Expert Systems with Applications 39 (2012) 7067–7083

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition

Souhaib Ben Taieb [a,*], Gianluca Bontempi [a], Amir F. Atiya [c], Antti Sorjamaa [b]

[a] Machine Learning Group, Département d'Informatique, Faculté des Sciences, Université Libre de Bruxelles, Belgium
[b] Environmental and Industrial Machine Learning Group, Adaptive Informatics Research Centre, Altoo University School of Science, Finland
[c] Faculty of Engineering, Cairo University, Giza, Egypt

ARTICLE INFO

Keywords:
Time series forecasting
Multi-step ahead forecasting
Long-term forecasting
Strategies of forecasting
Machine learning
Lazy Learning
NN5 forecasting competition
Friedman test

ABSTRACT

Multi-step ahead forecasting is still an open challenge in time series forecasting. Several approaches that deal with this complex problem have been proposed in the literature but an extensive comparison on a large number of tasks is still missing. This paper aims to fill this gap by reviewing existing strategies for multi-step ahead forecasting and comparing them in theoretical and practical terms. To attain such an objective, we performed a large scale comparison of these different strategies using a large experimental benchmark (namely the 111 series from the NN5 forecasting competition). In addition, we considered the effects of deseasonalization, input variable selection, and forecast combination on these strategies and on multi-step ahead forecasting at large. The following three findings appear to be consistently supported by the experimental results: Multiple-Output strategies are the best performing approaches, deseasonalization leads to uniformly improved forecast accuracy, and input selection is more effective when performed in conjunction with deseasonalization.

1. Introduction

Time series forecasting is a growing field of interest playing an important role in nearly all fields of science and engineering, such as economics, finance, meteorology and telecommunication (Palit & Popovic, 2005). Unlike one-step ahead forecasting, multi-step ahead forecasting tasks are more difficult (Tiao & Tsay, 1994), since they have to deal with various additional complications, like accumulation of errors, reduced accuracy, and increased uncertainty (Sorjamaa, Hao, Reyhani, Ji, & Lendasse, 2007; Weigend & Gershenfeld, 1994).

The forecasting domain has been influenced, for a long time, by linear statistical methods such as ARIMA models. However, in the late 1970s and early 1980s, it became increasingly clear that linear models are not adapted to many real applications (De Gooijer & Hyndman, 2006). In the same period, several useful nonlinear time series models were proposed such as the bilinear model (Poskitt & Tremayne, 1986), the threshold autoregressive model (Tong, 1983, 1990; Tong & Lim, 1980) and the autoregressive conditional heteroscedastic (ARCH) model (Engle, 1982) (see De Gooijer & Hyndman, 2006; De Gooijer & Kumar, 1992 for a review). Nowadays, Monte Carlo simulation or bootstrapping methods are used to compute nonlinear forecasts. Since no assumptions are made about the distribution of the error process, the latter approach is preferred (Clements, Franses, & Swanson, 2004; De Gooijer & Hyndman, 2006). However, the study of nonlinear time series analysis and forecasting is still in its infancy compared to the development of linear time series (De Gooijer & Hyndman, 2006).

In the last two decades, machine learning models have drawn attention and have established themselves as serious contenders to classical statistical models in the forecasting community (Ahmed, Atiya, El Gayar, & El-Shishiny, 2010; Palit & Popovic, 2005; Zhang, Eddy Patuwo, & Hu, 1998). These models, also called black-box or data-driven models (Mitchel, 1997), are examples of nonparametric nonlinear models which use only historical data to learn the stochastic dependency between the past and the future. For instance, Werbos found that Artificial Neural Networks (ANNs) outperforms the classical statistical methods such as linear regression and Box–Jenkins approaches (Werbos, 1974, 1988). A similar study has been conducted by Lapedes and Farber (1987) who conclude that ANNs can be successfully used for modeling and forecasting nonlinear time series. Later, others models appeared such as decision trees, support vector machines and nearest neighbor regression (Alpaydin, 2010; Hastie, Tibshirani, & Friedman, 2009). Moreover, the empirical accuracy of several machine learning models has been explored in a number of forecasting competitions under different data conditions (e.g. the NN3, NN5, and the annual ESTSP competitions (Crone, 2009a, 2009b; Lendasse, 2007, 2008)) creating interesting scientific debates in

* Corresponding author.
E-mail address: sbentaie@ulb.ac.be (S. Ben Taieb).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2012.01.039

the area of data mining and forecasting (Crone, 2009; Hand, 2008; Price, 2009).

In the forecasting community, researchers have paid attention to several aspects of the forecasting procedure such as model selection (Aha, 1997; Anders & Korn, 1999; Chapelle & Vapnik, 2000; Curry & Morga, 2006), effect of deseasonalization (Hylleberg, 1992; Makridakis, Wheelwright, & Hyndman, 1998; Nelson, Hill, Remus, & O'Connor, 1999; Zhang & Qi, 2005), forecasts combination (Bates & Granger, 1969; Clemen, 1989; Timmermann, 2006) and many other critical topics (De Gooijer & Hyndman, 2006). However, approaches for generating multi-step ahead forecasts for machine learning models did not receive as much attention, as pointed out by Kline: "*One issue that has had limited investigation is how to generate multiple-step-ahead forecasts*" (Kline, 2004).

To the best of our knowledge, five alternatives (or strategies) have been proposed in the literature to tackle an H-step ahead forecasting task. The *Recursive* strategy (Cheng, Tan, Gao, & Scripps, 2006; Hamzacebi, Akay, & Kutay, 2009; Kline, 2004; Sorjamaa et al., 2007; Tiao & Tsay, 1994; Weigend & Gershenfeld, 1994) iterates, H times, a one-step ahead forecasting model to obtain the H forecasts. After estimating the future series value, it is fed back as an input for the following forecast.

In contrast to the previous strategy which use a single model, the *Direct* strategy (Cheng et al., 2006; Hamzacebi et al., 2009; Kline, 2004; Sorjamaa et al., 2007; Tiao & Tsay, 1994; Weigend & Gershenfeld, 1994) estimates a set of H forecasting models, each returning a forecast for the ith value ($i \in \{1,\ldots,H\}$).

A combination of the two previous strategies, called *DirRec* strategy has been proposed in Sorjamaa & Lendasse (2006). The idea behind this strategy is to combine aspects from both, the Direct and the Recursive strategies. In other words, a different model is used at each step but the approximations from previous steps are introduced into the input set.

In order to preserve, between the predicted values, the stochastic dependency characterizing the time series, the Multi-Input Multi-Output (*MIMO*) strategy has been introduced and analyzed in Bontempi (2008), Bontempi and Ben Taieb (2011) and Kline (2004). Unlike the previous strategies where the models return a scalar value, the MIMO strategy returns a vector of future values in a single step.

The last strategy, called *DIRMO* (Ben Taieb, Bontempi, Sorjamaa, & Lendasse, 2009), aims to preserve the most appealing aspects of both the DIRect and miMO strategies. This strategy aims to find a trade-off between the property of preserving the stochastic dependency between the forecasted values and the flexibility of the modeling procedure.

In the literature, these five forecasting strategies have been presented separately, sometimes, using different terminologies. The *first contribution* of this paper is to present a thorough unified review as well as a theoretical comparative analysis of the existing strategies for multi-step ahead forecasting.

Despite the fact that many studies have compared between the different multi-step ahead approaches, the collective outcome of these studies regarding forecasting performance has been inconclusive. So the modeler is still left with little guidance as to which strategy to use. For example, research from Bontempi, Birattari, and Bersini (1999) and Weigend, Huberman, and Rumelhart (1992) provide experimental evidence in favor of Recursive strategy against Direct strategy. However, results from Zhang and Hutchinson (1994), Sorjamaa et al. (2007) and Hamzacebi et al. (2009) support the fact that the Direct strategy is better than the Recursive strategy. The work by Sorjamaa and Lendasse (2006) shows that the DirRec strategy gives better performance than Direct and Recursive strategies. The Direct and Recursive strategies have been theoretically and empirically compared in Atiya, El-shoura, Shaheen, and El-sherif (1999). In this study the authors obtained theoretical and experimental evidence in favor of Direct strategy. Concerning the MIMO strategy, Kline (2004) and Cheng et al. (2006) support the idea that MIMO strategy provides worse forecasting performance than Recursive and Direct strategies. However, in Bontempi (2008) and Bontempi and Ben Taieb (2011), the comparison between MIMO, Recursive, and Direct was in favor of MIMO. Finally, Ben Taieb et al. (2009) and Ben Taieb, Sorjamaa, and Bontempi (2010) show that the DIRMO strategy gives better forecasting results than Direct and MIMO strategies when the parameter controlling the degree of dependency between forecasts is correctly identified. These previous comparisons have been performed with different datasets in different configurations using different forecasting methods, such as Multiple Linear Regression, Artificial Neural Networks, Hidden Markov Models and Nearest Neighbors.

All the contradictory findings of these studies make it all the more necessary to investigate further to find the truth concerning the relative performance of these strategies. The *second contribution* of this paper is an experimental comparison of the different multi-step ahead forecasting strategies on the 111 time series of the NN5 international forecasting competition benchmark. These time series pose some of the realistic problems that one usually encounters in a typical multi-step ahead forecasting task, for example the existence of several times series of possibly related dynamics, outliers, missing values, and multiple overlying seasonalities. This experimental comparison is performed for a variety of different configurations (regarding seasonality, input selection and combination), in order to have the comparison as encompassing as can be. In addition, the methodology used for this experimental comparison is based on the guidelines and recommendations advocated in some of the methodological papers (Demšar, 2006; Garca & Herrera, 2009).

In other words, the aim of this paper is not to make a comparison of machine learning algorithms for forecasting (which was already conducted in Ahmed et al. (2010)) but rather to show for a given learning algorithm, how the choice of the forecasting strategy can sensibly influence the performance of the multi-step ahead forecasts. In this work, we adopted the Lazy Learning algorithm (Aha, 1997), a particular instance of local learning, which has been successfully applied to many real-world forecasting tasks (Bontempi, Birattari, & Bersini, 1998; McNames, 1998; Sauer, 1994).

Last but not least, the paper proposes also a Lazy Learning entry to the NN5 forecasting competition (Crone, 2009b). The goal is to assess how this model fares compared to the other computational intelligence models that were proposed for the competition (Bontempi & Ben Taieb, in press). This will give us an idea about the potential of this approach.

The paper is organized as follows. The next section presents a review of the different forecasting strategies. Section 3 describes the Lazy Learning model and the associated algorithms for the different forecasting strategies. Section 4 gives a detailed presentation of the datasets and the methodology applied for the experimental comparison. Section 5 presents the results and discusses them. Finally, Section 6 gives a summary and concludes the work.

2. Strategies for multi-step-ahead time series forecasting

A multi-step ahead (also called long-term) time series forecasting task consists of predicting the next H values $[y_{N+1},\ldots,y_{N+H}]$ of a historical time series $[y_1,\ldots,y_N]$ composed of N observations, where $H > 1$ denotes the forecasting horizon.

This section will first give a presentation of the five forecasting strategies and next, a subsection will be devoted to a comparative analysis of these strategies in terms of number and types of models to learn as well as forecasting properties.

We will use a common notation where f and F denote the functional dependency between past and future observations, d refers to the embedding dimension (Casdagli, Eubank, Farmer, & Gibson, 1991) of the time series, that is the number of past values used to predict future values and w represents the term that includes modeling error, disturbances and/or noise.

2.1. Recursive strategy

The oldest and most intuitive forecasting strategy is the *Recursive* (also called *Iterated* or *Multi-Stage*) strategy (Cheng et al., 2006; Hamzacebi et al., 2009; Kline, 2004; Sorjamaa et al., 2007; Tiao & Tsay, 1994; Weigend & Gershenfeld, 1994). In this strategy, a single model f is trained to perform a *one-step ahead* forecast, i.e.

$$y_{t+1} = f(y_t, \ldots, y_{t-d+1}) + w, \tag{1}$$

with $t \in \{d, \ldots, N-1\}$.

When forecasting H steps ahead, we first forecast the first step by applying the model. Subsequently, we use the value just forecasted as part of the input variables for forecasting the next step (using the same one-step ahead model). We continue in this manner until we have forecasted the entire horizon.

Let the trained one-step ahead model be $\hat{f}$. Then the forecasts are given by:

$$\hat{y}_{N+h} = \begin{cases} \hat{f}(y_N, \ldots, y_{N-d+1}) & \text{if } h = 1, \\ \hat{f}(\hat{y}_{N+h-1}, \ldots, \hat{y}_{N+1}, y_N, \ldots, y_{N-d+h}) & \text{if } h \in \{2, \ldots, d\}, \\ \hat{f}(\hat{y}_{N+h-1}, \ldots, \hat{y}_{N+h-d}) & \text{if } h \in \{d+1, \ldots, H\}. \end{cases} \tag{2}$$

Depending on the noise present in the time series and the forecasting horizon, the Recursive strategy may suffer from low performance in multi-step ahead forecasting tasks. Indeed, this is especially true if the forecasting horizon h exceeds the embedding dimension d, as at some point all the inputs are forecasted values instead of actual observations (Eq. (2)). The reason for the potential inaccuracy is that the Recursive strategy is sensitive to the accumulation of errors with the forecasting horizon. Errors present in intermediate forecasts will propagate forward as these forecasts are used to determine subsequent forecasts.

In spite of these limitations, the Recursive strategy has been successfully used to forecast many real-world time series by using different machine learning models, like recurrent neural networks (Saad, Prokhorov, & Wunsch, 1998) and nearest-neighbors (Bontempi et al., 1999; McNames, 1998).

2.2. Direct strategy

The *Direct* (also called Independent) strategy (Cheng et al., 2006; Hamzacebi et al., 2009; Kline, 2004; Sorjamaa et al., 2007; Tiao & Tsay, 1994; Weigend & Gershenfeld, 1994) consists of forecasting each horizon independently from the others. In other terms, H models f_h are learned (one for each horizon) from the time series $[y_1, \ldots, y_N]$ where

$$y_{t+h} = f_h(y_t, \ldots, y_{t-d+1}) + w, \tag{3}$$

with $t \in \{d, \ldots, N-H\}$ and $h \in \{1, \ldots, H\}$.

The forecasts are obtained by using the H learned models $\hat{f}_h$ as follows:

$$\hat{y}_{N+h} = \hat{f}_h(y_N, \ldots, y_{N-d+1}). \tag{4}$$

This implies that the Direct strategy does not use any approximated values to compute the forecasts (Eq. (4)), being then immune to the accumulation of errors. However, the H models are learned independently inducing a conditional independence of the H forecasts. This affects the forecasting accuracy as it prevents the strategy from considering complex dependencies between the variables $\hat{y}_{N+h}$ (Bontempi, 2008; Bontempi & Ben Taieb, 2011; Kline, 2004). For example consider a case where the best forecast is a linear or mildly nonlinear trend. The direct method could yield a broken curve because of the "uncooperative" way the H forecasts are generated. Also, this strategy demands a large computational time since there are as many models to learn as the size of the horizon.

Different machine learning models have been used to implement the Direct strategy for multi-step ahead forecasting tasks, for instance neural networks (Kline, 2004), nearest neighbors (Sorjamaa et al., 2007) and decision trees (Tran, Yang, & Tan, 2009).

2.3. DirRec strategy

The *DirRec* strategy (Sorjamaa & Lendasse, 2006) combines the architectures and the principles underlying the Direct and the Recursive strategies. DirRec computes the forecasts with different models for every horizon (like the Direct strategy) and, at each time step, it enlarges the set of inputs by adding variables corresponding to the forecasts of the previous step (like the Recursive strategy). However, note that unlike the two previous strategies, the embedding size d is not the same for all the horizons. In other terms, the DirRec strategy learns H models f_h from the time series $[y_1, \ldots, y_N]$ where

$$y_{t+h} = f_h(y_{t+h-1}, \ldots, y_{t-d+1}) + w, \tag{5}$$

with $t \in \{d, \ldots, N-H\}$ and $h \in \{1, \ldots, H\}$.

To obtain the forecasts, the H learned models are used as follows:

$$\hat{y}_{N+h} = \begin{cases} \hat{f}_h(y_N, \ldots, y_{N-d+1}) & \text{if } h = 1, \\ \hat{f}_h(\hat{y}_{N+h-1}, \ldots, \hat{y}_{N+1}, y_N, \ldots, y_{N-d+1}) & \text{if } h \in \{2, \ldots, H\}. \end{cases} \tag{6}$$

This strategy outperformed the Direct and the Recursive strategies on two real-world time series: Santa Fe and Poland Electricity Load data sets (Sorjamaa & Lendasse, 2006). Few research has been done regarding this strategy, so there is a need for further evaluation.

2.4. MIMO strategy

The three previous strategies (Recursive, Direct and DirRec) may be considered as Single-Output strategies (Ben Taieb, Sorjamaa, & Bontempi, 2010) since they model the data as a (multiple-input) Single-Output function (see Eqs. (2), (4) and (6)).

The introduction of the *Multi-Input Multi-Output* (MIMO) strategy Bontempi (2008); Bontempi and Ben Taieb (2011) (also called Joint strategy Kline (2004)) has been motivated by the need to avoid the modeling of Single-Output mapping, which neglects the existence of stochastic dependencies between future values and consequently affects the forecast accuracy Bontempi (2008); Bontempi and Ben Taieb (2011).

The MIMO strategy learns one Multiple-Output model F from the time series $[y_1, \ldots, y_N]$ where

$$[y_{t+H}, \ldots, y_{t+1}] = F(y_t, \ldots, y_{t-d+1}) + \mathbf{w}, \tag{7}$$

with $t \in \{d, \ldots, N-H\}$, $F : \mathbb{R}^d \to \mathbb{R}^H$ is a vector-valued function (Micchelli & Pontil, 2005), and $\mathbf{w} \in \mathbb{R}^H$ is a noise vector with a covariance that is not necessarily diagonal (Matías, 2005).

The forecasts are returned in one step by a Multiple-Output model $\widehat{F}$ where

$$[\hat{y}_{t+H}, \ldots, \hat{y}_{t+1}] = \widehat{F}(y_N, \ldots, y_{N-d+1}). \tag{8}$$

The rationale of the MIMO strategy is to preserve, between the predicted values, the stochastic dependency characterizing the time series. This strategy avoids the conditional independence assumption made by the Direct strategy as well as the accumulation of errors from which plagues the Recursive strategy. So far, this strategy has been successfully applied to several real-world multi-step ahead time series forecasting tasks (Ben Taieb et al., 2009, 2010; Bontempi, 2008; Bontempi & Ben Taieb, 2011).

However, the need to preserve the stochastic dependencies by using one model has a drawback as it constrains all the horizons to be forecasted with the same model structure. This constraint could reduce the flexibility of the forecasting approach (Ben Taieb et al., 2009). This was the motivation for the introduction of a new Multiple-Output strategy: DIRMO (Ben Taieb et al., 2009, 2010), presented next.

2.5. DIRMO strategy

The DIRMO strategy (Ben Taieb et al., 2009) aims to preserve the most appealing aspects of both the DIRect and miMO strategies. Taking a middle approach, DIRMO forecasts the horizon H in blocks, where each block is forecasted in a MIMO fashion. Thus, the H-step-ahead forecasting task is decomposed into n Multiple-Output forecasting tasks ($n = \frac{H}{s}$), each with an output of size s ($s \in \{1,\ldots,H\}$).

When the value of the parameter s is 1, the number of forecasting tasks n is equal to H which correspond to the Direct strategy. When the value of the parameter s is H, the number of forecasting tasks n is equal to 1 which correspond to the MIMO strategy. There are intermediate configurations between these two extremes depending on the value of a parameter s.

The tuning of the parameter s allows us to improve the flexibility of the MIMO strategy by calibrating the dimensionality of the outputs (no dependency in the case $s = 1$ and maximal dependency for $s = H$). This provides a beneficial trade off between the preserving a larger degree of the stochastic dependency between future values and having a greater flexibility of the predictor.

The DIRMO strategy, previously called MISMO strategy Ben Taieb et al. (2009) (renamed for clarity reason), learns n models F_p from the time series $[y_1,\ldots,y_N]$ where

$$[y_{t+p*s},\ldots,y_{t+(p-1)*s+1}] = F_p(y_t,\ldots,y_{t-d+1}) + \mathbf{w}, \tag{9}$$

with $t \in \{d,\ldots,N-H\}$, $p \in \{1,\ldots,n\}$ and $F_p : \mathbb{R}^d \to \mathbb{R}^s$ is a vector-valued function if $s > 1$.

The H forecasts are returned by the n learned models as follows:

$$[\hat{y}_{N+p*s},\ldots,\hat{y}_{N+(p-1)*s+1}] = \widehat{F}_p(y_N,\ldots,y_{N-d+1}). \tag{10}$$

The DIRMO strategy has been successfully applied to two forecasting competitions: ESTSP'07 (Ben Taieb et al., 2009) and NN3 (Ben Taieb et al., 2010).

2.6. Comparative analysis

To summarize, there are five possible forecasting strategies that perform a multi-step ahead forecasting task: *Recursive*, *Direct*, *DirRec*, *MIMO* and *DIRMO* strategies. Fig. 1 shows the different forecasting strategies with links indicating their relationships.

As we see, the DirRec strategy is a combination of the Direct and the Recursive strategy, while the DIRMO strategy is a combination of the Direct and the MIMO strategy.

Contingent on the selected strategy, a different number and type of models will be required. Before presenting the general comparison of the multi-step ahead forecasting strategies, let us highlight using an example the differences between the forecasting strategies.

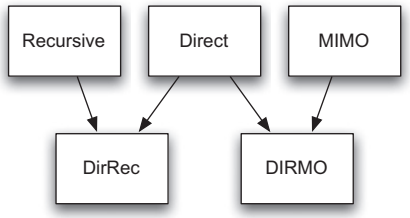

Fig. 1. The different forecasting strategies with the links showing their relationship.

Consider a multi-step ahead forecasting task for the time series $[y_1,\ldots,y_N]$ where the forecasting horizon H is 4. Table 1 shows, for each strategy, the different input sets and forecasting models involved in the calculation of the four forecasts $[\hat{y}_{N+1},\ldots,\hat{y}_{N+4}]$.

Let T_{SO} and T_{MO} denote the amount of computational time needed to learn (with a given learning algorithm) a Single-Output model and a Multiple-Output model, respectively. For a given H-step ahead forecasting task, we can see in Table 2 for each strategy the number and type of models to learn, the size of the output for each model as well as the computational time.

Suppose $T_{MO} = T_{SO} + \delta$, which is a reasonable assumption because learning a model with a vector-valued output takes more time than learning a model with a single-valued output. This allows us to rank the forecasting strategies according to their computation time for training given in Table 2. Indeed, we have

$$\underbrace{1 \times T_{SO}}_{Recursive} < \underbrace{1 \times T_{MO}}_{MIMO} < \underbrace{\frac{H}{s} \times T_{MO}}_{DIRMO} < \underbrace{H \times T_{SO}}_{Direct} < \underbrace{H \times (T_{SO} + \mu)}_{DirRec}, \tag{11}$$

where we suppose that the parameter s of DIRMO is not equal to 1 or H.

Note, in one hand, that the time needed to learn a SO model of the DirRec strategy equals $T_{SO} + \mu$ because the input size of each SO task increases at each step. On the other hand, if we need to select the value of the parameter s by on some tuning, the DIRMO strategy will take more time and hence will be the slowest one.

In the following, we conclude this section by summarizing the pros and cons of the five forecasting strategies as depicted on Table 3.

3. Lazy Learning for time series forecasting

Each of the forecasting strategies introduced in Section 2 demands the definition of a specific forecasting model or learning algorithm to estimate either the scalar-valued function f (see Eqs. (1), (3) and (5)) or the vector-valued function F (see Eqs. (7) and (9)) which represent the temporal stochastic dependencies. As the goal of the paper is not to compare forecasting models (as in Ahmed et al. (2010)) but rather multi-step ahead forecasting strategies, the choice of a underlying forecasting model is required to setup the experiments. In this paper, we adopted the Lazy Learning algorithm, which is a particular instance of local learning models, since it has been shown to be particularly effective in time series forecasting tasks (Ben Taieb et al., 2009, 2010; Bontempi, 1999, 2008; Bontempi & Ben Taieb, 2011; Bontempi et al., 1998).

Next section gives a general comparison of global models with local models. Section 3.2 presents the Lazy Learning algorithm in terms of learning properties. Sections 3.3 and 3.4 describe two Lazy Learning algorithms for two types of learning tasks, namely the Single-Output and Multiple-Output Lazy Learning algorithms. Finally, the a discussion is presented on the model combination.

Table 1
The different forecasting models used by each strategy to obtain the 4 forecasts needed.

	$\hat{y}_{N+1}$	$\hat{y}_{N+2}$	$\hat{y}_{N+3}$	$\hat{y}_{N+4}$
Recursive	$\hat{f}(y_N,\ldots,y_{N-d+1})$	$\hat{f}(\hat{y}_{N+1},y_N,\ldots,y_{N-d+2})$	$\hat{f}(\hat{y}_{N+2},\hat{y}_{N+1},\ldots,y_{N-d+3})$	$\hat{f}(\hat{y}_{N+3},\hat{y}_{N+2},\ldots,y_{N-d+4})$
Direct	$\hat{f}_1(y_N,\ldots,y_{N-d+1})$	$\hat{f}_2(y_N,\ldots,y_{N-d+1})$	$\hat{f}_3(y_N,\ldots,y_{N-d+1})$	$\hat{f}_4(y_N,\ldots,y_{N-d+1})$
DirRec	$\hat{f}_1(y_N,\ldots,y_{N-d+1})$	$\hat{f}_2(\hat{y}_{N+1},y_N,\ldots,y_{N-d+1})$	$\hat{f}_3(\hat{y}_{N+2},\hat{y}_{N+1},\ldots,y_{N-d+1})$	$\hat{f}_4(\hat{y}_{N+3},\hat{y}_{N+2},\ldots,y_{N-d+1})$
MIMO	$\widehat{F}(y_N,\ldots,y_{N-d+1})$			
DIRMO ($s=2$)	$\widehat{F}_1(y_N,\ldots,y_{N-d+1})$		$\widehat{F}_2(y_N,\ldots,y_{N-d+1})$	

Table 2
For each forecasting strategy: the number and type of models (Single-Output or Multiple-Output) to learn and the size of the output for each model.

	Number of models	Types of models	Size of output	Computational time
Recursive	1	SO	1	$1 \times T_{SO}$
Direct	H	SO	1	$H \times T_{SO}$
DirRec	H	SO	1	$H \times (T_{SO}+\mu)$
MIMO	1	MO	H	$1 \times T_{MO}$
DIRMO	$\frac{H}{s}$	MO	s	$\frac{H}{s} \times T_{MO}$

3.1. Global vs local modeling for supervised learning

Forecasting the future values of a time series using past observations can be reduced to a supervised learning problem or, more precisely, to a regression problem. Indeed, the time series can be seen as a dataset made of pairs where the first component, called input, is a past temporal pattern and the second, called output, is the corresponding future pattern. Being able to predict the unknown output for a given input is equivalent to forecasting the future values given the last observations of the time series.

Global modeling is the typical approach to the supervised learning problem. Global models are parametric models that describe the relationship between the inputs and the output values as an analytical function over the whole input domain. Examples of global models are linear models (Montgomery, Peck, & Vining, 2006), nonlinear statistical regressions (Seber & Wild, 1989) and Neural Networks (Rumelhart, Hinton, & Williams, 1986).

Another approach is the *divide-and-conquer* modeling which consists in relaxing the global modeling assumptions by dividing a complex problem into simpler problems, whose solutions can be combined to yield a solution to the original problem (Bontempi, 1999). The divide-and-conquer has evolved in two different paradigms: the *modular architectures* and the *local modeling* approach (Bontempi, 1999).

Modular techniques replace a global model with different modules covering different parts of the input space. Examples based on this approach are Fuzzy Inference Systems (Takagi & Sugeno, 1985), Radial Basis Functions (Moody & Darken, 1989; Poggio & Girosi, 1990), Local Model Networks (Murray-Smith, 1994), Trees (Breiman, Friedman, Olshen, & Stone, 1984) and Mixture of Experts (Jordan & Jacobs, 1994). The modular approach is in the intermediate scale between the two extremes, the global and the local approach. However, their identification is still performed on the basis of the whole dataset and requires the same procedures used for generic global models.

Local modeling techniques are at the extreme end of divide-and-conquer methods. They are nonparametric models that combine excellent theoretical properties with a simple and flexible learning procedure. Indeed, they do not aim to return a complete description of the input/output mapping but rather to approximate the function in a neighborhood of the point to be predicted (also called the query point). There are different examples of local models, for example nearest neighbor, weighted average, and locally weighted regression (Atkeson, Moore, & Schaal, 1997b). Each of these models use data points near the point to be predicted for estimating the unknown output. *Nearest neighbor* models simply find the closest point and uses its output value. *Weighted average* models combines the closest points by averaging them with weights inversely proportional to their distance to the point to be predicted. *Locally weighted regression* models fit a model to nearby points with a weighted regression where the weights are function of distances to the query point.

The effectiveness of local models is well-known in the time series and computational intelligence community. For example, the method proposed by Sauer (1994) gave good performance and ranked second best forecast for the Santa Fe A dataset from a forecasting competition organized by Santa Fe institute. Moreover, the two top-ranked entries of the KULeuven competition used local learning methods (Bontempi et al., 1998; McNames, 1998).

In this work, we will restrict to consider a particular instance of local modeling algorithms: the Lazy Learning algorithm (Aha, 1997).

3.2. The Lazy Learning algorithm

It is possible to encounter different degree of "laziness" in local learning algorithms. For instance, a k Nearest Neighbor (k-NN)

Table 3
A summary of the pros and cons of the different multi-step forecasting strategies.

	Pros	Cons	Computational time needed
Recursive	Suitable for noise-free time series (e.g. chaotic)	Accumulation of errors	+
Direct	No accumulation of errors	Conditional independence assumption	++++
DirRec	Trade-off between Direct and Recursive	Input set grows linearly with H	+++++
MIMO	No conditional independence assumption	Reduced flexibility: same model structure for all the horizons	++
DIRMO	Trade-off between total dependence and total independence of forecasts	One additional parameter to estimate	+++

algorithm, which learns the best value of k before the query is requested, is hardly a lazy approach since, after the query is presented, it requires only a reduced amount of learning procedure, only the computation of the neighbors and the average. On the contrary, a local method, which depends on the query to select the number of neighbors or other structural parameters presents a higher degree of "laziness".

The Lazy Learning (LL) algorithm, extensively discussed in Birattari, Bontempi, and Bersini (1999) and Birattari and Bersini (1997), is a *query-based* local modeling technique where *the whole learning procedure* is deferred until a forecast is required. When the query is requested, the learning procedure may start to select the best value of the number of neighbors (or other structural parameters) and next, the dataset is searched for the nearest neighbors of the query point. The nearest neighbors are then used for estimating a local model, which returns a forecast. The local model is then discarded and the procedure is repeated from scratch for subsequent queries.

The LL algorithm has a number of attractive features (Aha, 1997), namely, the reduced number of assumptions, the online learning capability and the capacity to model nonstationarity. LL assumes no a priori knowledge on the process underlying the data, which is particularly relevant in real datasets. These considerations motivate the adoption of the LL algorithm as a learning model in a multi-step ahead forecasting context.

Local modeling techniques require the definition of a set of model parameters namely *the number k of neighbors*, *the kernel function*, *the parametric family* and *the distance metric*[REF]. In the literature, different methods have been proposed to select automatically the adequate configuration (Atkeson et al., 1997b; Birattari et al., 1999). However, in this paper, we will limit the search on only the selection of the number of neighbors (also called or equivalent to the bandwidth selection). This is essentially the most critical parameter, as it controls the bias/variance trade-off. Bandwidth selection is usually performed by rule-of-thumb techniques (Fan & Gijbels, 1995), plug-in methods (Ruppert, Sheather, & Wand, 1995) or cross-validation strategies (Atkeson, Moore, & Schaal, 1997a). Concerning the other parameters, we use the tricubic kernel (Cleveland, Devlin, & Grosse, 1988) as kernel function, a constant model for the parametric family and the euclidean distance as metric.

Note that in order to apply local learning to a time series, we need to embed it into a dataset made of pairs where the first component is a temporal pattern of length d and the second component is either the future value (in the case of Single-Output Modeling) or the consecutive temporal pattern of length H (in the case of Multiple-Output Modeling). In the following sections, D will refer to the embedded time series with M input/output pairs.

3.3. Single-Output Lazy Learning algorithm

In the case of Single-Output learning (i.e with scalar output), the Lazy Learning procedure consists of a sequence of steps detailed in Algorithm 1. The algorithm assesses the generalization performance of different local models and compares them in order to select the best one in terms of generalization capability. To perform that, the algorithm associate a Leave-One-Out (LOO) error $e_{LOO}(k)$ to the estimation $y_q^{(k)}$ obtained with k neighbors (lines 4 and 5).

The LOO error can provide a reliable estimate of the generalization capability. However the disadvantage of such an approach is that it requires to repeat k times the training process, which means a large computational effort. Fortunately, in the case of linear models there exists a powerful statistical procedure to compute the LOO cross-validation measure at a reduced computational cost: the PRESS (Prediction Sum of Squares) statistic (Allen, 1974).

In case of constant model, the LOO error $e_{LOO}(k)$ for the estimation $y_q^{(k)}$ of the query point $\mathbf{x}_q$ is calculated as follows (Bontempi, 1999):

$$e_{LOO}(k) = \frac{1}{k}\sum_{j=1}^{k}(e_j(k))^2, \tag{12}$$

where $e_j(k)$ designates the error obtained by setting aside the jth neighbor of $\mathbf{x}_q$ ($j \in \{1,\ldots,k\}$). If we define the output of the k closest neighbors of $\mathbf{x}_q$ as $\{y_{[1]},\ldots,y_{[k]}\}$ then, $e_j(k)$ is defined as

$$e_j(k) = y_{[j]} - \frac{\sum_{i=1(i\neq j)}^{k} y_{[i]}}{k-1} \tag{13}$$

$$= \frac{(k-1)y_{[j]} - \sum_{i=1(i\neq j)}^{k} y_{[i]}}{k-1} \tag{14}$$

$$= \frac{(k-1)y_{[j]} + y_{[j]} - y_{[j]} - \sum_{i=1(i\neq j)}^{k} y_{[i]}}{k-1} \tag{15}$$

$$= \frac{k y_{[j]} - \sum_{i=1}^{k} y_{[i]}}{k-1} \tag{16}$$

$$= k\frac{y_{[j]} - y_q^{(k)}}{k-1}. \tag{17}$$

Note that if we use Eq. (13) to calculate the LOO error (Eq. (12)), the training process is repeated k times since the sum in Eq. (13) is performed for each index j. However, by using the PRESS statistic (Eq. (17)), we avoid this large computational effort since the sum is replaced by the previously computed $y_q^{(k)}$ which was already calculated. This makes the PRESS statistic an efficient method to compute the LOO error.

Algorithm 1. Single-Output Lazy Learning

Input: $D = \{(\mathbf{x}_i, y_i) \in (\mathbb{R}^d \times \mathbb{R})$ with $i \in \{1,\ldots,M\}\}$, dataset.
Input: $\mathbf{x}_q \in \mathbb{R}^d$, query point.
Input: $Kmax$, the maximum number of neighbors.
Output: $\hat{y}_q$, the prediction of the (scalar) output of the query point $\mathbf{x}_q$.
1 Sort increasingly the set of vectors $\{\mathbf{x}_i\}$ with respect to the distance to $\mathbf{x}_q$.
2 $[j]$ designate the index of the jth closest neighbor of $\mathbf{x}_q$.
3 for $k \in \{2,\ldots,Kmax\}$ **do**
4 $\quad y_q^{(k)} = \frac{1}{k}\sum_{j=1}^{k} y_{[j]}$.
5 $\quad$ Calculate $e_{LOO}(k)$ which is defined in Eq. (12).
6 end
7 $k^* = \arg\min_{k\in\{2,\ldots,Kmax\}} e_{LOO}(k)$.
8 $\hat{y}_q = y_q^{(k^*)}$.
9 return $\hat{y}_q$.

After evaluating the performance of local models with different number of neighbors k (lines 3 to 6), the best one which minimizes the LOO error (having index k^*) is selected (lines 7 and 8). Finally, the prediction of the output of $\mathbf{x}_q$ is returned (line 9).

3.4. Multiple-Output Lazy Learning algorithm

The adoption of Multiple-Output strategies requires the design of Multiple-Output (or equivalently multi-response) modeling techniques (Breiman & Friedman, 1997; Matías, 2005; Micchelli & Pontil, 2005) where the output is no more a scalar quantity but a vector of values. Like in the Single-Output case, we need criteria to assess and compare local models with different number of neighbors. In the following, we present two criteria: the first one is an

extension of the LOO error for the Multiple-Output case (Algorithm 2) (Ben Taieb et al., 2010; Bontempi, 2008) and the second one is a

Algorithm 2. Multiple-Output Lazy Learning (*LOO criterion*): MIMO-LOO

Input: $D = \{(\mathbf{x}_i, \mathbf{y}_i) \in (\mathbb{R}^d \times \mathbb{R}^l)$ with $i \in \{1,\ldots,M\}\}$, dataset.
Input: $\mathbf{x}_q \in \mathbb{R}^d$, query point.
Input: *Kmax*, the maximum number of neighbors.
Output: $\hat{\mathbf{y}}_q$, the prediction of the (vectorial) output of the query point $\mathbf{x}_q$.
1 Sort increasingly the set of vectors $\{\mathbf{x}_i\}$ with respect to the distance to $\mathbf{x}_q$.
2 $[j]$ will designate the index of the jth closest neighbor of $\mathbf{x}_q$.
3 **for** $k \in \{2,\ldots,Kmax\}$ **do**
4 $\quad \mathbf{y}_q^{(k)} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{y}_{[j]\cdot}$
5 $\quad E_{LOO}(k) = \frac{1}{l}\sum_{h=1}^{l} e_{LOO}^{l}(k)$ where $e_{LOO}^{l}(k)$ is defined in Eq. (12).
6 **end**
7 $k^* = \arg\min_{k\in\{2,\ldots,Kmax\}} E_{LOO}(k)$.
8 $\hat{\mathbf{y}}_q = \mathbf{y}_q^{(k^*)}$
9 **return** $\hat{\mathbf{y}}_q$.

criterion proper to the Multiple-Output modeling (Algorithm 3) (Ben Taieb et al., 2010; Bontempi & Ben Taieb, 2011). Note that, in the two algorithms, the output is a vector of size l (e.g. l will equal H with the MIMO strategy or s in the DIRMO strategy).

Algorithm 2 is an extension of the Algorithm 1 for vectorial outputs. We still use the LOO cross-validation measure as a criterion to estimate the generalization capability of the model but here, the LOO error is an aggregation of the errors obtained for each output (line 5). Note that the same number of neighbors is selected for all the outputs (e.g. MIMO strategy) unlike what could happen with different Single-Output tasks (e.g. Direct strategy).

The second criterion uses the fact that the forecasting horizon H is supposed to be large (multi-step ahead forecasting) and hence we have enough samples to estimate some descriptive statistics. Then, instead of using the Leave-One-Out error, we can use as criterion a measure of stochastic discrepancy between the forecasted values and the training time series. The lower the discrepancy between the descriptors of the forecasts and the training time series, the better is the quality of the forecasts (Bontempi & Ben Taieb, 2011).

Several measures of discrepancy can be defined, both linear and non-linear. For example, the autocorrelation can be used as linear statistics and the maximum likelihood as a non-linear one. In this work, we will consider only one linear measure using both the autocorrelation and the partial correlation.

The assessment of the quality of the estimation $y_q^{(k)}$ of the query point $\mathbf{x}_q$ is calculated as follows

$$E_\Delta(k) = \underbrace{1 - |cor[\rho(ts \cdot \mathbf{y}_q^{(k)}), \rho(ts)]|}_{} + \underbrace{1 - |cor[\pi(ts \cdot \mathbf{y}_q^{(k)}), \pi(ts)]|}_{}, \tag{18}$$

where the symbol "$\cdot$" represents the concatenation, ts represent the training time series and *cor* is the Pearson correlation. This discrepancy measure is composed of two parts where the first part uses the autocorrelation (noted ρ) and the second uses the partial autocorrelation (noted π).

For each part, we calculate the discrepancy (estimated with the correlation, noted *cor*) between, on one hand, the autocorrelation (or partial autocorrelation) of the concatenation of the training time series ts and the forecasted sequence $\mathbf{y}_q^{(k)}$ and, on the other hand, the autocorrelation (or partial autocorrelation) of the training time series ts (Ben Taieb et al., 2009; Bontempi, 2008).

In Algorithm 3, after evaluating the performance of local models

Algorithm 3. Multiple-Output Lazy Learning (*discrepancy criterion*): MIMO-ACFLIN

Input: $ts = [ts_1,\ldots,ts_N]$, time series.
Input: $D = \{(\mathbf{x}_i, \mathbf{y}_i) \in (\mathbb{R}^d \times \mathbb{R}^l)$ with $i \in \{1,\ldots,M\}\}$, dataset.
Input: $\mathbf{x}_q \in \mathbb{R}^d$, query point.
Input: *Kmax*, the maximum number of neighbors.
Output: $\hat{\mathbf{y}}_q$, the prediction of the (vectorial) output of the query point $\mathbf{x}_q$.
1 Sort increasingly the set of vectors $\{\mathbf{x}_i\}$ with respect to the distance to $\mathbf{x}_q$.
2 $[j]$ will designate the index of the jth closest neighbor of $\mathbf{x}_q$.
3 **for** $k \in \{2,\ldots,Kmax\}$ **do**
4 $\quad \mathbf{y}_q^{(k)} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{y}_{[j]\cdot}$
5 $\quad$ Calculate $E_\Delta(k)$ which is defined in Eq. (18).
6 **end**
7 $k^* = \arg\min_{k\in\{2,\ldots,Kmax\}} E_\Delta(k)$.
8 $\hat{\mathbf{y}}_q = \mathbf{y}_q^{(k^*)}$
9 **return** $\hat{\mathbf{y}}_q$.

with different number of neighbors k (lines 3–6), the best one, which minimizes the discrepancy between the forecasting sequence and the training time series (having index k^*), is selected (lines 7 and 8). In other words, the goal is to select the best number of neighbors k^* which preserve the stochastic properties of the time series in the forecasted sequence. Finally, the prediction of the output of $\mathbf{x}_q$ is returned (line 9).

3.5. Model selection or model averaging

Considering the Algorithm 1, we can see that we generate, for the query $\mathbf{x}_q$, a set of predictions $\left\{y_q^{(2)}, y_q^{(3)}, \ldots, y_q^{(Kmax)}\right\}$, each obtained with different number of neighbors. For each of these predictions, a testing error $\{e_{LOO}(2), e_{LOO}(3), \ldots, e_{LOO}(Kmax)\}$ has been calculated. Note that the next considerations are also applicable to Algorithms 2 and 3.

The goal of model selection is to use all this information (set of predictions and testing errors) to estimate the final prediction $\hat{y}_q$ of the query point $\mathbf{x}_q$. There exist two main paradigms mainly the *winner-take-all* and *combination* approaches.

In the Algorithm 1, we presented the winner-take-all approach (noted WINNER) (Maron & Moore, 1997) which consists of comparing the set of models $y_q^{(k)}$ and selecting the best one in terms of testing error $e_{LOO}(k)$ (see line 7).

Selecting the best model according to the testing error is intuitively the approach which should work the best. However, results in machine learning show that the performance of the final model can be improved by combining models having different structures (Breiman, 1996; Jacobs, Jordan, Nowlan, & Hinton, 1991; Raudys & Zliobaite, 2006; Schapire, Freund, Bartlett, & Lee, 1998).

In order to apply the model averaging, lines 7 and 8 of the Algorithm 1 can be replaced by

$$\hat{y}_q = \frac{p_2 y_q^{(2)} + \cdots + p_{Kmax} y_q^{(Kmax)}}{\sum_{k=2}^{Kmax} p_k}, \tag{19}$$

where an average is calculated. The weights p_k will take different values depending on the combination approach adopted. If p_k equals $\frac{1}{Kmax}$, we are in the case of equally weighted combination

and $\hat{y}_q$ reduces to an arithmetic mean (noted COMB). Otherwise, if weights are assigned according to testing errors, p_k will equal $\frac{1}{e_{LOO}(k)}$ and $\hat{y}_q$ reduces to a weighted mean (noted WCOMB).

4. Experimental setup

4.1. Time series data

In the last decade, several time series forecasting competitions (e.g. the NN3, NN5, and the ESTSP competitions (Crone, 2009a, 2009b; Lendasse, 2007, 2008)) have been organized in order to compare and evaluate the performance of computational intelligence methods. Among them, the NN5 competition (Crone, 2009b) is one of the most interesting one since it includes the challenges of a real-world multi-step ahead forecasting task, namely multiple time series, outliers, missing values as well as multiple overlying seasonalities, etc. Fig. 2 shows four time series from the NN5 dataset.

Each of the 111 time series of this competition represents roughly two years of daily cash money withdrawal amounts (735 data points) at ATM machines at one of the various cities in the UK. For each time series, the competition required to forecast the values of the next 56 days, using the given historical data points, as accurately as possible. The performance of the forecasting methods over one time series was assessed by the symmetric mean absolute percentage of error (SMAPE) measure (Crone, 2009b), defined as

$$\text{SMAPE} = \frac{1}{H}\sum_{h=1}^{H}\frac{|\hat{y}_h - y_h|}{(\hat{y}_h + y_h)/2} \times 100, \tag{20}$$

where y_h is the target output and $\hat{y}_h$ is the prediction. Since this is a relative error measure, the errors can be averaged over all time series to obtain a mean SMAPE defined as

$$\text{SMAPE}^{*} = \frac{1}{111}\sum_{i=1}^{111}\text{SMAPE}_i, \tag{21}$$

where SMAPE_i denotes the SMAPE of the ith time series.

4.2. Methodology

The aim of the experimental study is to compare the accuracy of the five forecasting strategies in the context of the NN5 competition. Since the accuracy of a forecasting technique is known to be dependent on several design choices (e.g. the deseasonalization or the input selection) and we want to focus our analysis on the multi-step ahead forecasting strategies, we consider a number of different configurations in order to increase the statistical power of our comparison. Every configuration is composed of several preprocessing steps as sketched in Fig. 3. Since some of these steps (e.g. deseasonalization) can be performed in alternative ways (e.g. two alternatives for the deseasonalization, two alternatives for input selection, three alternatives for the model selection), we come up with 12 configurations. The details about each step are given in what follows.

Step 1: Gaps removal. The specificity of the NN5 series requires a preprocessing step called *gaps removal* where by gap we mean two types of anomalies: (i) zero values that indicate that no money withdrawal occurred and (ii) missing observations for which no value was recorded. About 2.5% of the data are corrupted by gaps. In our experiments we adopted the gap removal method proposed in Wichard (2011): if y_m is the gap sample, this method replaces the gap with the median of the set $[y_{m+365}, y_{m-365}, y_{m+7}$ and $y_{m-7}]$ among which are available.

Step 2: Deseasonalization. The adoption of deseasonalization may have a large impact on the forecasting strategies because the NN5 time series possess a variety of periodic patterns. For that reason we decided to consider tasks with and without deseasonalization in order to better account for the role of the forecasting strategy. We adopt the deseasonalization methodology discussed in Andrawis, Atiya, & El-Shishiny (in press) to remove the strong day of the week seasonality as well as the moderate day of the month seasonality. Of course after we deseasonalize and apply the forecasting model we restore back the seasonality.

Step 3: Embedding dimension selection. Every forecasting strategy requires the setting of the size d of the embedding dimension (see Eqs. (1)–(9)). Several approaches have been proposed in the literature to select this value (Kantz and Schreiber, 2004). Since this aspect is not a central theme in our paper we just applied the state-of-the-art approach reviewed in Crone and Kourentzes (2009), which consists of selecting the time-lagged realizations with significant partial correlation function (PACF). This method allows to select the value of the embedding dimension and then to identify the relevant variables within the window of past observations. We set the maximum lag of the PACF to 200 to provide a sufficiently comprehensive pool of features. However, note that the final dimensionality of the input vectors for all the time series is on average equal to 24.

Step 4: Input selection. We considered the forecasting task with and without input variable selection step. A variable selection procedure requires the setting of two elements: the *relevance criterion*, i.e. statistics which estimates the quality of the selected variables, and the *search procedure*, which describes the policy to explore the input space. We adopted the *Delta test*(DT) as relevance criterion. The DT has been introduced in time series forecasting domain by Pi and Peterson (1994) and later successfully applied to several forecasting task (Ben Taieb et al., 2009, 2010; Guillén et al., 2008; Liitiäinen and Lendasse, 2007; Mateo and Sovilj, 2010). This criterion is based on applying some kind of a noise variance estimator, and then selecting the set of input variables that yield the strongest and most deterministic dependence between inputs and outputs (Mateo and Lendasse, 2008).

Concerning the search procedure, we adopted a Forward–Backward Search (FBS) procedure which is a combination of forward selection (sequentially adding input variables) and backward search (sequentially removing some input variables). This choice was motivated by the flexibility of the FBS procedure which allows a deeper exploration of the input space. Note that the search is initialized by the set of variables defined in the previous step.

Step 5: Model selection. Concerning the model selection procedure, three approaches (see Section 3.5) are taken into consideration in our experiments:

WINNER:
This approach selects the model that gives best performance for the test set (winner-take-all approach).
COMB:
This approach combines all alternative models by simple averaging.
WCOMB:
This approach combines models by weighted averaging where weights are inversely proportional to the test errors.

4.2.1. The Compared forecasting strategies

Table 4 presents the eight forecasting strategies that we tested, showing also their respective acronyms.

4.2.2. Forecasting performance evaluation

This section describes the assessment procedure (Fig. 4) of the 8 forecasting strategies.

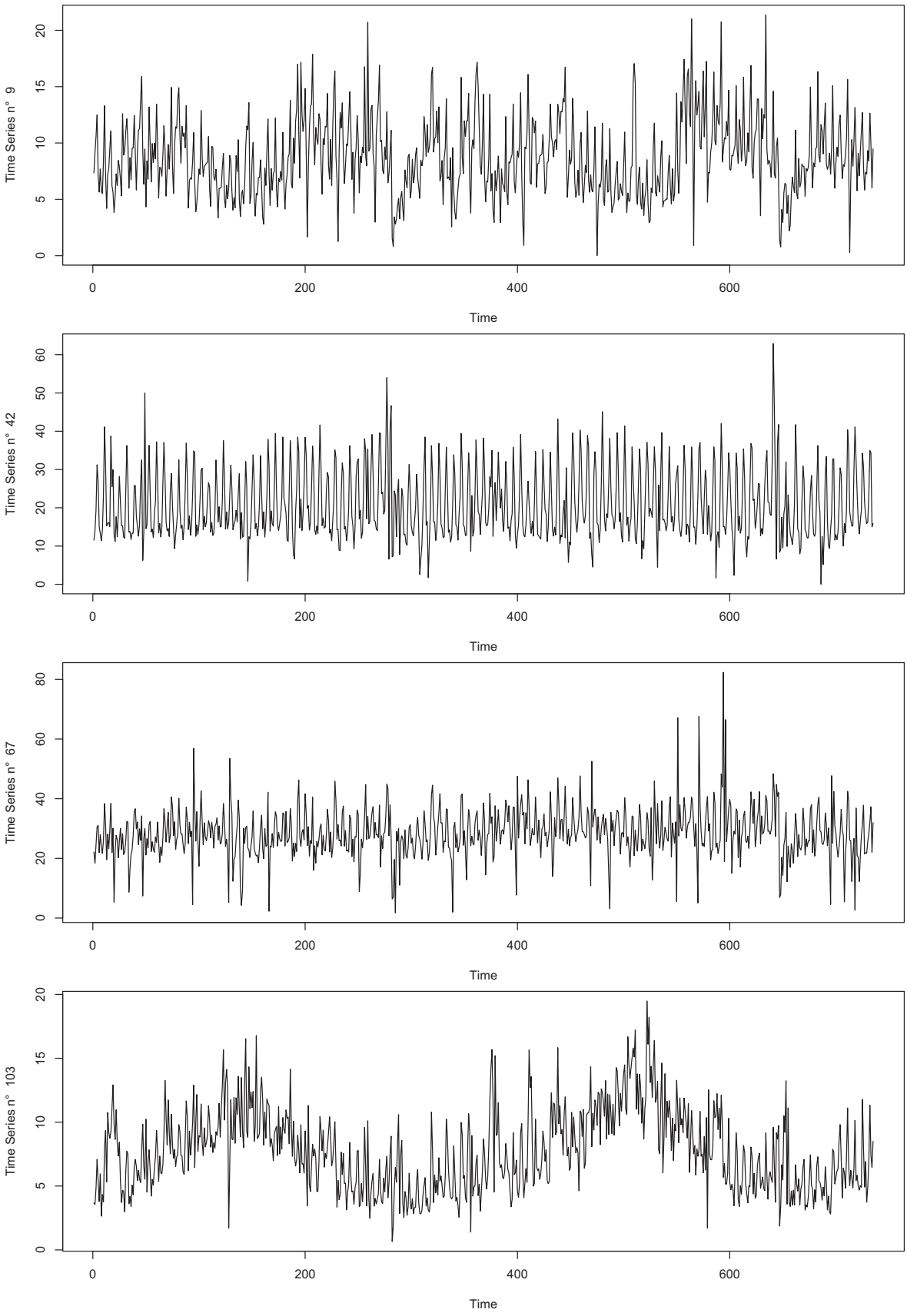

Fig. 2. Four time series from NN5 time series forecasting competition.

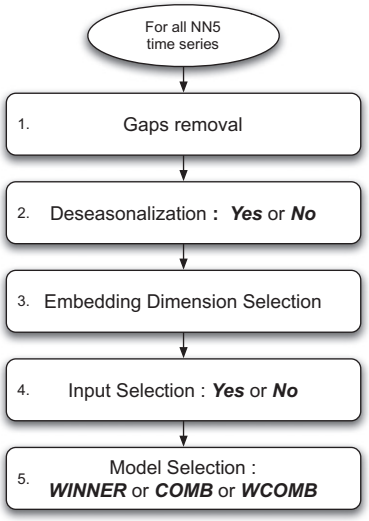

Fig. 3. The different preprocessing steps.

The procedure for comparing between the eight forecasting strategies is shown in Fig. 4. The accuracy of each forecasting strategy is first measured using the SMAPE* measure calculated over the 111 time series and defined in Eq. (21). To test if there are significant general differences in performance between the different strategies, we have to consider the problem of comparing multiple models on multiple data sets. For such case (Demšar, 2006; Garca & Herrera, 2009) in a detailed comparative study recommended using a two stage procedure: first to apply Friedman's or Iman and Davenport's tests to test if the compared models have the same mean rank. If this test rejects the null-hypothesis, then post hoc pairwise tests are to be performed to compare the different models. These tests adjust the critical values higher to ensure that there is at most a 5% chance that one of the pairwise differences will be erroneously found significant.

4.2.2.1. Friedman test. The Friedman test (Friedman, 1937, 1940) is a non-parametric procedure which tests the significance of differences between multiple ranks. It ranks the algorithms for each dataset separately: the rank of 1 will be given to the best performing algorithm, the rank of 2 to the second best and so on. Note that average ranks are assigned in case of ties.

After ranking the algorithms for each dataset, the Friedman test compares the average ranks of algorithms. Let r_j^i be the rank of the jth of k algorithms on the ith of N data sets, the average rank of the jth algorithm is $R_j = \frac{1}{N}\sum_i r_j^i$.

The null-hypothesis states that all the algorithms are equivalent and so their ranks R_j should be equal. Under the null-hypothesis, the Friedman statistic

$$Q = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{22}$$

is distributed according to a chi-squared with $k-1$ degrees of freedom (χ^2_{k-1}), when N and k are large enough (as a rule of a thumb, $N > 10$ and $k > 5$) (Demšar, 2006).

Iman & Davenport (1980), showing that Friedman's statistic is undesirably conservative, derived another improved statistic, given by

$$S = \frac{(N-1)Q}{N(k-1)-Q}, \tag{23}$$

which is distributed, under the null-hypothesis, according to the F-distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom.

4.2.2.2. Post-hoc test. When the null-hypothesis is rejected, i.e. there is a significant difference between at least 2 strategies, a post hoc test is performed to identify significant pairwise differences among all the algorithms. The test statistic for comparing the ith and the jth algorithm is

$$z = \frac{(R_i - R_j)}{\sqrt{\frac{k(k+1)}{6N}}}, \tag{24}$$

which is asymptotically normally distributed under the null hypothesis. After the corresponding p-value is calculated, it is compared with a given level of significance α.

However, in multiple comparisons, as there are a possibly large number of pairwise comparisons, there is a relatively high chance that some pairwise test are incorrectly rejected. Several procedures exist to adjust the value of α to compensate for this bias, for instance Nemenyi, Holm, Shaffer as well as Bergmann and Hommel

Table 4
The five forecasting strategies with their respective variants which gives eight forecasting strategies.

1. REC	The *Recursive* forecasting strategy	
2. DIR	The *Direct* forecasting strategy	
3. DIRREC	The *DirRec* forecasting strategy	
MIMO	4. MIMO-LOO	A variant of the *MIMO* forecasting strategy with the LOO selection criteria
	5. MIMO-ACFLIN	A variant of the *MIMO* forecasting strategy with the autocorrelation selection criteria
DIRMO	6. DIRMO-SEL	The *DIRMO* forecasting strategy which select the best value of the parameter s
	7. DIRMO-AVG	A variant of the *DIRMO* strategy which calculates a simple average of the forecasts obtained with different values of the parameter s
	8. DIRMO-WAVG	The DIRMO-AVG with a weighted average where weights are inversely proportional to testing errors

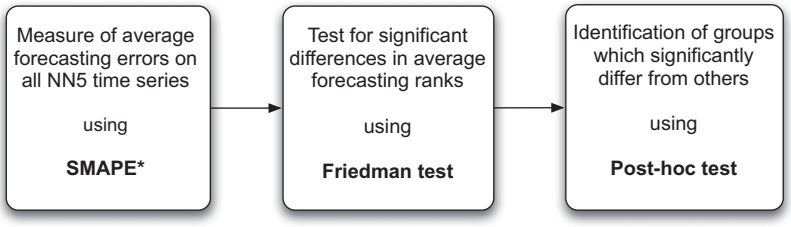

Fig. 4. Different steps of the forecasting performance evaluation.

(Demšar, 2006). Based on the suggestion of Garca and Herrera (2009) we adopted Shaffer's correction. The reason is that Garca and Herrera (2009) showed that Shaffer's procedure has the same complexity Holm's procedure, but with the advantage of using information about logically related hypothesis.

.4.3. Experimental phases

In order to reproduce the same context of the NN5 forecasting competition the experimental setting is made of two phases: the *pre-competition* and the *competition* phase.

4.3.1. Pre-competition phase

The pre-competition phase is devoted to the comparison of the different forecasting strategies using the available observations of 111 time series. The goal is to learn the different parameters and then estimate the forecasting performance and compare between the different strategies.

To estimate the forecasting performance of each strategy, we used a learning scheme with training-validation-testing sets. Each time series (containing 735 observations) is partitioned in three mutually exclusive sets (A, B and C) as shown in Fig. 5: training (Day 1 to Day 623: 623 values), validation (Day 624 to Day 679: 56 values) and testing (Day 680 to Day 735: 56 values).

The validation set (B in Fig. 5) is used to build and tune the models. Specifically, as we use a Lazy Learning approach, we need to select, for each model, the range of number of neighbors ($[2,\ldots,Kmax]$) to use in performing the forecasting task.

The test set (C in Fig. 5) is used to measure the performances of each forecasting strategy. To make the utmost use of the available data, we adopt a multiple time origin test as suggested by Tashman (2000), where the time origin denotes the point from which the multi-step ahead forecasts are generated.

The time origin and corresponding forecast intervals are given as:

1. Day 680 to Day 735 (56 data points);
2. Day 687 to Day 735 (49 data points);
3. Day 694 to Day 735 (42 data points).

In other words, we perform the forecast three times starting from the three different starting points, each time forecasting a number of steps ahead till the end of the interval. Note that we used the same test period and evaluation criterion (i.e. the SMAPE) as used by Andrawis et al. (in press). This allows us to compare our results with several other machine learning models tested in this article.

4.3.2. Competition phase

In the competition phase we generate the final forecasts, made up of 56 future observations, which would have been submitted to the competition. This phase takes advantage of the lessons learned and the design choices made in the pre-competition phase. Here, we combine the training set with the test set (A + B in Fig. 6) to re-train the models of the different strategies and then generate the final forecast (which will be submitted to the competition). The training set (A + B in Fig. 6) is now made of 679 data points and the validation set (C in Fig. 6) is composed of the next 56 data points, as shown in Fig. 6. In other words, the 735 values are then used to build and tune the models, which will next return the forecasted values.

5. Results and discussion

This section presents and discusses the prediction results of the forecasting strategies for the *pre-competition* and *competition* phase. For each phase, we report the results obtained in the 12 different configurations introduced in Section 4.2.

The forecasting performance of the strategies is measured using the criteria discussed in Section 4.2.2 and presented by means of

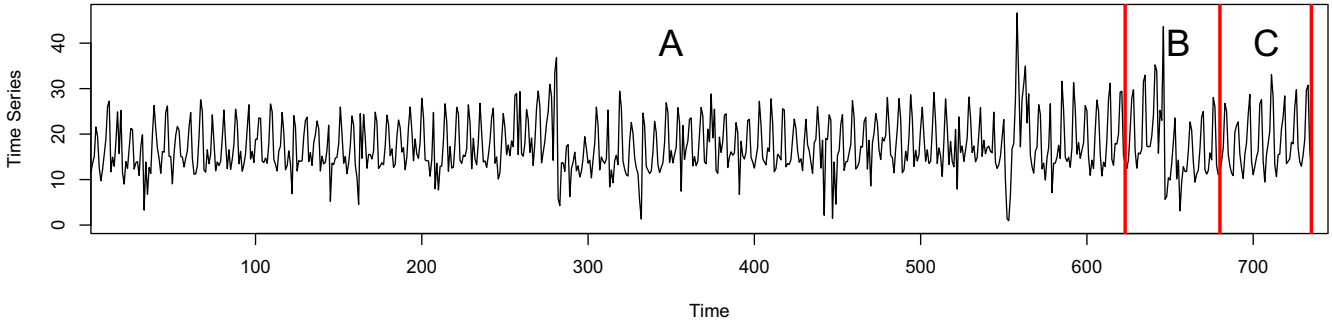

Fig. 5. Learning with three mutually exclusive sets for training (A), validation (B) and testing (C).

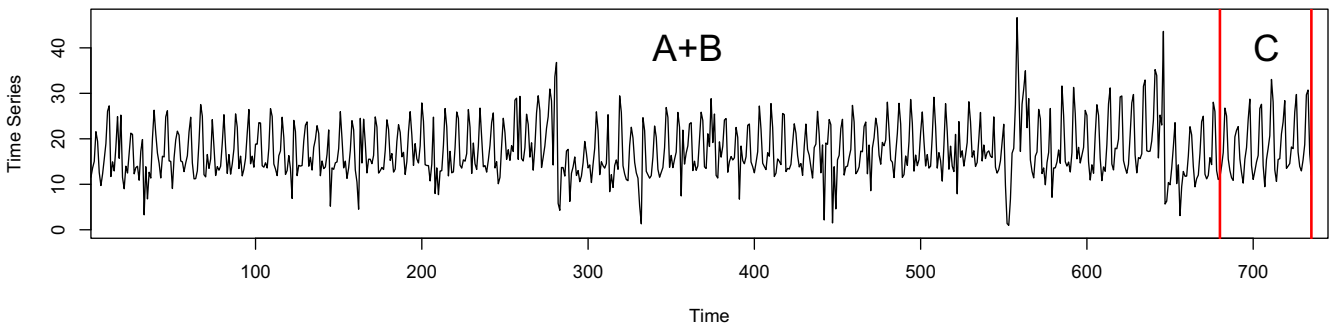

Fig. 6. Forecasting with two mutually exclusive sets for training (A + B) and validation (C).

Table 5
Pre-competition phase. Average forecasting errors (SMAPE*) with average ranking for each strategy in the 12 different configurations. The numbers in round bracket represent the ranking within the column.

Strategies	WINNER		COMB		COMBW	
	SMAPE* [Std err]	Avg. rank	SMAPE* [Std err]	Avg. rank	SMAPE* [Std err]	Avg. rank
(a) Deseasonalization : No – input selection : No						
DIR	22.37[0.55](7)	5.19(7)	22.19[0.54](7)	4.96(6)	22.61[0.55](5)	5.47(7)
REC	21.41[0.59](6)	3.98(5)	21.95[0.58](5)	4.63(5)	22.91[0.62](7)	4.73(5)
DIRREC	45.25[0.89](8)	8.00(8)	40.73[0.83](8)	8.00(8)	38.93[0.77](8)	7.99(8)
MIMO-LOO	21.17[0.60](2)	3.64(2)	**20.61**[0.57](1)	**2.77**(1)	**20.61**[0.58](1)	**2.71**(1)
MIMO-ACFLIN	21.40[0.59](5)	4.48(6)	20.69[0.58](2)	3.17(2)	20.61[0.58](2)	2.71(2)
DIRMO-SEL	21.21[0.56](3)	3.76(3)	22.15[0.56](6)	5.40(7)	22.68[0.56](6)	5.32(6)
DIRMO-AVG	21.27[0.56](4)	3.88(4)	20.90[0.56](3)	3.36(3)	21.16[0.56](3)	3.42(3)
DIRMO-WAVG	**21.12**[0.56](1)	**3.07**(1)	20.96[0.56](4)	3.71(4)	21.25[0.57](4)	3.65(4)
(b) Deseasonalization : No – input selection : Yes						
DIR	22.80[0.50](4)	3.20(4)	22.73[0.52](4)	3.27(4)	23.37[0.54](4)	3.27(4)
REC	21.17[0.55](2)	2.20(2)	22.21[0.52](3)	2.96(3)	23.20[0.56](3)	3.01(3)
DIRREC	45.21[0.93](5)	5.00(5)	40.57[0.83](5)	4.96(5)	39.03[0.75](5)	4.96(5)
MIMO-LOO	**21.06**[0.57](1)	**2.02**(1)	**20.60**[0.59](1)	**1.79**(1)	**20.64**[0.59](1)	**1.88**(1)
MIMO-ACFLIN	21.40[0.57](3)	2.58(3)	20.65[0.59](2)	2.01(2)	20.64[0.59](2)	1.88(2)
(c) Deseasonalization : Yes – input selection : No						
DIR	22.29[0.57](7)	6.42(8)	20.41[0.57](7)	6.23(8)	20.49[0.58](7)	6.43(8)
REC	21.61[0.61](6)	5.35(6)	19.39[0.59](6)	3.87(4)	19.31[0.59](5)	3.68(2)
DIRREC	22.61[0.68](8)	6.33(7)	20.67[0.64](8)	6.15(7)	20.61[0.61](8)	6.04(7)
MIMO-LOO	19.81[0.59](4)	3.83(4)	19.21[0.60](4)	**3.69**(1)	19.25[0.60](3)	3.82(4)
MIMO-ACFLIN	**19.34**[0.59](1)	3.21(3)	19.19[0.59](3)	4.00(5)	19.25[0.60](4)	3.82(5)
DIRMO-SEL	19.49[0.57](2)	**2.90**(1)	**18.98**[0.57](1)	3.72(2)	**19.04**[0.58](1)	**3.66**(1)
DIRMO-AVG	20.39[0.56](5)	4.74(5)	19.36[0.58](5)	4.61(6)	19.43[0.58](6)	4.77(6)
DIRMO-WAVG	19.71[0.58](3)	3.21(2)	19.02[0.57](2)	3.72(3)	19.11[0.58](2)	3.79(3)
(d) Deseasonalization : Yes – input selection : Yes						
DIR	21.87[0.47](4)	3.95(5)	20.07[0.49](4)	3.82(5)	20.20[0.51](4)	3.97(5)
REC	20.52[0.53](3)	2.95(3)	19.14[0.55](3)	2.57(3)	19.20[0.55](3)	2.61(3)
DIRREC	23.06[0.84](5)	3.77(4)	21.03[0.68](5)	3.77(4)	20.95[0.66](5)	3.74(4)
MIMO-LOO	20.02[0.53](2)	2.59(2)	18.86[0.54](2)	2.45(2)	**18.88**[0.54](1)	**2.34**(1)
MIMO-ACFLIN	**18.95**[0.54](1)	**1.76**(1)	**18.81**[0.54](1)	**2.38**(1)	18.88[0.54](2)	2.34(2)

two tables. The first one provides the average SMAPE as well as the ranking for each forecasting strategy. Since the null-hypothesis stating that all the algorithms are equivalent has been rejected (using the Iman-Davenport statistic) for all the configurations, we proceeded with the post hoc test. The second table presents the results of this post hoc test, which partitions the set of strategies in several groups which are statistically significantly different in terms of forecasting performance.

Note that the configurations which require the input selection do not contain the DIRMO results since combining the selection of the inputs with the selection of the parameter s would have needed an excessive amount of computation time.

Table 6
Pre-competition phase. Group of strategies statistically significantly different (sorted by increasing ranking) provided by Friedman and post-hoc tests for the 12 configurations.

WINNER	COMB	WCOMB
(a) Deseasonalization : No – input selection : No		
1. "DIRMO-WAVG" (3.07) "MIMO-LOO" (3.64) "DIRMO-SEL" (3.76) "DIRMO-AVG" (3.88) "REC" (3.98) "MIMO-ACFLIN" (4.48) "DIR" (5.19)	1. "MIMO-LOO" (2.77) "MIMO-ACFLIN" (3.17) "DIRMO-AVG" (3.36) "DIRMO-WAVG" (3.71)	1. "MIMO-LOO" (2.71) "MIMO-ACFLIN" (2.71) "DIRMO-AVG" (3.42) "DIRMO-WAVG" (3.65)
2. "DIRREC" (8.00)	2. "REC" (4.63) "DIR" (4.96) "DIRMO-SEL" (5.40)	2. "REC" (4.73) "DIRMO-SEL" (5.32) "DIR" (5.47)
	3. "DIRREC" (8.00)	3. "DIRREC" (7.99)
(b) Deseasonalization : No – input selection : Yes		
1. "MIMO-LOO" (2.02) "REC" (2.20) "MIMO-ACFLIN" (2.58)	1. "MIMO-LOO" (1.79) "MIMO-ACFLIN" (2.01)	1. "MIMO-LOO" (1.88) "MIMO-ACFLIN" (1.88)
2. "DIR" (3.20)	2. "REC" (2.96) "DIR" (3.27)	2. "REC" (3.01) "DIR" (3.27)
3. "DIRREC" (5.00)	3. "DIRREC" (4.96)	3. "DIRREC" (4.96)
(c) Deseasonalization : Yes – input selection : No		
1. "DIRMO-SEL" (2.90) "DIRMO-WAVG" (3.21) "MIMO-ACFLIN" (3.21) "MIMO-LOO" (3.83)	1. "MIMO-LOO" (3.69) "DIRMO-SEL" (3.72) "DIRMO-WAVG" (3.72) "REC" (3.87) "MIMO-ACFLIN" (4.00) "DIRMO-AVG" (4.61)	1. "DIRMO-SEL" (3.66) "REC" (3.68) "DIRMO-WAVG" (3.79) "MIMO-LOO" (3.82) "MIMO-ACFLIN" (3.82)
2. "DIRMO-AVG" (4.74) "REC" (5.35)	2. "DIRREC" (6.15) "DIR" (6.23)	2. "DIRMO-AVG" (4.77)
3. "DIRREC" (6.33) "DIR" (6.42)		3. "DIRREC" (6.04) "DIR" (6.43)
(d) Deseasonalization : Yes – input selection : Yes		
1. "MIMO-ACFLIN" (1.76)	1. "MIMO-ACFLIN" (2.38) "MIMO-LOO" (2.45) "REC" (2.57)	1. "MIMO-LOO" (2.34) "MIMO-ACFLIN" (2.34) "REC" (2.61)
2. "MIMO-LOO" (2.59) "REC" (2.95)	2. "DIRREC" (3.77) "DIR" (3.82)	2. "DIRREC" (3.74) "DIR" (3.97)
3. "DIRREC" (3.77) "DIR" (3.95)		

Table 7
Forecasting errors for the different forecasting models.

Model	SMAPE*
GPR-ITER	19.90
GPR-DIR	21.22
GPR-LEV	20.19
NN-ITER	21.11
NN-LEV	19.83
MULT-REGR1	19.11
MULTI-REGR2	18.96
MULT-REGR3	18.94
MOV-AVG	19.55
Holt's Exp Sm	23.77
Combined	18.95

5.1. Pre-competition results

The SMAPE and ranking results for the *pre-competition* phase are presented in Table 5 while the results of the post hoc test are summarized in Table 6.

The availability of the SMAPE* results, obtained according to the procedure used in Andrawis et al. (in press), makes possible the comparison of our pre-competition results with those of several others learning methods available in Andrawis et al. (in press). For the sake of comparison, Table 7 reports the forecasting errors for some of the techniques considered in Andrawis et al. (in press), notably Gaussian Process Regression (GPR), Neural Network (NN), Multiple Regression (MULT-REGR), Simple Moving Average (MOV-AVG), Holt's Exponential Smoothing and a combination (Combined) of such techniques. The comparison shows that the best configuration of Table 5d, that is the MIMO-ACFLIN strategy, is competitive with all these models with a SMAPE* amounting to 18.81%.

5.2. Competition results

The SMAPE and ranking results for the *competition* phase are presented in Table 8 while the results of the post hoc test are summarized in Table 9.

The pre-competition results presented in the previous section suggest us to use the MIMO-ACFLIN strategy with the `Comb` model selection approach by removing the seasonality and applying the input selection procedure, since this configuration obtains the smallest forecasting error (18.81%).

By using the MIMO-ACFLIN strategy and the corresponding configuration in the competition phase, we would generate forecasts with a SMAPE* equals to 20.28% which is quite good compared to the best computational intelligence entries of the competition as shown in Table 10. Fig. 7 shows the forecasts of the MIMO-ACFLIN strategy versus the actual values for four NN5 time series to illustrate the forecasting capability of this strategy.

5.3. Discussion

From all presented results one can deduce the following observations below. These findings refer mainly to the pre-competition results. But, one can easily see that they mostly also apply to the competition phase results.

- The overall best method is MIMO-ACFLIN, used with input selection, deseasonalization and equal weight combination (COMB).

Table 8
Competition phase. Average forecasting errors (SMAPE*) with average ranking for each strategy in the 12 different configurations. The numbers in round bracket represent the ranking within the column.

Strategies	WINNER		COMB		COMBW	
	SMAPE* [Std err]	Avg. rank	SMAPE* [Std err]	Avg. rank	SMAPE* [Std err]	Avg. rank
(a) Deseasonalization : No – input selection : No						
DIR	24.48[0.52](7)	5.58(7)	23.06[0.50](6)	5.21(6)	22.89[0.48](6)	5.20(7)
REC	24.22[0.62](6)	4.96(6)	23.71[0.52](7)	5.24(7)	23.54[0.54](7)	5.05(6)
DIRREC	44.97[0.85](8)	7.93(8)	40.37[0.80](8)	7.97(8)	38.17[0.72](8)	7.95(8)
MIMO-LOO	**21.92**[0.56](1)	**2.90**(1)	**21.55**[0.50](1)	**2.84**(1)	**21.64**[0.49](1)	**2.79**(1)
MIMO-ACFLIN	22.45[0.54](3)	3.65(3)	21.57[0.50](2)	3.08(2)	21.64[0.49](2)	2.79(2)
DIRMO-SEL	22.60[0.54](5)	3.99(5)	22.27[0.49](5)	4.46(5)	22.50[0.50](5)	4.84(5)
DIRMO-AVG	22.55[0.53](4)	3.78(4)	21.73[0.49](4)	3.68(4)	21.93[0.49](4)	3.84(4)
DIRMO-WAVG	22.36[0.54](2)	3.21(2)	21.70[0.49](3)	3.52(3)	21.87[0.49](3)	3.55(3)
(b) Deseasonalization : No – input selection : Yes						
DIR	24.43[0.52](4)	3.28(4)	23.14[0.48](4)	3.22(4)	23.25[0.48](4)	3.24(4)
REC	22.73[0.69](3)	2.28(2)	22.45[0.67](3)	2.53(3)	22.57[0.65](3)	2.42(3)
DIRREC	44.91[0.86](5)	4.94(5)	39.88[0.77](5)	4.93(5)	38.04[0.71](5)	4.95(5)
MIMO-LOO	**22.18**[0.53](1)	**2.12**(1)	**21.78**[0.49](1)	**2.09**(1)	**21.84**[0.49](1)	**2.19**(1)
MIMO-ACFLIN	22.62[0.51](2)	2.39(3)	21.83[0.50](2)	2.24(2)	21.84[0.49](2)	2.19(2)
(c) Deseasonalization : Yes – input selection : No						
DIR	23.98[0.55](8)	6.36(8)	21.65[0.46](8)	6.05(8)	21.75[0.47](7)	6.14(8)
REC	23.12[0.59](6)	5.47(6)	21.39[0.49](6)	5.04(6)	21.86[0.49](8)	5.40(6)
DIRREC	23.51[0.60](7)	6.15(7)	21.58[0.52](7)	5.78(7)	21.57[0.51](6)	5.67(7)
MIMO-LOO	21.11[0.54](4)	3.69(4)	20.27[0.47](4)	3.77(3)	20.34[0.47](3)	3.69(3)
MIMO-ACFLIN	**20.25**[0.47](1)	**3.05**(1)	20.18[0.46](2)	3.67(2)	20.34[0.47](4)	3.69(4)
DIRMO-SEL	20.85[0.51](2)	3.45(3)	20.18[0.46](3)	3.77(4)	**20.23**[0.45](1)	**3.45**(1)
DIRMO-AVG	21.66[0.51](5)	4.47(5)	20.38[0.46](5)	4.34(5)	20.48[0.46](5)	4.38(5)
DIRMO-WAVG	20.97[0.51](3)	3.35(2)	**20.15**[0.46](1)	**3.59**(1)	20.23[0.46](2)	3.59(2)
(d) Deseasonalization : Yes – input selection : Yes						
DIR	23.91[0.50](5)	4.14(5)	21.54[0.47](5)	3.94(5)	21.58[0.48](5)	3.91(5)
REC	22.57[0.64](3)	3.05(3)	21.39[0.63](3)	2.96(3)	21.45[0.64](3)	2.93(3)
DIRREC	23.66[0.58](4)	3.79(4)	21.48[0.52](4)	3.50(4)	21.47[0.51](4)	3.50(4)
MIMO-LOO	20.74[0.51](2)	2.14(2)	**20.28**[0.53](1)	**2.27**(1)	**20.28**[0.52](1)	**2.33**(1)
MIMO-ACFLIN	**20.39**[0.54](1)	**1.87**(1)	20.28[0.53](2)	2.32(2)	20.28[0.52](2)	2.33(2)

Table 9
Competition phase. Group of strategies statistically significantly different (sorted by increasing ranking) provided by Friedman and post-hoc tests for the 12 configurations.

WINNER	COMB	WCOMB
(a) Deseasonalization : No – input selection : No		
1. "MIMO-LOO" (2.90) "DIRMO-WAVG" (3.21) "MIMO-ACFLIN" (3.65) "DIRMO-AVG" (3.78) "DIRMO-SEL" (3.99)	1. "MIMO-LOO" (2.84) "MIMO-ACFLIN" (3.08) "DIRMO-WAVG" (3.52) "DIRMO-AVG" (3.68) "DIRMO-SEL" (4.46) "DIR" (5.21) "REC" (5.24)	1. "MIMO-LOO" (2.79) "MIMO-ACFLIN" (2.79) "DIRMO-WAVG" (3.55) "DIRMO-AVG" (3.84)
2. "REC" (4.96) "DIR" (5.58)	2. "DIRREC" (7.97)	2. "DIRMO-SEL" (4.84) "REC" (5.05) "DIR" (5.20)
3. "DIRREC" (7.93)		3. "DIRREC" (7.95)
(b) Deseasonalization : No – input selection : Yes		
1. "MIMO-LOO" (2.12) "REC" (2.28) "MIMO-ACFLIN" (2.39)	1. "MIMO-LOO" (2.09) "MIMO-ACFLIN" (2.24) "REC" (2.53)	1. "MIMO-LOO" (2.19) "MIMO-ACFLIN" (2.19) "REC" (2.42)
2. "DIR" (3.28)	2. "DIR" (3.22)	2. "DIR" (3.24)
3. "DIRREC" (4.94)	3. "DIRREC" (4.93)	3. "DIRREC" (4.95)
(c) Deseasonalization : Yes – input selection : No		
1. "MIMO-ACFLIN" (3.05) "DIRMO-WAVG" (3.35) "DIRMO-SEL" (3.45) "MIMO-LOO" (3.69) "DIRMO-AVG" (4.47)	1. "DIRMO-WAVG" (3.59) "MIMO-ACFLIN" (3.67) "MIMO-LOO" (3.77) "DIRMO-SEL" (3.77) "DIRMO-AVG" (4.34) "REC" (5.04) "DIRREC" (5.78) "DIR" (6.05)	1. "DIRMO-SEL" (3.45) "DIRMO-WAVG" (3.59) "MIMO-LOO" (3.69) "MIMO-ACFLIN" (3.69) "DIRMO-AVG" (4.38)
2. "REC" (5.47) "DIRREC" (6.15) "DIR" (6.36)		2. "REC" (5.40) "DIRREC" (5.67) "DIR" (6.14)
(d) Deseasonalization : Yes – input selection : Yes		
1. "MIMO-ACFLIN" (1.87) "MIMO-LOO" (2.14)	1. "MIMO-LOO" (2.27) "MIMO-ACFLIN" (2.32)	1. "MIMO-LOO" (2.33) "MIMO-ACFLIN" (2.33)
2. "REC" (3.05)	2. "REC" (2.96)	2. "REC" (2.93)
3. "DIRREC" (3.79) "DIR" (4.14)	3. "DIRREC" (3.50) "DIR" (3.94)	3. "DIRREC" (3.50) "DIR" (3.91)

- The Multiple-Output strategies (MIMO and DIRMO) are invariably the best strategies. They beat the Single-Output strategies, such as DIR, REC, and DIRREC. Both MIMO and DIRMO give comparable performance. For DIRMO, the selection of the parameter s is critical, since it has a great impact on the performance. Should there be an improved selection approach, this strategy would have a big potential.
- Both versions of MIMO are comparable. Also the versions of DIRMO give close results, with perhaps DIRMO-WAVG a little better than the other two versions.
- Among the Single-Output strategies, the REC strategy has almost always a smaller SMAPE and a better ranking than the DIR strategy. DIRREC is the worse strategy overall, and gives especially low accuracy when no deseasonalization is performed.
- Deseasonalization leads to consistently better results (in 38 out of 39 models). This result is consistent with some other studies, such as Zhang and Qi (2005). The possible reason for this is that when no deseasonalization is performed, we are putting a higher burden on the model to forecast the future seasonal pattern plus the trend and the other aspects, which apparently is hard to simultaneously satisfy.

- Input selection is especially beneficial when we perform a deseasonalization. Absent deseasonalization, the results are mixed (as to whether input selection improves the results or not). The possible explanation is that when no deseasonalization is performed, the model needs all the previous cycle to construct the future seasonal pattern. Performing an input selection will deprive it from essential information.
- Concerning the model selection aspect, both combination approaches (COMB and WCOMB) are superior to the winner-take-all (WINNER). Both COMB and WCOMB are comparable, and the results do not differ by much. This is consistent with much of the findings in forecast combination literature, e.g. Andrawis et al. (in press), Clemen (1989), Timmermann (2006) and Andrawis, Atiya, and El-Shishiny (2011a).
- The relative performance and ranking of the different strategies is persistent. Most findings that are based on the pre-competition results are true for the competition phase results. This is also true for the findings concerning the deseasonalization, input selection, and model selection. This persistence is reassuring, as we can have some confidence in relying on the test or validation results for selecting the best strategies.
- The best strategy based on the pre-competition data, the MIMO-ACFLIN method, would have topped all computational intelligence entries of the NN5 competition in the true competition hold-out data.

Table 10
Forecasting errors for different computational intelligence forecasting models which participate to the NN5 forecasting competition.

Model	SMAPE*
MIMO-ACFLIN	**20.28**
Andrawis	20.4
Vogel	20.5
D'yakonov	20.6
Rauch	21.7
Luna	21.8
Wichard	22.1
Gao	22.3
Puma-Villanueva	23.7
Dang	23.77
Pasero	25.3

6. Conclusion

Forecasting a time series many steps into the future is a very hard problem because the larger the forecast horizon, the higher is the uncertainty. In this paper we presented a comparative review of existing strategies for multi-step ahead forecasting, together with an extensive comparison, applied on the 111 time series of the NN5 forecasting competition. The comparison gave some interesting lessons that could help researchers channel their experiments into the most promising approaches. The most consistent findings are that Multiple-Output approaches are invariably better than Single-Output approaches. Also, deseasonalization had a very considerable positive impact on the performance. Finally, the results are clearly quite persistent. So, selecting the best strategy based on testing performance is a very potent approach.

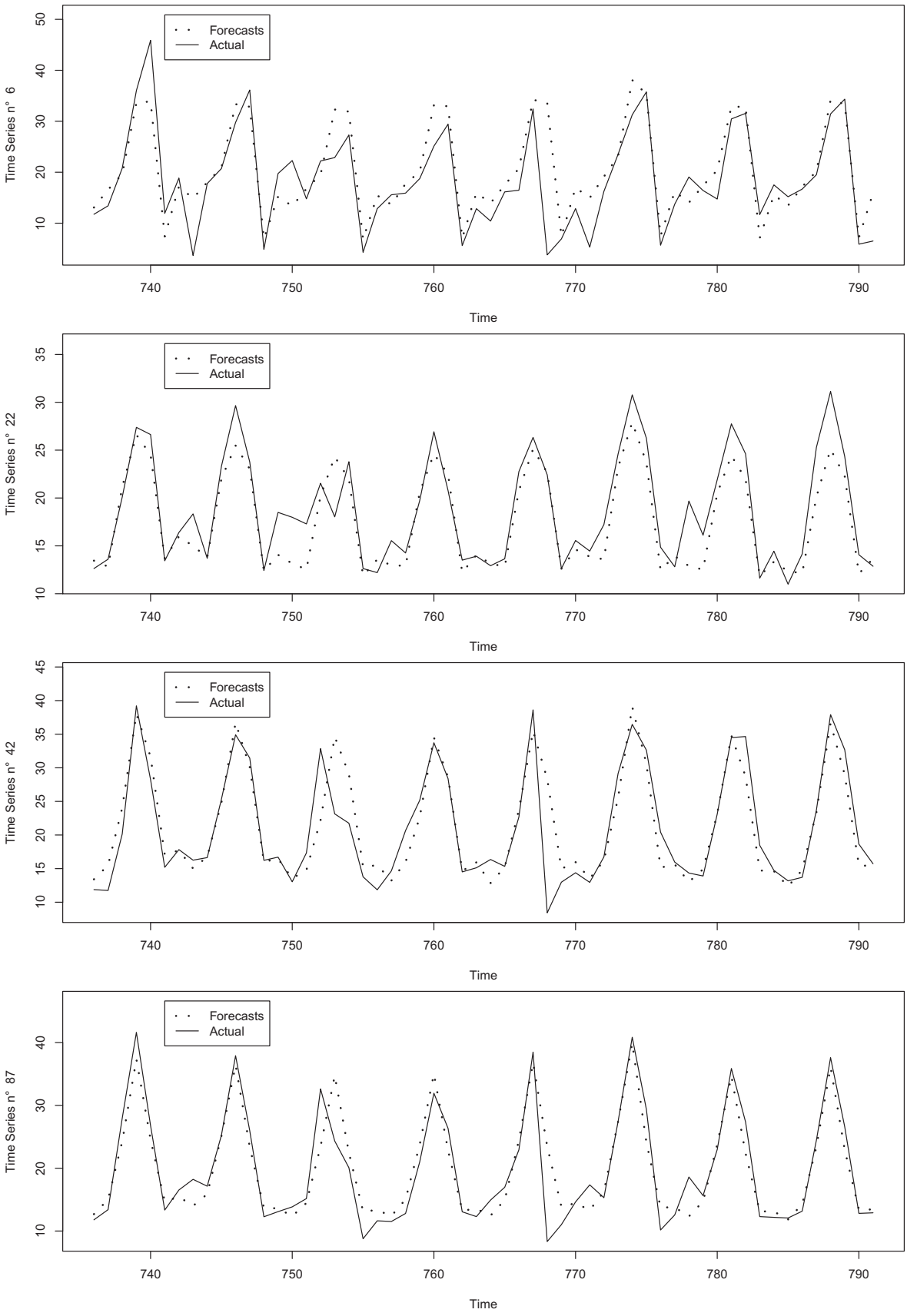

Fig. 7. The forecast versus the actual of the MIMO-ACFLIN strategy for four NN5 time series.

A possible direction for future research could therefore be developing other new improved Multiple-Output strategies. Also, possibly tailoring deseasonalization methods specifically for Multiple-Output strategies could also be a promising research point.

Acknowledgments

We would like to thanks the authors of the paper Garca and Herrera (2009) for making their methods available at http://sci2s.ugr.es/keel/multipleTest.zip.

References

Aha, D. W. (Ed.). (1997). *Lazy learning*. Norwell, MA, USA: Kluwer Academic Publishers. ISBN: 0-7923-4584-3.

Ahmed, N. K., Atiya, A. F., El Gayar, N., & El-Shishiny, H. (2010). An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews (to appear), 29*(5-6).

Allen, D. M. (1974). The relationship between variable selection and data agumentation and a method for prediction. *Technometrics, 16*(1), 125–127. ISSN: 00401706. <http://www.jstor.org/stable/1267500>.

Alpaydin, E. (2010). *Introduction to machine learning. Adaptive computation and machine learning* (2nd ed.,). The MIT Press. ISBN: 978-0-262-01243-0. <http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0262012111>..

Anders, U., & Korn, O. (1999). Model selection in neural networks. *Neural Networks, 12*(2), 309–323. ISSN: 0893-6080.

Andrawis, R. R., Atiya, Amir F., & El-Shishiny, H. (2011a). Combination of long term and short term forecasts, with application to tourism demand forecasting. *International Journal of Forecasting, 27*(3), 870–886.

Andrawis, R. R., Atiya, A. F., & El-Shishiny, H. (in press). Forecast combinations of computational intelligence and linear models for the nn5 time series forecasting competition. *International Journal of Forecasting* (Corrected Proof). ISSN: 0169-2070. <http://www.sciencedirect.com/science/article/B6V92-51WV6JD-2/2/110d69a3e7fdea1d853ee3152755f99a>.

Atiya, A., El-shoura, S. M., Shaheen, S. I., & El-sherif, M. S. (1999). A comparison between neural-network forecasting techniques – Case study: River flow forecasting. *IEEE Transactions on Neural Networks, 10*, 402–409.

Atkeson, C. G., Moore, A. W., & Schaal, S. (1997a). Locally weighted learning. *Artificial Intelligence Review, 11*(1–5), 11–73.

Atkeson, C. G., Moore, A. W., & Schaal, S. (1997b). Locally weighted learning. *Artificial Intelligence Review, 11*(1), 11–73. URL http://www.springerlink.com/index/G8280541763Q0223.pdf.

Bates, J. M., & Granger, C. W. J. (1969). The combination of forecasts. *OR, 20*(4), 451–468. ISSN: 14732858. <http://www.jstor.org/stable/3008764>.

Ben Taieb, S., Bontempi, G., Sorjamaa, A., & Lendasse, A. (2009). Long-term prediction of time series by combining direct and mimo strategies. In *International joint conference on neural networks*. <http://eprints.pascal-network.org/archive/00004925/>.

Ben Taieb, S., Sorjamaa, A., & Bontempi, G. (2010). Multiple-output modeling for multi-step-ahead time series forecasting. *Neurocomputing, 73*(10-12), 1950–1957. ISSN: 0925-2312. <http://www.sciencedirect.com/science/article/B6V10-4YJ6GCW-4/2/8429b80db7773717c9d455b485fb7c4d> (Subspace learning/selected papers from the european symposium on time series prediction).

Birattari, B., & Bersini, M. (1997). Lazy learning for local modeling and control design. <http://citeseer.ist.psu.edu/bontempi97lazy.html>.

Birattari, M., Bontempi, G., & Bersini, H. (1999). Lazy learning meets the recursive least-squares algorithm. In M. S. Kearns, S. A. Solla, & D. A. Cohn (Eds.). *NIPS* (Vol. 11, pp. 375–381). Cambridge: MIT Press.

Bontempi, G. (1999). *Local learning techniques for modeling, prediction and control*. Ph.d., IRIDIA-Université Libre de Bruxelles, BELGIUM.

Bontempi, G. (2008). Long term time series prediction with multi-input multi-output local learning. In *Proceedings of the 2nd European symposium on time series prediction (TSP), ESTSP08, Helsinki, Finland* (pp. 145–154).

Bontempi, G., & Ben Taieb, S. (2011). Conditionally dependent strategies for multiple-step-ahead prediction in local learning. *International Journal of Forecasting, 27*(3), 689–699.

Bontempi, G., Birattari, M., & Bersini, H. (1999). Local learning for iterated time-series prediction. In I. Bratko & S. Dzeroski (Eds.), *Machine learning: Proceedings of the sixteenth international conference* (pp. 32–38). San Francisco, CA: Morgan Kaufmann Publishers.

Bontempi, G., Birattari, M., & Bersini, H. (1998). Lazy learning for iterated time series prediction. In J.A.K. Suykens, & J. Vandewalle (Eds.), *Proceedings of the international workshop on advanced black-box techniques for nonlinear modeling* (pp. 62–68). Belgium: Katholieke Universiteit Leuven.

Breiman, L. (1996). Bagging predictors. *Machine Learning, 24*(2), 123–140. ISSN: 08856125. <http://www.springerlink.com/index/10.1007/BF00058655>.

Breiman, L., & Friedman, J. H. (1997). Predicting multivariate responses in multiple linear regression. *Journal of the Royal Statistical Society, Series B, 59*(1), 3–54.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Belmont, CA: Wadsworth International Group.

Casdagli, M., Eubank, S., Farmer, J. D., & Gibson, J. (1991). State space reconstruction in the presence of noise. *Physica D, 51*, 52–98.

Chapelle, O., & Vapnik, V. (2000). Model selection for support vector machines. *Advances in neural information processing systems* (Vol. 12). MIT Press.

Cheng, H., Tan, P.-N., Gao, J., & Scripps, J. (2006). Multistep-ahead time series prediction. In W. K. Ng, M. Kitsuregawa, J. Li, & K. Chang (Eds.), *PAKDD. Lecture notes in computer science* (Vol. 3918, pp. 765–774). Springer. ISBN: 3-540-33206-5.

Clemen, R. T. (1989). Combining forecasts: A review and annotated bibliography. *International Journal of Forecasting, 5*(4), 559–583. ISSN: 0169-2070.

Clements, M. P., Franses, P. H., & Swanson, N. R. (2004). Forecasting economic and financial time-series with non-linear models. *International Journal of Forecasting, 20*(2), 169–183. ISSN: 0169-207.

Cleveland, W. S., Devlin, S. J., & Grosse, E. (1988). Regression by local fitting: Methods, properties, and computational algorithms. *Journal of Econometrics, 37*(1), 87–114. ISSN: 0304-4076. <http://www.sciencedirect.com/science/article/B6VC0-4582FT0-1K/2/731cd0c23ef342f8d074fdd4e9c41325>.

Crone, S. F. (2009). Mining the past to determine the future: Comments. *International Journal of Forecasting, 25*(3), 456–460. ISSN: 0169-2070. <http://www.sciencedirect.com/science/article/B6V92-4WN8H50-1/2/44b2ded1c6387e7f124db526162db270> (Special section: time series monitoring).

Crone, Sven., 2009a. NN3 Forecasting Competition. http://www.neural-forecasting-competition.com/NN3/index.htm, a. Last update 26/05/2009. Visited on 05/07/2010.

Crone, Sven., 2009b. NN5 Forecasting Competition. http://www.neural-forecasting-competition.com/NN5/index.htm, b. Last update 27/05/2009. Visited on 05/07/2010.

Crone, S. F., & Kourentzes, N. (2009). Input-variable specification for neural networks: an analysis of forecasting low and high time series frequency. In *Proceedings of the 2009 international joint conference on Neural Networks, IJCNN'09 Piscataway, NJ, USA* (pp. 3221–3228). IEEE Press. ISBN: 978-1-4244-3549-4. <http://portal.acm.org/citation.cfm?id=1704555. 1704739>.

Curry, B., & Morga, P. H. (2006). Model selection in neural networks: Some difficulties. *European Journal of Operational Research, 170*(2), 567–577. URL http://ideas.repec.org/a/eee/ejores/v170y2006i2p567-577.html.

De Gooijer, J. G., & Hyndman, R. J. (2006). 25 years of time series forecasting. *International Journal of Forecasting, 22*(3), 443–473. ISSN: 0169-2070.

De Gooijer, J. G., & Kumar, K. (1992). Some recent developments in non-linear time series modelling, testing, and forecasting. *International Journal of Forecasting, 8*(2), 135–156. ISSN: 0169-2070. <http://www.sciencedirect.com/science/article/B6V92-469244K-1/2/cb5cbac7df80324a85e47c96f4a1e290>.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research, 7*, 1–30. ISSN: 1532-4435.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica, 50*(4), 987–1007. ISSN: 00129682. <http://www.jstor.org/stable/1912773>.

Fan, J., & Gijbels, I. (1995). Adaptive order polynomial fitting: Bandwidth robustification and bias reduction. *Journal of Computational and Graphical Statistics, 4*, 213–227.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association, 32*(200), 675–701. ISSN: 01621459. <http://www.jstor.org/stable/2279372>.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics, 11*(1), 86–92. ISSN: 00034851. <http://www.jstor.org/stable/2235971>.

Garca, S., & Herrera, F. (2009). An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research, 9*, 2677–2694. URL http://www.jmlr.org/papers/volume9/garcia08a/garcia08a.pdf.

Guillén, A., Sovilj, D., Mateo, F., Rojas, I., & Lendasse, A. (2008). New methodologies based on delta test for variable selection in regression problems. In *Workshop on parallel architectures and bioinspired algorithms, Toronto, Canada*.

Hamzacebi, C., Akay, D., & Kutay, F. (2009). Comparison of direct and iterative artificial neural network forecast approaches in multi-periodic time series forecasting. *Expert Systems with Applications, 36*(2, Part 2), 3839–3844. ISSN: 0957-4174. <http://www.sciencedirect.com/science/article/B6V03-4S03RD5-6/2/8520e0674be6409b24eb1aca953bdb09>.

Hand, D. (2008). Mining the past to determine the future: Problems and possibilities. *International Journal of Forecasting*. ISSN: 01692070. <http://dx.doi.org/10.1016/j.ijforecast.2008.09.004>.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining* inference and prediction* (2nd ed.). Springer. URL http://www.stat.stanford.edu/tibs/ElemStatLearn/.

Hylleberg, S. (1992). *Modelling seasonality* (2nd ed.). Oxford, UK: Oxford University Press.

Iman, R. L., & Davenport, J. M. (1980). Approximations of the critical region of the friedman statistic. *Communications in Statistics*, 571–595.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive Mixtures of Local Experts. *Neural Computation, 3*(1), 79–87. ISSN: 08997667. <http://www.mitpressjournals.org/doi/abs/10.1162/neco.1991.3.1.79>.

Jordan, M. J., & Jacobs, R. A. (1994). Hierarchical mixtures of experts and the em algorithm. *Neural Computation, 6*, 181–214.

Kantz, H., & Schreiber, T. (2004). *Nonlinear time series analysis*. New York, NY, USA: Cambridge University Press.

Kline, D. M. (2004). Methods for multi-step time series forecasting with neural networks. In G. P. Zhang (Ed.), *Neural networks in business forecasting* (pp. 226–250). Information Science Publishing.

Lapedes, A., & Farber, R. (1987). *Nonlinear signal processing using neural networks: prediction and system modelling*. Technical Report LA-UR-87-2662, Los Alamos National Laboratory, Los Alamos, NM.

Lendasse, A. (Ed.) (2007). *ESTSP 2007: Proceedings*. ISBN: 978-951-22-8601-0.

Lendasse, A., (Ed.) (2008). *ESTSP 2008: Proceedings, Multiprint Oy/ Otamedia*. ISBN: 978-951-22-9544-9.

Liitiäinen, E., & Lendasse, A. (2007). Variable scaling for time series prediction: Application to the ESTSP07 and the NN3 forecasting competitions. In *IJCNN 2007, international joint conference on neural networks, Orlando, Florida, USA* (pp. 2812 – 2816). Eau Claire, Wisconsin, USA: Documation LLC.

Makridakis, S., Wheelwright, S. C., & Hyndman, R. J. (1998). *Forecasting: Methods and Applications*. John Wiley & Sons.

Maron, O., & Moore, A. W. (1997). The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Review, 11*(1), 193–225. <http://dx.doi.org/10.1023/A:1006556606079>.

Mateo, F., & Lendasse, A. (2008). A variable selection approach based on the delta test for extreme learning machine models. In M. Verleysen (Ed.), *Proceedings of the European symposium on time series prediction* (pp. 57–66). Evere, Belgium: d-side publ..

Mateo, F., & Sovilj, D. (2010). Approximate k-NN delta test minimization method using genetic algorithms: Application to time series. *Neurocomputing, 73*(10-12), 2017–2029.

Matías, J. M. (2005). Multi-output nonparametric regression. In C. Bento, A. Cardoso, & G. Dias (Eds.), *EPIA. Lecture notes in computer science* (Vol. 3808, pp. 288–292). Springer. ISBN: 3-540-30737-0.

McNames, J. (1998). A nearest trajectory strategy for time series prediction. In *Proceedings of the internationalworkshop on advanced black-box techniques for nonlinear modeling, Belgium* (pp. 112–128). K.U. Leuven.

Micchelli, C. A., & Pontil, M. A. (2005). On learning vector-valued functions. *Neural Computation, 17*(1), 177–204. ISSN: 0899-7667.

Mitchel, T. M. (1997). *Machine learning*. New York: McGraw-Hill.

Montgomery, D. C., Peck, E. A., & Vining, G. G. (2006). *Introduction to Linear Regression Analysis* (4th ed.). Hoboken: Wiley & Sons. ISBN: 0471754951. <http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0471754951>..

Moody, J., & Darken, C. J. (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation, 1*(2), 281–294.

Murray-Smith, R. (1994). *A local model network approach to nonlinear modelling*. PhD thesis, Department of Computer Science, University of Strathclyde, Strathclyde, UK.

Nelson, M., Hill, T., Remus, W., & O'Connor, M. (1999). Time series forecasting using neural networks: should the data be deseasonalized first? *Journal of Forecasting, 18*(5), 359–367. URL http://www.sciencedirect.com/science/article/B6V92-469244K-1/2/cb5cbac7df80324a85e47c96f4a1e290.

Palit, A. K., & Popovic, D. (2005). *Computational intelligence in time series forecasting: Theory and engineering applications (Advances in industrial control)*. Secaucus, NJ: Springer-Verlag New York, Inc.. ISBN: 1852339489.

Pi, H., & Peterson, C. (1994). Finding the embedding dimension and variable dependencies in time series. *Neural Computation, 6*, 509–520. ISSN: 0899-7667. <http://portal.acm.org/citation.cfm?id=1362347.1362357>.

Poggio, R., & Girosi, F. (1990). Regularization algorithms for learning that are equivalent to multilayer networks. *Science, 247*, 978–982.

Poskitt, D. S., & Tremayne, A. R. (1986). The selection and use of linear and bilinear time series models. *International Journal of Forecasting, 2*(1), 101–114. ISSN: 0169-2070.

Price, S. (2009). Mining the past to determine the future: Comments. *International Journal of Forecasting, 25*(3), 452–455. <http://ideas.repec.org/a/eee/intfor/v25y2009i3p452-455.html>.

Raudys, S., & Zliobaite, I. (2006). The multi-agent system for prediction of financial time series. In *Artificial intelligence and soft computing, ICAISC 2006* (pp. 653–662).

Rumelhart, D. E., Hinton, G. E., & Williams, R. K. (1986). Learning representations by backpropagating errors. *Nature, 323*(9), 533–536.

Ruppert, D., Sheather, S. J., & Wand, M. P. (1995). An effective bandwidth selector for local least squares regression. *Journal of the American Statistical Association, 90*(432), 1257–1270. ISSN: 01621459. <http://www.jstor.org/stable/2291516>.

Saad, E., Prokhorov, D., & Wunsch, D. (1998). Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks. *IEEE Transactions on Neural Networks, 9*(6), 1456–1470. <http://dx.doi.org/10.1109/72.728395>.

Sauer, T. (1994). Time series prediction by using delay coordinate embedding. In A. S. Weigend & N. A. Gershenfeld (Eds.), *Time series prediction: Forecasting the future and understanding the past* (pp. 175–193). Harlow, UK: Addison Wesley.

Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics, 26*(5), 1651–1686. ISSN: 00905364. <http://projecteuclid.org:80/Dienst/getRecord?id=euclid.aos/1024691352/>.

Seber, G. A. F., & Wild, C. J. (1989). *Nonlinear regression*. New York: Wiley.

Sorjamaa, A., & Lendasse, A. (2006). Time series prediction using dirrec strategy. In M. Verleysen (Ed.), ESANN06, *European symposium on artificial neural networks, Bruges, Belgium* (pp. 143–148).

Sorjamaa, A., Hao, J., Reyhani, N., Ji, Y., & Lendasse, A. (2007). Methodology for long-term prediction of time series. *Neurocomputing, 70*(16-18), 2861–2869.

Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics, 15*(1), 116–132.

Tashman, L. J. (2000). Out-of-sample tests of forecasting accuracy: An analysis and review. *International Journal of Forecasting, 16*(4), 437–450. ISSN: 0169-2070.

Tiao, G. C., & Tsay, R. S. (1994). Some advances in non-linear and adaptive modelling in time-series. *Journal of Forecasting, 13*(2), 109–131.

Timmermann, A. (2006). Forecast combinations. In G. Elliott, C. Granger, & A. Timmermann (Eds.), *Handbook of economic forecasting* (pp. 135–196). Elsevier Pub..

Tong, H. (1983). *Threshold models in Nonlinear Time Series Analysis*. Berlin: Springer Verlag.

Tong, H. (1990). *Non-linear Time Series: A Dynamical System Approach*. Oxford University Press.

Tong, H., & Lim, K. S. (1980). Threshold autoregression, limit cycles and cyclical data. *Journal of the Royal Statistical Society. Series B (Methodological), 42*(3), 245–292. ISSN: 00359246. <http://www.jstor.org/stable/2985164>.

Tran, V. T., Yang, B.-S., & Tan, A. C. C. (2009). Multi-step ahead direct prediction for the machine condition prognosis using regression trees and neuro-fuzzy systems. *Expert Systems with Applications, 36*(5), 9378–9387. ISSN: 0957-4174.

Weigend, A. S., & Gershenfeld, N. A. (Eds.) (1994). Time series prediction: Forecasting the future and understanding the past. <http://adsabs.harvard.edu/cgi-bin/nph-bib_query?bibcode=1994tspf.conf.W>.

Weigend, A. S., Huberman, B. A., & Rumelhart, D. E. (1992). Predicting sunspots and exchange rates with connectionist networks. In M. Casdagli & S. Eubank (Eds.), *Nonlinear modeling and forecasting* (pp. 395–432). Addison-Wesley.

Werbos, P. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. PhD thesis, Harvard University, Cambridge, MA.

Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks, 1*(4), 339–356. ISSN: 0893-6080. <http://www.sciencedirect.com/science/article/B6T08-485RHDS-7/2/037e956cda49bd2d2c66085cfccd7de4>..

Wichard, J. D. (2011). Forecasting the nn5 time series with hybrid models. *International Journal of Forecasting, 27*(3), 700–707.

Zhang, G., Eddy Patuwo, B., & Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting, 14*(1), 35–62. ISSN: 0169-2070.

Zhang, X., & Hutchinson, J. (1994). Simple architectures on fast machines: Practical issues in nonlinear time series prediction. In A. S. Weigend & N. A. Gershenfeld (Eds.), *Time series prediction: Forecasting the future and understanding the past* (pp. 219–241). Addison Wesley.

Zhang, G. P., & Qi, M. (2005). Neural network forecasting for seasonal and trend time series. *European Journal of Operational Research, 160*(2), 501–514. ISSN: 0377-2217. <http://www.sciencedirect.com/science/article/B6VCT-4B1SMWY-9/2/24d67e60c11bd47d4d6d6eeac708caeb> (Decision support systems in the internet age).