning models can be found in Chapter 5. Also the discussion of

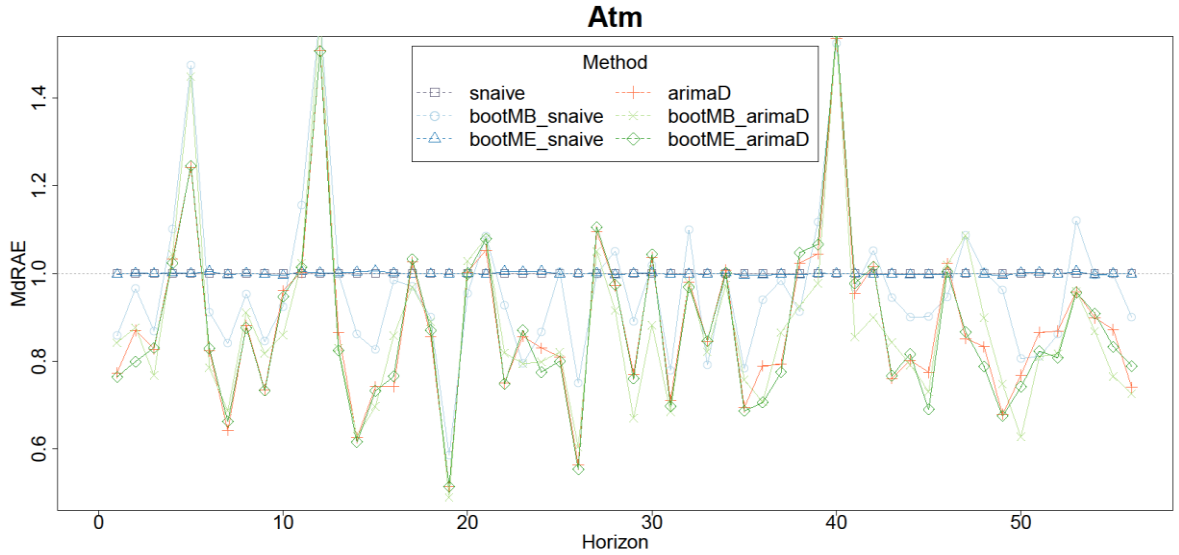

Figure 3.11: *Performance of bagging approaches for snaive and arimaD models applied on NN5 data.*

the performance of an ARIMA model with covariates is postponed to this chapter as well as the application of the models onto the Arimasim data.

Several ARIMA and ETS variants are tested to account for nonlinearity as well as seasonality. Generally the same notation as in previous chapters for distinguishing the alternatives is used. For ARIMA based models a hardcoded seasonal differencing variant, identifiable by a suffix *D,* is tested as alternative to the auto detection of seasonal differencing by the *auto.arima* R-function having no suffix in its name. On the other hand, the STL alternative first strips off the seasonal component and rewinds this at the end as explained in Chapter 3.3. Beside the auto detection of an ETS model denoted *ets,* also a forced damped variant is used (*etsd*) which is motivated in Chapter 3.2. Also for the *ets* model a STL version (*etsSTL*) is used. Last but not least all alternatives are applied on Box-Cox transformed series leading to a *bc* suffix in its names with lambda detected as explained in Chapter 3.1.1.

Additionally some ensembling variants of these classical models are tested. E.g. *ens_mean* denotes a model that averages the forecast over all classical ARIMA and ETS models excluding the very bad performing *thetaf* and the ARIMA based ones without hard-differencing, i.e. *arima* and *arima_bc* (as these exhibit really bad performance for NN5 time series, cf. Chapter 3.5), ending up in averaging 11 models: *arimaD, arimaD_bc, arimaSTL, arimaSTL_bc, ets, ets_bc, etsd, etsd_bc, etsSTl, etsSTL_bc, tbats*. As Kourentzes et al. (2014) suggest using the median for aggregating ensembles in the time series forecasting context, also this approach named *ens_median* is added to the tested portfolio.

Beside the already mentioned (cf. Chapter 3.4) bagged approaches for *snaive* and *arimaD* based on the MB and ME bootstrap alternatives (*bootMB_snaive*, *bootME_snaive*, *bootMB_arimaD*, *bootME_arimaD*), a random selection of one of above mentioned 11 models is conducted for every bootstrap sample in order to decorrelate the results per bag sample (*bootMB_randsel*, *bootME_randsel*). This should additionally reduce the variance for the bagged forecast pretty much following the idea behind random forests explained in Chapter 4.3. For all experiments B=30 bagged versions per time series are drawn following the MBB and ME boostrap sample strategies (cf. Chapter 3.4).

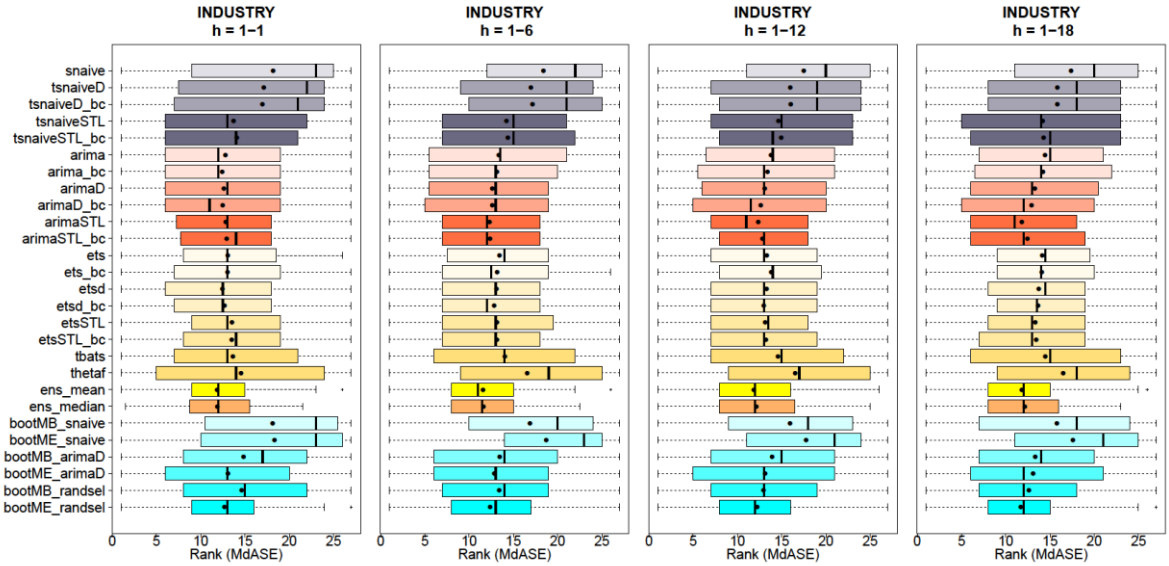

Figure 3.12: *Ranked MdASE by horizon range of classical forecast models together with bagging approaches for M3-INDUSTRY competition series.*

Method	INDUSTRY h=1-1	INDUSTRY h=1-6	INDUSTRY h=1-12	INDUSTRY h=1-18
snaive	18.18	18.41	17.49	17.39
tsnaiveD	17.12	16.98	15.97	15.83
tsnaiveD_bc	16.98	17.17	16.04	15.83
tsnaiveSTL	13.72	14.23	14.63	14.19
tsnaiveSTL_bc	14.09	14.39	14.96	14.29
arima	12.80	13.35	13.80	14.45
arima_bc	12.42	13.15	13.39	14.19
arimaD	12.60	12.62	13.08	13.28
arimaD_bc	12.47	12.63	12.63	12.92
arimaSTL	12.83	12.32	12.35	11.81
arimaSTL_bc	12.94	12.36	12.82	12.45
ets	13.06	13.43	13.31	14.13
ets_bc	13.02	13.18	13.81	14.06

Method	INDUSTRY h=1-1	INDUSTRY h=1-6	INDUSTRY h=1-12	INDUSTRY h=1-18
etsd	12.46	13.09	13.30	13.73
etsd_bc	12.69	12.83	12.98	13.65
etsSTL	13.53	13.15	13.12	13.33
etsSTL_bc	13.50	13.15	13.22	13.43
tbats	13.63	14.05	14.58	14.47
thetaf	14.55	16.58	16.53	16.50
ens_mean	**11.84**	**11.58**	**11.84**	11.82
ens_median	11.88	11.63	12.13	12.15
bootMB_snaive	18.13	16.89	15.96	15.77
bootME_snaive	18.33	18.73	17.76	17.59
bootMB_arimaD	14.83	13.45	13.94	13.33
bootME_arimaD	13.09	12.85	13.13	13.11
bootMB_randsel	14.64	13.41	12.96	12.62
bootME_randsel	12.69	12.39	12.26	**11.71**

Table 3.3: *Average Ranked MdASE corresponding to dots in Figure 3.12. Best model metric per horizon range (with vertical split just for visual convenience) is bold & underlined and shading denote Top20% accordingly.*

As always some naïve benchmark results (cf. Chapter 2.3.1) are added in order to help assessing the general benefit of the advanced models.

Figure 3.12, Figure 3.13 and Figure 3.14 together with Table 3.3, Table 3.4 and Table 3.5 show the results for the classical approaches together with the bagged models.

Main findings, apart from the ones already discussed in previous chapters, are:

- ARIMA based models generally outperform ETS approaches. This can be seen for all 3 benchmark series.
- If one knows about existing seasonality and does not decide to use STL for deseasoning, hard-differencing is usually better than letting *auto.arima* decide for seasonal differencing. This result is also confirmed by all benchmarks most striking for the NN5 data (see Figure 3.14).

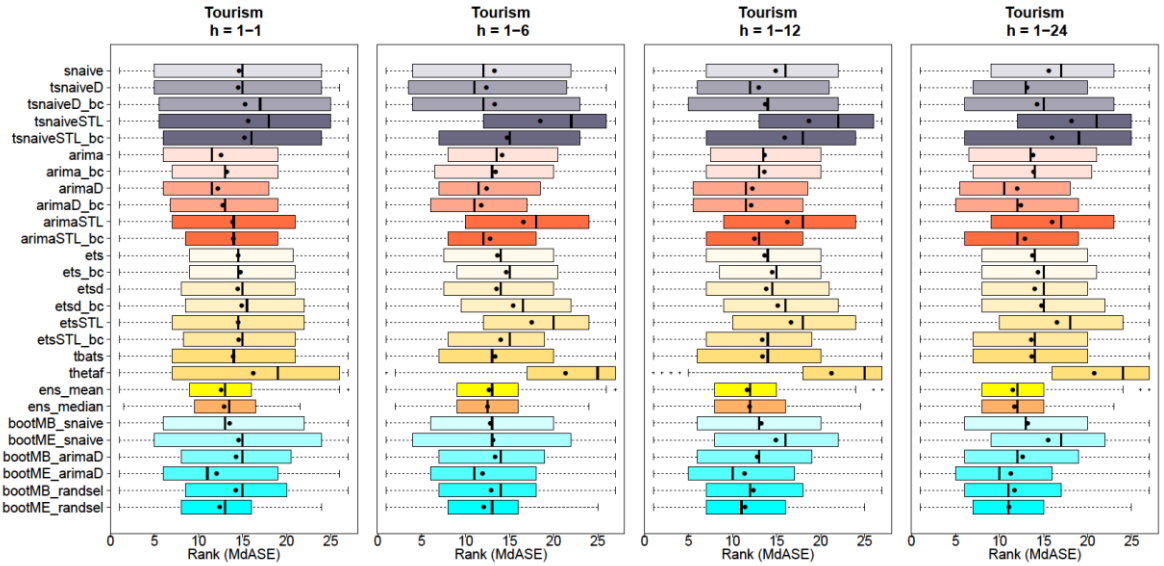

Figure 3.13: *Ranked MdASE by horizon range of classical forecast models together with bagging approaches for Tourism competition series.*

Method	Tourism h=1-1	Tourism h=1-6	Tourism h=1-12	Tourism h=1-24	Method	Tourism h=1-1	Tourism h=1-6	Tourism h=1-12	Tourism h=1-24
snaive	14.60	13.27	14.90	15.61	etsd	14.45	13.53	13.83	14.00
tsnaiveD	14.49	12.35	12.99	13.13	etsd_bc	14.89	15.41	15.13	14.75
tsnaiveD_bc	15.30	13.31	13.71	14.24	etsSTL	14.50	17.52	16.62	16.51
tsnaiveSTL	15.62	18.47	18.69	18.16	etsSTL_bc	14.56	13.98	13.37	13.61
tsnaiveSTL_bc	15.20	14.74	15.93	15.97	tbats	13.93	13.36	13.40	13.66
arima	12.57	14.16	13.64	13.81	thetaf	16.22	21.36	21.23	20.75
arima_bc	13.20	13.42	13.62	13.85	ens_mean	12.55	12.66	11.63	11.50
arimaD	12.18	12.40	12.25	12.00	ens_median	12.89	12.49	11.92	11.69
arimaD_bc	12.73	**11.77**	12.12	12.43	bootMB_snaive	13.53	12.79	13.27	13.20
arimaSTL	13.88	16.58	16.21	15.97	bootME_snaive	14.54	13.12	14.93	15.51
arimaSTL_bc	13.95	12.79	12.48	12.87	bootMB_arimaD	14.26	13.35	12.81	12.61
ets	14.51	13.63	13.64	13.72	bootME_arimaD	**12.06**	11.95	**11.37**	11.30
ets_bc	14.75	14.62	14.51	14.35	bootMB_randsel	14.25	12.92	12.38	11.70
					bootME_randsel	12.38	12.07	11.42	**11.08**

Table 3.4: *Average Ranked MdASE corresponding to dots in Figure 3.13. Best model metric per horizon range (with vertical split just for visual convenience) is bold & underlined and shading denote Top20% accordingly.*

- If using STL for deseasoning one might test a Box-Cox variant to account for multiplicative seasonality. This situation mostly happens for the Tourism data and shows advantages there (see Figure 3.13). On the other hand, for NN5 time series only the 1-step-ahead forecast profits from this approach but is contra productive in the long run (see Figure 3.14).
- When averaging over several models, using the median as aggregating metric does not have an advantage over standard mean. Generally the ensemble shows good forecast capability but is not always better than a single model (cf. Table 3.4 and Table 3.5). A tuned weighting of the ensemble parts should help here (which is not tested). But the variability clearly reduces which can be seen by the narrower box plots compared to the single models for all three benchmarks. Furthermore the forecasts are also more constant over different horizon ranges, i.e. the ranking does not change much by different horizon ranges. Generally one should give a simple ensembling always a try when having several good performing models.

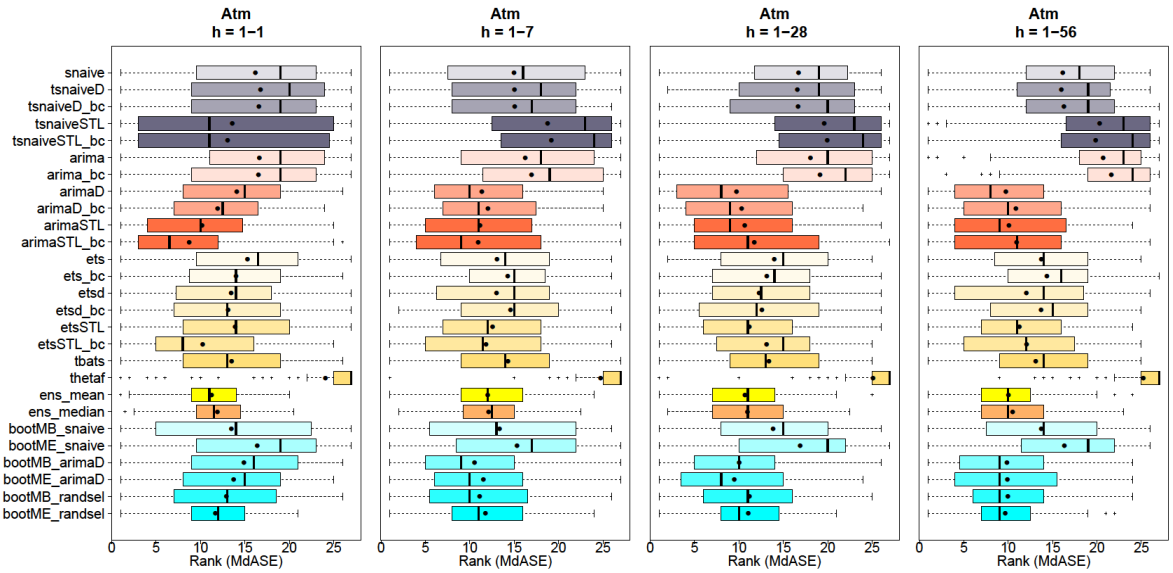

Figure 3.14: *Ranked MdASE by horizon range of classical forecast models together with bagging approaches for NN5 competition series.*

Method	Atm h=1-1	Atm h=1-7	Atm h=1-28	Atm h=1-56	Method	Atm h=1-1	Atm h=1-7	Atm h=1-28	Atm h=1-56
snaive	16.18	15.00	16.72	16.14	etsd	13.44	13.02	12.28	12.05
tsnaiveD	16.76	15.04	16.58	16.00	etsd_bc	13.11	14.60	12.59	13.72
tsnaiveD_bc	16.59	15.07	16.65	16.28	etsSTL	13.91	12.57	11.17	11.25
tsnaiveSTL	13.59	18.78	19.61	20.30	etsSTL_bc	10.23	11.82	13.10	12.08
tsnaiveSTL_bc	13.05	19.19	19.94	19.86	tbats	13.51	14.31	13.38	13.09
arima	16.62	16.25	18.06	20.69	thetaf	24.07	24.75	25.10	25.22
arima_bc	16.52	16.96	19.14	21.61	ens_mean	11.25	12.02	10.66	10.01
arimaD	14.07	11.36	9.72	9.74	ens_median	11.91	12.13	10.96	10.50
arimaD_bc	11.92	12.03	10.31	10.86	bootMB_snaive	13.47	13.36	13.86	13.72
arimaSTL	10.17	11.15	10.65	10.07	bootME_snaive	16.39	15.33	16.90	16.34
arimaSTL_bc	**8.71**	10.93	11.72	10.95	bootMB_arimaD	14.90	**10.54**	10.05	9.86
ets	15.29	13.07	13.98	13.73	bootME_arimaD	13.75	11.54	**9.48**	9.93
ets_bc	13.99	14.28	13.14	14.35	bootMB_randsel	12.92	11.12	11.20	9.96
					bootME_randsel	11.68	11.77	11.06	**9.68**

Table 3.5: *Average Ranked MdASE corresponding to dots in Figure 3.14. Best model metric per horizon range (with vertical split just for visual convenience) is bold & underlined and shading denote Top20% accordingly.*

- Ensembling by bagging mostly results in an improvement for Tourism and NN5. But processing time obviously increases. The ME bootstrap exhibit some advantages over the well-known MBB approach for the non-naïve models and is at least much easier to use series (see also the comments at the end of the previous chapter).
- Further ensembling the bagging approach by randomly selecting one out of a group of nice performing model for every bootstrap sample further reduces the variance of the bagged forecasts by decorrelating the results. Actually for all 3 benchmark series the *bootME_randsel* is the winner for the competition horizon range objective.
- Table 3.3 shows that the runner-up of *bootME_randsel* for the M3-INDUSTRY competition horizon range is represented by *arimaSTL* together with *ens_mean*. The latter is also the best method for 1-step-ahead forecast, half season and full season forecast for this data. In the NN5 competition *arimaD* is close to the winner for the long horizon range but interestingly really bad for the 1-step-

ahead prediction, see Table 3.5. Here the *arimaSTL_bc* reveals a remarkable performance with an average rank of 8.71.

- Even though already explained in previous chapters it should again be stated that the *thetaf* implementation of the promising (regarding official M3 competition, Makridakis & Hibon (2000)) Theta method is clearly the worst classical model for Tourism, NN5 and M3-INDUSTY benchmarks.

Due to several reasons already mentioned a comparison with the original benchmark results for the Tourism and the M3-INDUSTRY series is only possible on a qualitative level.

Even though applied on the whole M3 monthly series, a similar result as the one presented above, revealed the already mentioned study from Bergmeir et al. (2014) showing an improved performance with respect to the sMAPE metric of a bagged ETS model on Box-Cox transformed data utilizing MB-bootstrapping. The best performing models in the original competition for all M3-INDUSTRY time series were some commercial packages using combinations of Arima and ETS models (see Makridakis & Hibon (2000).

Athanasopoulos & Hyndman (2011) initially compared several classical models on the Tourism series before starting the official competition. ForecastPro, a commercial package combining Arima and ETS models (also used in the M3 competition) showed best performance for all horizon ranges regarding MdASE metric. The winner of the final competiton (comprising additional quarterly series) was using an ensemble of Arima, ETS and naïve methods (Brierly (2011b)).

As can be seen on Crone (2009b) no classical model was under the top performing ones for the NN5 series (apart from Wildi (2008) but which is framed by the frequency domain approach and therefore not related to the classical Arima and ETS models). When using the winner (*arimaSTL*) of this thesis benchmark regarding sMAPE, the competition evaluation metric, a value of 20.7% results which is better than the best statistical method submitted in the original benchmark! In Chapter 5.3 it will be shown that this can still be beaten by a Machine Learning approach.

4 Machine Learning Approaches

This chapter introduces several Machine Learning prediction algorithms. The selection of models is motived in Chapter 1.1. Simultaneously first results of applying these models onto the benchmark data are given, including a comparison with the best naïve and classical model to get an impression of the forecast performance. A more comprehensive result overview is postponed to Chapter 5. In fact, several variants of each Machine Learning model are tested which partly gets motivated by a basic simulation study presented in Chapter 4.5 that has the intention to better understand the forecast capability of several model variants (or covariate sets) for some phenotypical time series.

Basically each model is provided with an initial covariate set of lagged target variables comprising the last 2 seasons to account for correlations and a time covariate in order to enable the algorithm to model a trend. In order to handle the seasonality in the time series, seasonal dummies are created. In Chapter 3.1.1 the seasonal differencing is presented as a standard approach for deseasoning in the ARIMA context but can be applied in advance for any model and is therefore added as a possible variant. A further alternative is represented by an initial STL decomposition to strip of the seasonal influence and rewind it again at the end. These three mutually exclusive approaches are denoted by the suffix “SD” (seasonal dummies), “D” (seasonal differencing) and “STL”. Actually the simulation study in Chapter 4.5 will suggest letting the Machine Learning algorithm model the seasonality just with the lagged target information which is therefore the 4^{th} basic approach. Furthermore it might be advantageous to reduce the covariate set of lagged target values, i.e. the *lagset*, for some models. The best approach in terms of model selection would be to tune the composition of the lagset. As this increases the processing time drastically, it was decided to just apply each variant also with a fixed reduced lagset of just the most recent lag and the first seasonal lag. This alternative is marked with a “RL” in the name. Together with a Box-Cox transformation option (denoted by the suffix “bc”), which is important in order to keep the models comparable to the classical approaches comprising also this option, a total of 16 variants for each Machine Learning are tested. For example the algorithm *nnetSTL_bc_RL* denotes a Neural Net model with STL decomposition chosen for handling seasonality applied on Box-Cox transformed time series and using a time variable (default for all models) and a reduced lagset as covariates. Alternatively *nnet_bc* stands for a Neural Net approach with the full lagset which is also responsible for modeling the seasonality of the Box-Cox transformed data.

As explained in the following, every algorithm additionally has one or more tuning parameter for regularization. The exact parameter values used are listed in the corresponding chapters. Notice that these listed values are the fine grid extraction of an initial rough grid search comprising also several above mentioned modeling variants applied on the different benchmark datasets!

4.1 Neural Nets

Artificial Neural Nets (ANN, also named *nnet* in this thesis) are by far the most used algorithm for forecasting by the Machine Learning community (Krollner et al. (2010)). Indeed they represent the only approach from this field applied already in the M3 competition conducted in 2000 (Makridakis & Hibon (2000)). Though, Gooijer & Hyndman (2006) summarize that “ANNs had been oversold as a miracle forecasting technique” but indeed suffer from the “curse of model complexity and model over-parametrization”.

Basically ANNs try to mimic the function of the human brain. E.g. for the vanilla ANN in Figure 4.1 comprising just one hidden layer, the units in each layer represent a neuron and the connections

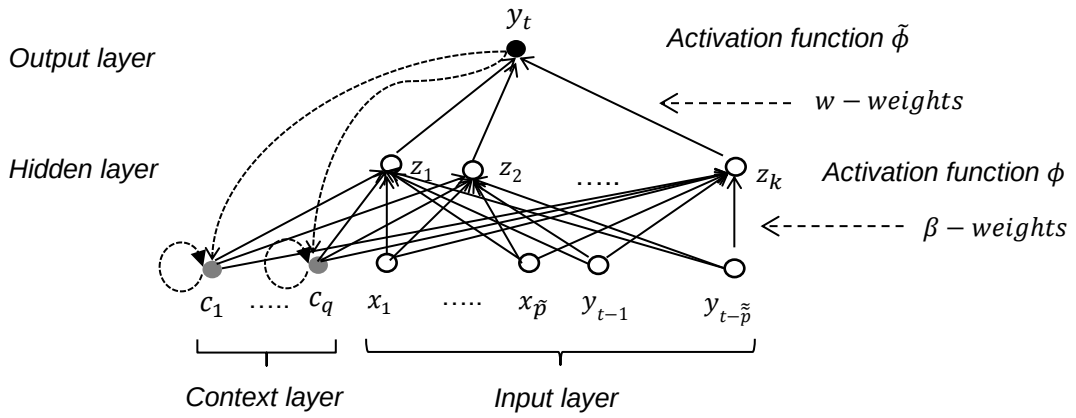

Figure 4.1: *Idea of Vanilla (one hidden layer) Neural Net for Time Series forecasting with autoregressive components in the input layer and a context layer for "error feedback" (Jordan network).*

between them stand for the synapses. From a statistical viewpoint they represent just a nested combination of several logistic regressions in case the sigmoid activation function $f(a) = 1/(1 + e^{-a})$ is used for the functions ϕ and $\tilde{\phi}$ in the model equation

$$y = \tilde{\phi}\left(w_0 + \sum_{l=1}^{k} w_l \, \phi\left(\beta_{l0} + \sum_{j=1}^{p} \beta_{lj} x_j\right)\right) \tag{4.1}$$

with the "inner equation" $z_l = \phi\left(\beta_{l0} + \sum_{j=1}^{p} \beta_{lj} x_j\right)$ modeling the influence of the inputs on the units in the hidden layer. Notice that equation (4.1) does not comprise the context layer effects (described below) of Figure 4.1. Also when using the identity for $\tilde{\phi}$ in the regression case, a very flexible nonlinear model results already with just one hidden layer. In fact such a model, given enough units in the hidden layer, is in theory (!) a universal approximator, which means that it can model any smooth function $y(\boldsymbol{x})$ (cf. Murphy (2012)). This ability, together with the idea of imitating the human brain, are main reasons for the high credits ANNs received in the community. On the other hand and as usual for very flexible (and nonlinear) models, the model interpretation or quantification of variable effects is at least extremely difficult, moving ANNs into the "black box" model edge.

The model parameters (or weights) $\boldsymbol{\theta} = (\boldsymbol{\beta}, \boldsymbol{w})$ are determined by a gradient descent algorithm called *backpropagation,* applied to minimize quadratic loss in the regression case. Importantly, this minimization is not a convex problem; it can happen that it ends in just a local minimum. Therefore several random starting weights should be used accompanied by averaging the results. This approach is also utilized in the benchmarks of this thesis using the *avNNet* function of the *caret* R-package (Kuhn (2014)).

The flexible modeling is prone to overfitting and needs regularization. One approach is called *early stopping* and stands for ending the backpropagation, with respect to performance on a test set, before reaching its minimum. A more reasonable alternative that directly solves the problem of determining the number of hidden units is represented by a L2-penalization of the weights, called *weight decay*, in conjunction with choosing a quite high number of hidden units. This is the approach of choice in the benchmarks using 20 units in 1 hidden layer.

Figure 4.1 shows how an autoregressive component is incorporated just by adding lagged target values to the input layer. Past forecast errors or a moving average component (cf. Chapter 3.1.1) can

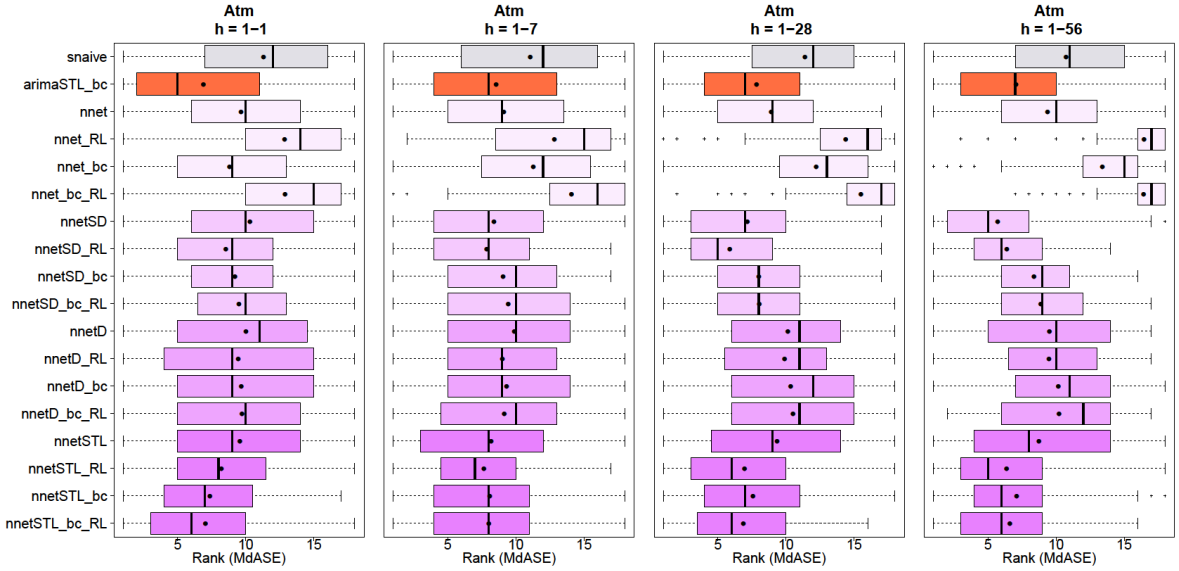

Figure 4.2: *Ranked MdASE by horizon range of Neural Net based models for NN5 competition. Results of best naïve (snaive) and one of the best classical models (arimaSTL_bc) are added.*

Method	Atm h=1-1	Atm h=1-7	Atm h=1-28	Atm h=1-56
snaive	11.30	11.05	11.40	10.74
arimaSTL_bc	**6.88**	8.55	7.83	7.05
nnet	9.66	9.11	8.91	9.38
nnet_RL	12.86	12.83	14.38	16.45
nnet_bc	8.81	11.29	12.22	13.40
nnet_bc_RL	12.88	14.08	15.50	16.42
nnetSD	10.31	8.39	7.17	**5.71**
nnetSD_RL	8.53	7.86	**5.88**	6.39
nnetSD_bc	9.22	9.07	8.01	8.38
nnetSD_bc_RL	9.50	9.44	8.04	8.88
nnetD	10.02	9.89	10.14	9.51
nnetD_RL	9.45	9.02	9.90	9.48
nnetD_bc	9.68	9.32	10.35	10.17
nnetD_bc_RL	9.73	9.15	10.52	10.23
nnetSTL	9.57	8.18	9.35	8.74
nnetSTL_RL	8.21	**7.66**	6.95	6.36
nnetSTL_bc	7.37	8.09	7.58	7.11
nnetSTL_bc_RL	7.05	8.02	6.87	6.60

Table 4.1: *Average Ranked MdASE corresponding to dots in Figure 4.2. Best model metric per horizon range is bold & underlined and shading denote Top20% accordingly.*

be modeled by a *context* layer resulting in a so-called *recurrent* ANN. More precise, the variant presented in Figure 4.1 is called a *Jordan* network. Unfortunately the R support for recurrent ANNs is not very comfortable. Some tests for the NN5 data utilizing the *jordan*-function of the *RSNNS* R-package (Bergmeir & Benitez (2014)) were not very promising (cf. Chapter 5). Therefore it was decided to use the standard Neural Network (without a context layer), together with lagged target values being part of the input layer.

Despite the universal approximation property of the vanilla network, there are several applications where a more complicated network structure might be advantageous. In image detection the *convolutional* ANNs are popular. These networks exploit the 2D structure of the images by their layer architecture (see e.g. Hastie et al. (2009)). Furthermore *deep* ANNs are of current research interest and have shown remarkable performances in speech recognition for instance (see Murphy (2012)).

Benchmark Insights

As will be shown in Chapter 5, only for the NN5 benchmark time series a neural net approach gets better results than the best naïve forecast. Therefore Figure 4.2 and Table 4.1 show the performance comparison for these series. The initial tuning parameter tests revealed that averaging over 3 neural nets (each using different starting values) with 1 hidden layer comprising 20 units and a weight decay varying between 1, 10, 20 and 30 can generally be applied to all benchmark data. Some noticeable results which can be identified in Figure 4.2 and Table 4.1 are the following:

- An initial Box-Cox transformation only results in some improvement if a STL decomposition is used to tackle seasonality; for other deseasoning approaches it is contra productive. It should already be mentioned that the latter is true for all deseasoning approaches for the M3-INDUSTRY benchmark and can be best seen in the plots presented in Chapter 5.1.
- Using solely a reduced lagset (*RL*) for modeling the seasonality ends up with drastically bad forecasts. This finding (for the NN5 data) will be confirmed in the following chapters by all other Machine Learning approaches and indicates that the other lags hold at least seasonal information for this data. But accounting for seasonality by a STL decomposition profits from the lag reduction in the NN5 case.
- Generally the seasonality is best handled by a STL decomposition combined with a Box-Cox transformation for the 1-step-ahead prediction; but in the long run the *SD* variant is the winner. This does not hold for Tourism and M3-INDUSTRY as can be seen in Chapter 5. For these benchmarks hard seasonal differencing in conjunction with a reduced lagset (*D_RL* variant) is best. In fact it will pop up in the following that for the Tourism data hard differencing is often the best deseasoning variant also for other ML methods!

4.2 Kernel-Machines: SVMs and Gaussian Processes

SVMs

Historically kernel machines are originated from Support Vector Machine (SVM) models for classification. The central idea behind these SVM models is to maximize the predictor space, the so-called *margin*, between the data points of different target labels but allow also separation violating *slack* points (see left graph in Figure 4.3). It can be shown (cf. Hastie et al. (2009)) that the separating hyperplane can be expressed as $f(\boldsymbol{x}) = \boldsymbol{\beta}'\boldsymbol{x} + \beta_0 = \sum_{i=1}^{N\_Support} \alpha_i \tilde{y}_i \boldsymbol{x}_i'\boldsymbol{x} + \beta_0$ (with $\tilde{y}_i \in \{-1,1\}$), i.e. a sum just over the support vectors that are lying exactly on the margin edges (or also on the false side of the hyperplane if slack points are allowed) indicating that data points outside the margin (and on the right side) do not contribute to the solution.

The same result is obtained when solving $\min_{\boldsymbol{\beta},\beta_0} \sum_{i=1}^{N} [1 - y_i f(\boldsymbol{x_i})]_+ + \frac{\lambda}{2}\|\boldsymbol{\beta}\|^2$ (with $[\quad]_+$ returning only positive values and 0 otherwise), which is equal to a regression problem with L2 penalization utilizing the so-called *Hinge-loss* $[1 - y_i f(\boldsymbol{x_i})]_+$ instead of the familiar squared error loss.

The regression analogon of a SVM is represented by using the ε-insensitive loss

$$L\big(y, f(\boldsymbol{x})\big) = \begin{cases} 0 & if\ |y - f(\boldsymbol{x})| < \varepsilon \\ |y - f(\boldsymbol{x})| - \varepsilon & otherwise \end{cases} \tag{4.2}$$

resulting in

$$f(\boldsymbol{x}) = \boldsymbol{\beta}'\boldsymbol{x} + \beta_0 = \sum_{i=1}^{N\_Support} (\alpha_i^* - \alpha_i)\boldsymbol{x}_i'\boldsymbol{x} + \beta_0 \tag{4.3}$$

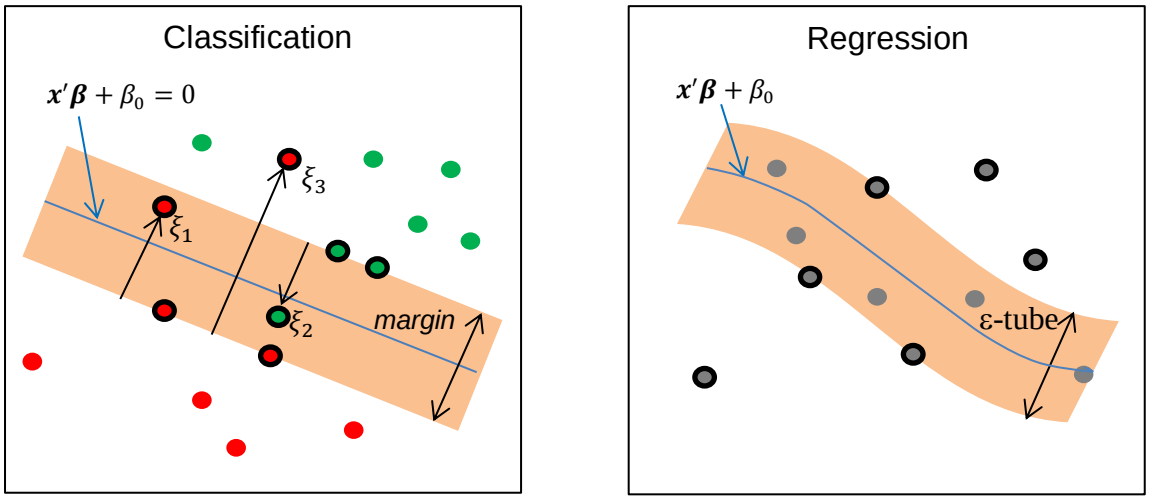

Figure 4.3: *Idea of SVM showing rough analogy between classification (left plot, including slack points) and regression (right plot) problem. Support vectors are marked with black border. In the left plot the separating hyperplane lies in the middle of the margin; on the right plot the regression curve is surrounded by the ε-tube.*

denoting now the regression curve in the hyperspace (with parameters α_i^*, α_i depending on the target values y_i, see e.g. Schölkopf & Smola (2002)). Here the support vectors are lying on the edge and outside of the ε-tube (see right plot in Figure 4.3). Therefore the solution (4.3) now ignores data points inside(!) the tube, closing the rough analogy to the classification case.

Crucial for using a kernel machine is the scalar product $\boldsymbol{x_i}'\boldsymbol{x}$ in (4.3). If one is replacing it by some basis functions $\boldsymbol{x_i} \rightarrow \boldsymbol{h}(\boldsymbol{x_i}) = \left(h_1(\boldsymbol{x}_i), \ldots, h_M(\boldsymbol{x}_i)\right)$ in order to model also nonlinear dependencies (just like it is done in semi-parametric spline regression case), the solution $f(\boldsymbol{x}) = \sum_{i=1}^{N\_Support}(\alpha_i^* - \alpha_i)h(\boldsymbol{x}_i')h(\boldsymbol{x}) + \beta_0 = \sum_{i=1}^{N\_Support}(\alpha_i^* - \alpha_i)K(\boldsymbol{x}_i', \boldsymbol{x}) + \beta_0$ now only depends on the so-called kernel $K(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = \boldsymbol{h}(\boldsymbol{x})'\boldsymbol{h}(\tilde{\boldsymbol{x}})$ which can be calculated very cheaply for a special choice of $\boldsymbol{h}$. Popular representatives are the radial Gaussian kernel $K(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = exp(-\gamma\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|^2)$ or the polynomial kernel of grade d: $K(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = (1 + \boldsymbol{x}'\tilde{\boldsymbol{x}})^{\boldsymbol{d}}$. For the latter it can be shown that it constitutes of 6 basis functions $\boldsymbol{h}(\boldsymbol{x}) = \left(1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2\right)$ in the 2-dimensional case ($\boldsymbol{x} = (x_1, x_2)$) and assuming a grade of 2. However, for the Gaussian kernel it is impossible to specify all basis functions as these are spanning an infinite dimensional space even though ending up in the finite dimensional solution (4.3) just depending on the evaluations of the Kernel at the support vectors. By this so-called *kernelization* a solution in an infinite dimensional space is found in which any smooth function of finite dimensional data can be "easily" linearly fitted. This circumstance might explain the initial euphoria regarding SMVs, up to the assumption that they can beat the curse of dimensionality or can even handle an arbitrary number of irrelevant dimensions. Why the latter is not true, is nicely argued in Hastie et al. (2009), chapter 12.3.4.

Kernel Machines

In addition, the kernelization is, contrary to early assumptions, not an explicit property of the SVM, but can be applied in all cases for which the solution of an optimization problem comprises a scalar

product of the input data which is known as *kernel trick*. E.g. also the parameter vector of a standard ridge regression $\boldsymbol{\beta} = (\boldsymbol{X}'\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}'\boldsymbol{y}$ can be written as $\boldsymbol{\beta} = \boldsymbol{X}'(\boldsymbol{X}\boldsymbol{X}' + \lambda\boldsymbol{I})^{-1}\boldsymbol{y} = \boldsymbol{X}'\boldsymbol{a} = \sum_{i=1}^{N} a_i \boldsymbol{x}_i$ (with $\boldsymbol{a} := (\boldsymbol{X}\boldsymbol{X}' + \lambda\boldsymbol{I})^{-1}\boldsymbol{y}$) when utilizing the Sherman-Morrison-Woodbury fourmula (for $\boldsymbol{E}$ and $\boldsymbol{H}$ invertible holds: $(\boldsymbol{E} - \boldsymbol{F}\boldsymbol{H}^{-1}\boldsymbol{G})^{-1}\boldsymbol{F}\boldsymbol{H}^{-1} = \boldsymbol{E}^{-1}\boldsymbol{F}(\boldsymbol{H} - \boldsymbol{G}\boldsymbol{E}^{-1}\boldsymbol{F})^{-1}$, see e.g. Murphy (2012)). Therefore the solution $f(\boldsymbol{x}) = \boldsymbol{\beta}'\boldsymbol{x} = \sum_{i=1}^{N} a_i \boldsymbol{x}_i'\boldsymbol{x}$ depends on a scalar product!

Actually it can be shown that this kernelization is applicable for a much broader class of problems. It is possible to express the infinite dimensional solution $f(\boldsymbol{x}) = \sum_{k=1}^{\infty} \beta_k h_k(\boldsymbol{x}_i)$ of the general optimization problem

$$\operatorname{argmin}_{\{\beta_k\}_{k=1}^{\infty}} \{ \sum_{i=1}^{N} L(y_i, \sum_{k=1}^{\infty} \beta_k h_k(\boldsymbol{x}_i)) + \lambda \sum_{k=1}^{\infty} \beta_k^2 \} \tag{4.4}$$

with an arbitrary convex loss function and L2-penalization, as a finite-dimensional kernel $f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x})$ (cf. Hastie et al. (2009)). This remarkable fact is a result of the *Representer Theorem* of the *Reproducing Kernel Hilbert Spaces* (RKHS), the mathematical theory behind the kernels. Very important in this context is the L2-penalization ("Ridge penalty") indicating that e.g. a L1-penalization ("Lasso penalty") cannot be kernelized! Applying the kernel solution to (4.4) results in $\operatorname{argmin}_{\{\alpha_i\}_{i=1}^{N}} \{ \sum_{j=1}^{N} L(y_j, \sum_{i=1}^{N} \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}_j)) + \lambda \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}_j)\alpha_j \}$ (notice the $i \to j$ shift) which can be compactly written together with (4.4) as two equivalent representations of the optimization problem:

$$\min_{\boldsymbol{\alpha}} \{ L(\boldsymbol{y}, \boldsymbol{K}\boldsymbol{\alpha}) + \lambda \boldsymbol{\alpha}'\boldsymbol{K}\boldsymbol{\alpha} \} \quad \triangleq \quad \min_{\boldsymbol{\beta}} \{ L(\boldsymbol{y}, \boldsymbol{H}\boldsymbol{\beta}) + \lambda \boldsymbol{\beta}'\boldsymbol{\beta} \} \tag{4.5}$$

The compact penalty shape $\boldsymbol{\alpha}'\boldsymbol{K}\boldsymbol{\alpha}$ is a consequence of the *Reproducing* property of the kernel in the framework of the RKHS. The finite-dimensional matrix $\boldsymbol{K} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is called *Gram matrix* and has to be positive definite to be the empirical version of a so-called *Mercer kernel* which allows the decomposition into basis functions. The reduction from an infinite-dimensional to a finite-dimensional problem is closely related to a principal component analysis applied on the $h_{k=1..\infty}$ features (see Hastie et al. (20009), chapter 18.5.2). Alternatively, the impressive kernel trick can be seen in the context of an initially $p \gg N$ problem (with $p \approx \infty$), which can always be casted to a N-dimensional problem (cf. again Hastie et al. (20009), chapter 18.3.5).

Gaussian Processes

As mentioned above, the positive definiteness of the Gram matrix is crucial for the decomposition into basis functions and therefore the whole kernel trick. This circumstance, together with the principle that the Gram matrix represents a similarity measure (like the scalar product it originates from), builds the ground for an extension of the kernel approach to other structured features and even really abstract features. In principle, one can hand over an arbitrary positive definite matrix as Gram matrix and predict the target. In order to end up with a meaningful, i.e. well predicting model, it is important that the structure of the data points in the sense of an implicit similarity is recognized. An extreme example can be found in text categorization which is using string kernels that just encode the number of coinciding letter series between each two documents (see e.g. Murphy (2012)).

Basically a similar approach is used for Gaussian Processes. It is assumed that the target is drawn from a multivariate Gaussian distribution

$$\boldsymbol{y} = (y_1, .., y_T, y_{T+1}) = (\boldsymbol{y_T}, y_{T+1}) \sim N\left(\begin{pmatrix}\boldsymbol{\mu_T}\\ \mu_{T+1}\end{pmatrix}, \boldsymbol{\Sigma} = \begin{bmatrix}\boldsymbol{\Sigma_T} & \boldsymbol{\Sigma_{T,T+1}}\\ \boldsymbol{\Sigma_{T+1,T}} & \sigma^2_{T+1}\end{bmatrix}\right) \tag{4.6}$$

with a covariance matrix $\boldsymbol{\Sigma}$ that functions as the positive definite Gram-Matrix encoding the implicit similarity of data points. Best predictor for y_{T+1} is then the expected value conditional on $\boldsymbol{y_T}$: $E(y_{T+1}|\boldsymbol{y_T})$. Also the whole predictive distribution for y_{T+1} can be calculated and is a result of the properties of the Multivariate Gaussian distribution (see e.g. Fahrmeir et al. (2013):

$$y_{T+1}|\boldsymbol{y_T} \sim N\big(\mu_{T+1} + \boldsymbol{\Sigma_{T+1,T}} \cdot \boldsymbol{\Sigma_T^{-1}}(\boldsymbol{y_T} - \boldsymbol{\mu_T}),\ \sigma^2_{T+1} - \boldsymbol{\Sigma_{T+1,T}} \cdot \boldsymbol{\Sigma_T^{-1}} \cdot \boldsymbol{\Sigma_{T,T+1}}\big) \tag{4.7}$$

Remarkably, in classical Geo-Statistics the value of $E(y_{T+1}|\boldsymbol{y_T})$ is known as the *kriging* predictor and can be shown to be the best linear predictor also for arbitrarily distributed data (cf. Cressie (1993))! Moreover, the whole approach has strong relationships to classical weighted regression modeling and longitudinal regression models. Especially for the latter, part of the correlation of data points is tried to be modeled by random effects inducing nontrivial correlation structures. Something similar is done in the machine learning field by combining different kernels to new Gram (covariance) matrices which e.g. also result in periodic behavior (see Ebden (2008)). But this is often not an easy task like for Lloyd (2014) in the GEFcom2012 competition (Hong (2012)) who states that “the forms and parameters of the kernel functions (…) were chosen by trial and error (observing which kernels gave reasonable looking predictions and computing public test scores)”.

Also a tight relationship to simple AR(1) modeling exist. Using a simple exponential relationship in the correlation function for a simple time series without any covariates is equal to a standard AR(1) model as $corr(y_t, y_{t'}) = exp(-|(t-t')/\phi|) = \phi_1^{|t-t'|}$ (using $\phi_1 := exp(-|1/\phi|)$) is exactly the correlation of an AR(1) process.

A more popular choice for the kernel comprises a Gaussian correlation function dependent on the isotropic distance in the predictor space $cov(y_t, y_{t'}) = k(\boldsymbol{x}, \boldsymbol{x'}) = \sigma^2 exp(-\|(\boldsymbol{x} - \boldsymbol{x'})/\phi\|^2)$. Notice that utilizing this kernel results just on first sight in nearly the same model as above described SVM using a exponential kernel! The only difference is that the lagged target values are now not part of the predictor set but are implicitly used as they determine the empirical correlation. But for the SVM the kernel is modeling the covariate effects whereas for a Gaussian process it just encodes the correlation matrix. In general, modeling a Gaussian process as kernel machine can be seen as weighted (or correlated) Ridge-Regression (due to the L2-penalization) with a different loss function (ε-insensitive-loss instead of squared-loss) using the covariates solely (!) for determining the data correlation. This finding makes this model type not very promising. Nonetheless, due to the motivation described in Chapter 1.1, it is tested in the benchmarks using above mentioned Gaussian kernel.

For all benchmarks the same tuning ranges are applied which are a result of a rough grid search over all benchmark series! For the SVM with a radial kernel the cost parameter is set to C=2 whereas σ varies over the values of 2^{-9}, 2^{-8} …, 2^{-3} which are also the σ-values for the Gaussian Process. The polynomial kernel tests degrees of 1 to 4 and let the cost parameter vary over the values of 1, 2, 3, 5 and 10; the internal scale parameter is set to 0.01 (see also the R-documentation of the kernlab R-package Karazoglou et al. (2015)).

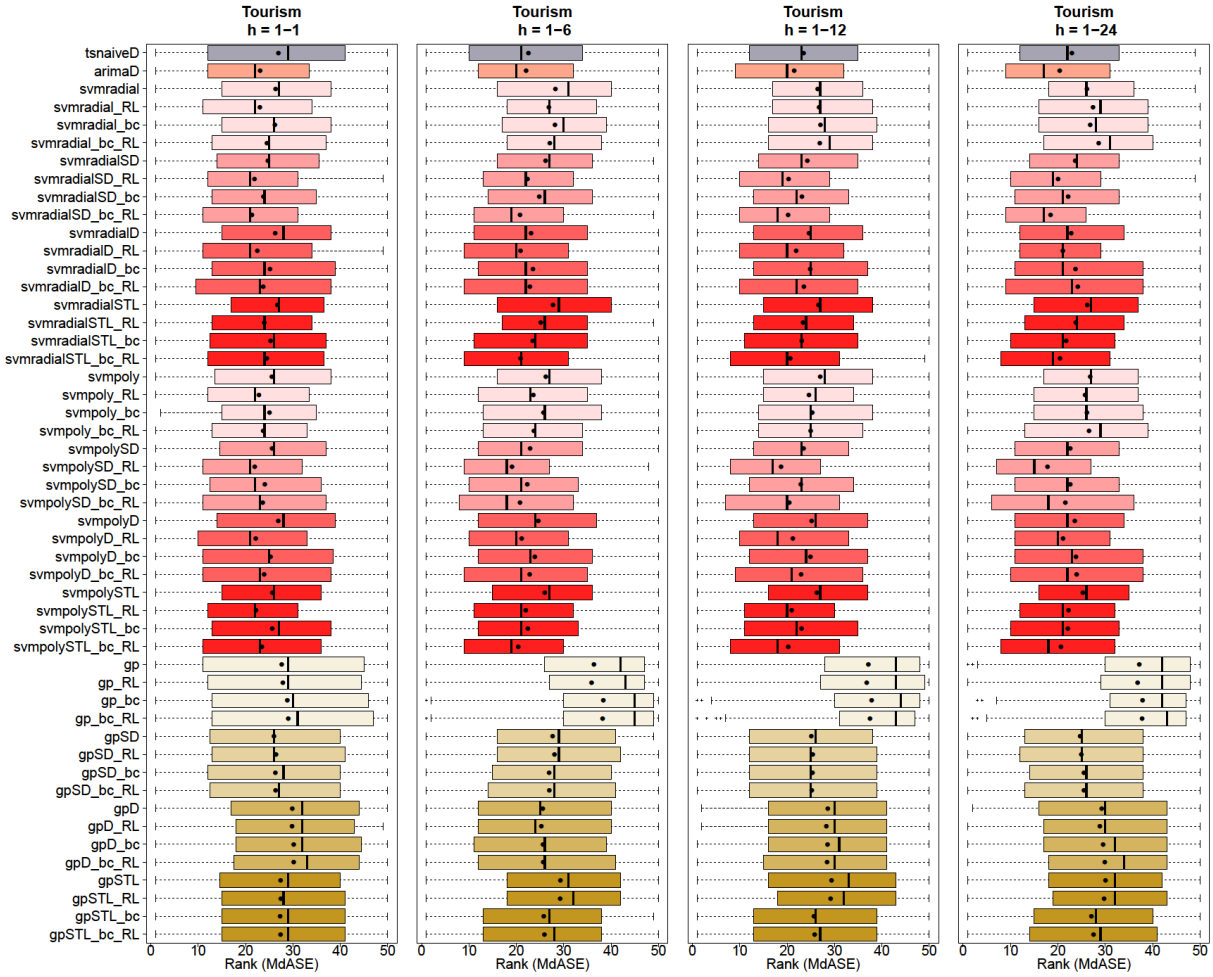

Figure 4.4: *Ranked MdASE by horizon range of Kernel machines for Tourism competition. Results of best naïve (tsnaiveD) and one of the best classical models (arimaD) are added.*

Benchmark Insights

Figure 4.4 and Table 4.2 compare the performance of the Kernel Machines applied on the Tourism competition data. Actually the SVM method is the only used Machine Learning method that beats *arimaD* in the long run for these time series (cf. Chapter 5.2)!

Basically there are just slight differences between using a polynomial (*svmpoly*) or a radial (*svmradial*) kernel. But the Gaussian Process (*gp*) approach is clearly inferior. Furthermore, as will be shown later in Chapter 5, *gp* is consistently the worst Machine Learning method over all benchmarks (except for the M3-MACRO time series). Additionally the following can be identified in Figure 4.4 and Table 4.2:

- Using a STL decomposition for deseasoning profits from a Box-Cox transformation starting from half season horizon range on. Such a behavior has already been noticed in the previous chapter in conjunction with the ANN model. Notice that for the NN5 benchmark data there is just an advantage in the very long run (cf. Figure 5.3 in Chapter 5.2).
- The reduced lag variants are consistently superior. This holds even for the “plain” versions (which are using only the lag information for modeling seasonality) indicating that the autoregressive influence is very limited for the Tourism time series. Further this confirms the forthcoming simulation results for Kernel Machines (cf. Chapter 4.5) that seem to profit more from removing irrelevant variables than other approaches. As already mentioned, further improvement might be

Method	Tourism h=1-1	Tourism h=1-6	Tourism h=1-12	Tourism h=1-24
tsnaiveD	26.94	22.56	23.48	22.98
arimaD	23.06	22.04	21.50	20.37
svmradial	26.37	28.26	26.45	26.13
svmradial_RL	23.03	26.87	26.76	27.43
svmradial_bc	26.21	28.17	27.02	26.82
svmradial_bc_RL	24.53	27.08	26.90	28.61
svmradialSD	24.71	26.18	24.26	23.66
svmradialSD_RL	21.87	22.35	20.32	20.02
svmradialSD_bc	23.74	24.85	23.14	22.14
svmradialSD_bc_RL	**21.33**	20.78	20.22	18.47
svmradialD	26.26	23.09	24.64	22.81
svmradialD_RL	22.48	20.87	21.90	21.05
svmradialD_bc	25.17	23.48	24.85	23.68
svmradialD_bc_RL	23.70	22.87	23.54	24.22
svmradialSTL	26.68	27.72	26.62	26.18
svmradialSTL_RL	23.90	25.13	23.35	23.73
svmradialSTL_bc	25.25	23.43	23.08	21.72
svmradialSTL_bc_RL	24.48	20.88	20.63	20.42
svmpoly	25.57	26.25	27.01	26.83
svmpoly_RL	22.80	23.60	24.64	25.72
svmpoly_bc	25.04	25.74	25.32	26.13
svmpoly_bc_RL	23.72	23.71	24.95	26.55
svmpolySD	25.59	22.92	23.46	22.63
svmpolySD_RL	21.92	**19.10**	**18.70**	**17.82**
svmpolySD_bc	24.06	22.32	22.87	22.62
svmpolySD_bc_RL	23.62	20.79	20.47	21.56

Method	Tourism h=1-1	Tourism h=1-6	Tourism h=1-12	Tourism h=1-24
svmpolyD	26.89	24.63	25.16	23.54
svmpolyD_RL	22.12	21.14	21.22	21.05
svmpolyD_bc	25.26	23.90	24.92	23.78
svmpolyD_bc_RL	23.91	22.78	22.95	23.93
svmpolySTL	25.67	25.97	26.27	25.24
svmpolySTL_RL	22.19	21.97	20.94	22.23
svmpolySTL_bc	25.59	22.42	23.01	22.08
svmpolySTL_bc_RL	23.43	20.36	20.19	20.63
gp	27.64	36.38	37.14	37.17
gp_RL	27.89	35.90	36.80	36.84
gp_bc	28.81	38.35	37.85	37.87
gp_bc_RL	29.04	38.23	37.51	37.79
gpSD	26.01	27.66	25.13	24.69
gpSD_RL	26.43	28.04	25.41	24.96
gpSD_bc	26.30	26.92	25.36	25.46
gpSD_bc_RL	26.32	26.95	25.22	25.50
gpD	29.88	25.56	28.60	29.25
gpD_RL	29.80	25.29	28.30	28.84
gpD_bc	30.17	25.63	28.52	29.56
gpD_bc_RL	30.16	25.70	28.44	29.88
gpSTL	27.38	29.27	29.38	30.05
gpSTL_RL	27.42	29.23	29.18	29.77
gpSTL_bc	27.28	25.79	25.64	27.08
gpSTL_bc_RL	27.41	25.92	25.79	27.50

Table 4.2: *Average Ranked MdASE corresponding to dots in Figure 4.4. Best model metric per horizon range is bold & underlined and shading denote Top20% accordingly.*

received by lag tuning for the SVM models. Notice in this context that for Gaussian process a lag reduction is irrelevant as the lagged target variables are not used as covariates. For the M3-INDUSTRY and the NN5 data the situation is similar for the non-plain versions (cf. Figure 5.1 & Figure 5.3 and Table 5.1 & Table 5.3 in Chapter 5).

- Regarding the best approach to tackle seasonality no clear direction can be given as the performance highly depends on the variant according Box-Cox transformation and lag reduction as well as the examined horizon range. But actually *svmpoly_SD_RL* is the best overall method. Remarkably this is also the case for the NN5 benchmark series for the competition horizon range (see Table 5.3)!

4.3 Random Forests

A Random Forest builds an ensemble vote from several CART (classification and regression tree) models applied to bootstrapped training data.

A CART model is *the* standard tree classifier for which at each split a criterion, e.g. squared loss in the regression case, is evaluated for all covariates and possible split-points. The best covariate split-point combination is used to define the split. This process can be stopped if a node has fewer observations than a defined value, or if the splitting criterion stays above a threshold, i.e. for example when no improvement can be achieved by an additional split. But usually a tree is fully grown and pruned again due to a complexity criterion, which can be optimized by cross-validation. This process assures an intrinsic variable selection, robustness to outliers in input space and insensitiveness to monotone covariate transformations. Furthermore missing input values are elegantly treated by using surrogate variables, best mimicking the “orginal” split variable, as replacements for just the missing data.

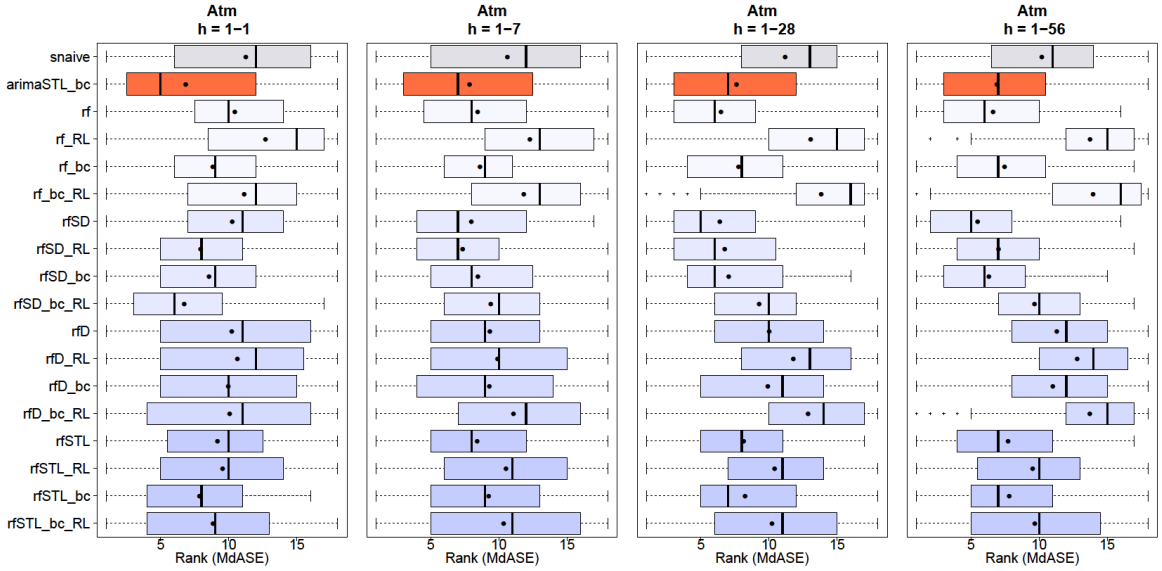

Figure 4.5: *Ranked MdASE by horizon range of Random Forest based models for NN5 competition data. Results of best naïve (snaive) and one of the best classical models (arimaSTL_bc) are added.*

Method	Atm h=1-1	Atm h=1-7	Atm h=1-28	Atm h=1-56	Method	Atm h=1-1	Atm h=1-7	Atm h=1-28	Atm h=1-56
snaive	11.26	10.62	11.19	10.21	rfD	10.23	9.33	10.02	11.31
arimaSTL_bc	6.83	7.85	7.60	6.90	rfD_RL	10.64	9.91	11.78	12.79
rf	10.45	8.44	6.48	6.63	rfD_bc	9.96	9.30	9.91	11.01
rf_RL	12.70	12.27	13.06	13.75	rfD_bc_RL	10.06	11.08	12.87	13.73
rf_bc	8.83	8.62	7.77	7.46	rfSTL	9.18	8.41	8.14	7.72
rf_bc_RL	11.13	11.83	13.83	13.97	rfSTL_RL	9.55	10.52	10.41	9.53
rfSD	10.26	7.98	**6.39**	**5.50**	rfSTL_bc	7.85	9.25	8.25	7.81
rfSD_RL	7.94	**7.35**	6.75	7.03	rfSTL_bc_RL	8.85	10.36	10.23	9.68
rfSD_bc	8.56	8.47	7.04	6.31					
rfSD_bc_RL	**6.73**	9.41	9.28	9.67					

Table 4.3: *Average Ranked MdASE corresponding to dots in Figure 4.5. Best model metric per horizon range is bold & underlined and shading denote Top20% accordingly.*

The main disadvantage of CARTs is a high prediction variance due to totally different trees that might already result from small changes in input data. This can be tackled by averaging over several CARTs which are trained on bootstrapped data. Importantly, the predictions of the different models are identically distributed but not independent, giving rise to a decorrelation step for further variance reduction as illustrated in the following equation:

$$Var\left(\frac{1}{B}\sum_{i=1}^{B} x_i\right) = \frac{1}{B^2}\left(\sum_{i=1}^{B} Var(x_i) + \sum_{i\neq j} Cov(x_i, x_j)\right) = \frac{\sigma^2}{B} + \frac{B^2-B}{B^2}\rho\sigma^2 = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 \tag{4.8}$$

This relationship shows that the variance of the average of identically distributed but dependent random variables (with variance σ^2) can be reduced not only by increasing the sample size B, but also by reducing ρ, here correspondingly the correlation between different trees. This decorrelation is achieved in a Random Forest algorithm by providing just a randomly chosen subset of m out of all p covariates in each split (for all trees). As this covariate reduction also increases the bias, the tree

number *m* is a tuning parameter which can also directly evaluated through the performance for the out-of-bag (OOB) bootstrap folds. Importantly, the trees in the ensemble are not pruned in order to avoid further bias growth. The number of trees should be sufficiently large; for all benchmarks some pretests have shown that the default value of 500 used by the *randomForest* function of the R-package *randomForest* (Liaw & Wiener (2014)) is adequate. For the number of randomly chosen predictors in each split a grid of 1, 3, 5, 10 and 20 is tested (and automatically reduced in case of less predictors available).

Before discussing some benchmark results, some disadvantages of Random Forests should be mentioned. First of all they are incapable of predicting a future trend. In fact this problem resists for all tree-based learners, e.g. also for the tree based *gbm* model introduced in the next chapter. An explicit example showing this problem for time series forecasting can be found in the simulation study of Chapter 4.5.

Secondly the interpretation capability of a single tree gets lost. But even though the individual covariate influence per observation cannot be determined an overall so-called importance plot might help. In this plot the sum of all split criterion improvements for each covariate is ranked. Alternatively the OOB prediction difference resulting from randomly permuting the values of one covariate can be calculated. The latter “importance” metric has the disadvantage that 2 highly correlated predictors result in low importance for both whereas they kind of share their importance in the first version.

Benchmark Insights

Actually the Random Forest is not capable of beating the best naïve method for the Tourism and M3-INDUSTRY time series apart from the 1-step-ahead prediction in the Tourism case (see Chapter 5). Therefore Figure 4.6 and Table 4.3 show the results for the NN5 benchmark. The used tuning parameter values are discussed above.

- Very drastic is the performance reduction shown over most horizon ranges and variants when reducing the lagset!
- A Box-Cox transformation is only advantageous for the 1-step-ahead forecast in conjunction with a STL decomposition or seasonal dummies (*SD*).
- Generally, the seasonality is best handled by using seasonal dummies. Tough, its performance highly depends on the additional step variants. E.g. the winner in the long run *rfSD* performs really bad for a 1-step-ahead forecast as already mentioned in Chapter 2.3.2!

It must be mentioned that these results are different for the Tourism and M3-INDUSTRY time series as it can be identified in the figures and tables of Chapter 5. But this finding should not get too much attention as the Random Forest is performing worse than the best naïve method for these series.

4.4 Boosting

Boosting is, similar to Random Forests, an ensemble method. One main difference is that for boosting each model in the ensemble is applied to different data. E.g. for the first boosting algorithm invented for classification problem called *AdaBoost*, the misclassified observations in each run of the boosting algorithm get a higher weight in the following run. Each run is represented by the fit of a so-called *weak learner*, i.e. a model which is slightly better than guessing or equivalently spoken has high bias but low variance. for example tree stumps in case of AdaBoost. This can also be seen as the second main difference to bagging which aims to reduce the high variance of a low-biased learner by ensembling. Interestingly it can be shown that AdaBoost is equivalent to forward stagewise additive

modeling using exponential loss $L(y, f(\boldsymbol{x})) = e^{-yf(x)}$ (with y coded as $\{-1; 1\}$) (see e.g. Hastie et al. (2009)). Here the solution of the empirical risk minimization in each step

$$(\hat{\beta}_m, \hat{\boldsymbol{\gamma}}_m) = arg\,\min_{\beta,\boldsymbol{\gamma}} \sum_{i=1}^{N} L(y_i, f_{m-1}(\boldsymbol{x}_i) + \beta b(\boldsymbol{x}_i; \boldsymbol{\gamma})) \tag{4.9}$$

is used to update the model in an additive greedy way:

$$f_m(\boldsymbol{x}) = f_{m-1}(\boldsymbol{x}) + \hat{\beta}_m b(\boldsymbol{x}; \hat{\boldsymbol{\gamma}}_m)\ , \tag{4.10}$$

with $b(\boldsymbol{x}; \boldsymbol{\gamma})$ denoting the weak base learner dependent on the model parameter $\boldsymbol{\gamma}$. Obviously this can be overtaken to the regression case with continuous target y and a quadratic loss which would result in repetitive fitting of residuals! Alternatively more robust (i.e. less sensitive to outliers) loss functions are absolute error $L(y, f(\boldsymbol{x})) = |y - f(\boldsymbol{x})|$ or the Huber loss, with the latter basically using quadratic loss for small and absolute loss for higher values of $|y - f(\boldsymbol{x})|$.

Even though the Huber loss is differentiable on the whole support, the optimization becomes much more complicated which gave rise to the very feasible G*radient Boosting* algorithm, inspired by numerical optimization approaches and applicable to any differentiable loss function and arbitrary base learner. For this algorithm the pseudo residuals $r_{im} = -\left[\frac{\partial L(y_i, f(\boldsymbol{x}_i))}{\partial f(\boldsymbol{x}_i)}\right]_{f=f_{m-1}}$ are always (also in the classification case!) fitted with a so-called regression base learner, i.e. using a quadratic loss for determining the model parameter

$$\hat{\boldsymbol{\gamma}}_m = arg\,\min_{\boldsymbol{\gamma}} \sum_{i=1}^{N} (r_{im} - b(\boldsymbol{x}_i; \boldsymbol{\gamma}))^2 \tag{4.11}$$

whereas the update parameter $\hat{\beta}_m$ for (4.10) is afterwards calculated using the desired loss function:

$$\hat{\beta}_m = arg\,\min_{\beta} \sum_{i=1}^{N} L(y_i, f_{m-1}(\boldsymbol{x}_i) + \beta b(\boldsymbol{x}_i; \hat{\boldsymbol{\gamma}}_m)) \tag{4.12}$$

Additionally if the base learner is not weak by design, e.g. a deeper tree than a stump, it can always be casted to have this property by multiplying the base learner with a regularization parameter $0<\nu<1$ to prevent the model from overfitting. Usually this parameter is set to a small value of 0.1, 0.01 or 0.001 leaving the number of model updates m as the only tuning parameter.

Furthermore some improvement both in accuracy as well as processing speed can be usually achieved by sampling a fraction of the input data in each iteration, resulting in *stochastic Gradient Boosting*.

Now different base learner can be used for $b(\boldsymbol{x}; \boldsymbol{\gamma})$. Very popular are trees, comprising around 6 levels defining the interaction depth (cf. Hastie et al. (2009)). Boosting this learner combines the ability of trees to model arbitrary interactions, comfortable dealing with missing values and the intrinsic feature selection property with improved nonlinear fitting capabilities and the resistance to overfitting, resulting in one of the most popular and easy-to-use predictive algorithms. This approach is named *gbm* in this thesis, following the R-package name providing this algorithm. One disadvantage of this approach is the lack of interpretability which can be tackled to some extent by an *importance plot*, describing the gain regarding the internal split criterion (e.g. quadratic loss for regression trees) per covariate aggregated over all splits (see also Chapter 4.3). Additionally partial dependence plots can be calculated for which in general the univariate, but controlled for other covariates, partial dependence of

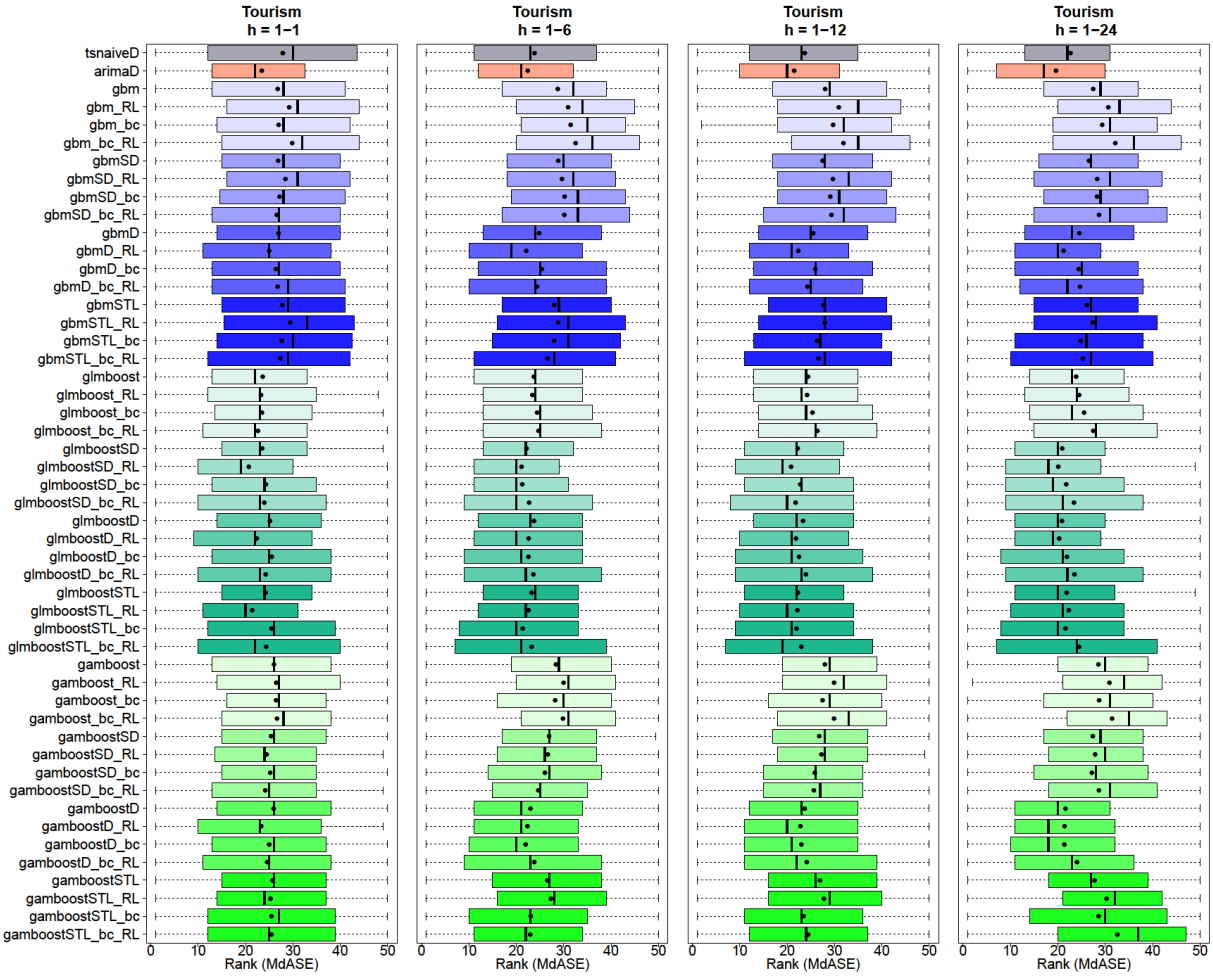

Figure 4.6: *Ranked MdASE by horizon range of Boosting models for Tourism competition. Results of best naïve (tsnaiveD) and one of the best classical models (arimaD) are added.*

$f(X)$ on the kth covariate X_k can be defined as the expected value $E_{X_{\setminus k}} f(X_k, X_{\setminus k})$ with $X_{\setminus k}$ denoting the covariable vector without the kth variable. This metric can be estimated at every empirical value of X_k by $\frac{1}{N}\sum_{i=1}^{N} f\left(X_k, x_{\setminus k_i}\right)$, see Hastie (2009). Even though this idea is applicable to every black-box algorithm, it is very popular for tree-based boosting algorithms for which the dependencies can be cheaply calculated.

Other boosting models provide more direct interpretability capabilities. For instance, it is also possible to boost linear regression models. If the base learner is further chosen from a pool of base learners, each restricted to just one covariate, an implicit variable selection process can be realized. This component-wise boosting, named *glmboost,* provides covariate effect estimations, i.e. the familiar "betas", but obviously without any confidence intervals. Furthermore it is possible to plot coefficient paths nicely illustrating the effect per covariate in dependence of the shrinkage parameter ν. Interestingly, it can be shown (see Hastie et al. (2009)) that the resulting paths are equal to the paths resulting from a lasso regression, i.e. a linear regression with L1 regularization, in case all lasso coefficients increase monotonically (as boosting paths are monotone due to the greedy construction principle).

Method	Tourism h=1-1	Tourism h=1-6	Tourism h=1-12	Tourism h=1-24
tsnaiveD	27.89	23.85	23.75	22.66
arimaD	23.40	22.41	21.48	**19.61**
gbm	26.79	28.77	28.02	27.44
gbm_RL	29.21	30.95	30.90	30.62
gbm_bc	26.99	31.45	29.74	29.34
gbm_bc_RL	29.86	32.52	31.92	32.14
gbmSD	26.88	28.87	27.51	26.58
gbmSD_RL	28.43	29.63	29.67	28.30
gbmSD_bc	27.21	30.21	29.14	28.26
gbmSD_bc_RL	26.58	30.12	29.39	28.72
gbmD	27.01	24.83	25.52	24.55
gbmD_RL	25.01	22.09	22.37	21.20
gbmD_bc	26.47	25.38	25.94	24.42
gbmD_bc_RL	26.77	24.37	24.32	24.71
gbmSTL	27.79	27.99	27.73	26.18
gbmSTL_RL	29.48	28.85	28.01	27.39
gbmSTL_bc	27.70	28.03	26.37	24.86
gbmSTL_bc_RL	27.36	26.64	26.63	25.26
glmboost	23.63	23.69	24.43	23.83
glmboost_RL	23.28	23.41	24.20	24.51
glmboost_bc	23.47	24.41	25.35	25.53
glmboost_bc_RL	22.65	24.70	26.40	27.45
glmboostSD	23.54	22.16	22.26	20.94
glmboostSD_RL	**20.68**	**21.11**	**20.86**	20.10
glmboostSD_bc	24.32	21.25	22.80	21.77
glmboostSD_bc_RL	23.97	22.70	21.77	23.42
glmboostD	25.16	23.73	23.39	20.86
glmboostD_RL	22.38	22.59	21.90	20.31
glmboostD_bc	25.54	22.58	22.56	21.90
glmboostD_bc_RL	24.28	23.58	23.96	23.50
glmboostSTL	24.27	23.25	22.29	21.85
glmboostSTL_RL	21.44	22.61	22.17	22.30
glmboostSTL_bc	25.52	21.35	21.98	21.63
glmboostSTL_bc_RL	24.38	23.23	23.04	24.51
gamboost	26.01	28.37	27.99	28.57
gamboost_RL	26.49	30.00	29.94	30.89
gamboost_bc	26.43	28.21	27.48	28.71
gamboost_bc_RL	26.66	29.85	29.93	31.44
gamboostSD	25.43	26.93	26.78	27.36
gamboostSD_RL	24.48	26.65	27.23	27.88
gamboostSD_bc	25.22	26.05	25.87	27.19
gamboostSD_bc_RL	24.18	24.65	25.68	28.64
gamboostD	25.94	23.00	23.72	21.62
gamboostD_RL	23.34	22.34	22.84	21.40
gamboostD_bc	24.99	21.98	23.01	21.38
gamboostD_bc_RL	24.54	23.81	24.15	24.05
gamboostSTL	25.70	26.55	26.92	27.75
gamboostSTL_RL	25.29	27.33	27.76	30.28
gamboostSTL_bc	25.50	23.03	23.49	28.60
gamboostSTL_bc_RL	25.50	22.97	24.45	32.59

Table 4.4: *Average Ranked MdASE corresponding to dots in Figure 4.6. Best model metric per horizon range is bold & underlined and shading denote Top20% accordingly.*

One can make gamboost models (see Fahrmeir et al. (2013) for a nice introduction into spline modeling).

These kind of models have shown promising performance in several studies; see the comments on related work in Chapter 1.2.

Further variants exist which for instance are treating dummy coded categorical covariates as one base learner (“Blockwise Boosting”) or using likelihood based loss functions applicable also for corresponding generalized linear models (“Likelihood Boosting”); cf. Tutz (2012).

As always a common grid of tuning parameters is used for all benchmarks which comprise a tree sequence of 100, 500, 900, 1300 for the *gbm* with a constant “interaction depth” of 6 and a shrinkage of 0.01. The number of boosting steps for *glmboost* is varied from 100 to 3100 with a step size of 500. The *gamboost* iterations are the only parameter set that get different values depending on the benchmark, i.e. 100 for Toursim and M3, 500 for NN5 and the simulations of Chapter 4.5 and 3000 for the Arimasim series. All other possible parameter values are kept to the default values (see R-documentation of gbm (Ridgeway (2015)), glmboost and gamboost (Hothorn et al. (2015))).

Benchmark Insights

Some initial findings are given for the Tourism benchmark data in Figure 4.6 and Table 4.4. Actually all boosting approaches are less performant than classical methods for the M3-INDUSTRY data as can be seen in Chapter 5.1.

Figure 4.6 and Table 4.4 exhibit the following insights.

- Initial Box-Cox transformations show only sporadic effects. This holds also for the NN5 data (see Figure 5.3 in Chapter 5.3).
- Reducing the lagset has a slight positive effect for *glmboost* for the 1-step-ahead forecast. But for *gbm* in conjunction with seasonal differencing (*SD*) it improves which is kind of surprising due to the automatic relevance detection which should be capable of eliminating the irrelevant lags. Actually this is not the case for the NN5 competition (see Chapter 5.3) which corresponds more to the expectations.
- For *gbm* and *gamboost* the hard differencing (*D*) is the best deseasoning method in the long run. Again it should already be mentioned that this changes in case of the NN5 competition data showing best performance for the STL approaches (see again Chapter 5.3). But for the Tourism time series *glmboostSD_RL* is the best method over all horizon ranges and also beats *arimaD* except for the competition horizon objective of 2 years. In the NN5 case also a *glmboost* model (*glmboostSTL_bc*) is the winner in the long run, see Table 5.3.

4.5 Simulation Study

The following small simulation study is intended to shed some light on basic questions regarding the forecasting capability of Machine Learning algorithms for typical time series consisting of just trend and seasonality without using any initial detrending or deseasonalizing. In detail it should be answered whether the algorithms have general problems with trend or seasonality, which predictors are needed to model these typical time series shapes and which algorithms are most promising (in the simulated phenotypic situation).

Figure 4.7 shows the simulated time series inspired by Pregels (1969) classification regarding trend and seasonality combinations. No explicit autoregressive or moving average component is included in the data generating process. Furthermore it was resigned to add any noise. The 15 seasons with a period of 12 might represent monthly time series comprising 15 years which models a seasonality between weekly (period = 7) and hourly (period = 24) as a compromise for typical time series periods. A similar study was conducted by Crone et al. (2009) showing superior performance of SVM over ANN and a classical ARIMA-ETS combination for nonlinear time series (progressive/degressive trend or multiplicative seasonality). Furthermore Robinzonov et al. (2010) compare boosting approaches for simulated time series with nonlinear autoregressive components, but no trend or seasonality.

Even though the different simulations are not exactly on the same scale (due to increasing trend and season components) the MAE is chosen as only a comparison per time series is needed and no aggregation over simulations is conducted (cf. also Chapter 2.3.1).

The investigated algorithms are ANN (*nnet*), SVM with radial (*svmradial*) and polynomial (*svmpoly*) kernel, Gaussian processes (*gp*), Random Forests (*rf*), Gradient Boosted Trees (*gbm*) and component-wise boosted linear (*glmboost*) and spline (*gamboost*) models. All these models are introduced in previous chapters. For *rf*, *gbm* and *gamboost* a special variant is also applied using an initial detrending (*rfdetrend*, *gbmdetrend*, *gamboostdetrend*) due to their incapability of forecasting a trend, see also the forthcoming explanations.

All algorithms are tuned with a parameter set defined by some initial pretests to assure a reasonable parameter grid.

Possible predictors for modeling a trend are the time variable itself, named *t* in the following (ranging from *t*=0-180 in the examples), a season variable counting up the seasons, i.e. season=1-15

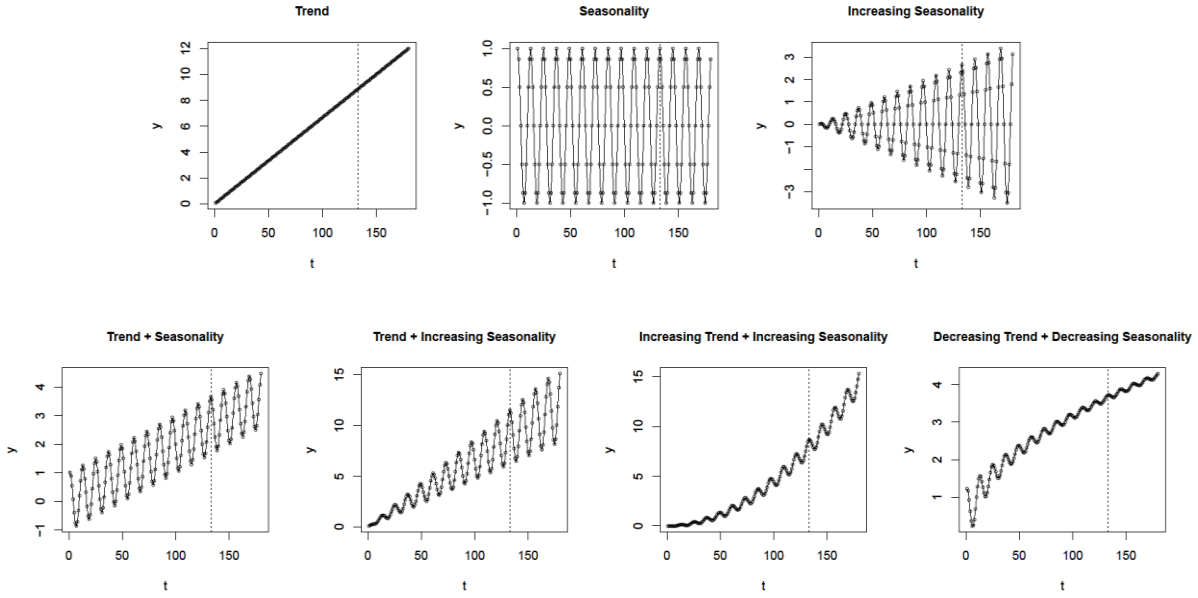

Figure 4.7: *Simulated linear and nonlinear phenotypic time series. Reference line denotes split of train (with first 12 time points removed due to lag creation) and test data. Only time series with increasing/decreasing trend or seasonality ("multiplicative seasonality") are "non-linear" in the time series context.*

(comprising 12 time points each) or the lag-1 information of the target variable (*y_lag1*). The *season* variable in contrast to the more fine grained *t*-variable might be advantageous in case of additional seasonality which can be modeled solely by a flexible nonlinear algorithm just using *t* (but not *season*) as a predictor and therefore "stealing some effect" from the seasonal predictors. Some additional tests have shown that all used algorithms, which are in general capable of modeling the season with the *t*-variable, are robust against this competition of *t*-variable and seasonal predictors. Therefore it was decided to just use *t* and *y_lag1* as possible trend predictors.

For catching the seasonal variation a numeric variable representing the time point inside a season (*S*=1-12), or corresponding 12 seasonal dummy variables (*SD*), or the seasonal lagged target (*y_lag12*) can be utilized.

Furthermore it is of interest how the algorithms perform in each situation when using a typical predictor set comprising variables for trend and season plus additional lag variables, which should catch any autoregressive influence (even though not present in the simulated data) by using at least *y_lag1* & *y_lag12* up to all lags of several seasons. It was decided to restrict the lag information to one season, i.e. *y_lag1, y_lag2, … y_lag12* and compare the performance to a reduced lag-set of *y_lag1* & *y_lag12* to test the sensitivity of the algorithms regarding irrelevant (lag-)variables.

The model names are marked in bold in the following to allow a fast identification of the corresponding explanations for the model.

Trend

The top left plot in Figure 4.8 shows some of the problems machine learning algorithms exhibit when trying to predict a trend.

Due to the sigmoid activation function the ***nnet*** predicts a damped trend.

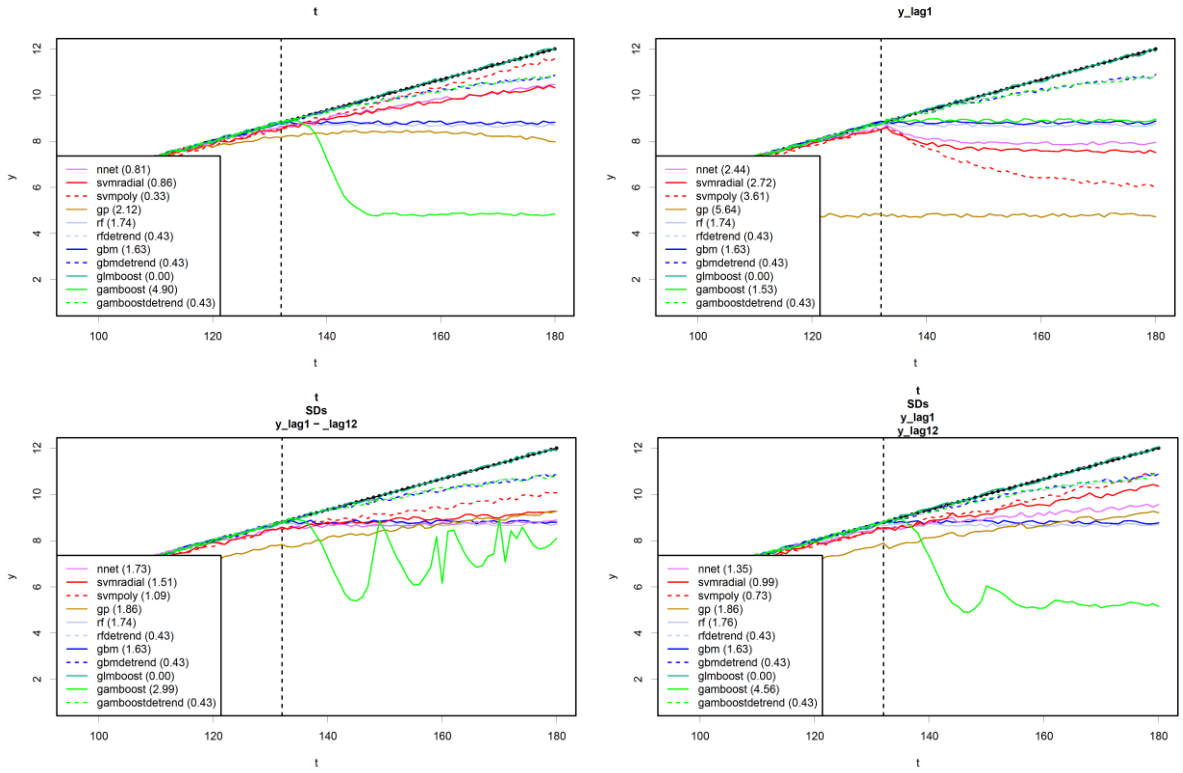

Figure 4.8: *Forecast capability of ML methods for time series consisting solely of a trend. The title in each plot denotes the used predictor set (see text for more explanation). A reference line splits off the test fold for the recursive multi-step forecast. The prediction inside the training fold (ranging from t=12-132) represents a 1-step-ahead forecast. The legend also shows the MAE (mean absolute error) for each method in brackets. A small jittering is applied to the lines in the plot to rectify possible overlay.*

Also the SVM with a Gaussian kernel (***svmradial***) gets a damped prediction. This is a result of the locality property of this kernel. A polynomial kernel (***svmpoly***) is better suited in this example. But when assuming that the trend will not continue (similar to the “philosophy” of an ETS model), the more damped shape of ***svmradial*** might fit better.

The Gaussian process (***gp***) shows a strong damping which actually reverts to a downward trend.

Tree based algorithms (***rf*** and ***gbm***) cannot predict a trend. Actually, due to the construction principle of trees, they are incapable of predicting any value outside the training range of the target. A possible approach could be to detrend the data by a STL decomposition and use an exponential smoothing model (ETS(A,A,N)) for the trend in an additive way before rewinding it to the algorithms forecast of the rest, i.e. season + remainder. The resulting models ***rfdetrend*** and ***gbmdetrend*** are exactly overlaid as in this example the whole prediction is due to the ETS model.

The ***glmboost*** using simple linear predictors is best suited to predict a linear trend resulting in a perfect fit.

The ***gamboost*** prediction returns to the mean training value due to the locality of the basis-functions (P-Splines) used by the boosting base learner (gam). Here also an initial detrending is helping (***gamboostdetrend***).

Using only y_lag1 as predictor (top right plot in Figure 4.8), results in catastrophic prediction for ***nnet***, ***svmradial, svmpoly***.

The ***gp*** prediction is an artefact as no predictors are left in fact (see Chapter 4.2).

When applying the typical predictor set (bottom left plot in Figure 4.8), the performance is lowered for ***nnet***, ***svmradial and svmpoly*** and nearly unchanged for the rest, except interestingly for ***gp*** with a slight improvement and ***gamboost*** now running into a weird wiggly behavior. The shortage for ***nnet***, ***svmradial and svmpoly*** can be soften by reducing the number of lags used (bottom right plot in Figure 4.8). The ***glmboost*** again is not affected at all and still shows the best fit!

Important results for the forthcoming benchmark can be summarized as follows:

- Tree based methods and component-wise spline boosting need detrending.
- A trend predictor variable is needed as y_lag1 cannot do the job of forecasting a trend for some algorithms.
- A reduced lagset might improve performance for some algorithms.

Seasonality

When trying to model a strict seasonality with the numeric *season* covariate (S = 1, 2, …, 12) ***rf***, ***gbm*** and ***gamboost*** result in a perfect fit (top left plot in Figure 4.9). All other algorithms (***nnet***, ***svmradial, svmpoly, gp***, ***glmboost***) show a bad fit, most striking for the more linear approaches ***glmboost*** and ***svm_poly***. Notice that any detrending does not have an influence in this case.

This changes dramatically when using seasonal dummies instead of one numeric covariate. Now all approaches show a good or perfect fit. Some underfitting of peaks occur for ***nnet***, ***svmradial, svmpoly***, ***gp***.

The latter degradation for ***nnet*** and ***svmpoly*** is more emphasized when using *y_lag12* as the only predictor. The bad performance of ***gp*** is again an artefact as no predictors are now used. Interestingly the ***rf*** benefits from using *y_lag12* instead of seasonal dummies. All in all, the lag12 can nearly equally be used instead of seasonal dummies for most algorithms.

With the typical predictor set only the kernel-based approaches (***svmradial, svmpoly***, ***gp***) show a remarkable underfit which can be lowered for ***svmpoly*** by reducing the lag-set. All other algorithms (***nnet***, ***gbm***, ***glmboost***, ***gamboost***) show a perfect fit, slightly reduced for nnet.

Important results are:

- A numeric season variable is not sufficient, seasonal dummies or seasonal lags are better suited for modeling stric seasonality.
- Kernel based methods have more problems than other algorithms with modeling strict seasonality.

Increased Seasonality (Multiplicative Seasonality)

Figure 4.10 (top left plot) shows that using the trend variable together with seasonal dummies, except for ***svmpoly*** all algorithms (***nnet***, ***svmradial***, ***gp***, ***rf***, ***gbm***, ***glmboost***, ***gamboost***) exhibit a bad forecast, for ***gbm*** and rf mostly due to an extreme underfit for the lower peaks. This adapts to the less extreme underfit for the upper peaks when using *season* instead of *t* as a predictor (not shown here) and represents one of the rare situations where this interchanging has an effect. Remarkably the detrending has a negative effect now and hints to the limited capability of the STL decomposition in such data situations (nonlinear trend or seasonality).

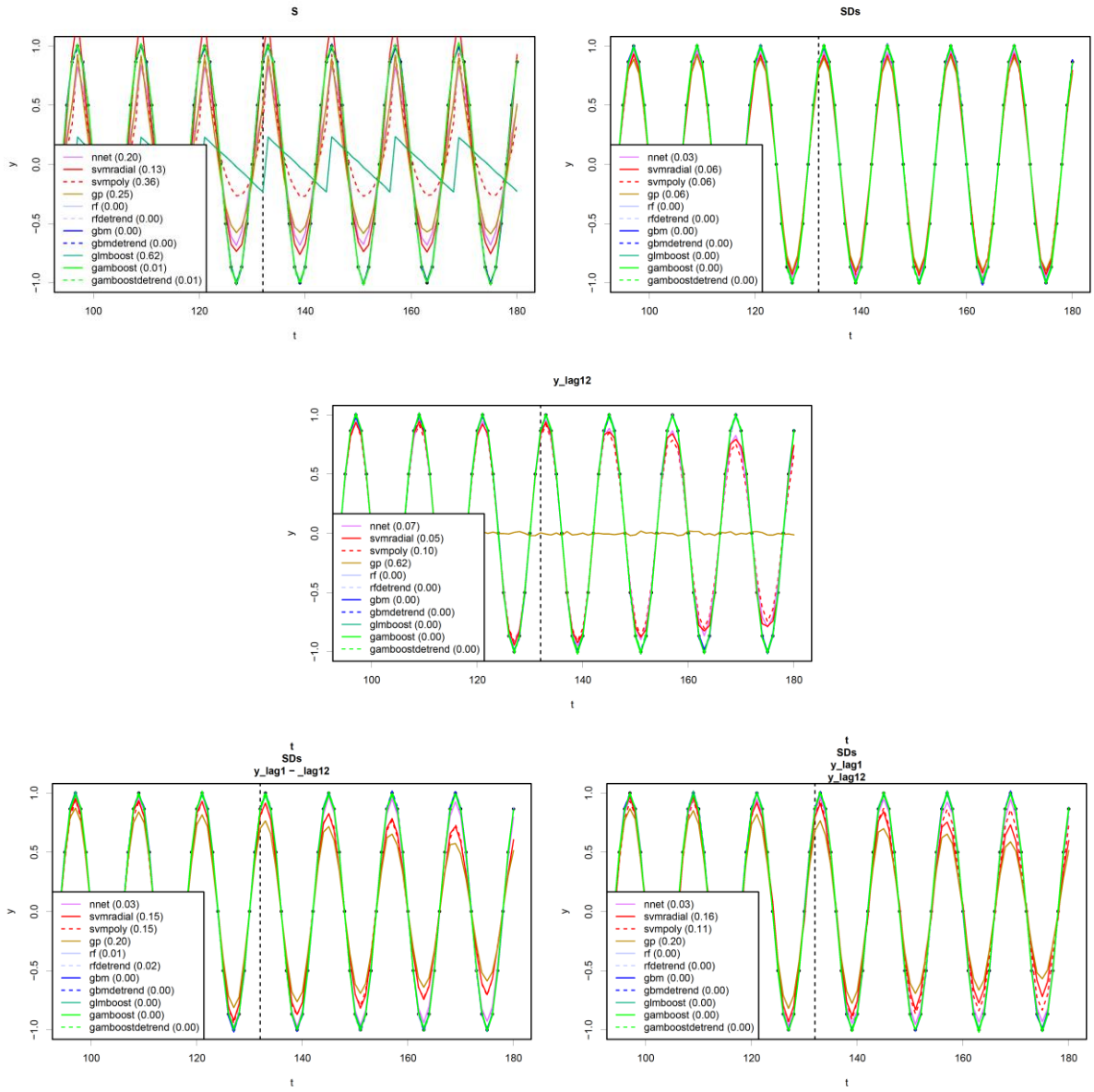

Figure 4.9: *Forecast capability of ML methods for time series consisting solely of seasonal variation (see Figure 4.8 for description of plot elements).*

When using *y_lag12* for modeling the seasonality, a totally different picture results. The fit for all low-performing models improve (***nnet***, ***svm_radial***, ***rf***, ***gbm***, ***glmboost***, ***gamboost***) except for ***gp*** (which now has only *t* left as a predictor). Furthermore the detrending now has a negative influence most striking for ***gamboostdetrend*** ending up with a weird wiggly behavior, whereas the other detrended algorithms ***rfdetrend*** and ***gbmdetrend*** benefit from this switch. The error for ***sympoly*** is nearly the same but the peaks are now more outgrown than underfitted as previously. ***Svmpoly*** still exibit the best fit together with ***nnet*** followed by ***gbm***, all beating now ***glmboost*** which outgrows the peaks to much.

Using the typical predictor set (middle left plot in Figure 4.10) ***glmboost*** returns to a perfect fit. All other models underfit the data most extreme for ***gp***. The same holds for the reduced lagset.

When removing the seasonal dummies and let *y_lag1* - *y_lag12* do the job (bottom left plot in Figure 4.10) it has only a remarkable positive effect on ***svmpoly*** despite the positive effect of using *y_lag12* instead of seasonal dummies in previous discussion. But the effect for ***svmpoly*** lowered again when

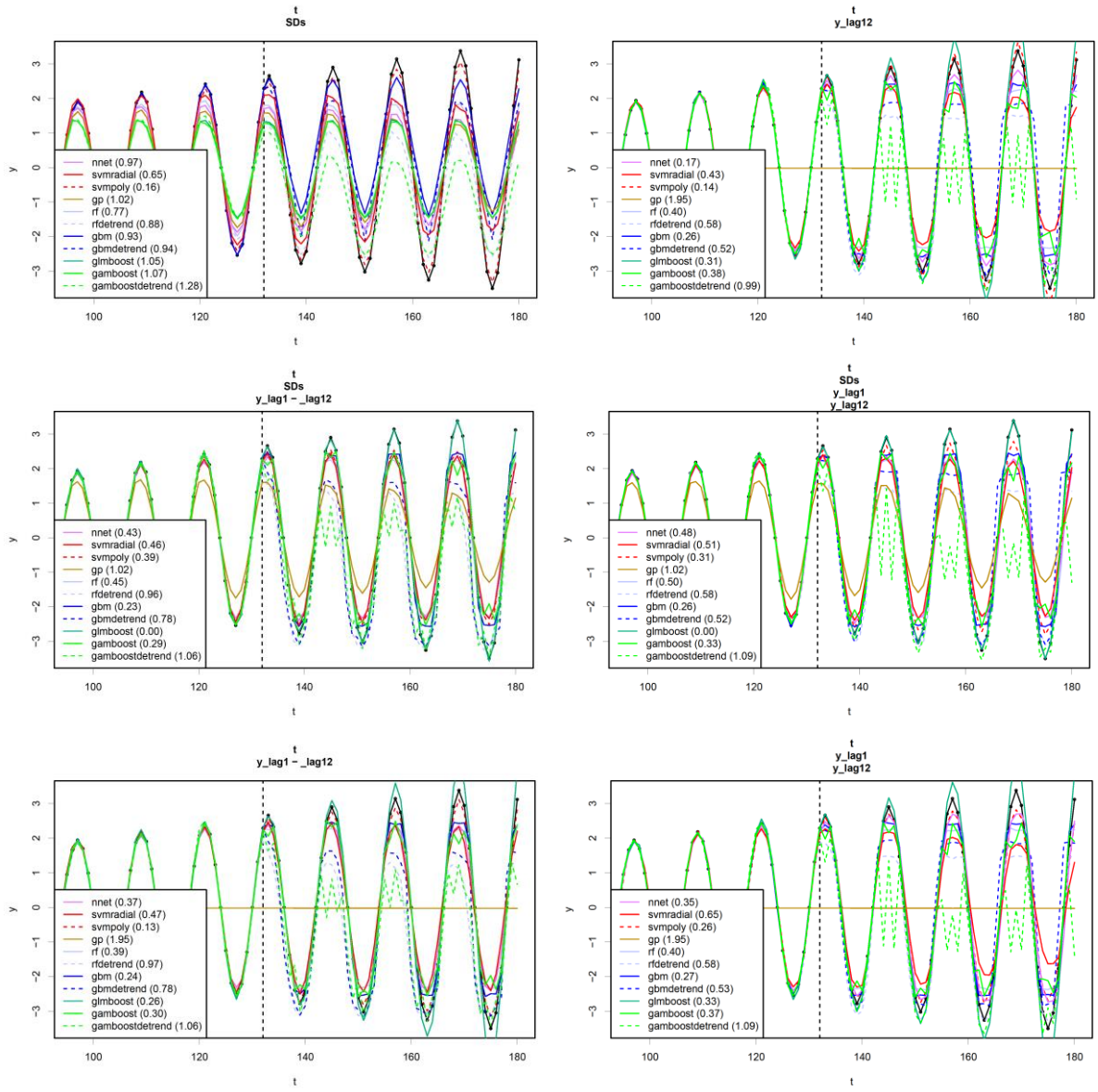

Figure 4.10: *Forecast capability of ML methods for time series consisting solely of increasing seasonal variation, i.e. multiplicative seasonality (see Figure 4.8 for description of plot elements).*

reducing the lagset. On the other hand the ***glmboost*** suffers from this change as it outgrows the peaks too much.

Most remarkable are the following points:

- The linear ***glmboost*** can perfectly model increased seasonality when provided with seasonal and lagged information.
- For all other models an initial Box-Cox transformation would cast the predictive properties to the strict seasonality case which mostly would be advantageous especially for the detrended algorithms.
- The seasonal dummies are only needed for ***glmboost***, for all other algorithms the lagset is sufficient.

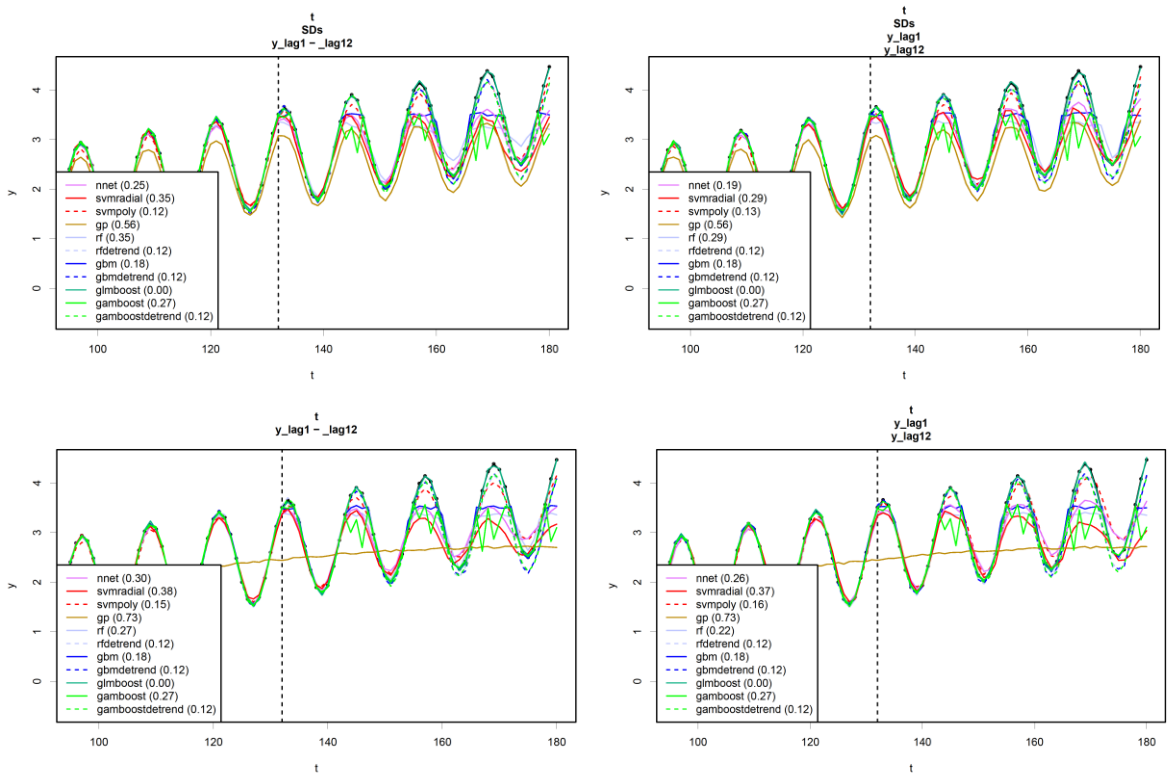

Figure 4.11: *Forecast capability of ML methods for time series consisting of trend and seasonal variation (see Figure 4.8 for description of plot elements).*

Trend + Seasonality

From now on only the behavior using the typical predictor set (with and without the seasonal dummies) is discussed as trend and seasonality are occurring together.

Figure 4.11 shows that a linear combination of trend and seasonality is best fit again with ***glmboost*** when using the typical predictor set. The ***svmpoly***, now not influenced by reducing the lags in contrast to ***nnet*** and ***svmradial***, is the runner-up together with the detrended nonlinear models ***rfdetrend***, ***gbmdetrend*** and ***gamboostdetrend***. Interestingly now the ***gamboost*** shows the weird wiggly behavior. As usual ***nnet*** and ***svmradial*** benefit from lag reduction. ***Gp*** again shows an extreme underfit already starting in the training fold.

Using only the lagged information together with the trend does in general not improve the performance.

Mind-keeping should be the fact that ***svmpoly*** performs nicely in this typical situation but is still clearly beaten by ***glmboost***.

Trend + Increasing Seasonality

All findings from the previous situation can be overtaken except that detrending is now counterproductive as already mentioned in the “Increasing Seasonality” simulation, making ***svmpoly*** the only runner-up of ***glmboost*** (see Figure 4.12) followed by ***gbmdetrend***. Again an initial BoxCox transformation would melt down the predictions to above simulation (trend + seasonality).

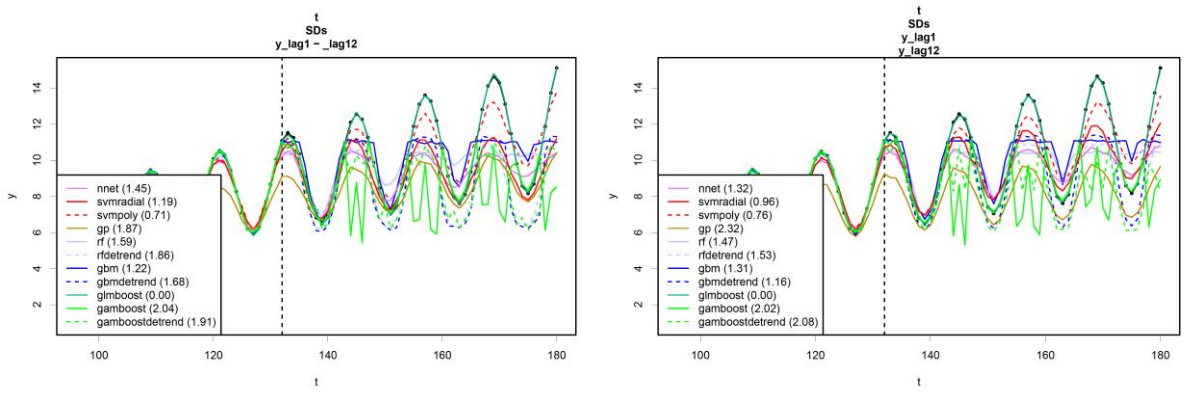

Figure 4.12: *Forecast capability of ML methods for time series consisting of trend and multiplicative seasonality (see Figure 4.8 for description of plot elements).*

Using the lagset without the seasonal dummies (not shown in plot) improves the ***svmpoly*** forecast and decreases the ***glmboost*** performance similar to the above "Increasing Seasonality" simulation. Also the ***svmradial*** suffers from this change.

Increasing/Decreasing Trend + Increasing/Decreasing Seasonality

Figure 4.13 shows in the top row the situation of increasing trend + increasing seasonality and in the bottom row the corresponding results for decreasing trend + decreasing seasonality, each again just for the typical predictor sets. Interestingly the ***glmboost*** is still the best model in both cases only reached by the detrended models and ***svmpoly*** in the decreasing case. For the increasing case the ***svmpoly*** performance now suffers a lot now from reducing the lags. All in all lots of the interesting behaviors already mentioned for the other simulations are occurring again, e.g. the wiggly ***gamboost*** prediction. The situation does not change when removing the seasonal dummies (not shown in Figure 4.13).

All above findings additionally suggest the following variants when applying the machine learning algorithms to the benchmark data:

- Applying detrended versions of random forest, gradient boosted tree and gamboost models.
- Usage of initial BoxCox transformation especially for increasing seasonality problems.
- Removing of seasonal dummies and letting the lagged target value model the seasonality.
- A reduced lag variant especially for neural net and kernel based methods.

Furthermore it is expected that ***glmboost*** will outperform all other machine learning algorithms when the benchmark data is near to the phenotypic situations of the simulations. Also the classical ARIMA and ETS approaches together with the usual accompanying transformation (e.g. Box-Cox, Differencing, …) will then be more than competitive as they are capable of fitting such data situations.

But it must be emphasized again that the simulations are solely caring about trend and seasonality but no autoregressive or covariate effects are regarded. Especially the latter point might push the gain for the NN5 competition data for other algorithms, especially the well-performing ***svmpoly***, due to the important covariate interactions (described in Chapter 2.1) which cannot be automatically caught through a ***glmboost*** model as this model is incapable of automatic interaction modeling.

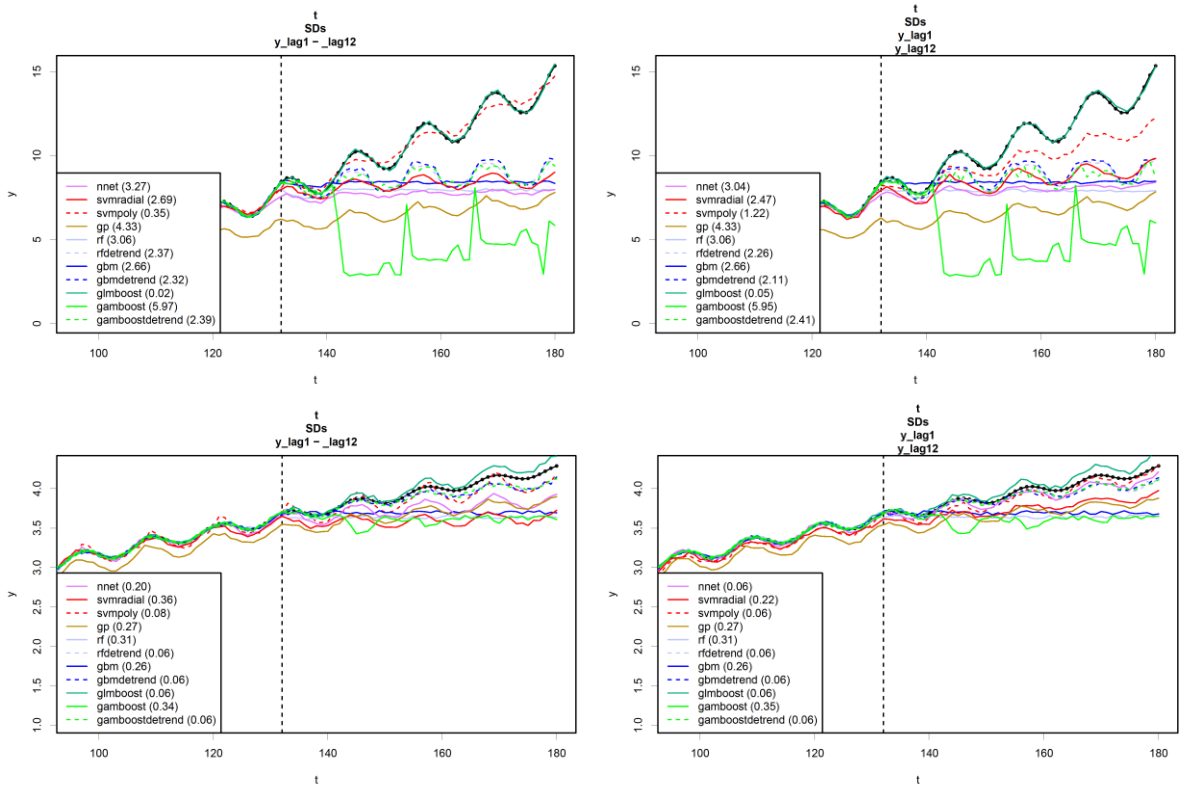

Figure 4.13: *Forecast capability of ML methods for time series consisting of increasing trend and increasing season (top row) and decreasing trend and decreasing seasonality (see Figure 4.8 for description of plot elements).*

5 Overall Benchmark Results

This chapter discusses main results from all benchmarks regarding the comparison of classical and Machine Learning approaches for time series forecasting. Be also aware of the benchmark insights given throughout Chapter 4 concentrating on the Machine Learning algorithms as well as Chapter 3.5 discussing solely the results for the classical models (including Bagging approaches). Furthermore Chapter 3 additionally comprises some analysis attached to the Figures included in this chapter for some special classical models.

Moreover, this chapter presents results for models incorporating covariates for the NN5 benchmark. In fact the best NN5 models are tested again as a variant, denoted through the additional suffix *COVARS*, utilizing the covariates discussed in Chapter 2.1, i.e.: *monthday, month, holiday, dayBefHoliday, dayAftHoliday, weekBefEaster, weekAftEaster*. Furthermore a possible improvement by applying the sRecDir multi-horizon strategy (as opposed to the standard recursive strategy), introduced in Chapter 2.2, for these models is investigated. Notice that this strategy also allows to compare a strict direct forecasting strategy for the first season as explained in Chapter 2.2. The notation for the models using this different forecasting strategy consists of an additional suffix *sRecDir*. The identifiers for all other variants follow the procedure already used in previous chapters: *D*, *STL* and *SD* (and "none") distinguish the handling of the seasonal component denoting seasonal hard-differencing, STL decomposition and usage of seasonal dummies respectively. No suffix in this context stands for models trying to model the seasonality just with the target variable lags comprising all lags for the last two seasons which are added by default to all Machine Learning approaches (in order to model the autoregressive component) except for the *RL* flagged variants getting just a reduced lagset of the most recent and the first seasonal lag. As explained in Chapter 4.2 the Gaussian Process represents an exception in the sense that all lagged target variable information is just used for estimating the covariance structure. Furthermore the *bc* suffix identifes a model applied on a Box-Cox transformed series (cf. Chapter 3.1.1).

For the tree-based *rf* (Random Forests) and *gbm* (boosted trees) models as well as for the *gamboost* (spline boosting) approach also the detrended approaches suggested in the Chapter 4.5 are tested. As always the results for several naïve models are added (cf. Chapter 2.3). For further explanations of the other models see the previous chapters.

5.1 M3

Figure 5.1 and Table 5.1 show the overall results for the M3-INDUSTRY benchmark data. Results for the other M3 categories are put into the Appendix due to the reasons discussed in Chapter 2.3, but some interesting findings for these categories are discussed at the end of this section.

Even though the differences in performance between all models are small compared to the Tourism and NN5 time series, the following points are remarkable:

- Generally all classical models outperform all Machine Learning approaches. Only for the 1-step-ahead forecast *glmboostSD_bc* shows a remarkable performance ending up as the winner for this horizon.
- ETS approaches are competitive with the ARIMA based model. Though, *arimaSTL* is the best overall non-ensemble method even though not the best for the 1-step-ahead forecast.
- Classical Ensemble models show a good performance in general with *bootME_randsel* as the winner regarding the competition objective horizon range.
- Under the Machine Learning algorithms linear boosting (*glmboost*) with its variants is the best approach.
- Highly remarkable is the extreme bad performance Neural Nets (*nnet*) in any variant reveal for M3-INDUSTRY time series. They rarely even beat the *tsnaiveSTL* naïve method.

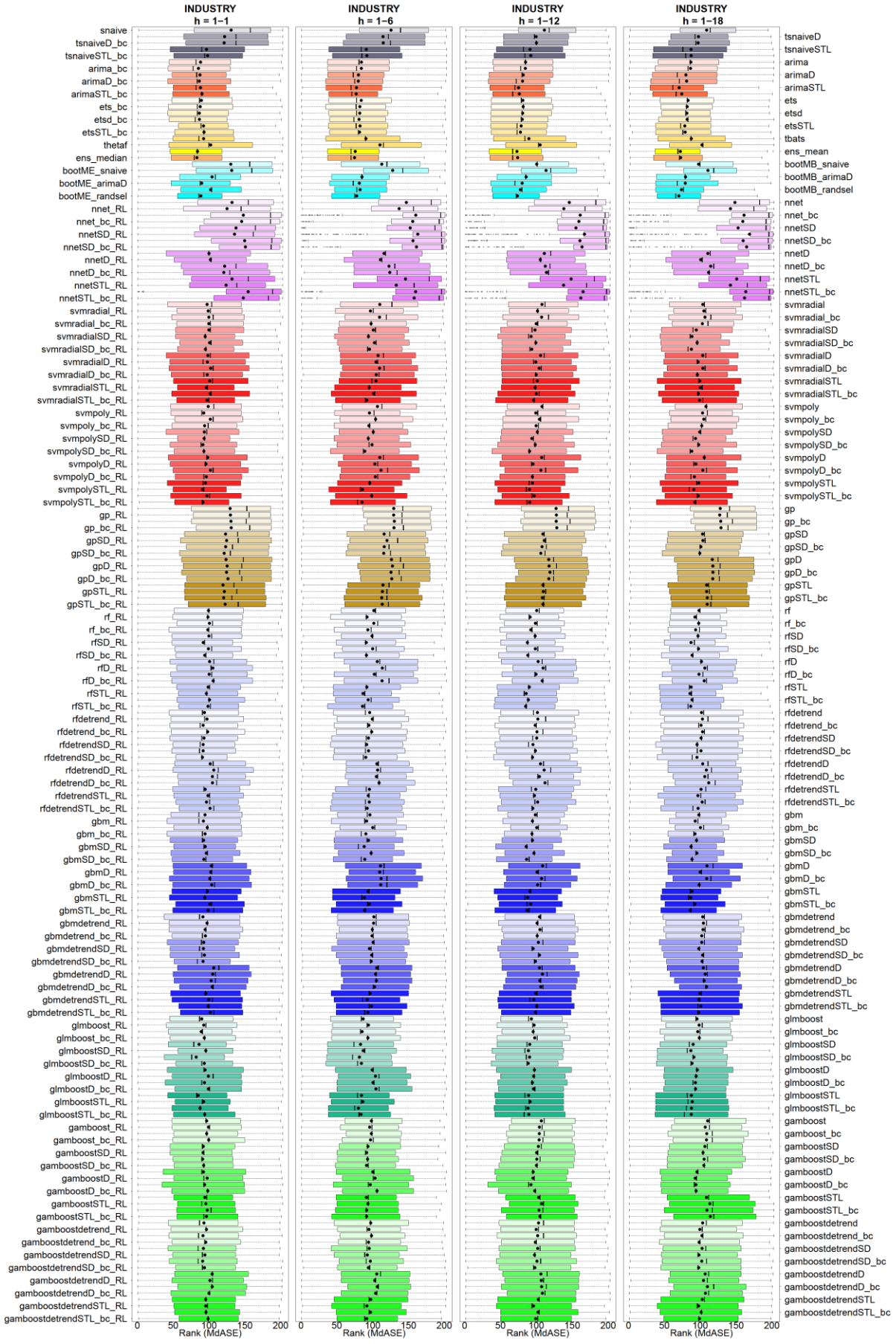

Figure 5.1: *Overall Result (Ranked MdASE) by horizon range for M3-INDUSTRY competition.*

Method	INDUSTRY h=1-1	INDUSTRY h=1-6	INDUSTRY h=1-12	INDUSTRY h=1-18
snaive	131.11	125.66	111.21	109.13
tsnaiveD	121.98	114.50	99.62	97.73
tsnaiveD_bc	121.35	114.92	100.08	97.14
tsnaiveSTL	96.68	91.13	91.12	87.32
tsnaiveSTL_bc	98.25	92.04	92.26	87.40
arima	88.35	84.39	84.93	87.08
arima_bc	85.51	84.21	84.62	86.58
arimaD	87.57	80.20	81.67	80.17
arimaD_bc	86.60	79.91	80.72	81.16
arimaSTL	88.28	77.41	75.06	70.61
arimaSTL_bc	90.27	77.09	76.19	74.30
ets	89.55	84.11	81.21	83.22
ets_bc	87.89	82.15	81.58	81.99
etsd	86.63	81.98	80.86	81.52
etsd_bc	86.81	81.06	80.34	81.81
etsSTL	92.55	82.36	78.68	78.54
etsSTL_bc	93.27	81.67	77.69	79.17
tbats	92.58	90.69	89.53	87.70
thetaf	102.50	110.21	105.24	102.70
ens_mean	84.12	75.68	**72.86**	72.33
ens_median	83.31	**74.73**	73.82	72.81
bootMB_snaive	130.58	112.83	100.47	97.87
bootME_snaive	131.89	128.04	113.46	110.87
bootMB_arimaD	104.43	85.25	85.72	79.69
bootME_arimaD	89.91	81.39	80.20	79.37
bootMB_randsel	102.51	82.51	77.76	73.92
bootME_randsel	88.82	77.80	73.65	**70.35**
nnet	132.25	147.08	145.93	148.62
nnet_RL	125.31	136.96	138.29	142.30
nnet_bc	148.05	160.20	161.19	161.33
nnet_bc_RL	145.80	155.95	159.38	159.62
nnetSD	137.83	152.50	155.22	153.01
nnetSD_RL	135.77	163.03	167.11	168.96
nnetSD_bc	149.87	156.37	160.81	160.05
nnetSD_bc_RL	150.99	161.16	163.69	164.90
nnetD	100.23	116.00	111.10	110.79
nnetD_RL	102.10	110.88	105.50	100.88
nnetD_bc	121.96	122.89	112.46	114.74
nnetD_bc_RL	120.83	123.94	114.09	111.83
nnetSTL	132.35	145.91	148.57	151.08
nnetSTL_RL	124.04	132.98	138.01	142.32
nnetSTL_bc	154.76	160.01	165.22	163.77
nnetSTL_bc_RL	148.14	157.89	162.04	162.01
svmradial	97.49	110.01	107.75	103.80
svmradial_RL	98.98	96.97	101.60	105.24
svmradial_bc	99.93	109.68	107.29	106.74
svmradial_bc_RL	99.84	97.86	99.72	103.19
svmradialSD	99.98	100.84	98.01	94.69
svmradialSD_RL	94.83	94.17	92.61	88.53
svmradialSD_bc	100.93	102.46	99.34	95.88
svmradialSD_bc_RL	95.20	95.69	93.58	87.76
svmradialD	98.60	107.58	105.86	103.57
svmradialD_RL	97.82	104.61	99.12	97.23
svmradialD_bc	102.52	109.67	103.86	103.77
svmradialD_bc_RL	97.63	105.06	99.79	96.06
svmradialSTL	100.99	104.47	101.22	99.39
svmradialSTL_RL	96.43	95.11	98.13	100.20
svmradialSTL_bc	101.72	101.29	100.05	97.59
svmradialSTL_bc_RL	97.27	91.80	95.96	99.11
svmpoly	99.05	106.66	107.76	108.18
svmpoly_RL	92.83	95.58	99.21	103.33
svmpoly_bc	102.02	104.18	104.22	105.05
svmpoly_bc_RL	94.11	94.80	99.59	102.30
svmpolySD	93.57	100.90	101.38	99.37
svmpolySD_RL	93.31	93.96	93.68	93.55
svmpolySD_bc	91.69	99.23	98.44	98.09
svmpolySD_bc_RL	93.16	89.09	90.57	88.27
svmpolyD	98.37	110.03	107.32	105.65
svmpolyD_RL	95.92	103.06	95.24	94.00
svmpolyD_bc	102.12	111.57	106.29	103.71
svmpolyD_bc_RL	96.39	103.74	94.39	91.98
svmpolySTL	95.18	95.88	94.42	97.95
svmpolySTL_RL	90.89	85.77	90.23	91.07
svmpolySTL_bc	97.28	98.52	97.01	97.29
svmpolySTL_bc_RL	91.71	85.23	90.70	92.64
gp	129.63	129.42	127.46	128.06
gp_RL	130.07	129.37	127.97	127.40
gp_bc	130.86	130.34	127.72	128.84
gp_bc_RL	131.16	129.89	128.44	129.11
gpSD	123.31	116.07	109.13	103.44
gpSD_RL	124.79	120.10	110.92	103.90
gpSD_bc	123.41	116.17	107.88	101.02
gpSD_bc_RL	121.28	115.44	106.20	99.66
gpD	123.91	126.35	117.26	116.98
gpD_RL	125.44	127.62	117.79	117.69
gpD_bc	124.63	125.53	118.97	117.86
gpD_bc_RL	126.52	126.71	117.36	117.56
gpSTL	119.63	114.38	109.20	109.14
gpSTL_RL	121.62	113.63	109.57	109.06
gpSTL_bc	120.67	112.68	108.38	109.91
gpSTL_bc_RL	122.66	113.45	109.22	109.84
rf	99.45	101.64	100.45	98.35
rf_RL	98.67	92.28	91.30	92.77
rf_bc	100.77	101.56	98.63	98.04
rf_bc_RL	98.46	93.31	92.17	93.83
rfSD	100.05	99.23	98.33	97.06
rfSD_RL	92.20	91.10	87.49	87.38
rfSD_bc	99.36	100.06	98.51	97.83
rfSD_bc_RL	94.17	91.43	88.03	88.78
rfD	101.03	106.53	102.46	101.95
rfD_RL	105.54	113.50	109.41	106.56
rfD_bc	100.40	102.33	99.57	98.43
rfD_bc_RL	106.07	113.11	108.20	105.67
rfSTL	98.99	91.97	90.13	87.51
rfSTL_RL	96.49	86.58	85.78	85.89
rfSTL_bc	100.96	93.87	88.80	88.67
rfSTL_bc_RL	98.72	85.89	85.10	86.86

Method	INDUSTRY h=1-1	INDUSTRY h=1-6	INDUSTRY h=1-12	INDUSTRY h=1-18
rfdetrend	93.90	95.90	101.48	101.95
rfdetrend_RL	97.38	99.56	102.12	103.21
rfdetrend_bc	92.03	94.80	98.33	101.32
rfdetrend_bc_RL	98.35	98.75	100.87	103.37
rfdetrendSD	93.24	94.23	100.52	101.30
rfdetrendSD_RL	92.08	91.70	95.30	95.67
rfdetrendSD_bc	92.02	94.34	98.73	101.15
rfdetrendSD_bc_RL	91.08	90.48	94.70	95.56
rfdetrendD	101.94	106.02	105.78	103.65
rfdetrendD_RL	106.92	106.97	110.87	108.71
rfdetrendD_bc	104.92	105.23	104.24	106.42
rfdetrendD_bc_RL	105.02	109.06	112.04	112.07
rfdetrendSTL	94.94	95.52	99.19	101.31
rfdetrendSTL_RL	98.76	93.93	97.38	96.96
rfdetrendSTL_bc	96.65	95.39	101.86	102.96
rfdetrendSTL_bc_RL	101.68	92.45	94.98	97.19
gbm	94.54	96.21	98.54	98.25
gbm_RL	92.36	91.61	94.55	93.21
gbm_bc	97.71	100.14	100.55	100.46
gbm_bc_RL	94.74	90.56	93.93	92.98
gbmSD	92.62	94.27	94.49	95.02
gbmSD_RL	94.92	88.23	85.72	87.30
gbmSD_bc	96.40	97.96	96.65	95.53
gbmSD_bc_RL	93.00	89.13	86.44	88.71
gbmD	103.53	110.98	108.77	109.64
gbmD_RL	102.62	109.89	100.98	100.05
gbmD_bc	101.43	111.85	107.29	109.18
gbmD_bc_RL	103.98	111.35	101.74	98.69
gbmSTL	98.04	93.72	91.08	88.75
gbmSTL_RL	94.47	88.56	87.90	87.07
gbmSTL_bc	101.54	95.58	92.10	92.67
gbmSTL_bc_RL	99.04	88.95	87.02	86.40
gbmdetrend	91.71	101.69	104.41	103.58
gbmdetrend_RL	97.36	101.40	101.03	104.55
gbmdetrend_bc	94.82	99.53	104.97	103.97
gbmdetrend_bc_RL	95.23	99.17	101.17	102.26
gbmdetrendSD	92.93	100.60	102.81	102.33
gbmdetrendSD_RL	92.40	95.53	95.47	98.18
gbmdetrendSD_bc	93.72	98.68	104.20	103.54
gbmdetrendSD_bc_RL	91.95	98.11	98.68	102.68
gbmdetrendD	106.86	105.74	104.40	104.50
gbmdetrendD_RL	105.39	104.11	108.48	107.21
gbmdetrendD_bc	103.26	105.23	104.77	105.63
gbmdetrendD_bc_RL	104.69	102.67	106.18	108.98
gbmdetrendSTL	95.85	96.65	99.22	99.73
gbmdetrendSTL_RL	99.81	92.56	96.44	98.47
gbmdetrendSTL_bc	99.16	98.32	100.58	100.98
gbmdetrendSTL_bc_RL	102.05	93.29	98.97	98.28
glmboost	90.06	86.78	93.05	95.69
glmboost_RL	92.88	94.17	96.89	98.65
glmboost_bc	89.33	85.37	95.49	96.71
glmboost_bc_RL	93.55	93.45	97.65	98.91
glmboostSD	86.27	83.11	90.71	90.24
glmboostSD_RL	95.67	86.90	88.90	87.10
glmboostSD_bc	**82.25**	81.51	90.53	91.34
glmboostSD_bc_RL	93.64	84.36	88.55	88.85
glmboostD	94.70	99.39	97.37	96.15
glmboostD_RL	99.37	103.90	96.29	94.48
glmboostD_bc	93.87	100.57	94.44	93.39
glmboostD_bc_RL	99.34	104.62	96.02	93.65
glmboostSTL	85.10	84.49	89.19	87.80
glmboostSTL_RL	92.81	86.46	91.22	89.01
glmboostSTL_bc	87.72	80.26	88.31	87.98
glmboostSTL_bc_RL	94.20	83.90	89.53	87.66
gamboost	96.93	98.31	106.89	110.47
gamboost_RL	99.28	95.91	104.28	107.14
gamboost_bc	96.86	98.19	104.12	109.26
gamboost_bc_RL	99.74	97.14	103.86	108.84
gamboostSD	92.24	93.62	102.81	106.64
gamboostSD_RL	91.75	91.59	100.39	104.16
gamboostSD_bc	91.29	92.95	100.65	105.58
gamboostSD_bc_RL	92.78	91.47	100.19	105.24
gamboostD	92.42	100.17	95.50	95.58
gamboostD_RL	97.62	103.45	95.24	93.22
gamboostD_bc	94.06	96.85	92.44	94.14
gamboostD_bc_RL	98.51	106.17	97.75	94.41
gamboostSTL	94.37	91.31	103.14	108.88
gamboostSTL_RL	95.68	93.47	107.03	113.61
gamboostSTL_bc	97.77	91.71	103.61	109.51
gamboostSTL_bc_RL	96.64	91.63	104.87	114.38
gamboostdetrend	93.14	97.39	103.08	103.57
gamboostdetrend_RL	96.61	94.93	99.80	99.58
gamboostdetrend_bc	91.84	98.39	101.82	102.55
gamboostdetrend_bc_RL	95.69	94.32	98.49	98.60
gamboostdetrendSD	92.42	95.25	101.88	102.83
gamboostdetrendSD_RL	94.40	92.63	97.41	98.35
gamboostdetrendSD_bc	91.32	96.93	100.81	102.34
gamboostdetrendSD_bc_RL	93.91	94.03	97.95	98.08
gamboostdetrendD	104.61	105.73	106.28	107.04
gamboostdetrendD_RL	101.48	102.97	106.86	104.75
gamboostdetrendD_bc	104.72	106.43	107.29	110.22
gamboostdetrendD_bc_RL	100.04	104.20	108.38	107.42
gamboostdetrendSTL	95.56	97.53	102.66	102.97
gamboostdetrendSTL_RL	95.58	91.95	95.95	97.89
gamboostdetrendSTL_bc	95.86	97.06	102.35	101.47
gamboostdetrendSTL_bc_RL	98.85	91.83	98.90	99.12

Table 5.1: *Average Ranked MdASE corresponding to dots in Figure 5.1. Best model metric per horizon range is bold & underlined and shading denote Top10% accordingly.*

- Detrending for Random Forests (*rf*), boosted trees (*gbm*) and boosted splines (*gamboost*) does not improve the forecasts.
- Gaussian Processes (*gp*) perform really bad especially in the long run.

To keep results for the M3 competition together, also some findings just shown in the Appendix for the other categories should be listed here. Keep in mind that except for the MICRO category all other time series are best or at least very satisfying forecasted with a naïve method making any conclusions regarding the model performances highly questionable as already discussed in Chapter 2.3.2 (cf. especially Figure 2.5):

- M3-DEMOGRAPHIC: Apart from the1-step-ahead forecast *tsnaiveSTL* is clearly the best method for all other horizon ranges.
- M3-FINANCE: In the long run *gamboostdetrendSD_RL* ends up as the winner.
- M3-MACRO: The detrended variants perform well but also *tsnaiveSTL* does, indicating that the decomposition is good in extracting the trend component and is therefore mainly responsible for the forecast for these time series.
- M3-MICRO: Gaussian Processes performs remarkably good with *gpSTL_bc* be the overall winner for the 1-season and the runner up for the half season and the competition horizon range. Interestingly ARIMA does not perform nice, actually even worse than ETS model for this category, which, together with the result for Gaussian Processes, indicates that the covariance structure is complicated and highly responsible for the outcome.

5.2 Tourism

For the Tourism time series some different findings pop up when examining Figure 5.2 and the corresponding Table 5.2:

- Generally Support Vector Machines (*svmpoly* and *svmradial*) together with boosted linear regression models (*glmboost*) are now competitive to classical methods. This confirms the result of the simulation study and indicates that many of the Tourism time series are similar to the phenotypic ones used in the simulation of Chapter 4.5.
- Moreover *svmpolySD_RL* is now the best overall method getting the first position starting from the half-season forecast horizon range on.
- ETS models are now inferior to ARIMA based ones. Especially the hard-differenced *arimaD* performs quite well and is by itself competitive to the classical bagging approaches (cf. also the findings in Chapter 3.5).
- Modeling seasonality solely with the lagged target information is clearly worse for the Machine Learning approaches at least in the long run.
- Box-Cox transformations have only a positive influence on the SVM in conjunction with a STL decomposition.
- Detrending the models that cannot model a future trend ends up with clearly worse forecast. This finding indicates that splitting the forecast into two models, i.e. ETS(A,A,N) for the trend and the Machine Learning approach for the rest, can be contra productive which might also result from the suboptimal decomposition. Actually a better detrending algorithm might help here.
- It seems that hard-differencing is a good initial step to tackle seasonality for these time series. At least in the long run only this deseasoning approach enables *gbm* and *gamboost* to beat the *tsnaiveD* model.

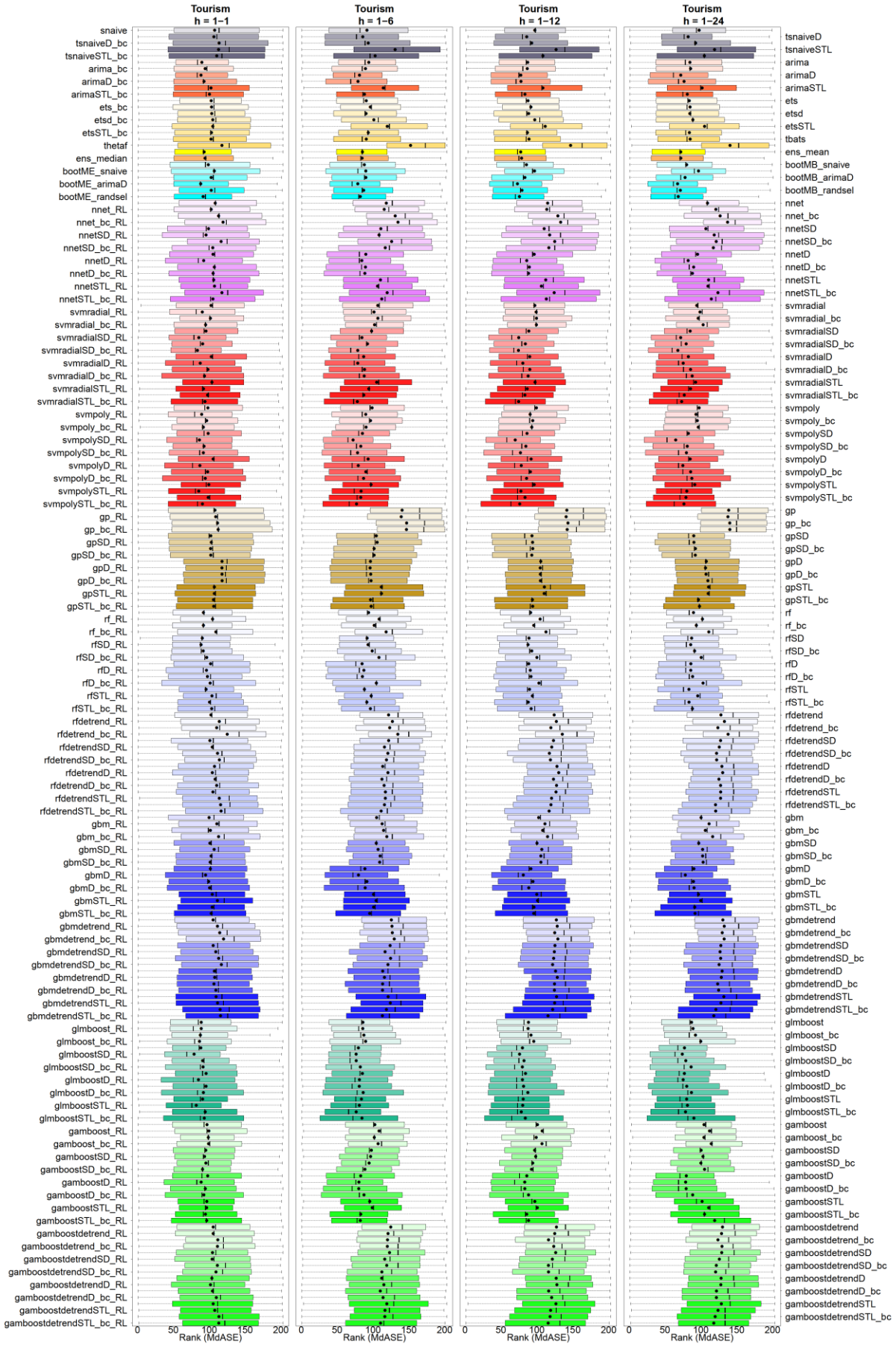

Figure 5.2: *Overall Result (Ranked MdASE) by horizon range for Tourism competition.*

Method	Tourism h=1-1	Tourism h=1-6	Tourism h=1-12	Tourism h=1-24
snaive	107.84	91.69	97.16	97.85
tsnaiveD	106.73	85.75	85.94	82.42
tsnaiveD_bc	114.05	93.45	92.88	92.95
tsnaiveSTL	113.28	130.88	126.91	119.36
tsnaiveSTL_bc	110.80	103.07	108.49	105.10
arima	89.92	94.16	87.21	84.93
arima_bc	94.64	89.58	86.60	85.85
arimaD	88.79	81.20	77.63	72.18
arimaD_bc	92.99	79.14	77.92	76.72
arimaSTL	102.57	114.37	108.40	101.95
arimaSTL_bc	100.58	88.10	83.33	81.20
ets	102.94	90.63	87.78	84.06
ets_bc	103.45	96.37	91.79	85.18
etsd	103.68	90.39	88.57	84.77
etsd_bc	105.22	101.21	97.08	89.01
etsSTL	105.19	119.95	111.68	105.38
etsSTL_bc	103.33	93.41	86.86	84.22
tbats	102.75	90.61	89.04	85.22
thetaf	117.92	152.45	147.06	140.89
ens_mean	93.15	85.33	77.47	72.14
ens_median	94.35	84.78	78.73	72.19
bootMB_snaive	99.02	88.04	85.48	80.42
bootME_snaive	107.29	90.06	96.84	96.96
bootMB_arimaD	102.98	90.23	83.39	78.18
bootME_arimaD	88.32	78.83	73.01	67.80
bootMB_randsel	103.11	87.20	79.34	71.70
bootME_randsel	91.80	82.35	75.82	68.72
nnet	108.54	118.59	115.10	109.51
nnet_RL	102.95	115.74	113.83	121.19
nnet_bc	113.31	131.10	129.31	127.09
nnet_bc_RL	119.48	134.98	133.16	137.40
nnetSD	99.18	111.00	110.28	107.19
nnetSD_RL	95.81	108.54	117.91	118.84
nnetSD_bc	116.86	126.05	124.91	121.66
nnetSD_bc_RL	105.11	117.20	116.79	117.66
nnetD	105.57	90.10	96.40	95.58
nnetD_RL	92.52	84.43	85.94	82.43
nnetD_bc	107.34	89.54	89.55	90.07
nnetD_bc_RL	105.60	88.70	88.50	88.47
nnetSTL	106.56	110.46	112.40	110.93
nnetSTL_RL	107.60	106.31	106.38	110.07
nnetSTL_bc	117.97	119.86	124.25	124.08
nnetSTL_bc_RL	105.35	112.31	113.25	114.88
svmradial	102.68	106.70	97.24	94.53
svmradial_RL	90.58	101.54	99.05	98.96
svmradial_bc	101.84	106.80	99.48	96.44
svmradial_bc_RL	95.15	102.19	99.39	103.37
svmradialSD	95.70	98.14	88.86	85.29
svmradialSD_RL	85.69	84.60	74.73	71.95
svmradialSD_bc	91.08	92.32	84.01	79.64
svmradialSD_bc_RL	83.73	78.79	74.37	68.11
svmradialD	103.13	86.99	90.00	82.70
svmradialD_RL	87.67	78.70	80.37	75.50
svmradialD_bc	98.38	88.21	90.14	85.70
svmradialD_bc_RL	93.62	87.80	87.87	88.05
svmradialSTL	103.92	105.10	97.42	92.84
svmradialSTL_RL	92.19	94.04	86.52	85.81
svmradialSTL_bc	97.89	87.08	83.19	76.95
svmradialSTL_bc_RL	94.21	78.16	74.59	73.38
svmpoly	98.33	98.86	99.38	97.75
svmpoly_RL	89.66	89.97	90.58	93.39
svmpoly_bc	96.63	96.62	94.28	95.27
svmpoly_bc_RL	92.09	90.41	92.90	96.54
svmpolySD	98.67	85.34	86.31	82.69
svmpolySD_RL	86.60	**72.28**	**69.89**	**65.07**
svmpolySD_bc	92.99	83.05	84.69	81.22
svmpolySD_bc_RL	91.97	78.54	77.09	79.99
svmpolyD	105.71	93.19	91.95	85.21
svmpolyD_RL	87.16	79.54	78.19	75.09
svmpolyD_bc	98.00	90.08	91.01	85.90
svmpolyD_bc_RL	94.65	86.88	85.62	87.62
svmpolySTL	99.59	97.33	96.28	91.60
svmpolySTL_RL	85.53	83.31	77.70	80.48
svmpolySTL_bc	98.75	82.83	83.41	79.28
svmpolySTL_bc_RL	90.67	77.07	76.05	76.56
gp	108.16	140.62	141.88	139.07
gp_RL	109.28	139.45	140.82	137.89
gp_bc	111.89	146.99	143.48	140.50
gp_bc_RL	112.68	146.74	142.38	140.64
gpSD	102.37	104.71	93.18	90.22
gpSD_RL	103.13	105.81	94.09	90.53
gpSD_bc	102.18	101.64	94.25	92.54
gpSD_bc_RL	102.35	101.58	92.93	92.43
gpD	118.12	96.48	105.53	107.73
gpD_RL	117.99	95.96	104.76	106.62
gpD_bc	118.11	96.89	104.88	107.41
gpD_bc_RL	118.19	97.34	105.00	109.94
gpSTL	107.33	111.68	110.44	110.89
gpSTL_RL	107.34	111.86	110.20	110.41
gpSTL_bc	106.08	96.68	93.78	96.85
gpSTL_bc_RL	106.53	97.45	94.22	98.33
rf	92.12	93.98	91.35	89.88
rf_RL	104.93	108.66	104.65	102.37
rf_bc	92.12	101.74	95.80	93.94
rf_bc_RL	109.48	118.44	112.76	111.31
rfSD	90.60	91.90	89.01	87.35
rfSD_RL	88.81	93.21	87.67	85.98
rfSD_bc	91.93	98.92	92.98	91.42
rfSD_bc_RL	96.73	108.36	100.13	100.73
rfD	102.08	84.88	88.65	86.25
rfD_RL	96.47	87.38	90.85	86.15
rfD_bc	97.73	85.25	91.86	88.66
rfD_bc_RL	101.46	104.76	102.92	103.32
rfSTL	95.94	88.16	89.79	83.56
rfSTL_RL	104.08	97.56	93.83	95.89
rfSTL_bc	100.67	91.52	88.15	83.72
rfSTL_bc_RL	103.64	96.55	92.12	88.37

Method	Tourism h=1-1	Tourism h=1-6	Tourism h=1-12	Tourism h=1-24
rfdetrend	102.57	121.77	124.15	128.17
rfdetrend_RL	114.23	127.19	127.24	133.07
rfdetrend_bc	110.65	123.43	119.55	123.98
rfdetrend_bc_RL	125.35	134.68	135.54	138.18
rfdetrendSD	101.11	121.90	123.20	127.75
rfdetrendSD_RL	104.47	116.07	120.78	125.89
rfdetrendSD_bc	111.80	120.83	117.54	122.16
rfdetrendSD_bc_RL	114.44	118.93	119.04	122.25
rfdetrendD	107.55	113.59	128.13	129.48
rfdetrendD_RL	104.54	120.90	130.58	130.61
rfdetrendD_bc	108.41	112.57	123.26	125.41
rfdetrendD_bc_RL	110.68	115.66	127.50	127.72
rfdetrendSTL	105.41	117.45	126.79	127.83
rfdetrendSTL_RL	114.35	117.33	120.61	127.76
rfdetrendSTL_bc	116.07	116.06	119.73	120.71
rfdetrendSTL_bc_RL	116.84	111.41	117.17	120.96
gbm	100.03	105.09	102.53	100.45
gbm_RL	110.87	112.79	111.37	111.70
gbm_bc	101.87	114.89	108.20	106.47
gbm_bc_RL	113.29	119.37	114.99	116.66
gbmSD	101.21	104.69	100.26	97.32
gbmSD_RL	107.14	107.25	106.88	102.67
gbmSD_bc	102.87	110.00	105.62	102.61
gbmSD_bc_RL	101.49	109.42	105.76	103.34
gbmD	101.93	88.87	91.74	90.44
gbmD_RL	95.23	79.86	81.13	78.59
gbmD_bc	99.50	91.67	94.64	89.93
gbmD_bc_RL	100.80	89.22	88.83	90.78
gbmSTL	104.50	100.82	100.06	96.99
gbmSTL_RL	111.77	104.58	101.09	100.13
gbmSTL_bc	104.42	101.05	96.08	91.66
gbmSTL_bc_RL	103.15	95.57	95.72	92.21
gbmdetrend	105.64	125.73	127.26	130.85
gbmdetrend_RL	111.72	126.52	128.52	132.76
gbmdetrend_bc	114.89	127.08	124.95	129.97
gbmdetrend_bc_RL	120.26	129.66	129.30	132.71
gbmdetrendSD	105.83	124.00	125.25	128.10
gbmdetrendSD_RL	109.46	116.93	124.45	127.04
gbmdetrendSD_bc	113.41	124.64	123.48	127.85
gbmdetrendSD_bc_RL	117.19	120.92	121.99	125.65
gbmdetrendD	107.50	113.67	126.17	129.96
gbmdetrendD_RL	107.99	116.14	128.57	128.76
gbmdetrendD_bc	106.85	113.57	124.72	123.74
gbmdetrendD_bc_RL	109.50	113.45	124.53	124.99
gbmdetrendSTL	109.96	121.37	127.67	132.28
gbmdetrendSTL_RL	111.99	124.37	124.99	128.26
gbmdetrendSTL_bc	115.56	118.87	122.12	121.28
gbmdetrendSTL_bc_RL	116.14	113.28	115.73	118.30
glmboost	89.10	86.04	88.27	87.22
glmboost_RL	89.15	85.78	87.85	89.27
glmboost_bc	87.87	87.36	91.91	92.71
glmboost_bc_RL	86.67	89.87	95.72	100.14
glmboostSD	88.49	79.72	79.92	77.31
glmboostSD_RL	**79.08**	76.52	75.90	74.26
glmboostSD_bc	91.01	76.67	82.14	79.23
glmboostSD_bc_RL	91.67	82.42	79.86	86.65
glmboostD	95.95	85.28	84.21	77.22
glmboostD_RL	85.26	81.23	79.94	75.59
glmboostD_bc	95.74	80.89	81.46	80.47
glmboostD_bc_RL	92.12	86.46	87.47	87.11
glmboostSTL	91.09	83.96	81.04	80.35
glmboostSTL_RL	82.15	81.11	80.36	81.46
glmboostSTL_bc	94.68	76.62	78.41	78.83
glmboostSTL_bc_RL	93.36	84.69	84.01	90.50
gamboost	97.14	102.59	100.90	104.63
gamboost_RL	99.64	108.55	107.40	112.22
gamboost_bc	99.02	101.91	99.11	104.43
gamboost_bc_RL	99.31	107.16	107.20	114.76
gamboostSD	95.40	97.82	96.92	100.54
gamboostSD_RL	93.68	96.74	98.48	103.18
gamboostSD_bc	95.58	94.65	94.23	100.21
gamboostSD_bc_RL	90.90	88.87	92.59	105.27
gamboostD	98.06	82.93	86.07	79.88
gamboostD_RL	89.03	80.50	83.09	78.95
gamboostD_bc	94.81	80.19	83.22	79.61
gamboostD_bc_RL	92.75	87.49	88.49	88.79
gamboostSTL	96.94	95.55	97.17	101.91
gamboostSTL_RL	96.69	99.03	100.74	111.11
gamboostSTL_bc	94.89	82.97	85.85	105.35
gamboostSTL_bc_RL	96.79	82.40	88.47	119.47
gamboostdetrend	105.80	124.84	127.42	130.37
gamboostdetrend_RL	105.79	121.12	124.97	128.99
gamboostdetrend_bc	112.24	120.33	116.25	124.12
gamboostdetrend_bc_RL	111.72	120.15	123.55	129.69
gamboostdetrendSD	104.85	123.16	126.34	129.31
gamboostdetrendSD_RL	104.37	116.45	121.65	126.01
gamboostdetrendSD_bc	111.62	119.25	116.12	121.75
gamboostdetrendSD_bc_RL	109.70	112.50	116.28	120.77
gamboostdetrendD	103.97	112.18	126.97	128.47
gamboostdetrendD_RL	102.06	115.28	127.97	128.14
gamboostdetrendD_bc	104.84	110.26	116.55	121.95
gamboostdetrendD_bc_RL	110.11	114.19	120.70	121.89
gamboostdetrendSTL	105.74	119.41	126.68	128.77
gamboostdetrendSTL_RL	107.97	116.84	119.27	124.20
gamboostdetrendSTL_bc	113.80	116.63	118.44	120.22
gamboostdetrendSTL_bc_RL	113.56	110.57	115.84	118.17

Table 5.2: *Average Ranked MdASE corresponding to dots in Figure 5.2. Best model metric per horizon range is bold & underlined and shading denote Top10% accordingly.*

5.3 NN5

The NN5 benchmark datasets comprise most time related information compared to the Tourism and M3 data. This can be seen in Figure 5.3 and Table 5.3 by the low performance of the naïve forecasts which in turn means that the other algorithms exploit the time information more fine grained.

The following findings (apart from covariate effects and the sRecDir forecast strategy presented later) can be further identified in Figure 5.3 and Table 5.3:

- Naive forecast, including STL variants, are definitely worse which indicates that the data comprises more information than just season and trend.
- Classical models are now clearly inferior to some Machine Learning approaches.
- Even though hard-differencing the seasonality is inevitable for the ARIMA models here, it is clearly the worst deseasoning method for the Machine Learning models.
- Using just the lagged target value information for modeling seasonality results in drastically bad forecasts. The clearly reduced performance when even reducing the lagset indicates that further lags beside the most recent and the first seasonal one comprise prediction relevant information.
- Again, detrending in advance makes it even worse for *rf*, *gbm* and *gamboost* models, see also the explanations in previous finding list for Tourism benchmark.
- SVM models are now the best models (apart from the 1-step-ahead forecast, see following comment) without a big difference between a radial and a polynomial kernel.
- Even though *gamboostSTL_RL_bc* is best 1-step-ahead *glmboost* is clearly better in the long run which confirms again the simulation results from Chapter 4.5.
- Again, Gaussian Processes are not competitive.

The best models for the competition horizon range are further tested adding the covariate set discussed above. In fact the tested models are: *nnetSD, svmradialSD, svmradialSD_RL, svmpolySD, svmpolySD_RL, svmradialSD_bc_RL, svmpolySD_bc_RL, rfSD, gbmSD, glmboostSTL_bc, glmboostSTL_bc_RL*. Also a sRecDir variant is processed. Main findings regarding these alternatives are:

- For *arimaD* the covariates are internally processed as a linear model. Actually using the covariates makes forecasting worse even though *weekBefEaster* should have a positive influence on the prediction (cf. Chapter 2.1).
- For the other models the effect of adding the covariates differs. All in all the change compared to models without any covariates is just slight. But actually if investigating the sMAPE which is also the competition metric, a main change can be identified for *svmradialSD_COVARS*. The final value of sMAPE=19.6% clearly beats the best models used in the official competition. As can be seen on Crone (2009b) the top-5 performing teams get a sMAPE of 19.9%, 20.4%, 20.5%, 20.6%, 21.1%; see also the explanation to the top-2 models in Chapter 2.1, i.e. Wildi (2010) and Andrawis et al. (2011). Furthermore Taieb et al. (2012c) used this data and end up with a sMAPE of 20.3% . They applied and averaged different forecasting strategies (recursive, direct and combinations) using a nearest neighbor prediction algorithm and utilize the deseasoning approach from Andrawis et al. (2011). Even though the SVM models showed superior performance throughout the investigations of this thesis and is therefore a top candidate, it must be admitted that for a completely honest comparison with the old competition results the final selection of the best model had to be based solely on the training set. Anyway this result is still remarkable!
- The sRECDIR does not improve the forecasts; even not for the first season where this strategy equals a Direct forecast approach. This finding is in line with the well-known experience that direct strategies are often worse and the question for the best forecast strategy is often an empirical one, i.e. depends on the investigated time series.

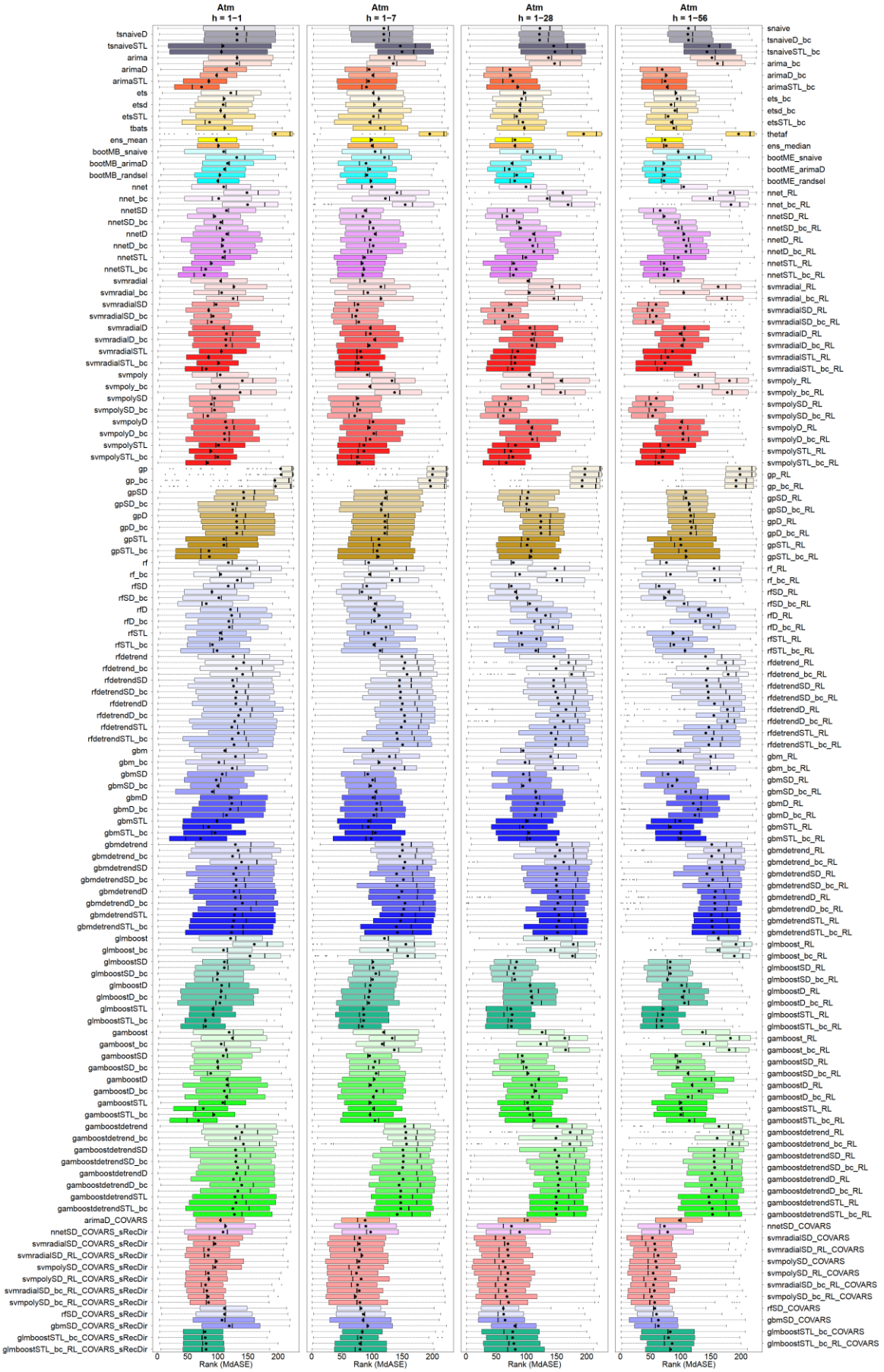

Figure 5.3: *Overall Result (Ranked MdASE) by horizon range for NN5 competition.*

Method	Atm h=1-1	Atm h=1-7	Atm h=1-28	Atm h=1-56
snaive	130.86	119.59	122.49	112.99
tsnaiveD	132.17	119.50	121.55	111.96
tsnaiveD_bc	131.98	119.73	122.04	113.14
tsnaiveSTL	108.85	146.78	145.34	147.02
tsnaiveSTL_bc	106.15	149.86	147.38	143.59
arima	132.13	128.59	136.76	151.75
arima_bc	131.69	135.26	146.78	160.89
arimaD	112.63	95.34	72.69	69.41
arimaD_bc	98.08	101.23	74.26	76.08
arimaSTL	85.36	94.25	77.59	74.26
arimaSTL_bc	73.21	90.75	85.63	77.67
ets	121.71	102.04	97.66	92.16
ets_bc	110.59	111.32	92.11	94.18
etsd	108.97	103.95	89.68	84.08
etsd_bc	104.88	112.58	88.87	90.44
etsSTL	111.00	102.36	83.95	79.27
etsSTL_bc	86.41	96.09	94.19	84.58
tbats	111.58	114.10	96.69	88.70
thetaf	195.35	195.56	194.83	196.03
ens_mean	97.53	99.41	81.05	74.01
ens_median	101.21	100.12	81.17	76.13
bootMB_snaive	109.69	105.09	101.51	95.97
bootME_snaive	131.43	120.86	123.25	113.62
bootMB_arimaD	116.15	90.05	76.14	72.19
bootME_arimaD	111.13	96.41	71.73	69.57
bootMB_randsel	103.57	91.68	84.12	73.42
bootME_randsel	100.34	98.14	80.62	72.35
nnet	109.89	99.33	99.43	105.04
nnet_RL	148.48	141.20	160.29	181.86
nnet_bc	101.68	122.14	134.50	148.27
nnet_bc_RL	149.43	154.89	169.21	183.33
nnetSD	114.38	88.59	78.99	65.95
nnetSD_RL	95.15	84.87	67.86	72.72
nnetSD_bc	105.05	96.87	87.78	91.52
nnetSD_bc_RL	103.60	101.90	90.75	96.10
nnetD	114.95	105.07	111.78	105.43
nnetD_RL	108.85	97.01	105.88	105.22
nnetD_bc	107.99	101.99	110.60	109.22
nnetD_bc_RL	111.51	98.60	112.59	108.74
nnetSTL	108.32	87.53	99.13	95.87
nnetSTL_RL	89.50	83.73	79.19	72.51
nnetSTL_bc	80.02	86.48	83.16	77.41
nnetSTL_bc_RL	77.61	84.71	78.44	72.58
svmradial	104.88	87.82	102.53	95.90
svmradial_RL	126.72	114.66	142.44	162.17
svmradial_bc	106.54	92.95	105.12	105.08
svmradial_bc_RL	125.54	114.75	145.67	168.50
svmradialSD	97.40	76.62	74.82	59.16
svmradialSD_RL	85.01	75.03	**61.31**	53.18
svmradialSD_bc	92.28	73.09	76.84	59.65
svmradialSD_bc_RL	89.51	78.04	64.54	54.43
svmradialD	109.41	97.08	105.77	106.17
svmradialD_RL	114.61	97.06	110.23	98.96
svmradialD_bc	113.09	104.02	108.41	105.59
svmradialD_bc_RL	113.68	95.33	109.58	101.78
svmradialSTL	105.86	80.61	85.68	86.20
svmradialSTL_RL	84.64	82.22	81.16	78.79
svmradialSTL_bc	101.36	76.87	81.11	74.00
svmradialSTL_bc_RL	80.83	77.14	76.67	68.14
svmpoly	104.14	92.17	105.04	123.79
svmpoly_RL	140.84	133.05	157.37	180.77
svmpoly_bc	103.05	96.18	103.77	129.54
svmpoly_bc_RL	137.09	137.34	156.74	177.34
svmpolySD	94.80	76.41	74.22	59.76
svmpolySD_RL	89.32	76.72	65.26	**50.28**
svmpolySD_bc	94.63	79.89	73.35	58.40
svmpolySD_bc_RL	83.56	**71.21**	61.82	53.54
svmpolyD	112.72	101.52	102.97	101.46
svmpolyD_RL	114.50	95.51	109.05	99.50
svmpolyD_bc	111.09	102.78	106.06	103.65
svmpolyD_bc_RL	111.35	96.72	109.81	103.74
svmpolySTL	101.32	86.41	82.00	79.42
svmpolySTL_RL	88.58	86.40	74.32	71.87
svmpolySTL_bc	99.78	75.80	77.50	70.05
svmpolySTL_bc_RL	83.11	78.67	66.37	63.67
gp	204.18	200.86	197.07	197.72
gp_RL	204.96	200.05	196.85	198.35
gp_bc	194.70	195.76	192.48	191.67
gp_bc_RL	195.63	196.99	192.27	191.76
gpSD	142.69	123.35	103.21	108.72
gpSD_RL	143.12	121.82	101.47	108.86
gpSD_bc	124.72	115.51	100.44	114.41
gpSD_bc_RL	125.06	115.15	104.26	114.92
gpD	131.54	121.74	123.25	116.76
gpD_RL	131.16	121.26	124.25	116.05
gpD_bc	131.73	121.64	123.34	117.91
gpD_bc_RL	131.28	121.13	124.45	116.65
gpSTL	109.80	111.22	102.56	99.50
gpSTL_RL	110.10	111.77	101.16	100.50
gpSTL_bc	85.43	108.41	107.91	108.77
gpSTL_bc_RL	86.21	110.19	107.50	109.09
rf	117.74	94.51	78.55	76.44
rf_RL	148.26	140.27	147.56	155.31
rf_bc	104.97	95.73	88.86	83.10
rf_bc_RL	132.56	133.15	150.55	156.55
rfSD	117.34	91.20	75.08	64.27
rfSD_RL	90.42	83.31	81.88	80.01
rfSD_bc	101.92	97.87	84.66	74.30
rfSD_bc_RL	80.96	105.02	105.56	105.74
rfD	121.20	103.29	116.79	129.55
rfD_RL	123.09	111.92	131.72	145.44
rfD_bc	118.40	103.89	113.33	124.58
rfD_bc_RL	119.22	123.16	143.82	154.99
rfSTL	104.06	94.05	91.50	87.53
rfSTL_RL	106.77	115.94	117.05	104.36
rfSTL_bc	91.19	102.93	93.04	88.63
rfSTL_bc_RL	99.04	113.41	115.91	107.24

Method	Atm h=1-1	Atm h=1-7	Atm h=1-28	Atm h=1-56
rfdetrend	125.53	149.35	145.86	141.19
rfdetrend_RL	143.04	154.47	170.00	174.20
rfdetrend_bc	130.98	152.51	149.21	144.74
rfdetrend_bc_RL	141.24	158.08	175.05	178.75
rfdetrendSD	124.56	145.70	145.30	142.27
rfdetrendSD_RL	126.23	145.18	144.98	143.98
rfdetrendSD_bc	131.42	147.19	148.58	145.85
rfdetrendSD_bc_RL	129.69	146.78	151.69	145.79
rfdetrendD	130.10	150.27	153.73	155.94
rfdetrendD_RL	137.73	148.51	165.58	177.36
rfdetrendD_bc	134.51	153.70	152.00	154.97
rfdetrendD_bc_RL	128.08	154.18	161.49	177.24
rfdetrendSTL	123.42	148.47	147.47	146.64
rfdetrendSTL_RL	134.08	140.86	141.12	141.70
rfdetrendSTL_bc	124.00	142.07	148.87	152.40
rfdetrendSTL_bc_RL	127.14	150.73	148.25	146.63
gbm	111.58	101.94	94.68	95.74
gbm_RL	129.42	128.79	140.30	149.71
gbm_bc	101.93	111.87	97.96	99.17
gbm_bc_RL	124.11	137.07	147.51	149.91
gbmSD	107.54	93.05	95.01	79.23
gbmSD_RL	97.57	100.86	105.70	94.12
gbmSD_bc	100.04	97.76	94.68	86.23
gbmSD_bc_RL	91.02	106.27	115.59	108.41
gbmD	122.62	101.56	116.36	133.59
gbmD_RL	123.30	107.95	118.63	120.92
gbmD_bc	120.75	106.93	116.79	128.95
gbmD_bc_RL	114.65	102.94	114.14	123.64
gbmSTL	98.87	94.30	101.71	98.78
gbmSTL_RL	84.94	93.95	94.12	83.09
gbmSTL_bc	95.69	105.50	104.04	99.81
gbmSTL_bc_RL	71.63	98.60	105.71	99.84
gbmdetrend	129.28	150.21	150.55	151.53
gbmdetrend_RL	133.86	150.41	155.36	163.10
gbmdetrend_bc	124.61	145.91	147.75	151.44
gbmdetrend_bc_RL	139.47	154.54	161.97	168.15
gbmdetrendSD	130.23	152.31	147.88	147.90
gbmdetrendSD_RL	126.15	140.74	150.85	143.55
gbmdetrendSD_bc	130.55	151.94	149.27	152.94
gbmdetrendSD_bc_RL	130.31	141.62	149.79	146.55
gbmdetrendD	126.68	149.31	151.77	157.52
gbmdetrendD_RL	129.27	144.28	156.07	156.06
gbmdetrendD_bc	140.95	154.73	152.17	156.55
gbmdetrendD_bc_RL	129.29	152.58	147.23	156.78
gbmdetrendSTL	127.50	149.51	154.05	150.83
gbmdetrendSTL_RL	126.55	147.36	153.90	151.68
gbmdetrendSTL_bc	124.25	140.24	150.47	152.33
gbmdetrendSTL_bc_RL	122.81	142.10	151.72	154.38
glmboost	121.30	120.73	133.34	162.44
glmboost_RL	160.65	156.04	177.77	191.57
glmboost_bc	109.57	126.04	140.21	161.72
glmboost_bc_RL	153.41	159.10	176.50	188.84
glmboostSD	111.05	101.32	83.87	82.78
glmboostSD_RL	110.95	101.76	81.69	82.32
glmboostSD_bc	99.98	106.30	79.22	83.05
glmboostSD_bc_RL	99.30	100.87	80.80	78.24
glmboostD	106.51	97.07	106.26	101.57
glmboostD_RL	105.94	96.31	107.79	105.91
glmboostD_bc	104.83	94.48	108.45	102.47
glmboostD_bc_RL	103.59	94.62	109.59	106.33
glmboostSTL	92.06	85.86	74.41	71.19
glmboostSTL_RL	92.48	86.30	76.38	70.31
glmboostSTL_bc	79.63	86.12	75.26	68.07
glmboostSTL_bc_RL	79.63	83.72	75.29	69.30
gamboost	118.99	119.04	126.19	136.18
gamboost_RL	124.34	133.49	163.70	182.62
gamboost_bc	105.78	116.80	123.70	138.43
gamboost_bc_RL	114.08	136.84	165.08	180.35
gamboostSD	109.15	96.05	93.23	93.59
gamboostSD_RL	99.77	104.85	95.07	99.02
gamboostSD_bc	100.15	102.27	99.99	95.53
gamboostSD_bc_RL	88.32	106.84	102.69	112.17
gamboostD	115.37	103.54	120.64	140.97
gamboostD_RL	115.40	97.54	108.85	118.78
gamboostD_bc	110.84	107.41	115.79	130.52
gamboostD_bc_RL	114.46	102.07	110.18	112.34
gamboostSTL	108.41	97.14	101.89	100.32
gamboostSTL_RL	76.17	102.59	102.03	100.20
gamboostSTL_bc	93.64	96.62	105.83	100.16
gamboostSTL_bc_RL	**68.20**	104.76	112.85	114.47
gamboostdetrend	131.98	154.42	151.25	163.46
gamboostdetrend_RL	140.07	156.17	172.61	187.28
gamboostdetrend_bc	129.64	155.69	148.99	160.55
gamboostdetrend_bc_RL	142.55	157.14	172.00	185.48
gamboostdetrendSD	129.40	151.14	147.28	155.55
gamboostdetrendSD_RL	130.77	151.97	153.68	155.41
gamboostdetrendSD_bc	129.87	151.75	150.33	155.48
gamboostdetrendSD_bc_RL	132.92	150.79	151.06	156.27
gamboostdetrendD	131.14	144.93	152.68	152.06
gamboostdetrendD_RL	126.09	151.40	155.43	157.26
gamboostdetrendD_bc	139.77	144.73	152.81	150.87
gamboostdetrendD_bc_RL	133.22	147.66	153.42	158.95
gamboostdetrendSTL	128.57	146.86	149.05	146.69
gamboostdetrendSTL_RL	130.27	147.61	150.32	147.58
gamboostdetrendSTL_bc	125.00	147.54	148.53	152.31
gamboostdetrendSTL_bc_RL	127.68	141.77	149.98	152.79
arimaD_COVARS	104.38	88.64	101.70	97.97
nnetSD_COVARS	112.95	89.79	75.39	73.07
nnetSD_COVARS_sRecDir	108.88	97.71	88.88	78.53
svmradialSD_COVARS	94.68	79.41	63.04	53.27
svmradialSD_COVARS_sRecDir	95.62	78.76	69.57	56.95
svmradialSD_RL_COVARS	84.83	79.43	68.97	57.85
svmradialSD_RL_COVARS_sRecDir	84.18	83.13	69.95	62.66
svmpolySD_COVARS	97.86	78.47	61.51	58.73
svmpolySD_COVARS_sRecDir	95.51	77.86	64.95	60.29
svmpolySD_RL_COVARS	84.35	74.09	69.33	54.32
svmpolySD_RL_COVARS_sRecDir	85.21	81.93	69.34	57.96
svmradialSD_bc_RL_COVARS	79.82	75.97	65.73	54.93
svmradialSD_bc_RL_COVARS_sRecDir	81.88	78.83	65.63	56.48
svmpolySD_bc_RL_COVARS	84.13	72.54	67.95	51.93
svmpolySD_bc_RL_COVARS_sRecDir	84.69	79.94	70.50	56.96
rfSD_COVARS	111.04	81.86	61.91	57.39
rfSD_COVARS_sRecDir	111.19	85.45	62.43	60.05
gbmSD_COVARS	107.04	85.22	64.85	63.10
gbmSD_COVARS_sRecDir	119.14	92.96	81.00	63.11
glmboostSTL_bc_COVARS	78.90	83.81	77.41	82.98
glmboostSTL_bc_COVARS_sRecDir	79.45	81.60	77.32	79.92
glmboostSTL_bc_RL_COVARS	80.67	80.10	79.43	85.87
glmboostSTL_bc_RL_COVARS_sRecDir	80.75	80.14	81.74	83.43

Table 5.3: *Average Ranked MdASE corresponding to dots in Figure 5.3. Best model metric per horizon range is bold & underlined and shading denote Top10% accordingly.*

Last but not least above results suggest testing some additional models explained in the following. But results were not promising and are therefore excluded from this thesis. Though, for completeness, this finding is important to be mentioned:

- Due to the limitation of tree based models to forecast a trend, a MARS (multivariate adaptive regression splines) model might help to circumvent this limitation as in its basis version is very similar to a CART but is able to model a future trend (see Hastie et al. (2009)).
- In order to account for moving average components a recurrent neural net utilizing the *jordan* function of the *RSNNS* R-package (Bergmeir & Benitez (2014)) was used as an alternative to the standard neural net.
- The good performance of SVM bases models and the *glmboost* suggests boosting the *svmradial* or *svmpoly* model. When using the quadratic loss as the internal optimization criterion boosting can be easily implemented as a recursive fitting of the residuals (cf. Chapter 4.4).
- A somewhat similar intention is behind an approach that applies a *gbm* model just on the residuals of a *glmboost* algorithm.
- For the classical ARIMA a possible approach to account for multiple seasonality is to strip of both *weekday* and *monthday* seasonality of the NN5 data by a double usage of the STL decomposition.

5.4 Arimasim

The last benchmark results to present refer to the simulated Arima series introduced in Chapter 2.1.

Figure 5.4 and Table 5.4 reveal the following findings:

- Not very surprising, *arima* is the best overall model even though not the winner in the long run.
- Also ETS based models are capable of creating a good forecast for ARIMA time series.
- Both classical model classes are clearly better than the ML approaches. But *glmboost* for shorter horizon ranges based models and remarkably *gbm* and *rf* in the long run are competitive.
- Most remarkable is that the bagging for classical models performs drastically bad for the 1-step-ahead forecast. This indicates that the ARMA structure of the series is seriously destroyed when bootstrapping by moving block or maximum entropy approaches!
- Somewhat surprising is the catastrophic 1-step-ahead performance of Gaussian Processes. This might be a result of the MA component of the series which kind of puzzles this algorithm.

5.5 Conclusions

All above findings together with some results presented in previous chapters can be condensed to some Lessons learned:

- Always apply several (!) naïve models in order to get a better evaluation of the forecast capability of more advanced models for the examined time series. Actually it also helps to get an impression of the forecastability of the data itself.
- Be aware of the complications different performance metrics and performance aggregation approaches reveal for time series benchmarks (cf. Chapter 2.3). Generally the no-free-lunch theorem must be enlarged regarding forecast horizons which mean that algorithm performances can drastically change when a different horizon range is investigated. These circumstances recommend clearly defining the forecast objective in terms of the evaluation metric (e.g. robust or outlier-sensitive) and the forecasting horizon.

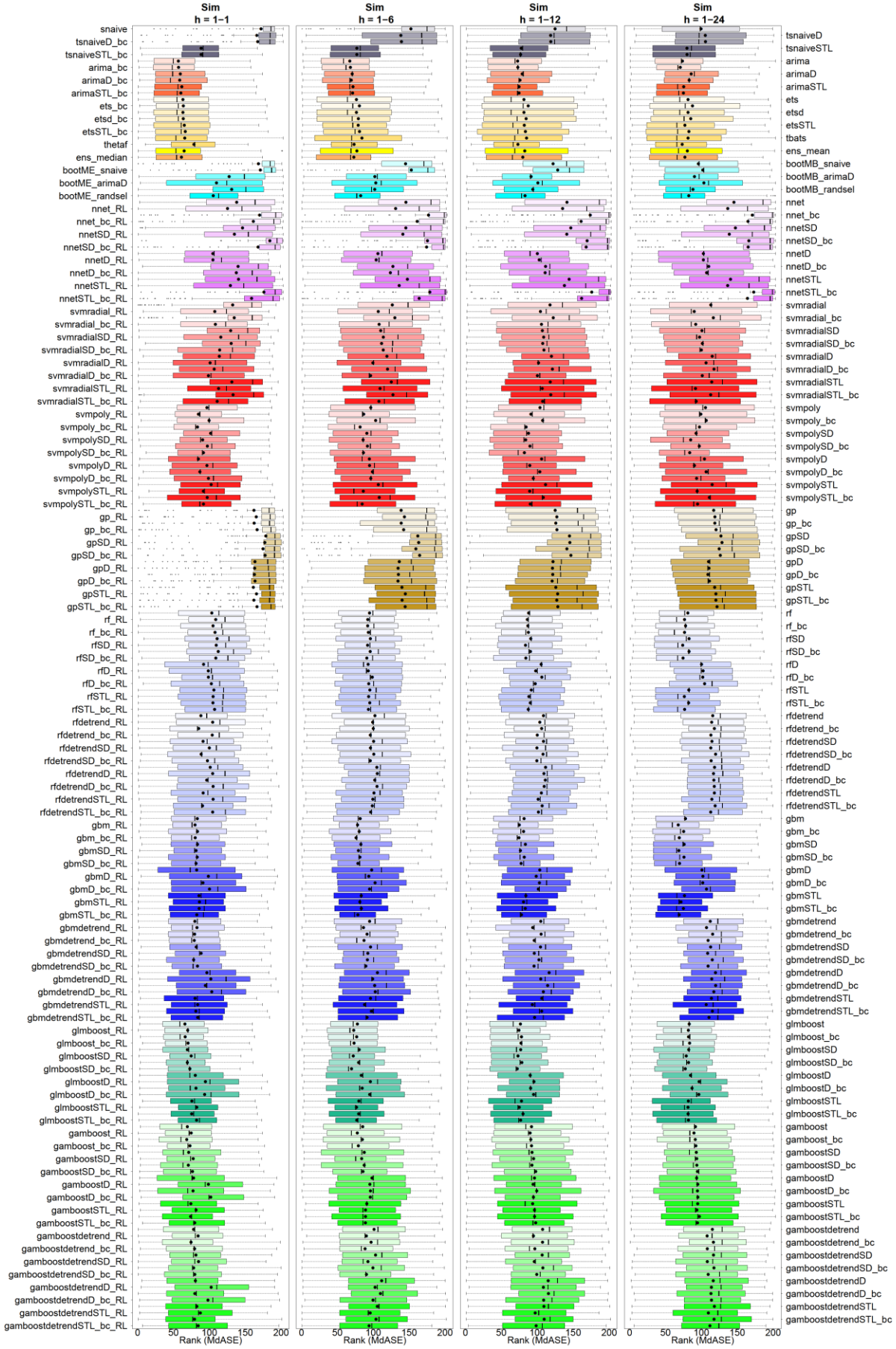

Figure 5.4: *Overall Result (Ranked MdASE) by horizon range for Arimasim competition.*

Method	Sim h=1-1	Sim h=1-6	Sim h=1-12	Sim h=1-24
snaive	172.17	152.47	124.87	99.64
tsnaiveD	166.31	138.45	118.64	105.53
tsnaiveD_bc	167.53	139.28	118.83	105.88
tsnaiveSTL	88.79	77.22	77.74	80.04
tsnaiveSTL_bc	88.71	77.21	76.73	80.14
arima	**56.96**	**67.32**	72.60	73.70
arima_bc	57.27	68.20	72.92	70.72
arimaD	59.73	70.59	78.98	85.92
arimaD_bc	58.77	69.37	75.96	83.03
arimaSTL	61.73	71.69	74.61	75.44
arimaSTL_bc	60.40	71.25	74.62	74.99
ets	63.37	76.53	81.42	81.47
ets_bc	63.87	80.75	87.40	87.73
etsd	63.37	76.53	81.42	81.47
etsd_bc	63.84	79.28	84.41	85.28
etsSTL	65.32	78.99	81.84	77.12
etsSTL_bc	66.61	80.75	83.44	82.72
tbats	65.92	84.20	84.99	81.45
thetaf	78.68	73.32	72.99	73.32
ens_mean	65.08	77.17	82.48	80.50
ens_median	61.26	72.94	79.84	77.11
bootMB_snaive	168.51	145.02	122.06	95.98
bootME_snaive	171.25	153.05	128.43	102.00
bootMB_arimaD	127.75	102.27	91.42	90.51
bootME_arimaD	110.10	103.52	100.80	103.62
bootMB_randsel	131.33	102.04	93.75	88.47
bootME_randsel	105.66	82.47	82.96	82.55
nnet	138.21	145.56	141.37	145.45
nnet_RL	125.44	131.12	135.30	136.88
nnet_bc	170.20	177.07	173.81	171.10
nnet_bc_RL	161.51	161.19	161.09	165.00
nnetSD	146.40	145.44	146.74	147.57
nnetSD_RL	135.08	141.21	141.12	138.97
nnetSD_bc	184.41	175.55	169.49	166.31
nnetSD_bc_RL	168.07	174.32	168.06	165.26
nnetD	105.09	106.48	100.13	102.86
nnetD_RL	105.21	104.02	102.71	102.88
nnetD_bc	140.17	127.41	111.62	110.28
nnetD_bc_RL	137.82	123.85	110.96	107.45
nnetSTL	140.74	147.60	144.40	140.71
nnetSTL_RL	129.50	136.20	137.78	136.77
nnetSTL_bc	176.18	178.82	175.89	173.05
nnetSTL_bc_RL	159.04	164.00	161.74	164.36
svmradial	132.70	126.50	117.77	113.30
svmradial_RL	107.76	107.00	104.35	90.12
svmradial_bc	134.85	130.12	122.30	116.73
svmradial_bc_RL	108.61	108.17	105.82	92.15
svmradialSD	129.85	110.39	107.06	100.82
svmradialSD_RL	116.07	114.03	108.93	97.50
svmradialSD_bc	130.60	111.56	107.60	101.48
svmradialSD_bc_RL	114.54	113.87	109.23	100.06
svmradialD	114.07	118.97	119.70	115.15
svmradialD_RL	101.19	99.45	101.68	106.53
svmradialD_bc	106.84	119.85	121.01	117.62
svmradialD_bc_RL	98.88	96.28	100.12	101.37
svmradialSTL	131.45	125.03	118.04	114.46
svmradialSTL_RL	112.96	109.43	106.82	91.86
svmradialSTL_bc	133.18	127.36	118.75	112.95
svmradialSTL_bc_RL	111.01	107.85	107.38	92.77
svmpoly	96.76	96.55	103.74	105.80
svmpoly_RL	84.94	86.62	91.07	99.40
svmpoly_bc	99.87	103.30	107.46	106.88
svmpoly_bc_RL	83.64	81.64	84.38	97.40
svmpolySD	101.92	91.09	87.78	92.72
svmpolySD_RL	90.79	85.80	83.86	85.11
svmpolySD_bc	97.39	91.87	89.80	97.20
svmpolySD_bc_RL	91.96	86.25	81.97	83.83
svmpolyD	84.83	94.52	106.02	104.31
svmpolyD_RL	96.82	94.81	89.56	90.42
svmpolyD_bc	86.94	98.78	103.49	106.68
svmpolyD_bc_RL	98.80	96.54	94.40	93.44
svmpolySTL	102.62	106.99	111.50	115.05
svmpolySTL_RL	91.82	86.03	89.32	94.12
svmpolySTL_bc	97.09	108.36	108.31	111.21
svmpolySTL_bc_RL	91.90	84.45	90.19	94.77
gp	162.27	138.84	125.00	117.30
gp_RL	165.49	143.40	127.30	118.96
gp_bc	162.60	138.98	125.78	118.79
gp_bc_RL	166.74	142.41	127.34	120.55
gpSD	178.83	162.06	144.93	127.38
gpSD_RL	177.45	163.25	145.37	128.79
gpSD_bc	174.93	159.38	141.29	124.93
gpSD_bc_RL	177.85	164.61	146.86	126.45
gpD	163.62	136.28	121.91	110.09
gpD_RL	163.25	134.53	121.20	110.67
gpD_bc	162.94	135.08	121.53	109.20
gpD_bc_RL	163.83	134.47	120.66	111.33
gpSTL	161.48	139.98	125.80	118.59
gpSTL_RL	165.88	144.63	128.31	120.33
gpSTL_bc	162.15	140.22	127.15	120.31
gpSTL_bc_RL	166.61	144.38	128.60	121.87
rf	103.58	94.77	88.02	81.24
rf_RL	109.04	92.42	86.12	76.63
rf_bc	105.34	92.19	86.88	78.04
rf_bc_RL	108.14	93.27	87.57	76.59
rfSD	111.06	96.01	90.84	82.95
rfSD_RL	109.81	92.11	83.64	74.12
rfSD_bc	112.53	95.72	90.54	82.95
rfSD_bc_RL	109.14	91.09	84.02	74.54
rfD	92.12	92.90	105.35	100.31
rfD_RL	98.66	93.06	97.85	102.54
rfD_bc	98.88	98.66	106.29	102.06
rfD_bc_RL	102.98	93.72	97.19	104.64
rfSTL	106.48	95.27	91.70	82.65
rfSTL_RL	105.45	93.61	88.33	76.40
rfSTL_bc	105.19	95.18	90.28	82.32
rfSTL_bc_RL	107.45	93.32	87.49	76.74

Method	Sim h=1-1	Sim h=1-6	Sim h=1-12	Sim h=1-24
rfdetrend	88.42	102.39	108.69	115.96
rfdetrend_RL	104.82	99.16	103.59	114.46
rfdetrend_bc	85.30	99.99	106.12	118.22
rfdetrend_bc_RL	104.16	95.87	100.06	113.36
rfdetrendSD	91.39	100.46	108.69	114.84
rfdetrendSD_RL	100.13	96.16	99.57	113.27
rfdetrendSD_bc	88.69	100.75	107.99	119.96
rfdetrendSD_bc_RL	97.51	95.85	99.37	113.74
rfdetrendD	101.44	104.97	111.59	118.49
rfdetrendD_RL	104.78	105.64	109.21	116.97
rfdetrendD_bc	96.47	102.06	111.22	117.54
rfdetrendD_bc_RL	105.45	105.26	109.62	118.08
rfdetrendSTL	91.51	101.00	105.68	118.20
rfdetrendSTL_RL	105.38	99.41	101.23	114.70
rfdetrendSTL_bc	90.66	98.66	106.76	119.33
rfdetrendSTL_bc_RL	104.78	96.35	101.59	113.07
gbm	83.35	81.77	81.62	77.91
gbm_RL	80.24	78.14	74.94	**67.89**
gbm_bc	83.65	80.40	80.94	75.39
gbm_bc_RL	80.46	76.11	74.44	69.32
gbmSD	83.58	82.74	83.47	76.12
gbmSD_RL	81.64	79.38	76.56	68.78
gbmSD_bc	83.26	81.41	81.87	75.84
gbmSD_bc_RL	81.74	78.60	77.55	69.75
gbmD	82.58	97.82	103.56	100.90
gbmD_RL	98.79	93.82	98.50	102.29
gbmD_bc	91.25	102.53	103.06	102.09
gbmD_bc_RL	100.41	94.97	101.52	107.29
gbmSTL	85.77	83.37	84.18	76.58
gbmSTL_RL	86.46	81.45	80.86	71.78
gbmSTL_bc	86.12	83.52	83.30	75.22
gbmSTL_bc_RL	82.81	78.62	77.98	69.37
gbmdetrend	79.93	94.95	104.83	112.52
gbmdetrend_RL	82.89	86.37	93.67	107.32
gbmdetrend_bc	79.59	91.42	105.37	115.76
gbmdetrend_bc_RL	79.09	87.10	95.79	109.47
gbmdetrendSD	81.93	96.26	104.49	112.95
gbmdetrendSD_RL	88.74	92.51	95.89	108.89
gbmdetrendSD_bc	78.30	91.93	102.30	115.34
gbmdetrendSD_bc_RL	83.66	89.67	95.50	109.53
gbmdetrendD	96.82	105.81	116.56	119.77
gbmdetrendD_RL	102.27	99.36	104.85	114.53
gbmdetrendD_bc	94.70	101.96	114.48	120.11
gbmdetrendD_bc_RL	103.56	102.81	108.51	117.46
gbmdetrendSTL	80.29	96.23	106.31	114.18
gbmdetrendSTL_RL	84.02	88.04	93.18	106.87
gbmdetrendSTL_bc	81.41	97.49	106.16	115.51
gbmdetrendSTL_bc_RL	84.04	91.48	96.38	110.78
glmboost	66.14	77.90	76.50	83.18
glmboost_RL	70.04	72.77	74.41	81.99
glmboost_bc	66.48	76.89	77.94	82.26
glmboost_bc_RL	70.53	73.26	77.31	83.73
glmboostSD	69.50	80.39	76.88	82.48
glmboostSD_RL	74.82	72.24	73.11	79.07
glmboostSD_bc	69.51	79.59	78.51	82.23
glmboostSD_bc_RL	73.03	69.71	**72.13**	77.67
glmboostD	80.57	83.40	90.16	85.04
glmboostD_RL	94.71	95.96	95.39	96.93
glmboostD_bc	81.52	84.63	90.72	87.14
glmboostD_bc_RL	93.55	95.01	94.91	96.58
glmboostSTL	75.85	79.91	78.05	81.95
glmboostSTL_RL	82.61	76.62	75.01	81.52
glmboostSTL_bc	75.60	79.77	80.18	81.59
glmboostSTL_bc_RL	82.37	77.06	76.89	82.16
gamboost	69.58	85.26	91.99	91.87
gamboost_RL	74.85	77.83	90.00	89.82
gamboost_bc	68.63	84.87	91.00	91.60
gamboost_bc_RL	73.19	79.18	91.92	92.44
gamboostSD	71.30	87.46	92.67	93.06
gamboostSD_RL	77.66	83.82	94.93	93.62
gamboostSD_bc	70.90	87.39	92.62	93.82
gamboostSD_bc_RL	76.73	85.62	97.87	94.68
gamboostD	78.14	98.18	96.07	93.58
gamboostD_RL	98.83	95.28	94.17	95.09
gamboostD_bc	77.47	96.18	99.25	94.64
gamboostD_bc_RL	102.18	95.90	94.70	95.29
gamboostSTL	74.36	90.97	93.57	94.94
gamboostSTL_RL	81.74	89.23	96.10	93.85
gamboostSTL_bc	74.04	89.16	96.23	97.33
gamboostSTL_bc_RL	79.72	89.06	97.76	94.74
gamboostdetrend	78.19	101.18	107.37	115.42
gamboostdetrend_RL	84.44	90.45	94.69	108.08
gamboostdetrend_bc	74.66	97.00	107.28	116.94
gamboostdetrend_bc_RL	79.65	88.57	96.65	108.37
gamboostdetrendSD	81.45	103.18	106.84	116.98
gamboostdetrendSD_RL	85.12	92.56	95.88	108.42
gamboostdetrendSD_bc	78.24	99.57	107.82	117.60
gamboostdetrendSD_bc_RL	79.81	90.51	99.32	109.65
gamboostdetrendD	80.71	111.69	115.21	116.64
gamboostdetrendD_RL	102.79	102.63	108.56	114.08
gamboostdetrendD_bc	79.93	110.02	115.63	113.44
gamboostdetrendD_bc_RL	98.29	99.64	108.93	114.25
gamboostdetrendSTL	82.80	105.67	109.19	118.18
gamboostdetrendSTL_RL	87.63	95.79	97.08	109.85
gamboostdetrendSTL_bc	79.53	103.85	109.66	117.79
gamboostdetrendSTL_bc_RL	84.60	94.19	98.44	112.01

Table 5.4: *Average Ranked MdASE corresponding to dots in Figure 5.4. Best model metric per horizon range is bold & underlined and shading denote Top10% accordingly.*

- The classical ARIMA modeling should always get a try at least for classical time series without any additional external covariates. ETS model are less promising not only because of they are incapable of using exogenous covariates.
- Using seasonal dummies seems to be a good approach to tackle seasonality. This is good news as it allows modeling the seasonal influence in a standard Machine Learning way in contrast to seasonal differencing or STL approaches.
- But if a STL decomposition is applied to the time series, use a Box-Cox transformation to account for increasing seasonality. At least this should always be tested as a variant if STL is used.
- Ensembling over different methods not only reduces variance and therefore MSE, but also makes forecast more consistent over different horizon ranges. Tuning the weights might be advantageous depending on the algorithm used.
- If classical methods are superior and time and hardware resources allow bagging, try this alternative utilizing easy-to-use ME bootstrap. If several good performing methods result from some extended tests then try a random selection of these for every bootstrap sample to decorrelate the bagged forecasts.
- Tree based methods as well as standard splines, i.e. splines exhibiting a locality, are incapable of forecasting time series comprising a trend that persists in future. This is the reason why gradient boosted trees (*gbm*) as one of the most successful Machine Learning algorithm, might perform badly in the time series forecasting context. But keep in mind that for series without a trend and especially lots of covariates this approach might regain its credits.
- Gaussian Processes for time series forecasting are overrated in the Machine Learning community.
- Actually a Support Vector Machine seems to be one of the best Machine Learning approaches for forecasting especially when external covariates are available. Tuning the lagset might further improve the prediction from this model class.
- Direct forecasts might not be worth the effort regarding implementation amount as well as resource consumption. Though the presented sRecDir approach limits these efforts to a certain extent.

6 Summary and Outlook

In the framework of this thesis extensive comparisons regarding forecast performance of classical forecast model and Machine Learning approaches are conducted for a variety of official benchmark competition time series, i.e. the *M3* (Makridakis & Hibon (2000)), concentrating on the series of category *INDUSTRY*, and *Tourism* competitions (Athanasopoulos et al. (2011)) and the *NN5* contest (Crone (2009b)). Tests for 100 simulated pure ARIMA(1,1,1) time series are added.

The classical used models are basically ARIMA and ETS which are partly combined with different ensembling techniques comprising also bagging approaches utilizing 2 bootstrap strategies (Moving Block and Maximum Entropy bootstrap).

The selection of Machine Learning algorithms is influenced by several former studies, shortly discussed in the introduction, and comprises Neural Nets, SVMs with different Kernels, Gaussian Processes, Random Forests and several Boosting models using linear models (*glmboost*), splines (*gamboost*) and trees (*gbm*).

For all algorithms several model variants are tested like initial Box-Cox transformations or different deseasoning strategies, e.g. seasonal differencing, STL decompositions.

Some important prerequisites are founded in Chapter 2 like the different strategies for multi-step forecasting, i.e. recursive and direct approaches. With the *sRecDir* a special combination strategy is introduced that basically represents a direct strategy for the first season that is recursively used for the whole horizon range. Applied to the best Machine Learning models for the NN5 benchmark no substantial improvement compared to the standard recursive strategy results.

As the performance evaluation for time series forecasting is more complicated than for (time-unrelated) prediction, a deeper introduction to forecast performance metrics is also given suggesting to constrain most of the analysis to the MdRAE and MdASE related performance. But it must be kept in mind that the “no free lunch” theorem must be extended for forecasting in the sense of a dependence of the performance also on the forecast horizon which is nicely illustrated in this chapter by an example showing that every method (even a naïve one) can be the best and the worst for different series of a benchmark.

Furthermore the importance of using several naïve forecast, e.g. a deseasonalized random walk with drift, is stressed and confirmed by a comparison for the M3 data showing that lots of these series (in fact most of the series belonging to the categories *DEMOGRAPHIC*, *FINANCE* and *MACRO*) just consist of a trend + seasonality + random variation which makes benchmarks of sophisticated models highly questionable, though this is extensively done by several authors (cf. Chapter 1.2).

The classical models (ARIMA and ETS) are introduced in Chapter 3 with some emphasis put on ETS approaches as this model class is kind of neglected in academic canon even though highly popularized by practitioners. Furthermore some light is shed on the ARIMAX model muddle clarifying which options are available when exogenous covariate effects should be incorporated for ARIMA models.

Developed in the framework of these classical models but also applicable in combination with Machine Learning algorithms are initial Box-Cox transformations and different deseasoning strategies. Especially for the Tourism data a hard-coded seasonal differencing has shown to be advantageous for instance. An important side note here is that the latter is also recommended in conjunction with the *auto.arima* function of the forecast R-package as the test for seasonal differencing is declined to often resulting in dramatic performance reduction for the NN5 data.

Bagging reduces prediction variance and might therefore improve MSE (if increase in bias is limited). Applying this model variant to classical forecast methods is not straight forward, i.e. random sampling

with replacement, as this would destroy the data inherent dependency. Two approaches are introduced, the well-known Moving Block Bootstrap (MBB) and the easy-to-use Maximum Entropy (ME) bootstrap. The latter tries to keep the basic shape and data dependency by holding the ergodic property of the time series whereas the MBB samples complete blocks of data which are appended in order to reduce breaks in the dependency structure. The ME variant seems to have some advantages also regarding forecast performance, at least the *BootME_randsel* model, which comprises an additional decorrelation step by randomly selecting one of the classical model variants for each of the 30 bootstrap samples, is the winner for the competition objective forecast horizon for all 3 benchmarks with respect to the classical models. But a big question mark for the dependency keeping property is set by the simulated ARIMA series showing really bad performance of the bagging models for the 1-step-ahead forecast.

The tested model variants for the benchmark comprise ARIMA and ETS models with and without STL decomposition. Additionally a hard seasonal differencing for ARIMA and a strict damped ETS are applied. Each of these models is also tested in conjunction with a Box-Cox transformation. Further the tbats model (Arima-ETS combination) and the Theta method are added. The two bagging approaches (with 30 bootstrap samples for each time series), i.e. MB and ME bootstrap, are used in combination with a naïve model and the hard seasonal differenced ARIMA (as one of the best classical models) beside the before mentioned decorrelated versions (*BootME_randsel and BootMB_randsel*). Last but not least the models used in the latter variants are also ensembled by a simple averaging (mean and median).

Main results, apart from the ones mentioned above, seem to be that ARIMA mostly outperforms ETS models. Further findings are that if STL is used as the deseasoning approach also an initial Box-Cox transformation should be tested. Simple ensembling often generates better forecasts and generally stabilizes the variability. Using bagging approaches can further improve the forecasts at least the *BootME_randsel* performs best for all 3 competitions as mentioned above.

Chapter 4 is dedicated to the Machine Learning algorithms used.

Apart from exogenous covariate the typical predictor set comprising the time related information consists of a time point variable itself to forecast a trend and seasonal dummies to model the seasonal component. Especially the latter can also be accounted for just by the lagged target variable which is by default added to the covariate set to care for the time related correlation. This alternative is a result of an extended simulation study conducted to better understand how the Machine Learning approaches can handle the trend and seasonal components of phenotypic time series. This simulation further nicely shows that all tree based algorithms cannot predict a future trend which is an easy but nonetheless crucial finding also explaining why e.g. of one of the best predictive Machine Learning algorithms, i.e. *gbm*, exhibit unexpectedly bad performance in the benchmarks. A similar consequence has the locality property of splines used with the *gamboost* model performing worse than the *glmboost* restricted to linear covariate effect, suggesting to use additional variants for this model class as well as for the tree based utilizing an initial detrending step by STL decomposition. The trend is therefore forecasted separately by an ETS model and rewound in the end. Due to this double modeling or the non-optimal detrending (or even the ETS forecast) these variants perform consistently worse in the benchmarks.

This brings other algorithms into play like Neural Nets, the most used Machine Learning approach in the forecasting context. Unfortunately this model class is only better than a naïve forecast for the NN5 benchmark data. It must be mentioned that a recurrent Neural Net variant also capable of modeling a possible MA (moving average) component was just tested for the NN5 series with no improvement.

Also for Random Forests a performance mostly worse than a naïve forecast for the Tourism and M3-INDUSTRY is uncovered.

Chapter 4 further kind of demystifies Gaussian Processes, which are theoretically founded in the context of Kernel Machines by the Machine Learning community, by showing the strong connection to

well-known statistical approaches like kriging. Additionally these models perform consistently bad (apart from M3-MICRO series) questioning the high credit they have in the Machine Learning community.

On the other hand, SVMs as the typical representative of Kernel Machines perform really well; at least this model represent the only Machine Learning algorithm beating hard differenced ARIMA (*arimaD*) for the Tourism series.

As the simulation study also suggests to shorten the set of lagged target information all Machine Learning algorithms are tested with a reduced lagset. Together with a variant using initially Box-Cox transformed data to detangle possible multiplicative seasonality and the deseasoning approaches, i.e. seasonal differencing and STL decomposition, suggested by the classical approaches, in total 8 versions of each Machine Learning algorithm are tested. The results are compactly presented in Chapter 5 comparing their performance directly with the classical approaches.

Main findings for the M3-INDUSTRY series are that generally all classical models outperform all Machine Learning approaches. Under the Machine Learning algorithms *glmboost* with its variants is the best approach which confirms the simulation study experiments for phenotypic series. Furthermore the bad performance of Neural Nets is remarkable.

For the Tourism series the SVMs with polynomial and radial Kernel are competitive with the classical models with *svmpolySD_RL* (SVM with polynomial kernel, seasonal dummies for deseasoning and a reduced lagset) ending up as the best overall method starting from the half-season horizon range on. Together with the good performance of a boosted linear regression model (*glmboost*) the results from the simulation study are rebuild indicating a more phenotypic character of the series.

Though, the most informative insight might be given by the results of the NN5 benchmark as here the naïve forecast (including STL variants) are definitely worse than the applied models. It came out that the classical models are now inferior. After utilizing the exogenous covariates for the best performing models on this data, the SVM with a radial kernel using seasonal dummies resulted in a sMAPE of 19.6% clearly outperforming the team results of the official competition (cf. Crone (2009b)). Using a *sRecDir* forecast strategy does not improve the results.

A condensed list of lessons learned is given in Chapter 5.5 summarizing the overall main findings helpful for practice.

As the number of exogenous covariates is somewhat limited also for the NN5 benchmark series, forthcoming analysis might comprise the application of the most successful models onto other benchmark data comprising more exogenous information. One candidate is represented by the *Bike* contest hosted on the Kaggle website (Fanaee-T & Gama (2013)) consisting of hourly bike rental information with several highly influential weather related covariates.

Furthermore explicit lag tuning was neglected even though especially SVM-based model performance might further improve. Remember that this model class was the best one for the NN5 competition already beating the official team solutions.

Moreover kNN models are completely exlucded from the benchmark studies due to some resource restrictions as well as the questionable decision how to define a distance over time related and exogenous covariates which would also lead to a fully different testing approach. Nonetheless other authors used this model class (cf. Chapter 1.2) suggesting to give it a try.

Last but not least it should be tested how to improve the detrending of the tree based models in order to check whether especially the *gbm* as one of the most powerful prediction algorithm can regain its credits which would be of interest in situations with a high number of exogenous covariates.

A Supporting Plots and Tables

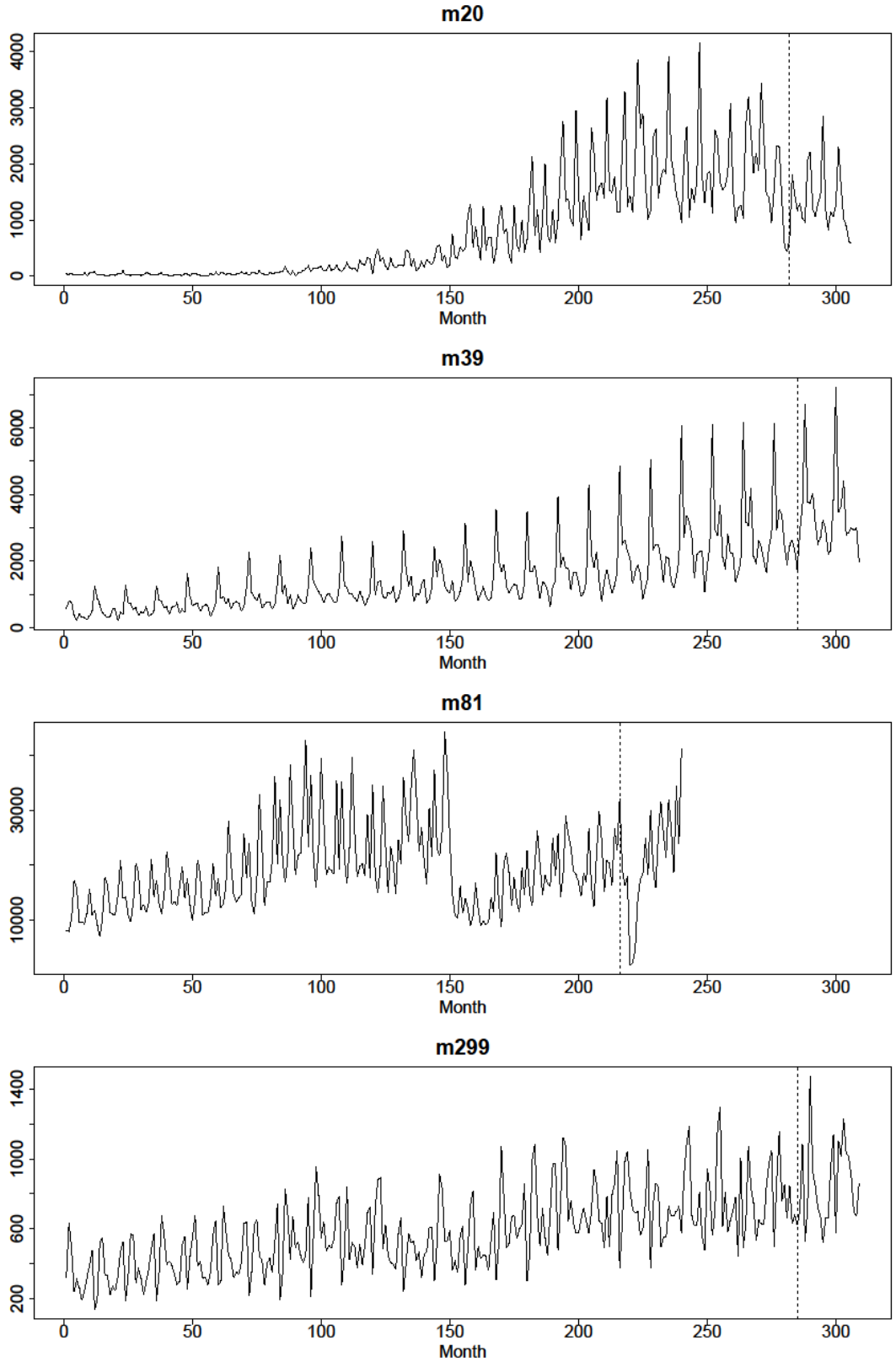

Figure A.1: *Example time series from Tourism data (reference line splits train and test data).*

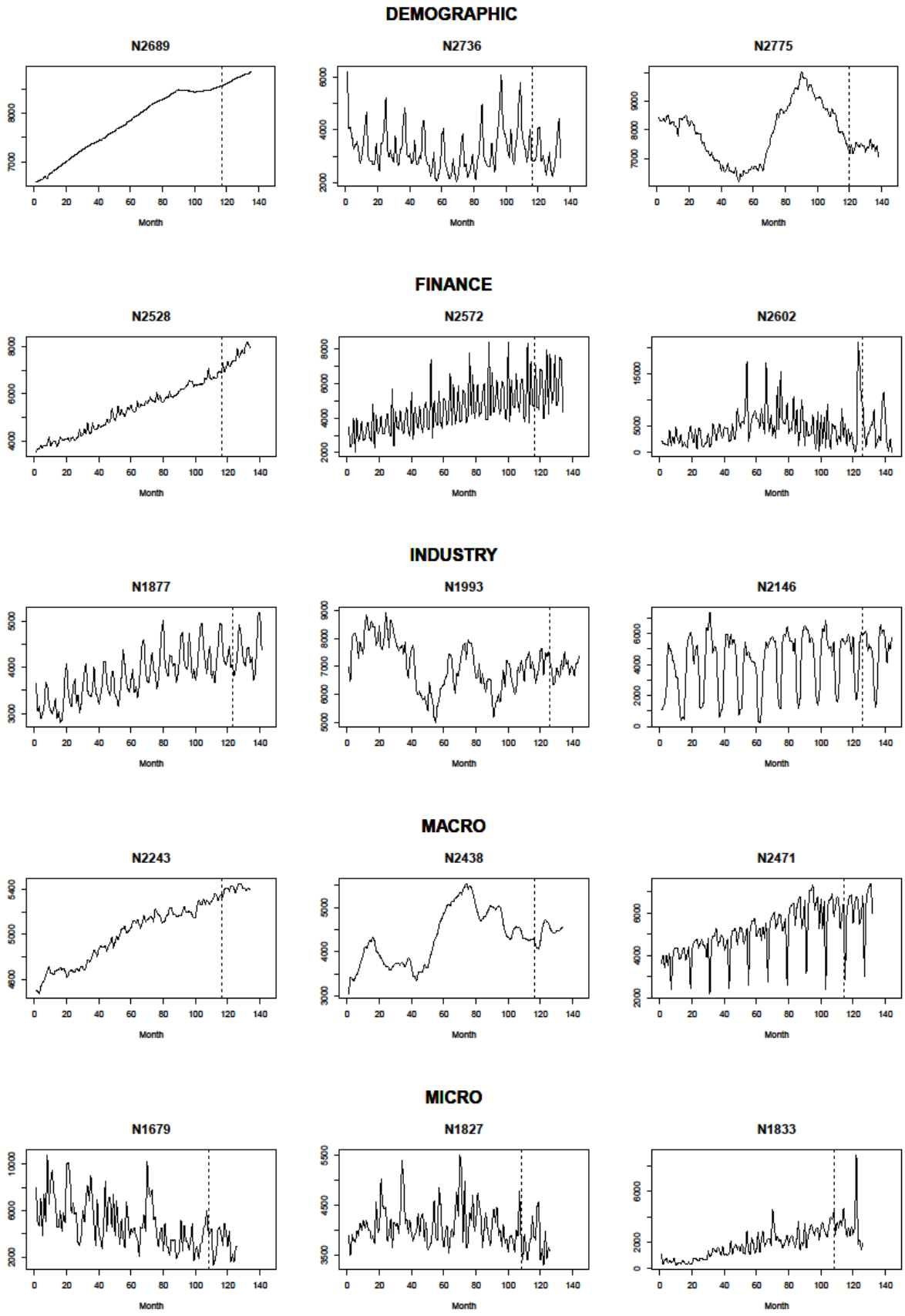

Figure A.2: *Example time series from M3 data (reference line splits train and test data).*

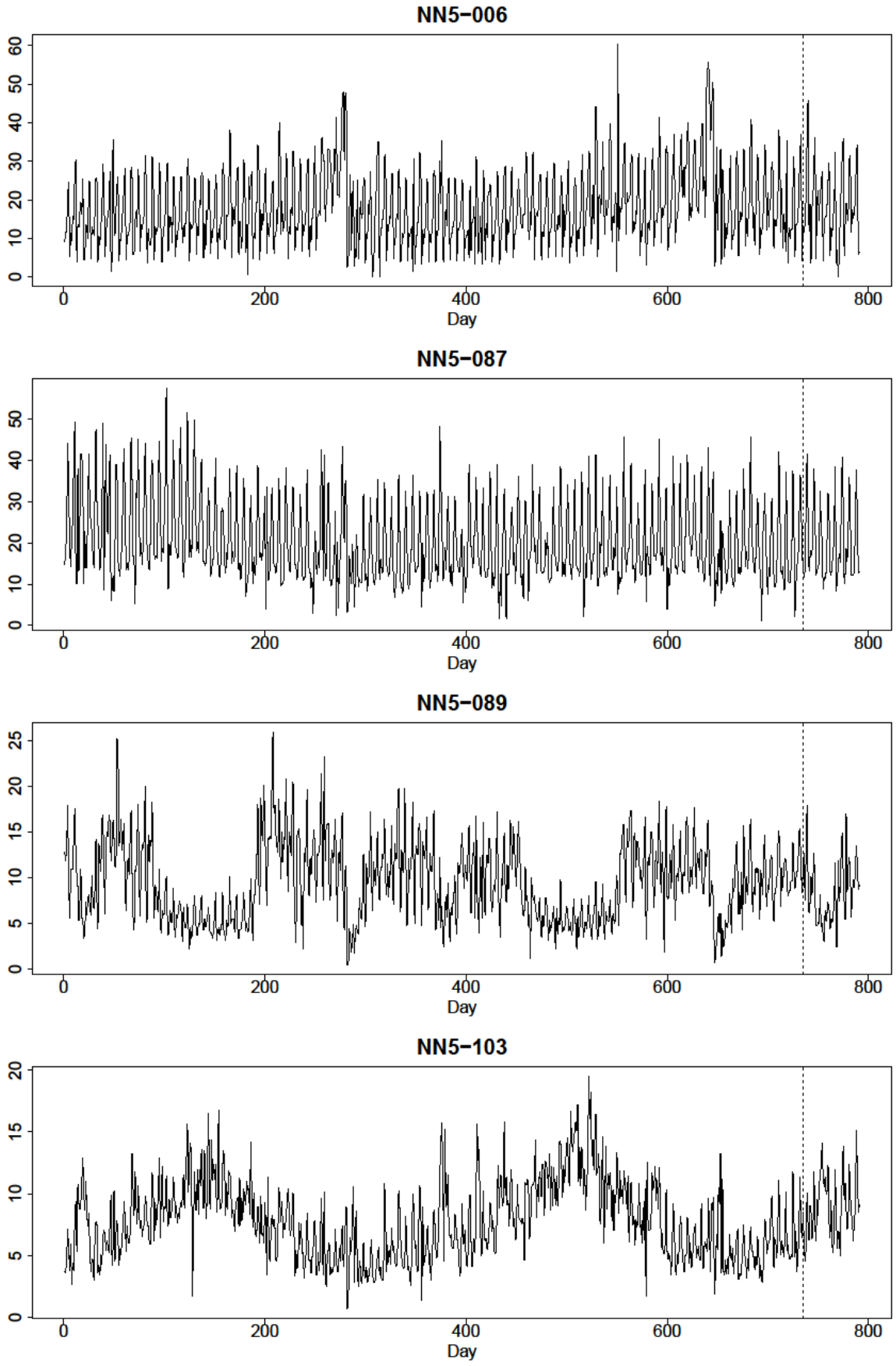

Figure A.3: *Example time series from NN5 data (reference line splits train and test data).*

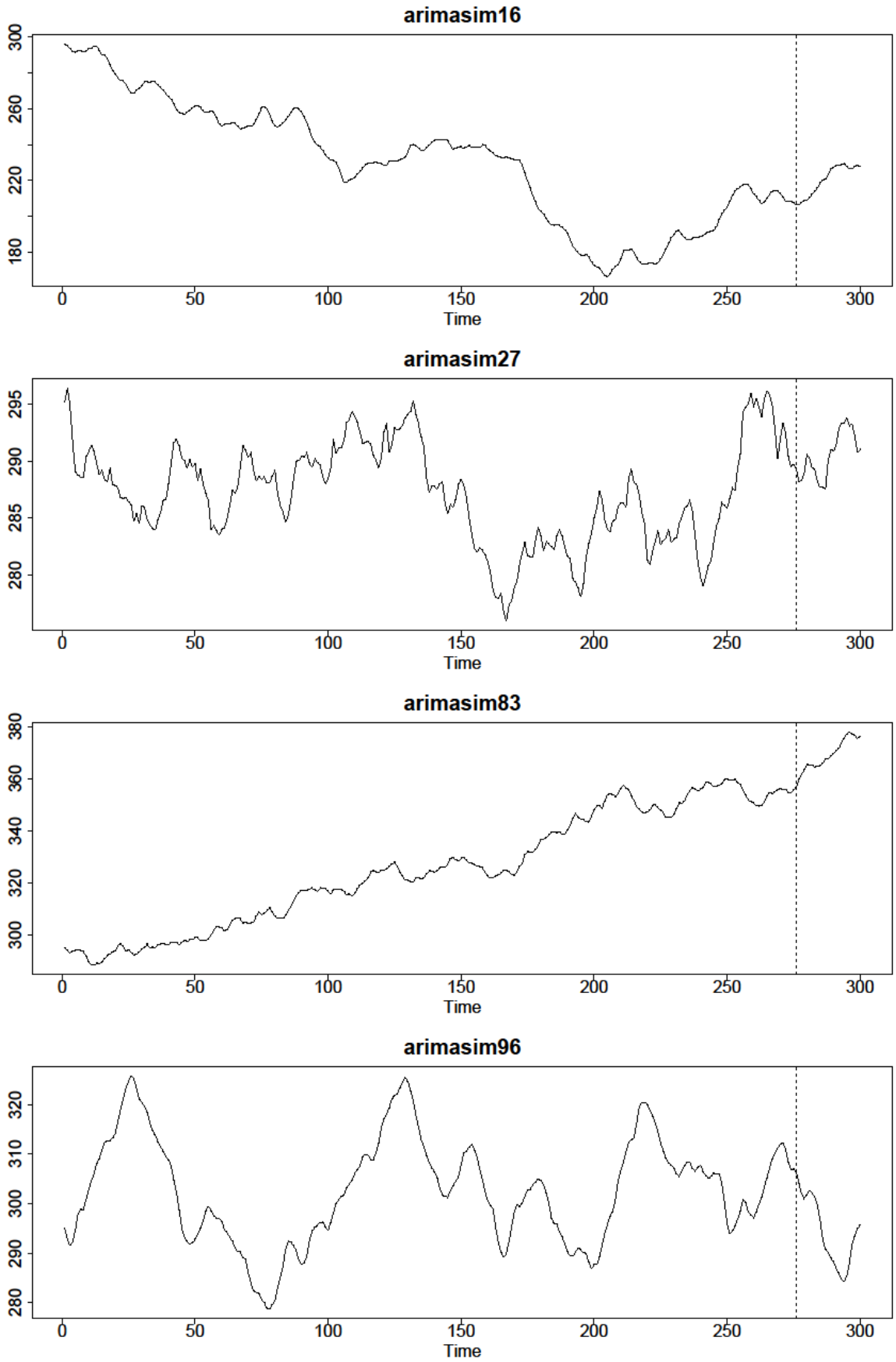

Figure A.4: *Example time series from Arimasim data (reference line splits train and test data).*

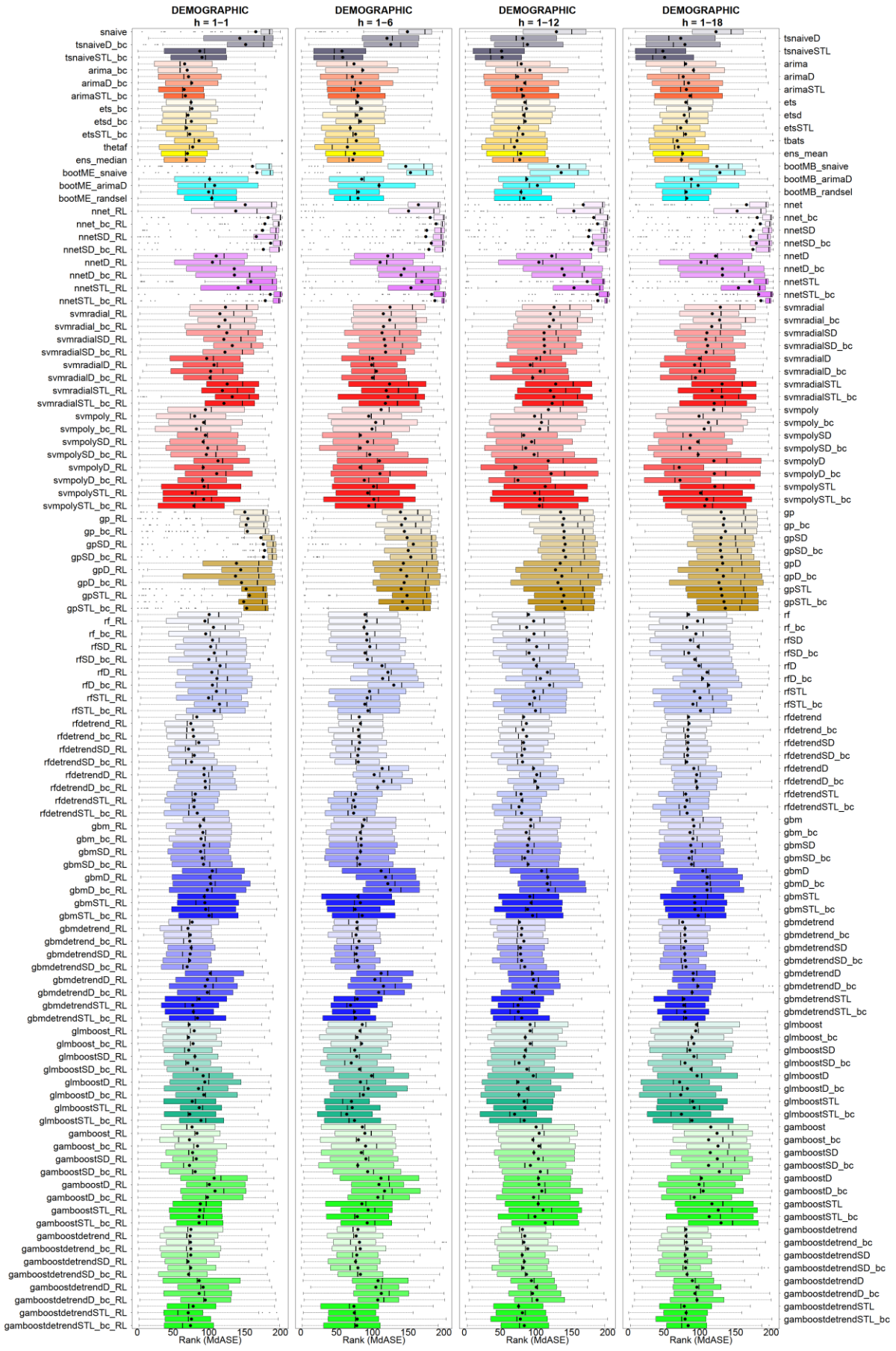

Figure A.5: *Overall Result (Ranked MdASE) by horizon range for M3-DEMOGRAPHIC competition.*

Method	DEMOGRAPHIC h=1-1	DEMOGRAPHIC h=1-6	DEMOGRAPHIC h=1-12	DEMOGRAPHIC h=1-18
snaive	165.73	149.06	128.60	122.98
tsnaiveD	143.49	120.44	81.54	73.66
tsnaiveD_bc	151.60	125.85	88.12	79.52
tsnaiveSTL	87.64	**57.53**	**52.00**	**48.80**
tsnaiveSTL_bc	90.69	58.69	52.63	51.19
arima	66.33	74.39	79.47	80.53
arima_bc	69.93	86.48	91.49	91.43
arimaD	71.72	72.06	74.38	77.06
arimaD_bc	76.11	83.38	84.34	84.52
arimaSTL	**65.62**	74.23	79.52	81.49
arimaSTL_bc	67.34	79.88	82.51	86.47
ets	75.32	79.06	84.44	80.82
ets_bc	76.57	84.68	86.57	86.58
etsd	71.03	78.49	82.88	78.57
etsd_bc	75.76	83.40	84.96	81.59
etsSTL	68.98	69.10	76.10	73.59
etsSTL_bc	73.41	76.61	81.59	80.37
tbats	86.46	77.61	73.49	68.69
thetaf	77.52	65.10	69.87	70.32
ens_mean	70.24	74.91	78.77	76.22
ens_median	68.68	72.79	77.22	74.63
bootMB_snaive	161.16	146.71	130.50	124.24
bootME_snaive	167.34	153.26	135.31	128.49
bootMB_arimaD	101.30	85.24	87.13	88.49
bootME_arimaD	108.28	109.36	102.03	98.01
bootMB_randsel	99.98	80.47	79.04	81.34
bootME_randsel	104.33	79.93	83.08	82.14
nnet	151.02	164.34	166.01	165.54
nnet_RL	137.81	150.66	152.89	152.56
nnet_bc	182.86	180.79	180.72	180.53
nnet_bc_RL	178.40	189.03	186.04	184.87
nnetSD	175.17	175.97	174.51	174.88
nnetSD_RL	166.43	174.76	173.41	170.99
nnetSD_bc	186.50	182.34	179.08	179.28
nnetSD_bc_RL	176.34	178.92	176.70	174.54
nnetD	110.49	121.60	122.10	122.31
nnetD_RL	105.13	110.87	104.03	102.02
nnetD_bc	135.67	144.37	136.48	131.86
nnetD_bc_RL	135.96	140.22	139.38	131.99
nnetSTL	158.74	169.12	171.62	169.96
nnetSTL_RL	141.11	153.86	153.14	154.40
nnetSTL_bc	186.04	182.87	185.29	182.50
nnetSTL_bc_RL	179.00	187.38	186.49	185.70
svmradial	123.46	124.69	125.28	129.06
svmradial_RL	115.64	115.41	119.93	117.79
svmradial_bc	122.74	124.21	124.49	128.01
svmradial_bc_RL	114.03	115.62	118.73	117.07
svmradialSD	125.26	113.51	111.31	110.18
svmradialSD_RL	121.08	116.60	111.17	108.71
svmradialSD_bc	132.91	118.09	111.86	111.14
svmradialSD_bc_RL	122.69	118.54	111.91	109.18
svmradialD	97.08	100.26	100.58	99.78
svmradialD_RL	107.39	99.06	92.06	93.07
svmradialD_bc	102.67	105.59	105.90	100.34
svmradialD_bc_RL	101.77	100.07	95.41	94.16
svmradialSTL	125.67	123.98	127.56	131.34
svmradialSTL_RL	119.03	119.31	120.10	117.48
svmradialSTL_bc	132.72	121.77	124.86	131.13
svmradialSTL_bc_RL	121.24	118.11	122.56	120.47
svmpoly	95.41	112.29	117.36	119.80
svmpoly_RL	80.31	94.88	98.08	99.31
svmpoly_bc	92.89	104.64	107.62	112.41
svmpoly_bc_RL	82.76	99.51	105.28	106.43
svmpolySD	95.28	83.22	82.89	87.17
svmpolySD_RL	92.29	93.21	93.98	97.47
svmpolySD_bc	98.64	83.06	85.68	86.39
svmpolySD_bc_RL	96.64	96.43	97.49	98.08
svmpolyD	112.90	109.92	117.21	119.92
svmpolyD_RL	92.23	82.79	71.77	71.53
svmpolyD_bc	111.10	110.57	121.14	120.63
svmpolyD_bc_RL	91.39	88.74	74.80	72.58
svmpolySTL	93.47	101.57	112.86	121.32
svmpolySTL_RL	76.83	93.49	98.18	101.32
svmpolySTL_bc	92.88	101.92	105.38	109.78
svmpolySTL_bc_RL	79.21	94.94	104.64	107.11
gp	150.41	139.29	134.41	130.20
gp_RL	154.72	145.99	138.79	134.20
gp_bc	152.33	140.74	137.22	133.18
gp_bc_RL	153.97	144.77	139.26	136.06
gpSD	172.96	148.59	138.99	130.14
gpSD_RL	176.18	157.24	140.77	129.00
gpSD_bc	178.12	150.13	138.57	129.23
gpSD_bc_RL	176.52	153.69	141.21	130.83
gpD	138.76	143.32	133.88	132.24
gpD_RL	144.63	139.67	127.24	124.48
gpD_bc	137.36	147.91	136.10	133.19
gpD_bc_RL	145.76	144.38	130.69	127.10
gpSTL	152.20	140.23	134.87	129.74
gpSTL_RL	156.79	148.46	138.79	131.14
gpSTL_bc	148.81	142.08	136.32	133.99
gpSTL_bc_RL	152.92	148.78	140.16	135.88
rf	100.69	89.76	89.53	84.63
rf_RL	94.66	91.79	96.87	97.30
rf_bc	106.67	88.46	87.00	82.24
rf_bc_RL	95.76	92.48	97.34	94.86
rfSD	105.34	92.21	90.28	87.19
rfSD_RL	102.73	96.44	101.00	98.01
rfSD_bc	107.84	89.84	90.44	84.83
rfSD_bc_RL	100.13	93.11	96.73	93.30
rfD	115.84	113.52	100.79	99.09
rfD_RL	104.34	121.78	116.11	110.49
rfD_bc	111.54	114.33	106.21	104.07
rfD_bc_RL	105.12	129.90	119.16	112.58
rfSTL	110.72	96.09	96.64	92.83
rfSTL_RL	99.87	92.84	98.34	100.55
rfSTL_bc	115.01	89.62	91.26	90.77
rfSTL_bc_RL	107.70	93.99	99.40	101.31

Method	DEMOGRAPHIC h=1-1	DEMOGRAPHIC h=1-6	DEMOGRAPHIC h=1-12	DEMOGRAPHIC h=1-18
rfdetrend	83.42	81.72	82.78	84.52
rfdetrend_RL	74.97	83.40	86.60	85.27
rfdetrend_bc	77.88	80.40	81.99	83.69
rfdetrend_bc_RL	78.78	81.09	86.84	83.82
rfdetrendSD	86.18	82.17	82.64	83.76
rfdetrendSD_RL	71.86	80.86	84.00	83.50
rfdetrendSD_bc	79.88	79.64	80.24	83.22
rfdetrendSD_bc_RL	76.03	80.30	81.29	82.13
rfdetrendD	93.60	113.82	96.60	91.93
rfdetrendD_RL	93.50	102.56	101.14	95.98
rfdetrendD_bc	95.07	115.82	99.02	95.27
rfdetrendD_bc_RL	95.27	107.28	102.61	96.60
rfdetrendSTL	81.38	76.60	79.23	80.61
rfdetrendSTL_RL	79.86	73.80	80.26	82.20
rfdetrendSTL_bc	79.44	75.82	76.39	79.80
rfdetrendSTL_bc_RL	84.03	73.93	79.92	82.62
gbm	92.88	88.76	92.89	90.71
gbm_RL	87.90	86.57	92.56	92.09
gbm_bc	91.96	83.09	86.38	86.66
gbm_bc_RL	89.36	84.22	90.32	90.68
gbmSD	93.27	84.43	88.38	87.67
gbmSD_RL	88.72	83.48	88.88	89.22
gbmSD_bc	90.98	78.92	84.26	85.42
gbmSD_bc_RL	92.77	82.36	89.11	88.78
gbmD	105.00	112.37	107.87	104.49
gbmD_RL	101.47	118.52	116.38	110.92
gbmD_bc	102.82	121.67	115.86	110.26
gbmD_bc_RL	98.10	125.22	117.28	110.43
gbmSTL	93.80	79.81	91.67	93.07
gbmSTL_RL	94.73	83.33	93.41	93.41
gbmSTL_bc	95.76	75.94	88.42	93.58
gbmSTL_bc_RL	100.51	85.89	95.37	98.24
gbmdetrend	76.82	78.60	76.93	76.41
gbmdetrend_RL	71.02	78.32	80.14	79.69
gbmdetrend_bc	74.23	77.27	79.60	79.42
gbmdetrend_bc_RL	73.60	81.40	83.12	80.41
gbmdetrendSD	75.45	78.53	78.41	78.09
gbmdetrendSD_RL	74.02	77.32	78.46	79.59
gbmdetrendSD_bc	73.27	78.83	79.93	79.92
gbmdetrendSD_bc_RL	69.94	80.88	84.34	81.07
gbmdetrendD	102.21	112.36	95.37	91.16
gbmdetrendD_RL	98.19	103.60	96.49	91.27
gbmdetrendD_bc	95.21	115.62	99.50	97.71
gbmdetrendD_bc_RL	98.11	108.94	95.08	89.70
gbmdetrendSTL	86.70	78.93	77.94	78.13
gbmdetrendSTL_RL	77.71	69.62	75.12	78.13
gbmdetrendSTL_bc	78.68	74.93	75.47	79.60
gbmdetrendSTL_bc_RL	84.32	76.61	80.22	80.81
glmboost	72.63	85.97	91.90	95.69
glmboost_RL	79.78	82.77	91.83	94.58
glmboost_bc	71.74	78.57	85.10	89.33
glmboost_bc_RL	78.17	85.01	91.96	92.00
glmboostSD	72.03	75.38	85.37	86.38
glmboostSD_RL	80.74	78.16	83.88	92.08
glmboostSD_bc	70.90	70.57	76.38	79.68
glmboostSD_bc_RL	83.76	81.98	87.33	87.97
glmboostD	92.28	98.91	96.41	96.61
glmboostD_RL	94.53	83.31	74.96	72.13
glmboostD_bc	85.80	94.21	88.08	82.99
glmboostD_bc_RL	93.09	87.39	76.04	73.69
glmboostSTL	76.99	70.80	83.43	90.31
glmboostSTL_RL	86.80	71.99	84.59	92.39
glmboostSTL_bc	73.50	64.29	70.20	74.71
glmboostSTL_bc_RL	89.24	75.12	83.54	89.36
gamboost	76.74	86.07	100.26	115.73
gamboost_RL	83.58	89.61	104.45	124.39
gamboost_bc	73.02	80.62	95.72	112.61
gamboost_bc_RL	84.37	90.57	104.03	125.76
gamboostSD	77.38	84.63	97.03	114.92
gamboostSD_RL	82.64	91.02	103.42	124.82
gamboostSD_bc	73.08	79.59	92.18	112.53
gamboostSD_bc_RL	81.08	93.42	106.07	127.54
gamboostD	107.60	112.10	103.47	102.48
gamboostD_RL	100.73	109.23	104.12	99.36
gamboostD_bc	108.78	117.07	108.17	105.09
gamboostD_bc_RL	98.23	107.67	96.70	92.52
gamboostSTL	88.32	85.42	103.16	117.23
gamboostSTL_RL	87.71	94.07	109.99	126.13
gamboostSTL_bc	86.51	78.98	98.68	113.52
gamboostSTL_bc_RL	86.52	92.89	112.98	130.08
gamboostdetrend	75.14	79.87	81.11	80.89
gamboostdetrend_RL	73.94	77.38	84.27	81.02
gamboostdetrend_bc	74.46	81.98	83.12	81.56
gamboostdetrend_bc_RL	74.91	83.17	88.59	82.75
gamboostdetrendSD	74.89	78.31	80.82	79.87
gamboostdetrendSD_RL	71.13	76.52	83.66	80.36
gamboostdetrendSD_bc	74.73	79.89	81.88	80.55
gamboostdetrendSD_bc_RL	71.78	83.38	86.77	82.47
gamboostdetrendD	86.68	107.99	93.67	89.89
gamboostdetrendD_RL	92.28	104.92	100.48	96.26
gamboostdetrendD_bc	86.73	112.50	95.30	93.51
gamboostdetrendD_bc_RL	95.20	107.61	101.64	96.40
gamboostdetrendSTL	78.26	74.23	75.70	78.62
gamboostdetrendSTL_RL	71.37	74.90	81.08	81.24
gamboostdetrendSTL_bc	75.82	78.92	78.26	79.78
gamboostdetrendSTL_bc_RL	74.48	78.28	83.66	84.03

Table A.1: *Average Ranked MdASE corresponding to dots in Figure A.5. Best model metric per horizon range is bold & underlined and shading denote Top10% accordingly.*

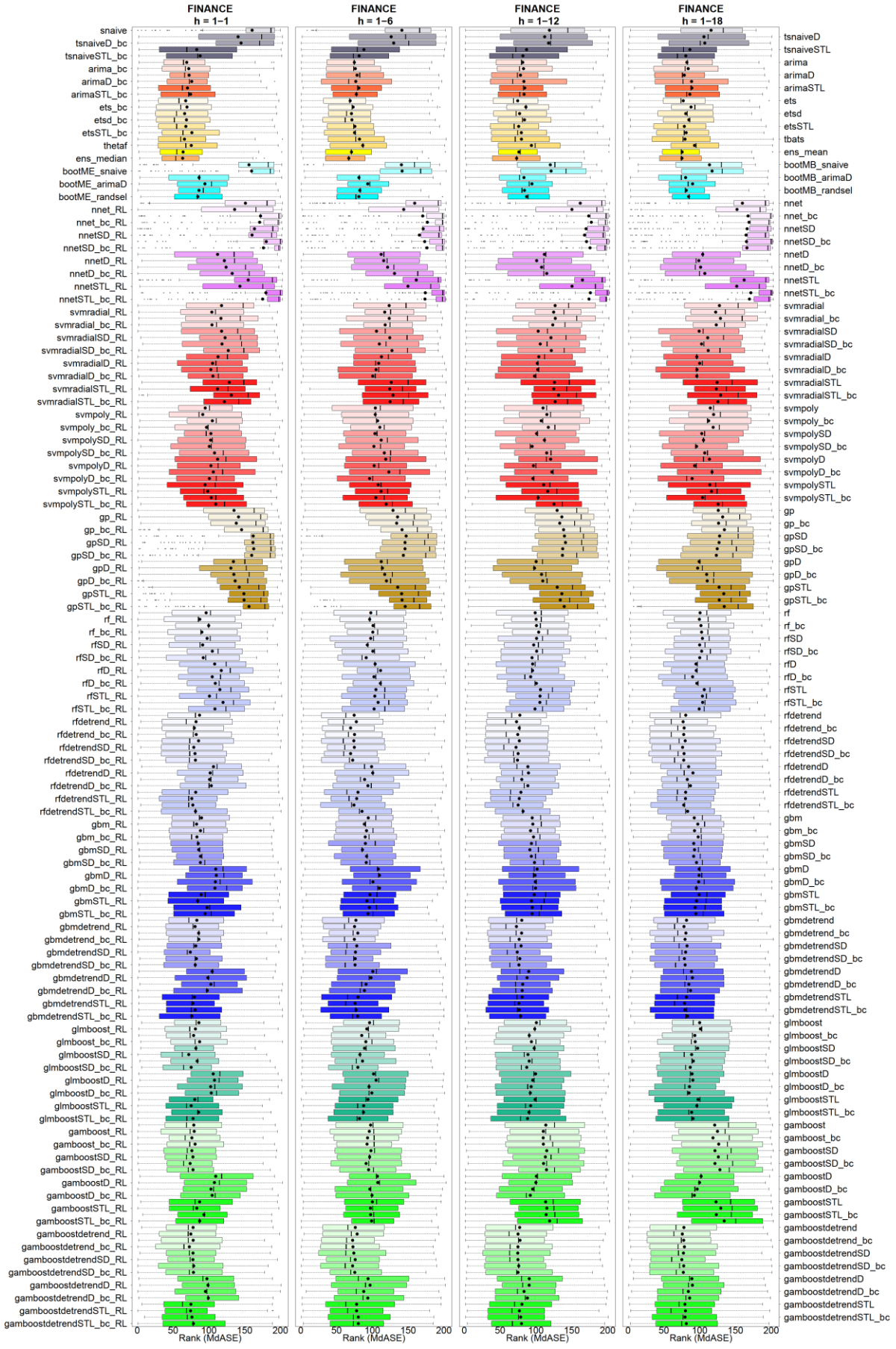

Figure A.6: *Overall Result (Ranked MdASE) by horizon range for M3-FINANCE competition.*

Method	FINANCE h=1-1	FINANCE h=1-6	FINANCE h=1-12	FINANCE h=1-18
snaive	160.60	141.26	118.83	116.05
tsnaiveD	141.07	126.25	112.12	106.04
tsnaiveD_bc	145.18	129.88	117.61	107.25
tsnaiveSTL	83.29	88.29	86.88	86.46
tsnaiveSTL_bc	87.98	81.49	81.73	80.79
arima	69.34	74.81	81.57	82.19
arima_bc	72.03	76.17	82.63	83.99
arimaD	72.59	78.68	78.40	78.71
arimaD_bc	76.18	76.80	83.11	88.82
arimaSTL	70.24	81.08	84.22	89.15
arimaSTL_bc	74.68	77.69	83.29	86.39
ets	67.84	69.00	74.42	77.28
ets_bc	69.61	73.16	85.84	88.05
etsd	66.31	70.89	77.08	80.62
etsd_bc	68.90	71.40	83.92	85.06
etsSTL	68.11	75.63	75.84	78.68
etsSTL_bc	76.47	75.07	79.53	81.30
tbats	66.06	82.04	79.70	79.64
thetaf	75.61	86.61	93.62	92.89
ens_mean	64.17	70.88	76.08	75.64
ens_median	**63.61**	**67.08**	73.03	75.39
bootMB_snaive	156.28	140.60	120.02	113.89
bootME_snaive	159.90	141.60	121.15	117.52
bootMB_arimaD	86.57	81.27	83.17	80.56
bootME_arimaD	94.66	93.61	94.42	90.28
bootMB_randsel	86.39	82.47	83.95	81.51
bootME_randsel	84.86	81.19	87.04	84.67
nnet	151.25	159.09	161.96	159.86
nnet_RL	136.21	143.87	150.25	152.27
nnet_bc	172.38	170.35	173.66	167.73
nnet_bc_RL	171.13	176.85	177.36	169.04
nnetSD	163.95	171.03	169.93	166.42
nnetSD_RL	160.46	165.69	167.90	164.96
nnetSD_bc	180.33	173.12	170.73	165.56
nnetSD_bc_RL	176.42	176.25	175.58	166.16
nnetD	112.36	112.28	111.76	104.35
nnetD_RL	121.78	115.88	101.04	98.96
nnetD_bc	124.26	121.36	107.80	101.39
nnetD_bc_RL	132.80	130.88	115.11	107.23
nnetSTL	155.10	161.35	164.90	162.28
nnetSTL_RL	143.54	149.72	150.31	151.96
nnetSTL_bc	179.89	174.04	175.98	171.61
nnetSTL_bc_RL	175.39	173.48	174.02	169.73
svmradial	117.89	123.52	126.73	127.72
svmradial_RL	104.33	117.32	124.68	122.53
svmradial_bc	117.11	123.59	126.95	129.22
svmradial_bc_RL	104.59	117.92	123.50	123.30
svmradialSD	117.96	105.80	103.11	99.33
svmradialSD_RL	122.84	124.30	120.92	111.27
svmradialSD_bc	118.54	109.81	105.92	102.80
svmradialSD_bc_RL	127.25	127.44	121.37	111.96
svmradialD	112.95	112.72	103.87	96.04
svmradialD_RL	105.26	108.83	101.81	100.11
svmradialD_bc	102.85	105.18	102.40	96.39
svmradialD_bc_RL	105.82	100.48	97.96	96.15
svmradialSTL	128.62	126.25	125.75	124.47
svmradialSTL_RL	112.18	124.23	124.86	123.32
svmradialSTL_bc	131.52	128.88	131.45	129.26
svmradialSTL_bc_RL	121.83	124.70	126.82	125.56
svmpoly	95.05	104.19	109.82	114.93
svmpoly_RL	91.61	104.07	115.25	119.62
svmpoly_bc	105.03	107.37	107.33	112.58
svmpoly_bc_RL	96.85	110.43	117.27	118.90
svmpolySD	102.93	103.61	100.84	103.01
svmpolySD_RL	102.67	112.33	111.79	105.45
svmpolySD_bc	100.75	102.16	94.66	95.71
svmpolySD_bc_RL	108.11	116.73	114.91	107.18
svmpolyD	112.32	118.59	120.41	113.72
svmpolyD_RL	102.92	102.43	96.53	93.04
svmpolyD_bc	106.36	123.06	122.15	117.43
svmpolyD_bc_RL	100.86	96.18	95.78	89.51
svmpolySTL	94.78	108.11	110.77	114.20
svmpolySTL_RL	98.56	112.10	116.42	116.56
svmpolySTL_bc	103.48	104.75	102.95	104.00
svmpolySTL_bc_RL	109.75	119.28	125.18	125.89
gp	135.33	128.83	129.61	126.14
gp_RL	141.61	135.22	136.10	132.25
gp_bc	138.53	133.97	133.26	126.48
gp_bc_RL	145.95	141.55	138.59	134.61
gpSD	162.08	147.22	139.80	127.94
gpSD_RL	161.68	145.67	141.24	127.18
gpSD_bc	162.77	145.61	137.33	124.70
gpSD_bc_RL	160.04	143.31	136.96	123.46
gpD	134.47	112.46	100.20	99.37
gpD_RL	131.00	113.99	97.94	97.30
gpD_bc	135.03	117.25	107.83	109.86
gpD_bc_RL	136.92	119.69	109.46	110.68
gpSTL	142.66	135.44	129.72	127.26
gpSTL_RL	149.04	140.85	136.12	133.81
gpSTL_bc	149.32	141.53	133.95	127.48
gpSTL_bc_RL	156.31	145.85	139.44	134.40
rf	96.35	97.82	98.95	100.72
rf_RL	87.50	96.29	100.29	99.75
rf_bc	99.99	100.98	100.75	101.94
rf_bc_RL	90.46	100.61	103.83	102.83
rfSD	97.73	97.77	100.83	103.96
rfSD_RL	91.53	92.98	96.82	100.09
rfSD_bc	105.02	100.69	101.03	102.95
rfSD_bc_RL	92.41	91.35	94.54	100.28
rfD	108.35	103.98	95.42	95.04
rfD_RL	117.54	111.42	95.32	95.38
rfD_bc	104.89	102.25	92.54	90.23
rfD_bc_RL	109.12	111.28	100.66	96.18
rfSTL	115.62	105.05	106.32	106.54
rfSTL_RL	101.00	103.40	105.68	103.96
rfSTL_bc	119.96	108.07	105.46	104.18
rfSTL_bc_RL	108.73	102.46	98.61	99.49

Method	FINANCE h=1-1	FINANCE h=1-6	FINANCE h=1-12	FINANCE h=1-18
rfdetrend	87.13	74.56	77.46	80.67
rfdetrend_RL	82.30	77.63	72.84	77.00
rfdetrend_bc	79.88	69.60	76.61	78.46
rfdetrend_bc_RL	82.60	75.10	74.59	77.85
rfdetrendSD	85.83	74.12	76.81	79.97
rfdetrendSD_RL	79.08	74.81	**72.18**	76.24
rfdetrendSD_bc	80.79	69.65	75.09	78.31
rfdetrendSD_bc_RL	81.42	72.54	74.25	77.82
rfdetrendD	106.68	98.91	88.75	84.68
rfdetrendD_RL	102.02	100.51	89.51	90.71
rfdetrendD_bc	101.20	89.24	80.17	82.82
rfdetrendD_bc_RL	103.53	93.83	88.75	87.24
rfdetrendSTL	81.75	79.97	78.94	80.51
rfdetrendSTL_RL	76.29	78.19	76.71	80.32
rfdetrendSTL_bc	77.85	74.35	75.05	77.79
rfdetrendSTL_bc_RL	81.51	85.54	81.96	83.32
gbm	89.90	94.32	94.76	92.54
gbm_RL	83.03	88.82	95.68	97.60
gbm_bc	88.54	91.93	92.50	92.95
gbm_bc_RL	83.49	90.04	96.64	98.05
gbmSD	85.28	90.36	93.73	91.82
gbmSD_RL	86.11	86.16	91.53	92.98
gbmSD_bc	88.52	91.97	93.61	91.77
gbmSD_bc_RL	88.40	91.54	98.00	95.69
gbmD	109.75	108.34	101.61	99.51
gbmD_RL	110.68	109.98	98.78	99.41
gbmD_bc	109.35	100.86	99.19	98.72
gbmD_bc_RL	108.68	110.01	99.45	95.54
gbmSTL	89.43	96.72	97.72	99.44
gbmSTL_RL	84.61	92.45	93.82	95.90
gbmSTL_bc	97.72	95.93	94.93	93.94
gbmSTL_bc_RL	95.08	94.13	94.76	95.05
gbmdetrend	83.40	77.05	80.13	81.79
gbmdetrend_RL	81.00	75.07	72.79	78.25
gbmdetrend_bc	85.92	79.73	79.95	80.93
gbmdetrend_bc_RL	86.13	74.80	76.53	79.69
gbmdetrendSD	81.99	78.30	79.26	82.42
gbmdetrendSD_RL	74.25	76.43	73.86	80.49
gbmdetrendSD_bc	82.73	76.03	77.50	78.84
gbmdetrendSD_bc_RL	81.16	75.78	76.18	79.82
gbmdetrendD	104.83	100.64	89.92	88.66
gbmdetrendD_RL	98.88	98.17	87.63	89.98
gbmdetrendD_bc	103.00	91.26	81.55	85.07
gbmdetrendD_bc_RL	98.02	88.98	80.82	87.34
gbmdetrendSTL	80.17	80.12	80.79	81.59
gbmdetrendSTL_RL	77.94	75.91	76.19	79.10
gbmdetrendSTL_bc	81.11	77.29	76.76	79.42
gbmdetrendSTL_bc_RL	76.38	79.34	78.64	83.23
glmboost	86.28	96.14	100.62	100.16
glmboost_RL	81.39	92.31	98.32	101.18
glmboost_bc	78.92	85.20	90.61	93.37
glmboost_bc_RL	87.31	90.85	93.69	93.93
glmboostSD	82.04	89.10	97.56	97.32
glmboostSD_RL	72.03	82.61	88.82	88.87
glmboostSD_bc	83.91	86.31	90.46	91.25
glmboostSD_bc_RL	75.50	79.67	87.17	86.98
glmboostD	106.26	101.79	99.48	89.48
glmboostD_RL	108.08	105.31	95.51	90.64
glmboostD_bc	102.70	95.42	93.30	85.89
glmboostD_bc_RL	103.61	99.34	92.11	85.32
glmboostSTL	80.50	93.97	98.78	97.96
glmboostSTL_RL	75.46	87.92	92.64	96.23
glmboostSTL_bc	85.89	87.63	90.23	88.92
glmboostSTL_bc_RL	78.12	82.09	88.18	90.86
gamboost	79.19	97.60	113.70	121.18
gamboost_RL	79.77	95.75	110.25	125.08
gamboost_bc	76.51	93.34	110.14	118.75
gamboost_bc_RL	80.96	92.58	110.39	126.73
gamboostSD	76.48	98.04	115.04	121.45
gamboostSD_RL	77.91	96.31	113.11	126.28
gamboostSD_bc	73.97	91.25	110.55	121.52
gamboostSD_bc_RL	77.99	94.81	114.72	128.48
gamboostD	109.82	107.14	100.56	102.11
gamboostD_RL	108.00	107.54	99.91	100.03
gamboostD_bc	103.13	96.45	95.39	96.70
gamboostD_bc_RL	104.82	99.53	92.06	92.72
gamboostSTL	87.34	100.04	113.45	122.93
gamboostSTL_RL	83.35	97.31	115.22	129.68
gamboostSTL_bc	93.28	97.61	114.23	123.48
gamboostSTL_bc_RL	87.45	98.88	119.06	134.35
gamboostdetrend	78.20	76.21	77.24	78.03
gamboostdetrend_RL	75.02	78.79	74.90	75.92
gamboostdetrend_bc	78.30	72.36	77.71	77.81
gamboostdetrend_bc_RL	72.97	72.88	74.84	78.77
gamboostdetrendSD	78.05	74.08	74.56	77.46
gamboostdetrendSD_RL	77.91	76.10	74.87	**75.12**
gamboostdetrendSD_bc	78.96	72.36	75.97	77.84
gamboostdetrendSD_bc_RL	78.64	75.50	74.97	77.34
gamboostdetrendD	97.45	94.09	90.68	89.28
gamboostdetrendD_RL	98.96	96.91	90.50	90.01
gamboostdetrendD_bc	95.43	87.95	83.31	84.32
gamboostdetrendD_bc_RL	99.45	93.75	88.13	86.15
gamboostdetrendSTL	75.11	77.96	80.45	79.51
gamboostdetrendSTL_RL	74.68	78.05	76.96	80.22
gamboostdetrendSTL_bc	76.64	80.40	78.73	79.82
gamboostdetrendSTL_bc_RL	78.61	80.45	79.81	81.44

Table A.2: *Average Ranked MdASE corresponding to dots in Figure A.6. Best model metric per horizon range is bold & underlined and shading denote Top10% accordingly.*

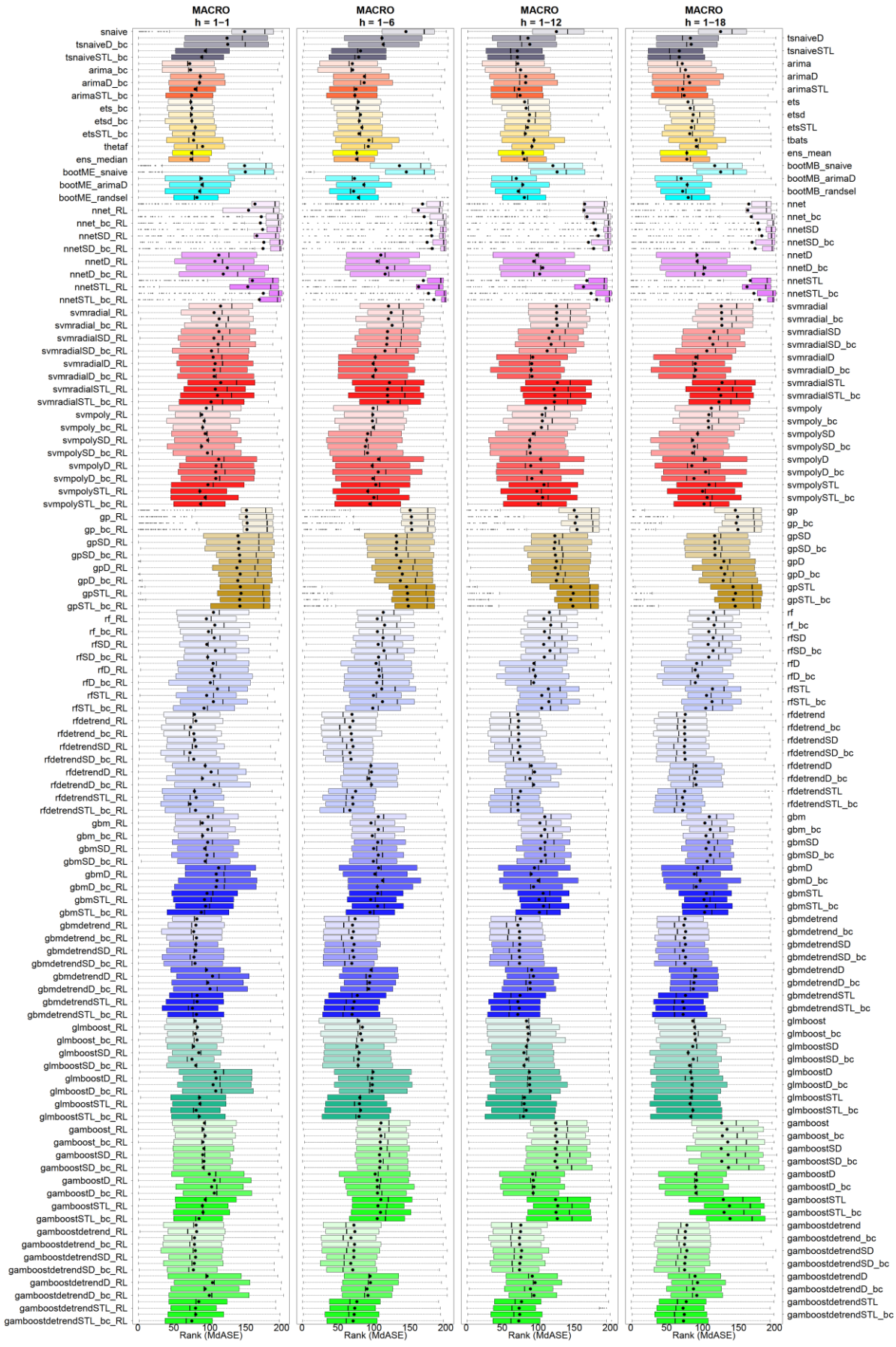

Figure A.7: *Overall Result (Ranked MdASE) by horizon range for M3-MACRO competition.*

Method	MACRO h=1-1	MACRO h=1-6	MACRO h=1-12	MACRO h=1-18
snaive	148.70	144.59	125.30	125.20
tsnaiveD	124.25	111.18	85.83	83.21
tsnaiveD_bc	125.21	113.00	88.29	84.29
tsnaiveSTL	94.22	81.47	70.94	67.72
tsnaiveSTL_bc	89.14	78.92	71.09	**67.66**
arima	**72.45**	70.01	71.71	71.74
arima_bc	73.58	70.43	75.40	76.25
arimaD	87.01	86.74	82.57	80.24
arimaD_bc	85.89	86.43	82.44	82.06
arimaSTL	80.03	75.09	73.13	72.47
arimaSTL_bc	75.05	73.49	74.71	74.43
ets	73.95	78.36	81.21	79.82
ets_bc	75.23	77.45	83.23	82.95
etsd	74.62	81.30	87.84	87.03
etsd_bc	75.25	79.66	86.80	85.80
etsSTL	80.14	83.21	85.13	84.38
etsSTL_bc	77.91	79.23	82.00	82.24
tbats	77.72	92.96	94.04	91.20
thetaf	90.23	92.08	91.26	91.45
ens_mean	74.83	76.22	78.66	78.18
ens_median	74.09	75.84	80.61	78.19
bootMB_snaive	148.72	135.53	120.46	117.01
bootME_snaive	149.69	145.20	126.24	125.22
bootMB_arimaD	88.67	73.01	**69.44**	70.25
bootME_arimaD	89.35	86.10	78.46	78.67
bootMB_randsel	86.32	71.81	72.33	72.36
bootME_randsel	82.39	78.57	80.79	80.17
nnet	163.01	167.70	164.59	164.34
nnet_RL	154.13	161.86	163.79	163.02
nnet_bc	171.97	169.92	167.83	167.82
nnet_bc_RL	170.56	179.19	177.03	177.08
nnetSD	173.76	179.61	179.49	179.05
nnetSD_RL	165.66	180.47	183.54	182.51
nnetSD_bc	175.41	174.03	169.95	168.94
nnetSD_bc_RL	174.30	180.97	176.97	173.04
nnetD	112.78	109.63	97.99	92.18
nnetD_RL	107.33	104.41	93.63	92.51
nnetD_bc	124.65	118.54	106.38	103.25
nnetD_bc_RL	119.08	115.62	102.05	101.47
nnetSTL	159.54	168.72	167.58	166.57
nnetSTL_RL	153.11	162.16	162.97	161.86
nnetSTL_bc	174.72	175.98	173.45	171.47
nnetSTL_bc_RL	169.22	183.63	181.35	179.41
svmradial	115.22	120.29	125.30	126.00
svmradial_RL	106.23	123.93	125.85	126.51
svmradial_bc	113.32	120.17	125.19	126.63
svmradial_bc_RL	110.15	125.28	126.46	126.89
svmradialSD	112.89	118.91	119.49	115.74
svmradialSD_RL	105.99	118.29	114.93	110.05
svmradialSD_bc	111.75	117.78	117.85	114.41
svmradialSD_bc_RL	102.64	115.34	112.44	105.92
svmradialD	104.88	101.80	92.37	90.71
svmradialD_RL	107.63	98.94	90.93	89.92
svmradialD_bc	106.31	102.43	90.26	90.49
svmradialD_bc_RL	106.23	98.42	91.20	89.08
svmradialSTL	115.38	121.58	126.66	127.06
svmradialSTL_RL	107.81	118.89	121.72	122.56
svmradialSTL_bc	110.89	118.75	123.13	125.30
svmradialSTL_bc_RL	101.99	119.44	122.38	122.90
svmpoly	95.49	98.97	110.12	112.29
svmpoly_RL	89.05	98.28	105.51	108.92
svmpoly_bc	92.50	97.94	105.75	108.00
svmpoly_bc_RL	90.36	99.12	104.46	108.51
svmpolySD	93.89	91.22	93.08	93.07
svmpolySD_RL	97.35	89.87	88.17	86.20
svmpolySD_bc	88.78	88.05	88.03	88.53
svmpolySD_bc_RL	97.07	91.00	89.08	86.13
svmpolyD	112.54	106.31	103.30	103.74
svmpolyD_RL	108.98	97.64	89.58	84.99
svmpolyD_bc	108.43	105.99	104.16	104.21
svmpolyD_bc_RL	108.55	98.59	91.18	88.10
svmpolySTL	97.67	102.53	107.87	108.95
svmpolySTL_RL	86.45	91.39	98.04	100.04
svmpolySTL_bc	93.79	99.91	106.10	106.29
svmpolySTL_bc_RL	87.37	94.22	100.11	102.02
gp	151.36	150.40	150.14	145.69
gp_RL	151.22	152.27	153.38	148.75
gp_bc	152.27	151.99	151.29	146.88
gp_bc_RL	152.02	152.52	153.85	149.25
gpSD	139.40	131.05	123.07	116.85
gpSD_RL	140.79	131.76	123.31	117.28
gpSD_bc	140.19	130.84	122.18	117.10
gpSD_bc_RL	141.69	131.38	123.87	117.35
gpD	142.18	137.54	125.39	129.63
gpD_RL	137.86	135.56	123.75	125.85
gpD_bc	142.45	139.45	127.21	130.70
gpD_bc_RL	139.25	136.72	125.14	128.67
gpSTL	142.70	145.44	145.41	142.54
gpSTL_RL	143.70	146.77	148.63	145.87
gpSTL_bc	141.72	146.45	145.24	142.90
gpSTL_bc_RL	142.09	147.83	148.20	145.46
rf	105.39	113.01	115.54	115.28
rf_RL	95.59	104.67	107.87	108.16
rf_bc	107.21	114.89	117.46	116.18
rf_bc_RL	98.45	104.92	108.72	108.80
rfSD	106.49	112.76	115.38	114.95
rfSD_RL	96.13	105.97	107.62	107.87
rfSD_bc	108.15	113.99	116.46	114.76
rfSD_bc_RL	97.43	106.47	108.46	108.74
rfD	104.86	103.09	94.10	92.01
rfD_RL	102.94	106.90	93.38	89.92
rfD_bc	106.55	107.59	95.99	93.39
rfD_bc_RL	101.08	104.11	93.66	90.00
rfSTL	110.91	110.85	114.08	113.81
rfSTL_RL	95.89	99.38	104.95	105.53
rfSTL_bc	105.31	111.91	114.79	113.30
rfSTL_bc_RL	92.28	98.40	104.81	104.44

Method	MACRO h=1-1	MACRO h=1-6	MACRO h=1-12	MACRO h=1-18
rfdetrend	78.87	69.54	71.63	76.02
rfdetrend_RL	80.82	70.87	72.92	74.63
rfdetrend_bc	73.53	66.57	71.97	75.32
rfdetrend_bc_RL	78.02	68.51	72.99	75.05
rfdetrendSD	79.26	69.20	72.41	75.76
rfdetrendSD_RL	80.95	71.01	74.60	74.76
rfdetrendSD_bc	72.74	66.82	72.04	75.01
rfdetrendSD_bc_RL	77.89	68.16	74.39	75.80
rfdetrendD	93.90	95.95	90.35	90.75
rfdetrendD_RL	102.09	96.58	94.92	91.66
rfdetrendD_bc	89.93	93.39	88.53	89.10
rfdetrendD_bc_RL	106.25	96.60	93.63	91.06
rfdetrendSTL	78.84	74.67	75.38	75.13
rfdetrendSTL_RL	81.28	70.29	72.52	72.13
rfdetrendSTL_bc	72.95	70.83	71.55	74.04
rfdetrendSTL_bc_RL	80.49	**66.55**	71.99	72.25
gbm	98.01	106.13	109.13	109.51
gbm_RL	89.99	96.08	102.36	103.24
gbm_bc	97.55	106.05	109.12	110.84
gbm_bc_RL	90.16	97.69	104.20	104.79
gbmSD	97.33	105.43	109.49	108.45
gbmSD_RL	93.24	99.63	103.19	105.00
gbmSD_bc	96.43	106.16	109.85	110.94
gbmSD_bc_RL	93.82	99.30	104.03	106.28
gbmD	112.62	106.60	94.76	93.54
gbmD_RL	109.25	101.41	90.21	88.61
gbmD_bc	110.32	112.20	99.98	96.76
gbmD_bc_RL	109.15	104.79	93.53	91.21
gbmSTL	96.55	105.41	106.89	105.58
gbmSTL_RL	93.04	95.60	101.34	102.22
gbmSTL_bc	94.54	105.26	107.38	105.59
gbmSTL_bc_RL	88.64	94.71	101.76	103.02
gbmdetrend	82.33	73.84	75.19	75.82
gbmdetrend_RL	80.93	70.20	71.72	74.03
gbmdetrend_bc	77.88	71.51	73.98	76.49
gbmdetrend_bc_RL	81.64	69.43	73.93	75.03
gbmdetrendSD	80.80	72.41	74.33	76.29
gbmdetrendSD_RL	80.83	70.49	73.43	73.27
gbmdetrendSD_bc	78.22	72.05	74.21	76.38
gbmdetrendSD_bc_RL	79.71	69.24	74.00	75.39
gbmdetrendD	95.66	96.16	91.04	89.98
gbmdetrendD_RL	104.11	94.46	93.49	90.72
gbmdetrendD_bc	97.47	93.96	88.74	88.18
gbmdetrendD_bc_RL	100.70	92.88	88.96	87.16
gbmdetrendSTL	82.65	77.01	74.83	76.37
gbmdetrendSTL_RL	82.63	72.68	71.79	72.66
gbmdetrendSTL_bc	76.25	70.66	73.62	74.39
gbmdetrendSTL_bc_RL	81.76	69.78	71.64	73.13
glmboost	80.13	78.87	83.76	86.48
glmboost_RL	82.89	83.98	85.02	89.05
glmboost_bc	80.10	81.39	86.14	89.33
glmboost_bc_RL	82.54	83.20	85.81	90.01
glmboostSD	77.73	76.72	83.64	86.46
glmboostSD_RL	85.01	79.65	80.14	80.02
glmboostSD_bc	75.48	78.28	83.91	87.19
glmboostSD_bc_RL	80.82	78.05	80.25	81.99
glmboostD	107.69	98.80	87.71	83.56
glmboostD_RL	109.16	97.31	88.36	84.86
glmboostD_bc	104.95	97.87	88.13	85.39
glmboostD_bc_RL	108.58	97.21	89.35	85.05
glmboostSTL	85.99	80.47	80.96	83.94
glmboostSTL_RL	86.52	79.81	80.57	82.75
glmboostSTL_bc	81.38	81.41	83.34	86.62
glmboostSTL_bc_RL	85.62	79.31	79.51	83.70
gamboost	92.87	109.83	124.12	126.85
gamboost_RL	90.98	108.08	125.41	134.06
gamboost_bc	93.69	109.82	124.01	127.55
gamboost_bc_RL	90.65	108.73	126.12	135.19
gamboostSD	91.97	109.55	123.74	125.99
gamboostSD_RL	90.42	107.84	125.79	135.65
gamboostSD_bc	92.12	108.64	123.54	126.79
gamboostSD_bc_RL	91.15	108.01	126.24	136.14
gamboostD	99.58	101.78	92.38	90.62
gamboostD_RL	106.61	104.62	93.18	91.74
gamboostD_bc	102.87	104.74	94.19	90.48
gamboostD_bc_RL	106.48	105.01	92.90	91.00
gamboostSTL	93.95	109.65	124.11	128.89
gamboostSTL_RL	90.01	105.61	127.09	137.42
gamboostSTL_bc	90.79	109.17	124.82	130.06
gamboostSTL_bc_RL	85.39	104.67	126.46	138.25
gamboostdetrend	81.17	72.22	75.86	78.18
gamboostdetrend_RL	81.90	73.55	75.47	76.18
gamboostdetrend_bc	79.03	68.09	73.83	75.64
gamboostdetrend_bc_RL	78.85	73.03	74.08	74.92
gamboostdetrendSD	80.05	72.12	76.92	78.16
gamboostdetrendSD_RL	80.51	72.00	75.97	76.32
gamboostdetrendSD_bc	78.62	67.85	74.33	75.34
gamboostdetrendSD_bc_RL	77.43	70.63	74.00	75.02
gamboostdetrendD	96.53	94.63	91.69	89.64
gamboostdetrendD_RL	104.04	95.13	95.68	93.07
gamboostdetrendD_bc	93.71	90.37	89.24	87.51
gamboostdetrendD_bc_RL	99.59	91.76	94.17	92.13
gamboostdetrendSTL	84.94	76.28	76.81	77.42
gamboostdetrendSTL_RL	80.52	73.43	73.46	72.96
gamboostdetrendSTL_bc	80.70	72.55	74.18	74.73
gamboostdetrendSTL_bc_RL	75.18	71.08	73.06	73.58

Table A.3: *Average Ranked MdASE corresponding to dots in Figure A.7. Best model metric per horizon range is bold & underlined and shading denote Top10% accordingly.*

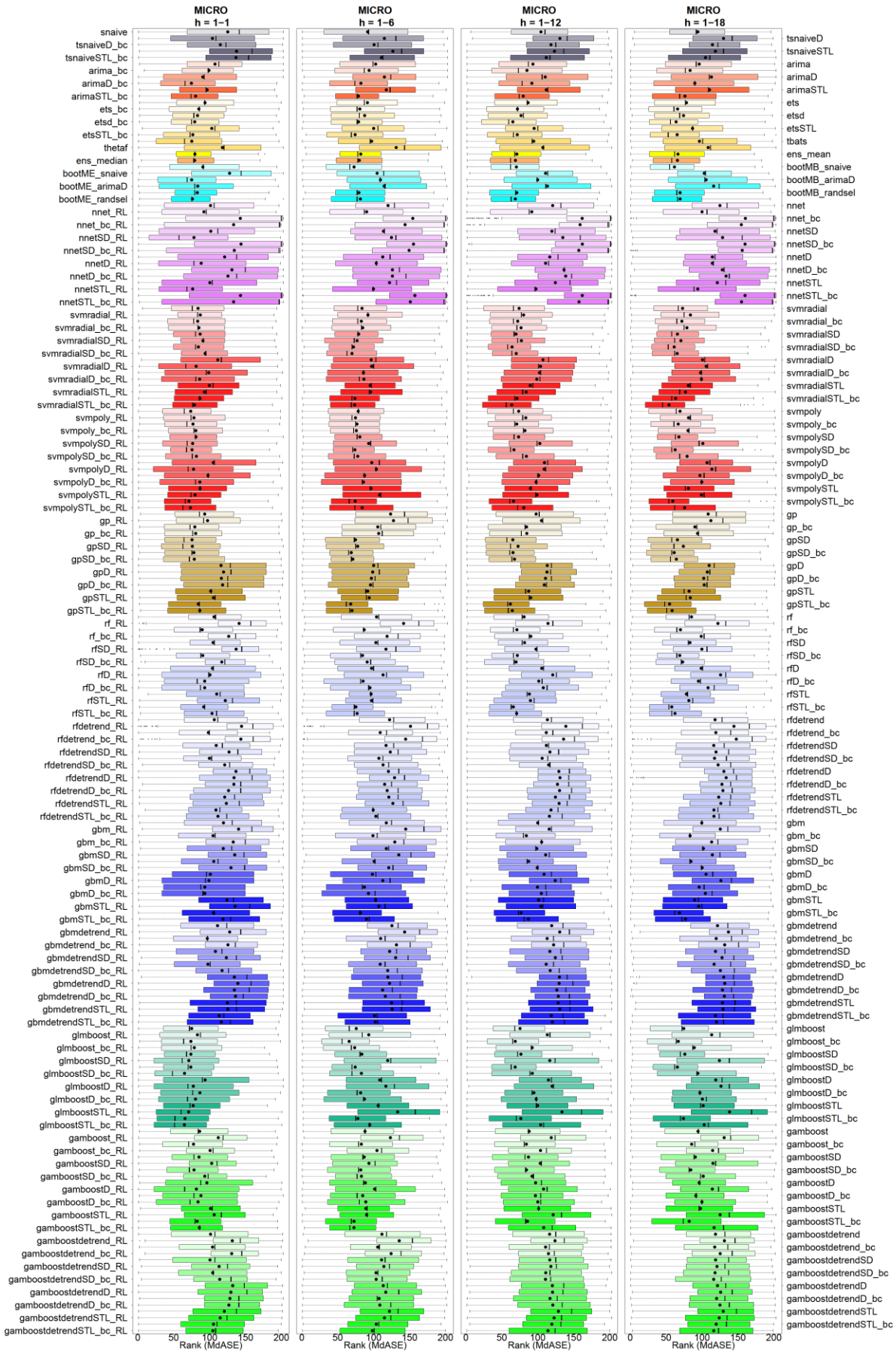

Figure A.8: *Overall Result (Ranked MdASE) by horizon range for M3-MICRO competition.*

Method	MICRO h=1-1	MICRO h=1-6	MICRO h=1-12	MICRO h=1-18
snaive	125.42	91.66	104.35	94.28
tsnaiveD	104.14	114.95	130.83	130.02
tsnaiveD_bc	115.09	100.66	118.31	114.60
tsnaiveSTL	137.74	126.85	123.24	118.73
tsnaiveSTL_bc	136.96	110.93	111.82	104.96
arima	107.70	102.77	93.24	96.23
arima_bc	99.28	93.73	84.95	83.45
arimaD	90.66	114.94	110.30	113.06
arimaD_bc	74.87	82.27	91.74	90.04
arimaSTL	96.78	117.62	111.86	110.43
arimaSTL_bc	80.78	78.77	79.75	76.24
ets	93.89	91.38	86.51	78.47
ets_bc	85.20	80.80	71.67	66.03
etsd	83.40	87.50	76.61	74.32
etsd_bc	79.46	79.03	65.23	64.03
etsSTL	103.21	100.01	94.96	86.89
etsSTL_bc	76.88	74.19	71.37	65.46
tbats	75.35	97.21	94.32	96.43
thetaf	118.03	131.42	107.05	108.58
ens_mean	79.76	82.00	70.79	66.59
ens_median	79.42	80.10	68.85	65.86
bootMB_snaive	90.85	72.78	70.24	61.66
bootME_snaive	128.01	104.91	110.84	103.19
bootMB_arimaD	75.28	109.52	99.91	105.83
bootME_arimaD	83.79	114.65	113.42	116.25
bootMB_randsel	83.21	79.57	70.79	69.59
bootME_randsel	76.30	81.52	68.70	69.53
nnet	101.53	120.08	120.71	125.06
nnet_RL	92.25	90.35	91.60	100.13
nnet_bc	142.98	154.58	161.83	160.45
nnet_bc_RL	133.65	143.80	158.92	154.79
nnetSD	101.90	113.70	119.59	118.08
nnetSD_RL	78.39	124.81	134.76	128.69
nnetSD_bc	143.92	155.47	161.98	159.98
nnetSD_bc_RL	134.61	149.46	157.04	156.11
nnetD	121.05	112.66	116.69	114.50
nnetD_RL	88.56	103.63	110.87	114.50
nnetD_bc	131.29	126.22	136.54	128.35
nnetD_bc_RL	125.71	126.23	138.78	133.56
nnetSTL	100.55	122.17	124.41	121.37
nnetSTL_RL	76.63	99.80	97.45	94.14
nnetSTL_bc	143.26	157.49	161.93	159.97
nnetSTL_bc_RL	133.58	151.09	157.44	155.06
svmradial	84.14	83.86	73.95	72.98
svmradial_RL	87.32	92.13	80.16	83.98
svmradial_bc	83.64	82.88	72.05	72.02
svmradial_bc_RL	85.31	84.82	76.68	79.10
svmradialSD	87.04	79.08	69.67	65.85
svmradialSD_RL	90.93	76.96	77.10	70.67
svmradialSD_bc	84.60	71.91	64.12	61.25
svmradialSD_bc_RL	94.01	69.83	69.98	65.49
svmradialD	111.60	96.59	107.17	100.75
svmradialD_RL	81.40	97.20	103.82	105.88
svmradialD_bc	98.95	85.83	102.27	98.02
svmradialD_bc_RL	86.30	86.42	98.59	99.28
svmradialSTL	99.79	95.14	89.26	80.91
svmradialSTL_RL	92.84	95.76	83.66	76.96
svmradialSTL_bc	86.83	73.88	70.95	63.31
svmradialSTL_bc_RL	78.97	73.50	63.63	**54.01**
svmpoly	73.91	78.53	73.31	69.43
svmpoly_RL	78.41	74.75	83.18	81.25
svmpoly_bc	77.06	76.75	70.68	66.92
svmpoly_bc_RL	81.07	76.00	82.10	80.57
svmpolySD	81.19	80.77	73.29	67.90
svmpolySD_RL	76.59	93.15	102.76	101.22
svmpolySD_bc	75.68	73.92	66.75	62.68
svmpolySD_bc_RL	81.77	77.84	84.06	79.36
svmpolyD	105.36	97.39	109.20	107.05
svmpolyD_RL	77.74	103.75	108.87	113.72
svmpolyD_bc	98.03	87.32	101.19	97.19
svmpolyD_bc_RL	86.60	85.78	97.57	99.75
svmpolySTL	86.95	96.24	89.66	81.02
svmpolySTL_RL	79.85	108.31	98.69	99.21
svmpolySTL_bc	71.53	74.22	66.10	59.24
svmpolySTL_bc_RL	73.39	84.03	80.43	75.73
gp	93.32	123.59	97.71	108.57
gp_RL	97.43	127.66	105.12	112.44
gp_bc	79.65	105.72	84.30	90.11
gp_bc_RL	80.89	107.13	84.68	93.90
gpSD	75.77	74.78	65.21	65.86
gpSD_RL	76.25	77.81	72.46	74.06
gpSD_bc	78.26	68.87	65.52	61.24
gpSD_bc_RL	78.91	71.05	67.55	64.29
gpD	116.10	100.07	113.13	109.99
gpD_RL	119.63	98.70	112.78	107.40
gpD_bc	116.65	96.92	110.66	102.36
gpD_bc_RL	118.36	95.62	109.03	103.15
gpSTL	101.79	91.75	87.20	82.12
gpSTL_RL	105.51	93.88	90.17	83.77
gpSTL_bc	84.55	68.01	**61.82**	54.88
gpSTL_bc_RL	86.53	69.85	64.48	58.34
rf	106.37	104.02	80.96	85.36
rf_RL	141.33	141.75	114.22	122.28
rf_bc	90.01	86.89	70.88	70.05
rf_bc_RL	126.78	118.81	90.25	99.03
rfSD	105.08	103.12	81.55	83.29
rfSD_RL	137.16	117.18	97.50	99.85
rfSD_bc	90.71	84.54	71.58	69.06
rfSD_bc_RL	117.17	91.02	69.85	73.06
rfD	104.17	97.16	105.43	98.88
rfD_RL	100.77	113.18	120.83	125.89
rfD_bc	93.32	85.35	101.34	95.12
rfD_bc_RL	93.38	94.89	107.68	108.49
rfSTL	110.13	96.55	88.20	78.32
rfSTL_RL	122.09	96.93	89.80	81.72
rfSTL_bc	92.07	75.28	65.89	57.85
rfSTL_bc_RL	103.56	77.00	70.54	62.42

Method	MICRO h=1-1	MICRO h=1-6	MICRO h=1-12	MICRO h=1-18
rfdetrend	106.80	122.13	113.36	118.26
rfdetrend_RL	144.64	151.32	139.05	144.51
rfdetrend_bc	98.53	109.10	111.78	119.32
rfdetrend_bc_RL	143.70	144.51	135.90	147.83
rfdetrendSD	109.42	117.26	111.87	116.40
rfdetrendSD_RL	127.35	123.23	116.89	119.51
rfdetrendSD_bc	99.96	107.34	106.05	117.53
rfdetrendSD_bc_RL	121.17	113.21	115.06	122.53
rfdetrendD	136.94	120.60	129.32	130.48
rfdetrendD_RL	134.21	129.05	130.93	133.68
rfdetrendD_bc	134.05	115.36	127.40	127.84
rfdetrendD_bc_RL	126.53	118.92	130.28	129.13
rfdetrendSTL	121.19	121.15	124.55	123.28
rfdetrendSTL_RL	123.54	126.76	129.70	126.23
rfdetrendSTL_bc	109.04	99.38	118.32	116.69
rfdetrendSTL_bc_RL	111.90	103.08	116.19	116.60
gbm	119.54	117.67	99.90	99.70
gbm_RL	140.50	144.65	115.79	125.56
gbm_bc	105.00	99.01	84.08	83.46
gbm_bc_RL	133.09	129.50	105.41	113.47
gbmSD	119.22	117.66	98.65	102.08
gbmSD_RL	135.18	135.00	110.98	114.51
gbmSD_bc	106.01	100.80	87.26	84.60
gbmSD_bc_RL	129.97	121.13	99.24	100.34
gbmD	101.30	98.32	108.80	105.72
gbmD_RL	99.38	112.85	124.38	126.34
gbmD_bc	93.59	87.11	99.50	96.12
gbmD_bc_RL	93.59	92.84	105.05	104.75
gbmSTL	124.42	102.88	101.24	89.86
gbmSTL_RL	135.76	107.30	104.36	95.49
gbmSTL_bc	105.76	81.65	76.70	68.78
gbmSTL_bc_RL	119.05	90.09	86.90	77.15
gbmdetrend	111.30	125.47	119.32	121.58
gbmdetrend_RL	128.10	143.26	130.53	136.78
gbmdetrend_bc	96.97	109.81	113.06	119.75
gbmdetrend_bc_RL	125.77	131.80	122.08	131.55
gbmdetrendSD	108.38	122.28	116.85	118.82
gbmdetrendSD_RL	123.51	130.73	124.50	127.91
gbmdetrendSD_bc	97.84	109.09	111.68	116.93
gbmdetrendSD_bc_RL	117.61	119.58	117.31	125.69
gbmdetrendD	134.77	119.86	130.19	130.24
gbmdetrendD_RL	139.61	122.18	129.74	131.68
gbmdetrendD_bc	134.78	112.51	126.49	128.55
gbmdetrendD_bc_RL	136.35	116.31	128.63	131.12
gbmdetrendSTL	125.17	124.97	128.31	128.70
gbmdetrendSTL_RL	125.61	125.99	130.64	129.40
gbmdetrendSTL_bc	113.72	101.21	118.82	118.87
gbmdetrendSTL_bc_RL	116.60	102.52	120.43	120.26
glmboost	75.28	75.81	75.31	74.66
glmboost_RL	83.31	93.25	113.26	113.43
glmboost_bc	74.14	**65.93**	68.66	67.02
glmboost_bc_RL	78.93	73.59	92.18	89.27
glmboostSD	74.17	83.55	76.54	76.09
glmboostSD_RL	71.29	119.59	116.64	124.26
glmboostSD_bc	74.04	74.42	68.54	65.86
glmboostSD_bc_RL	65.32	82.96	92.31	94.21
glmboostD	93.96	108.53	114.84	119.25
glmboostD_RL	77.56	117.11	119.92	126.52
glmboostD_bc	86.66	81.91	94.53	97.24
glmboostD_bc_RL	80.63	87.20	97.79	100.63
glmboostSTL	77.29	106.36	99.84	101.72
glmboostSTL_RL	70.93	133.48	133.67	138.40
glmboostSTL_bc	66.06	78.02	76.43	74.22
glmboostSTL_bc_RL	**65.11**	94.73	104.00	103.14
gamboost	86.30	88.04	87.93	94.56
gamboost_RL	112.40	123.35	118.78	130.93
gamboost_bc	77.75	82.84	84.13	85.58
gamboost_bc_RL	100.95	104.65	103.81	114.80
gamboostSD	85.42	86.87	86.96	90.69
gamboostSD_RL	103.14	93.46	103.47	115.28
gamboostSD_bc	78.54	82.07	84.20	84.37
gamboostSD_bc_RL	93.44	83.04	91.98	101.86
gamboostD	96.37	88.48	96.98	96.06
gamboostD_RL	81.86	101.41	108.04	114.40
gamboostD_bc	88.19	84.87	96.73	91.51
gamboostD_bc_RL	84.07	89.03	100.06	99.95
gamboostSTL	101.47	89.26	101.22	97.91
gamboostSTL_RL	106.56	90.26	121.46	125.31
gamboostSTL_bc	82.90	72.75	85.70	82.26
gamboostSTL_bc_RL	85.93	72.72	108.26	116.92
gamboostdetrend	101.57	111.57	116.43	118.79
gamboostdetrend_RL	131.85	135.66	124.00	131.29
gamboostdetrend_bc	104.26	105.62	110.81	117.76
gamboostdetrend_bc_RL	130.99	124.38	114.79	125.38
gamboostdetrendSD	100.73	111.12	116.67	118.93
gamboostdetrendSD_RL	113.47	114.23	118.24	120.87
gamboostdetrendSD_bc	104.54	103.35	111.06	118.28
gamboostdetrendSD_bc_RL	114.34	103.53	110.77	116.28
gamboostdetrendD	132.42	113.43	120.26	121.69
gamboostdetrendD_RL	129.79	116.96	120.61	126.00
gamboostdetrendD_bc	128.55	107.66	117.07	120.48
gamboostdetrendD_bc_RL	127.11	108.65	120.93	124.99
gamboostdetrendSTL	120.78	122.01	128.74	129.66
gamboostdetrendSTL_RL	114.98	115.08	122.76	124.05
gamboostdetrendSTL_bc	105.62	103.64	119.57	119.86
gamboostdetrendSTL_bc_RL	104.78	98.24	114.24	116.46

Table A.4: *Average Ranked MdASE corresponding to dots in Figure A.8. Best model metric per horizon range is bold & underlined and shading denote Top10% accordingly.*

Bibliography

Andrawis, R. R. & Atiya, A. F. & El-Shishiny, H. (2011). Forecasting combinations of computational intelligence and linear models for the NN5 time series forecasting competition. *International Journal of Forecasting 27(3), 672-688.*

Ahmed N. & Atiya, A. & Gayar, N. & El-shishiny, H. (2010). An Empirical Comparison of Machine Learning Models for Time Series Forecasting. *Journal of Econometric Reviews 29(5-6), 594-621.*

Athanasopoulos, G. & Hyndman, R. J. & Song, H. & Wu, D. C. (2011). The tourism forecasting competition. *International Journal of Forecasting 27(3), 822-844.*

Baker (2010). How I did it: Lee Baker on winning Tourism Forecasting Part One. http:// blog.kaggle. com/2010/09/27/how-i-did-it-lee-baker-on-winning-tourism-forecasting-part-one/ (visited on 01.12.2014).

Baker, L. C. & Howard, J. (2011). Winning methods for forecasting tourism time series. *International Journal of Forecasting 27(3), 850-852.*

Bergmeir, C. & Hnydman, R. J, & Benitez, J. M. (2014). Bagging Exponential Smoothing Methods using STL Decomposition and Box-Cox-Transformation. *Department of Econometrics and Business Statistics, Monash University, Working Paper 11/14, http://robjhyndman.com/papers/ BaggedETS-wpaper.pdf.*

Bergmeir, C. & Benitez, J. M. (2014). RSNNS: Neural Networks in R using the Stuttgart Neural Network Simulator (SNNS). *R package version 0.4-6.*

Bontempi, G. & Taieb, S. & Le Borgne, Y. (2013). Machine Learning Strategies for Time Series Forecasting. *Lecture Notes in Business Information Processing 138, 62-77.*

Brierly (2011a). Phil Brierley on winning tourism forecasting part two. *http://blog.kaggle.com/ 2011/03/22/phil-brierley-on-winning-tourism-forecasting-part-two/ (visited on 01.12.2014).*

Brierly (2011b). Winning methods for forecasting seasonal tourism time series. *International Journal of Forecasting 27(3), 853-854.*

Buchen, T. & Wohlrabe, K. (2010). Forecasting with many predictors - Is boosting a viable alternative? *Economics Letters 113(1), 16–18.*

Cleveland R. B. & Cleveland W. S. & McRae J. E. & Terpenning I. (1990). STL: A Seasonal-Trend Decomposition Procedure Based on Loess. *Journal of Official Statistics 6, 3–73.*

Cordeiro, C. & Neves, M. M. (2009). Forecasting Time Series with BOOT.EXPOS Procedure. *Revstat 7(2), 135-149.*

Cowpertwait, P. S. P. & Metcalfe A. V. (2009). Introductory Time Series with R. *Springer, New York*.

Cressie, N. (1993). Statistics for Spatial Data. *John Wiley & Sons*.

Crone, S. F. (2009a). Mining the past to determine the future: Comments. *International Journal of Forecasting, 25(3), 456-460.*

Crone, S. F. (2009b). NN5 forecasting competition. *http://www.neural-forecasting-competition.com/NN5 (visited on 01.12.2014).*

Crone, S. F. & Lessmann, S. & Pietsch, S. (2009). Forecasting with Computational Intelligence - An Evaluation of Support Vector Regression and Artificial Neural Networks for Time Series Prediction. *International Joint Conference on Neural Networks, ICJNN 2006, 3159-3166.*

De Livera, A. M. & Hyndman, R. J. & Snyder, R. D. (2011). Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association, 106(496), 1513-1527.*

Ebden, M. (2008). Gaussian processes for regression: A quick introduction. *The Website of Robotics Research Group in Department on Engineering Science, University of Oxford* (2008).

Fahrmeir, L. & Kneib, T. & Lang S. & Marx B. (2013). Regression. Models, Methods and Applications. *Springer, Heidelberg.*

Fan, S. & Hyndman, R. J. (2010). Short-term load forecasting based on a semi-parametric additive model. *IEEE Transactions on Power Systems 27(1).*

Fanaee-T, H. & Gama, J. (2013). Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence (2013), 1-15, Springer Berlin.*

Gooijer, J. G. & Hyndman, R. J. (2006). *25 years of time series forecasting. International Journal of Forecasting 22(1), 443-473.*

Guerrero, V.M. (1993). Time-series analysis supported by power transformations. *Journal of Forecasting 12, 37-48*.

Härdle, W. & Horowitz, J. & Kreiss, J.-P. (2003). Bootstrap Methods for Time Series. *International Statistical Review 71, 435-459.*

Hastie, T.& Tibshirani, R. & Friedman, J. (2009*).* The Elements of Statistical Learning. Data Mining, Inference and Prediction, Second edition. *Springer, New York.*

Hong, T. & Pinson, P. & Fan, S. (2014). Global Energy Forecasting Competition 2012. *International Journal of Forecasting 30(2), 357-363.*

Hong, T. (2012). Global Energy Forecasting Competition 2012. *http://www.drhongtao.com/gefcom/ 2012 (visited on 01.12.2014).*

Hothorn, T. & Buehlmann, P.. & Kneib, T. & Schmid, M. & Hofner, B. & Sobotka, F. & Scheipl, F. (2015). mboost: Model-Based Boosting. *R package version 2.4.2.*

Hyndman, R.J. & and Billah, B. (2003). Unmasking the Theta method. *International Journal of Forecasting, 19, 287-290.*

Hyndman, R. J & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting 22(4), 679-688.*

Hyndman R. J. & Koehler, A. B. & Ord, J. K. & Snyder R. D. (2008). Forecasting with Exponential Smoothing: The State Space Approach. *Springer, Berlin.*

Hyndman, R. J. & Athanasopoulos, G. (2014). Forecasting. Principles and Practice. *OTexts.*

Hyndman, R. J. (2015a). forecast: Forecasting functions for time series and linear models. *R package version 5.8.*

Hyndman, R. J. (2015b). Hyndsight. A blog by Rob J. Hyndman*. http://robjhyndman.com/hyndsight/ (visited on 01.12.2014).*

Kandananond, K. (2012). A Comparison of Various Forecasting Methods for Autocorrelated Time Series. *International Journal of Engineering Business Management 2012(4).*

Karatzoglou, A. & Smola, A. & Hornik, K. (2015). kernlab: Kernel-based Machine Learning Lab. *R package version 0.9.20.*

Kourentzes, N. & Barrow, D. K. & Crone, S. F. (2014). Neural Network Ensemble Operators for Time Series Forecasting. *Expert Systems with Applications 41(9), 4235-4244.*

Kreiss, J. P. & Lahiri, S. N. (2012). Bootstrap Methods for Time Series*. Handbook of Statistics 30, Time Series Analysis: Methods and Applications.*

Krollner, B. & Vanstone, B. & Finnie, G. (2010). Financial time series forecasting with machine learning techniques: A survey. *18th European Symposium on Artificial Neural Networks: Computational and Machine Learning, Bruges (Belgium), 25-30.*

Kuhn, M. & Johnson, K. (2013). Applied Predictive Modeling. *Springer, New York.*

Kuhn, M. (2014). caret: Classification and Regression Training. *R package version 6.0-35.*

Ladiray, D. & Quenneville, B. (2001). Seasonal Adjustment with the X-11 Method. *Lecture Notes in Statistics, 2001. Springer, New York.*

Liaw, A. & Wiener, M. (2014). randomForest: Breiman and Cutler's random forests for classification and regression. *R package version 4.6-10.*

Lloyd, J. R. (2014). GEFCom2012 hierarchical load forecasting: Gradient boosting machines and Gaussian processes. *International Journal of Forecasting 30(2), 369-374.*

Lora, A. T. & Santos, J. M. R. & Riquelme, J. C. & Exposito, A. G. & Ramos, J. L. M. (2004). Time-Series Prediction: Application to Short-Term Electric Energy Demand. *Current Topics in Artificial Intelligence 3040, 577-586.*

Makridakis, M. & Hibon, M. (2000). The M3-Competition: results, conclusions and implications. *International Journal of Forecasting 16(4), 451-476.*

Mammen, E. & Nandi, S. (2012). Bootstrap and Resampling. *Handbook of Computational Statistics: Concepts and Methods, 499-527.*

Mateo, F. & Carrasco, J. & Sellami, A & Millán-Giraldo, M. & Domínguez, M. & Soria-Olivas, E. (2013). Machine learning methods to forecast temperature in buildings. *Expert Systems with Applications 40(4),1061-1068.*

Murphy, K.P. (2012). Machine Learning. A Probabilistic Perspective. *MIT Press, Cambridge.*

Pegels, C. C. (1969). Exponential forecasting: some new variations. *Management Science, 12, 311-315.*

Politis, D. N. & Romano, J. P. (1994). The Stationary Bootstrap. *Journal of the American Statistical Association 89(428).*

Ridgeway, G. (2015). gbm: Generalized Boosted Regression Models. *R package version 2.1.1.*

Robinzonov, N. & Tutz, G. & Hothorn, T. (2010). Boosting Techniques for Nonlinear Time Series Models. *AStA Advances in Statistical Analysis 96, 99–122.*

Ruppert D. (2010). Statistics and Data Analysis for Financial Engineering. *Springer, New York.*

Schölkpf, B. & Smola, A. J. (2002). Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond. *MIT Press, Cambridge.*

Shafik, N. & Tutz, G. (2007). Boosting Nonlinear Additive Autoregressive Time Series. *Computational Statistics & Data Analysis 53(7), 2453–2464.*

Shumway, R. H. & Stoffer, D. S. (2011). Time Series Analysis and its Applications. With R Examples. Third Edition. *Springer, New York.*

Taieb, S. B. & Hyndman, R. J. (2012a). Recursive and direct multi-step forecasting: the best of both worlds. *Department of Econometrics and Business Statistics, Monash University, Working Paper 19/12, http://robjhyndman.com/papers/rectify.pdf.*

Taieb, S. & Hyndman, R. J. (2012b). A gradient boosting approach to the Kaggle load forecasting competition. *International Journal of Forecasting 30(2) 382-394.*

Taieb, S. B. & Bontempi, G. & Atiya, A. & Sorjamaa, A. (2012c). 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.*

Taieb, S. & Hyndman, R. J. (2014). Boosting multi-step autoregressive forecasts. *Proceedings of the 31st International Conference on Machine Learning 32, Beijing (China).*

Tutz, G. (2012). Regression for Categorical Data. *Cambridge University Press, New York.*

Verbeke, G. & Moolenbergs, G. (2008). Linear Mixed Model for Longitudinal Data. *Springer, New York.*

Vinod, H. D. (2006). Maximum Entropy Ensembles for Time Series Inference in Economics. *Journal of Asian Economics, 17(6).*

Vinod, H. D. & Lopez-de-Lacalle, J. (2009). Maximum Entropy Bootstrap for Time Series: The meboot R package. *Journal of Statistical Software, 29(5)*.

Wildi. M. (2010). Real-Time Signal Extraction: a Shift of Perspective (2010). *Estudios de Economia Applicada, 28 (3), 497-518.*

Zhang, L. & Zhou, W. & Chang, P. & Yang, J. & Li, F. (2013). Iterated time series prediction with multiple support vector regression models. *Neurocomputing 99, 411–422.*

Declaration

I hereby declare that I have written this thesis without help from others, using only the sources listed in the bibliography.

Munich, 27.05.2015

……..

Uwe Pritzsche