Cross-validation aggregation for combining autoregressive neural network forecasts

Barrow, DK & Crone, SF

Original citation & hyperlink:
Barrow, DK & Crone, SF 2016, 'Cross-validation aggregation for combining autoregressive neural network forecasts' *International Journal of Forecasting*, vol 32, no. 4, pp. 1120–1137
https://dx.doi.org/10.1016/j.ijforecast.2015.12.011

DOI 10.1016/j.ijforecast.2015.12.011
ISSN 0169-2070

Publisher: Elsevier

Cross-validation aggregation for combining autoregressive neural network forecasts

Devon K. Barrow[1] and Sven F. Crone[2]

[1]*School of Strategy and Leadership, Coventry University, Priory Street, Coventry, CV1 5FB*
[2]*Lancaster University Management School, Department of Management Science, Lancaster, Lancashire, LA1 4YX, UK*

Abstract

This paper evaluates *k*-fold and Monte Carlo cross-validation and aggregation (crogging) for combining neural network autoregressive forecasts. We introduce Monte Carlo crogging which combines bootstrapping and cross-validation in a single approach through repeated random splitting of the original time series into mutually exclusive datasets for training. As the training/validation split is independent of the number of folds, the algorithm offers more flexibility in the size, and number of training samples compared to *k*-fold cross-validation. The study also provides for crogging and bagging: (1) the first systematic evaluation across time series length and combination size, (2) a bias and variance decomposition of the forecast errors to understand improvement gains, and (3) a comparison to established benchmarks of model averaging and selection. Crogging can easily be extended to other autoregressive models. Results on real and simulated series demonstrate significant improvements in forecasting accuracy especially for short time series and long forecast horizons.

Key words: Forecast combination; bootstrapping, Monte Carlo, time series, cross-validation autoregression

[1]Corresponding author. Tel.: +44 (0) 779 626 76 74
Email address: devon.barrow@coventry.ac.uk (D.K. Barrow).
Present address: Coventry University, School of Strategy and Leadership, Faculty of Business and Law, Priory Street, Coventry, CV1 5FB

1. Introduction

Improving the accuracy of a univariate time series forecast remains important in many disciplines, from environmental sciences to business and finance. The approach of combining multiple forecasts has shown particular promise (Clemen and Winkler 1986; Timmermann 2006) as evidenced by various empirical studies (Aksu and Gunter 1992; Macdonald and Marsh 1994; Stock and Watson 2004; Clements and Hendry 2007; Jose and Winkler 2008; Kourentzes, Barrow, and Crone 2014) and objective forecasting competitions (Makridakis et al. 1982; Makridakis and Hibon 2000). The traditional approaches to forecast combination typically involve a set of independent, pre-specified forecasts from different algorithms, which are combined in a second step using a variety of different weighting schemes.

As an alternative to combining predictions of different algorithms, research in machine learning for predictive classification routinely apply repeated subsampling of the dataset on which a single algorithm is parameterised, creating diversity in data rather than in algorithms. Most widely studied, bagging (Breiman 1996a) and *k*-fold cross-validation ensembles (Krogh and Vedelsby 1995) adopt different data resampling techniques, bootstrapping and cross-validation respectively, to actively create diverse estimates of the same base learner algorithm for successive combination of the predictions. Their success in improving performance and robustness of predictions in classification has been empirically proven in a large number of research studies (see e.g. Dietterich 2000; Zhou, Wu, and Tang 2002), with their wide use reflected in published textbooks (see e.g. Perrone and Cooper 1992) and their availability in standard software packages (see e.g. Matlab and Salford Predictive Modeler Software Suite).

Despite both methods having been extended to regression in general, and time series forecasing in particular, this class of algorithms has received relatively limited attention in forecasting research. While bagging has been assessed in select studies, only recently have

Donate et al (2013) and Soric and Lolic (2013) studied cross-validation for time series forecast combination, with promising results (Donate et al. 2013; Sorić and Lolić 2013). However, both studies were constrained to the variant of *k*-fold cross-validation, applying a fixed and predetermined number of subsamples to create diversity. In contrast Monte Carlo cross-validation which combines the benefits of both cross-validation and bootstrapping – repeated random sampling with replacement – in a single approach has been relatively ignored for forecast combination.

In this study we use cross-validation for combining autoregressive forecasts. The forecast combination averages over a set of forecast models trained using mutually exclusive cross-validation replicates, sampled from a given learning set. Within the general framework of cross-validation and aggregating, or crogging for short, we introduce a new method of forecast combination, Monte Carlo crogging and evaluate against *k*-fold crogging and bagging for the first time in a single study. The contributions of this research study are therefore fourfold: (1) the first time application of Monte Carlo cross-validation for forecast combination; (2) the first systematic empirical evaluation of different cross-validation approaches and bagging across data conditions of time series length and equal number of samples using a simulated study on linear and nonlinear data as well as empirical data; (3) an assessment of performance in terms of a bias and variance decomposition of the mean squared error (MSE) of the forecasts; and (4) a comparison of cross-validation to bagging and established benchmark methods of model averaging and model selection utilising the 111 time series of the NN3 competition (Crone, Hibon, and Nikolopoulos 2011).

This paper is organised as follows: in Section 2 which follows, we review the literature on foreast combination, error estimation and data sampling, linking the three main areas of this research. In Section 3, we desribe how boostrapping and cross-validation are applied for forecast combination through bagging and the proposed crogging framework. We

describe several crogging strategies including the proposed combination based on Monte Carlo cross-validation and provide some theoretical insights into crogging to understand why it should be an effective strategy for forecast combination. In Section 4, we evaluate through extensive simulation, the difference between crogging and Bagging in terms of bias and variance, varying combination size and time series length, while Section 5 presents results of the empirical evaluation based on data of the NN3 competition. The final section provides a summary and concluding comments.

2. Forecast combination, error estimation and data sampling

In the 50 years since the seminal paper by Bates and Granger (1969) on forecast combination, the majority of papers have resorted to combining the results of multiple forecast models previously specified, or multiple training initializations thereof, each one parameterized on the same complete learning data. In contrast, recent methods based on bootstrapping and cross-validation focus on model estimation and actively creating diverse predictions over which to average. In this research we focus on cross-validation originally developed for the estimation of prediction error and to faciliate model selection. While our interest is in forecast combination, most existing research on cross-validation exists in the model selection literature (see review by Arlot and Celisse 2010). Here the estimation of predictive accuracy is important, both for evaluating the accuracy of statistical models, and for deciding the final model selected.

The statistical resampling technique of cross-validation (CV) assesses how the results of a statistical estimate will generalize to an independent data set (Stone 1974). Out-of-sample predictive accuracy is esimated by repeatedly spliting the original data into a training set for estimating the model, and a validation set for estimating the error in the predictions. This has the attractive feature of producing nearly unbiased estimates of the preidction error and provides a more representative estimation of the true ex ante performance of the model

(Efron 1983; Kohavi 1995). The technique is used most popurlarly in out-of-sample evaluations with a single hold-out dataset (Tashman 2000) and in specific application areas, such as climate forecasting (Michaelsen 1987), and financial forecasting with statistics and neural networks (Wolff 1987; Clarida et al. 2003; Hu et al. 1999). Despite the advantages in the approach, several research studies have also pointed out its limitations. For example, the advantage in obtaining an unbiased estimation is known to fail when the number of models grows exponentially with the number of observations. Birgé and Massart (2007) and Hardle and Marron (1985) showed that in the presence of outliers, cross-validation was prone to failure. Hart and Wehrly (1986) proved that cross-validation overfits for positively correlated data (see also Opsomer, Wang, and Yang 2001; Altman 1990; Hart 1991), although Burman and Nolan (1992) later showed it to be asymptotically optimal for stationary Markov process though within a specific framework. Less than persuasive early results were also obtained in the case of the leave-one-out cross-validation albeit for error estimation rather than forecast combination (see results of Burman and Nolan 1992; Burman, Chow, and Nolan 1994).

Recent research on cross-validation for time series forecast combination though very few, have produced promising results. Recently, Donate et al. (2013) employed a weighted *k*-fold cross-validation scheme for generating neural network ensembles in predicting six real world time series, improving accuracy for short and medium series in comparison to Holt-Winters exponential smoothing. Around the same time Sorić and Lolić (2013) propose the use of the leave-*h*-out cross-validation (Jackknife) combination for time series forecasting of euro area (EA) inflation, following the work on Jackknifing and model averaging by Hansen and Racine (2012). While the approach did not seem to offer any improvements in forecast accuracy at short horizons, results demonstrated that for the longer-horizons, forecasts were significantly more precise when using cross-validation.

In addition to the limited number studies which evaluate cross-validation for forecast combination, none of which consider Monte Carlo cross-valiation, existing studies also fail to compare cross-validation to the effective benchmark of bagging (short for "bootstrap aggregating"). Bagging preceeded the use of cross-validation for aggregation, employing instead bootstrapping for generating candidate forecasts. Like cross-validation, bootstrapping is an established statistical technique involving data resampling from observed data, used to assign measures of accuracy such as prediction error to a sample estimate (Efron 1979; Efron 1983; Efron and Tibshirani 1993). It is known to be particularly effective at reducing the variance of an estimator, but unlike cross-validation suffers from potentially large bias (Efron 1983; Kohavi 1995). Bagging has become widely applied and researched in time series forecasting, with recent applications in macro-economic forecasting (Watson 2005; Inoue and Kilian 2008), stock market volatility prediction (Hillebrand and Medeiros 2010), meteorological forecasting (Brenning, Andrey, and Mills 2011) and business forecasting (Kourentzes, Barrow, and Crone 2014) to name a few, and applications to new families of methods including exponential smoothing (Bergmeir and Hyndman 2014). This makes it a strong benchmark. Also by considering bagging we make this the first evaluation and comparision of bootstraping and cross-validation sampling strategies for forecast combination.

3. Cross validation and bootstrapping for forecast aggregation

In this section we describe cross-validation and the contender approach of bootstrapping within a general framework for forecast combination of autoregressive models, while introducing Monte Carlo cross-validation for combining forecasts. Given a a univariate time series $\boldsymbol{Y}_T = \{y_1, \dots, y_T\}$, our goal is to forecast at time T, the future H observations $\{y_{T+1}, \dots, y_{T+h}, y_{T+H}\}$ of some variable y. If we assume that the data comes from a possibly

nonlinear autoregressive (AR) process of order p, AR(p), a time series model m of the following form:

$$y_t = m(\mathbf{z}_{t-1}; \boldsymbol{\theta}) + e_t \quad (1)$$

where $\mathbf{z}_{t-1} = [y_{t-1}, \ldots, y_{t-p}]$, $\boldsymbol{\theta}$ are the model parameters to be estimated, and p is the order of the autoregressive lag, can be used to produce the required forecasts. The learning set L used for model parameter estimation consists of the set of $D = T - p$ input/output pairs $\{(y_t, \mathbf{z}_{t-1})\}_{t=p+1}^{T}$, where $\mathbf{z}_{t-1}$ is a sequence of p consecutive (past) observations, and y_t is the one-step-ahead observation in that same sequence to be forecasted. Given a set of K forecasts $\widehat{m}_k^h(\mathbf{z}_{t-1})$, the goal of forecast combination is to produce the combined forecast:

$$\widehat{M}^h(\mathbf{z}_{t-1}) = \sum_{k=1}^{K} w_k \widehat{m}_k^h(\mathbf{z}_{t-1}) \quad (2)$$

where w_k is the weight given to forecasts from model m_k and $\sum_k w_k = 1$.

3.1. Bootstrap aggregating

Breiman (1996a) in a milestone contribution proposed the use bootstrapping as a method for prediction aggregation introducing the bagging algorithm. Rather than use one model trained on only a single learning set L, bagging generates the set $\{L_K\}$ consisting of K repeated bootstraps samples from L, based on which multiple models are estimated and their predictions aggregated. Each bootstrap sample is formed by drawing a set of D pairs at random with replacement from L, according to a discrete uniform distribution, where each pair in L has equal probability of being chosen. In contrast to cross-validation, bootstrapping does not make use of a validation set. Rather the result of bootstrapping is a new dataset for model parameter estimation utilizing approximately $1-(1-(1/D))^D$=63.2% unique observations from the original learning set (Efron 1983). Each of the K bootstrap replicates are then used to estimate a model $\{m_k(\mathbf{z}_{t-1}, L_k)\}$. To aggregate the set of forecasts from these models, one takes the simple average as follows:

$$\widehat{M}^h(\mathbf{z}_{t-1}) = \frac{1}{K}\sum_{k=1}^{K} \widehat{m}_k^h(\mathbf{z}_{t-1}, L_k) \tag{3}$$

In this study we consider the ordinary bootstrap method (Efron 1979) where all the memory of the past required for predicting one-step-ahead is preserved in the lagged autoregressive vectors $\mathbf{z}_{t-1}$. Alternatively this may be viewed as adopting a moving block bootstrap approach (Kunsch 1989; Efron and Tibshirani 1993), where the overlapping bootstrap blocks, correspond to the lagged vectors $\mathbf{z}_{t-1}$, and the length of the bootstrap block is exactly p, the order of the autoregressive term, or larger.

Two popular approaches to the implementation of bagging are evaluated. In the first approach referred to in the classification literature as out-of-bag esimation (Breiman 1996b; Breiman 2001), we first bootstrap the entire learning set *L* to create the training set L^{Train}, and use the remaining observations not selected for training as a validation set L^{Valid}. In this situation the 'out-of-bag' observations, those not selected for training, form a validation set and change with every bootstrap sample. We label this as Bag_{Moob} for bagging using moving 'out-of-bag' observations. In the second approach, we first separate the learning set *L* into a training and validation set. We then bootstrap only the training set, keeping the validation set fixed. As such we call this approach Bag_{Foob}, for bagging with a fixed 'out-of-bag' or fixed validation set. The validation set will be used during neural network training to reduce overfitting. We also later use as a benchmark, the original bagging method without any validation set, bootstrapping the whole learning set.

3.2. Cross-validation and aggregating

Recently cross-validation has been applied to combining forecasts using *k*-fold cross-validation as an alternative to forecasting model selection. Donate et al. (2013) recently applied *k*-fold cross-validation to average forecasts from multiple models trained on different

data subsets created using cross-validation. In this study, we extend our consideration of cross-validaiton beyond *k*-fold cross-validatiion to include several other variants of cross-validation including Monte Carlo cross-validation. This is done within a single framework of cross-validation and aggregating or crogging for short. Each strategy differs depending on the number of learning set splits and whether the resulting training-validation dataset splits are mutually exclusive or overlapping

$\boldsymbol{Y}_T = \{y_1, \dots, y_T\}$: Time series with T observation
1: Generate a learning set $L = \{(y_t, \mathbf{z}_{t-1})\}_{t=p+1}^{T}$ from $\boldsymbol{Y}_T$
2: For a given CV strategy, generate $\{L_K^{Train}\}$, consisting of K CV training sets from L
3: For $k = 1\ toK\{$
   a. Select training set L_k^{Train} from $\{L_K^{Train}\}$
   b. Using set L_k^{Train} construct a model $m_k(\mathbf{z}_{t-1}, L_k^{Train})$
}
4: Combine the forecasts of all models m_k to obtain:

$$\widehat{M}^h(\mathbf{z}_{t-1}) = \frac{1}{K}\sum_{k=1}^{K} \widehat{m}_k^h(\mathbf{z}_{t-1}, L_k^{Train})$$

Figure 1. Pseudo code of Crogging

Figure 1 is a pseudo code of the crogging framework for combining forecasts. Given a time series we generate a set of input/output pairs as described in Section 3. Having selected a cross-validation strategy, described later, we generate a set of K training sets each used to estimate a single forecast model. Each training set is the result of a split of the learning set into a training dataset for estimating the forecast model, and a validation dataset for early stop training to reduce overfitting. Note that a separate test set will be used which will be the holdout sample on which out-of-sample accuracy is evaluated. The aggregate forecast is taken as the simple average of all K forecasts as follows:

$$\widehat{M}^h(\mathbf{z}_{t-1}) = \frac{1}{K}\sum_{k=1}^{K} \widehat{m}_k^h(\mathbf{z}_{t-1}, L_k^{Train}) \quad (4)$$

where L_k^{Train} is the k^{th} training dataset and $\widehat{m}_k(\mathbf{z}_{t-1}, L_k^{Train})$ is the model estimated using that dataset.

3.2.1. *k*-fold crogging

Within the general setting provided in Section 3.2, we define a *k*-fold cross-validation over the learning set *L*, as a division or splitting of *L* into *k* none-overlapping and mutually exclusive subsamples or folds of approximately equal size, with $k \leq D$. The procedure for doing this is depicted in Figure 2.

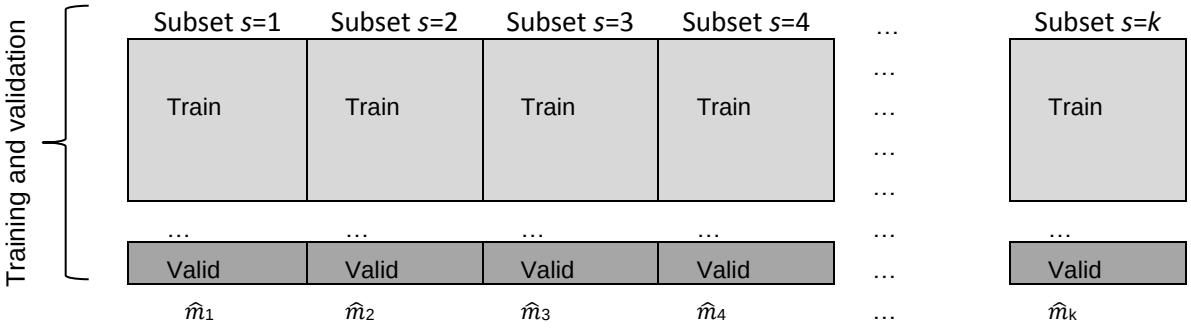

Figure 2. Example of *k*-fold cross-validation

Observations are drawn at random, but unlike in a bootstrap sample, without replacement. In each round k, we obtain a dataset L_k^{Train} comprised of *k*-1 subsamples, and use this to estimate the parameters of a forecast model $\widehat{m}_k(\mathbf{z}_{t-1}, L_k^{Train})$. This process is repeated *k*-times, so that each of the k subsamples are used exactly *k*-1 times as training data and once as validation data (see Figure 2). The combined forecast is then aggregated using Eq. (4), resulting in a combined forecast based on the given k-fold cross-validation strategy. In this case *k* the number of subset folds is also equal to *K* the sampling size. Where a single model is estimated on each sample then *K* is also the combination size.

Typically, *k*-fold cross-validation can apply a different number of folds with different properties depending on the value of *k*. For *k*= 2 we obtain a split of the learning set into 2 subsamples of approximately equal size, the first used for training and the second for

validation, and visa versa. The 10-fold cross-validation is the most commonly applied cross-validation strategy having obtained good results in practice (Kohavi 1995; Hu et al. 1999), and with some theoretical evidence (Bengio and Grandvalet 2004). The learning set is split into 10 approximately equal subsamples, training on a dataset of 9 subsamples, with one subsample for validation. Another common strategy is the leave-one-out (LOO) cross-validation, training on $N-1$ subsamples, with each subsample containing a single observation for validation. The general form of *k*-fold will be evaluated for different sample sizes including 2-fold and 10-fold, while the leave-one-out method will serve as a benchmark for our evaluation on real data.

An overall advantage of k-fold cross-validation, is that each observation is used both for training and validation, with equal weight of $k-1$ during training, and once for validation. A potential drawback is that k controls the trade-off between data available to train each model for a valid in-sample estimation, and data available for validation to estimate out-of-sample accuracy and control overfitting, LOO cross-validation being the extreme case.

3.2.2. Monte Carlo crogging

Each method suffers from one of more limitations as described previously. For example, *k*-fold and leave-one-out cross-validation are known to be asymptotically inconsistent (Efron 1983; Efron and Tibshirani 1986; Shao 1993; Stone 1977; Shao 1997). In the context of forecasting combination this means that both approaches may lead to overfitting, performing well in-sample but poorly out-of-sample. While *k*-fold cross validation has been found in some cases to perform better than leave-one-out (Breiman 1984; Burman 1989; Zhang 1993), it can suffer from unacceptably high variance leading to unreliable estimates (Efron 1983; Kohavi 1995).

In contrast the Monte Carlo cross-validation strategy (Picard and Cook 1984), sometimes referred to as repeated random subsampling validation is known to be assymptotically consistent, and less prone to overfitting (Shao 1993). The reduced overfitting is a consequence of the decoupling of the validation set and sampling size which enhances the potential impact of validation and therefore reduces the risk of overfitting. A larger validation set may however hinder the accurate estimate of the prediction error, due to the smaller training set, a trade-off between model parameter estimation and validation. Where predictive accuracy is the goal as is the case in forecasting, it was shown that Monte Carlo cross-validation provides a larger probability than Leave-one-out cross-validaton of selecting the model with best prediction ability (Shao 1993).

Monte Carlo cross-validation works by randomly splitting the learning set K times, each time randomly drawing without replacement J pairs to form the training set L^{Train}, and using the remaining $D - J$ pairs to form L^{Valid}. In this regard Monte Carlo cross-validation is similar to Bagging where 'out-of-bag' observations are used as validation set, the main difference being that with the later sampling is performed with replacement. On each round of Monte Carlo cross-validation, a forecast model is estimated, and the aggregate of all K forecasts is obtained using Eq. (4). Although the training and validation datasets are mutually exclusive for each round of Monte Carlo cross-validation as in k-fold cross-validation, between rounds an observation may appear in the training or validation dataset any number of times depending on the independent random sampling between rounds, as in bagging. This is because on each round, sampling is performed without replacement (therefore the same observation does not appear in both the training and validation sets), whereas between rounds, sampling is performed with replacement similar to the bootstrap sampling in each round of bagging. Another potential advantage over k-fold cross-validation is in the number of training sets. By decoupling sampling size K from the number of folds as in k-fold cross-

validation, Monte Carlo cross-validation is able to create forecast combinations (training sets) much larger than *k*, theoretically infinite, albeit at the expense of determining another metaparameter of the number of Monte Carlo samples. This study provides the first empirical results on the application of Monte Carlo cross-validation for forecast combination.

3.2.3. Holdout cross-validation

A special case of *k*-fold and Monte Carlo cross-validation widely is the holdout method which results in a single split of the learning set into a training and validation set (Bengio and Grandvalet 2004). With observations of a time series often split sequentially, with the validation data containing the most recent observations consecutively, holdout cross-validation is more similar to *k*-fold than Monte Carlo cross-validation, although non-sequential splitting as in Monte Carlo cross-validation is also feasible. One criticism of the holdout method is that it does not account for the variance with respect to the training set (Dietterich 1998). Research on the optimal number of observations to include in either dataset is also inconclusive, with heuristic rule of thumb, 70%:30% split into training and validation typically applied in practice. As there is only a single data split, the strategy cannot be applied directly for forecast aggregation. However due to its simplicity, the holdout method is widely applied in model selection (Arlot and Celisse 2010), and common in neural network training with early stopping to prevent overfitting. In this study we will use the holdout method as a benchmark for 1) forecast model selection referring to it as $\text{Holdout}_{\text{select}}$ and model averaging referring to it as $\text{Holdout}_{\text{avg}}$. We also evaluate model averaging without the use of a validation set. Each network is trained on the entire learning set and allowed to overfit, with the forecasts subsequently averaged to obtain the combined forecast. We refer to this method as $\text{Noholdout}_{\text{avg}}$.

3.3. Theoretical performance of crogging

In this section we apply the ambiguity decomposition of Brown et al. (2005) in assessing the predictive performance of crogging. We show that for a given observation, the squared error of the combined 1-step-ahead forecast is no more than the average squared error of the individual forecasts. This means that with no guarantee of selecting a forecast with error lower than the combined forecast, we are at least guaranteeed of having 1-step-ahead performance better on average than a forecast selected at random.

For crogging, we write the squared error (SE) of the combined forecast given by Eq. (2) for the 1-step-ahead forecast as $\text{SE} = \left(\widehat{M}(\mathbf{z}_{t-1}) - y_t\right)^2$. In comparison the average squared error of the constituent forecast models can be expressed as:

$$\begin{aligned}\sum_k \frac{1}{K}(\widehat{m}_k(\mathbf{z}_{t-1}) - y_t)^2 &= \sum_k \frac{1}{K}\left(\widehat{m}_k(\mathbf{z}_{t-1}) - \widehat{M}(\mathbf{z}_{t-1}) + \widehat{M}(\mathbf{z}_{t-1}) - y_t\right)^2 \\ &= \sum_k \frac{1}{K}\Big[\left(\widehat{m}_k(\mathbf{z}_{t-1}) - \widehat{M}(\mathbf{z}_{t-1})\right)^2 + \left(\widehat{M}(\mathbf{z}_{t-1}) - y_t\right)^2 \\ &\quad + 2\left(\widehat{m}_k(\mathbf{z}_{t-1}) - \widehat{M}(\mathbf{z}_{t-1})\right)\left(\widehat{M}(\mathbf{z}_{t-1}) - y_t\right)\Big]\end{aligned}$$

Using $\sum_k \frac{1}{K} = 1$ and $\widehat{M}(\mathbf{z}_{t-1}) = \sum_k \frac{1}{K}\widehat{m}_k(\mathbf{z}_{t-1})$ cross-terms disappear and we get:

$$\begin{aligned}\sum_k \frac{1}{K}(\widehat{m}_k(\mathbf{z}_{t-1}) - y_t)^2 &= \sum_k \frac{1}{K}\left(\widehat{m}_k(\mathbf{z}_{t-1}) - \widehat{M}(\mathbf{z}_{t-1})\right)^2 + \left(\widehat{M}(\mathbf{z}_{t-1}) - y_t\right)^2 \\ &= \textstyle\sum_k \frac{1}{K}\left(\widehat{m}_k(\mathbf{z}_{t-1}) - \widehat{M}(\mathbf{z}_{t-1})\right)^2 + \text{SE}\end{aligned}$$

Rearranging we obtain:

$$\text{SE}_h = \sum_k \frac{1}{K}(\widehat{m}_k(\mathbf{z}_{t-1}) - y_t)^2 - \sum_k \frac{1}{K}\left(\widehat{m}_k(\mathbf{z}_{t-1}) - \widehat{M}(\mathbf{z}_{t-1})\right)^2$$

Observe that the second term $\sum_k \frac{1}{K}(\widehat{m}_k(\mathbf{z}_{t-1}) - \widehat{M}(\mathbf{z}_{t-1}))^2$, the ambiguity term, is always positive and therefore reduces the first term, the average error of the individual forecasts. The larger the ambiguity term or equivalently the variance among the individual forecasts, the smaller the error of the combined forecast, SE. This means that the more variable the

individual 1-step-ahead forecasts $\hat{m}_k(\mathbf{z}_{t-1})$ generated based on the cross-validation replicates of L, the larger the potential improvement from crogging. Conversely if the 1-step-ahead forecasts are very similar, the ambiguity will be small and the error of the combined forecast will be close to the average error of the individual forecasts.

While an *h*-step-ahead analysis is beyond the scope of this study, it has been shown that the 1-step-ahead bias and variance affects the *h*-step-ahead bias and variance for the recursive multi-step forecasting strategy (Taieb and Atiya 2015). In particular they find that for complex model's such as neural networks (NNs), both the bias and variance tend to increase with the forecast horizon, in particular due to the large variance. We hypothesize that as cross-validation involves systematic resampling and training on the complete learning set, it will be effective at increasing ambiguity, while not adversely affecting the bias of the individual forecasts, and as a consequence improve the accuracy of the combined forecast. In the next section we investigate the performance of crogging from a bias and variance perpective estimated via Monte Carlo simulation. Future research should pursue a more detailed theoretical analysis of the *h*-step-ahead bias and variance in assessing the impact of each method.

4. Bias and variance decomposition of Crogging

4.1. Overview

In order to assess the efficacy of *k*-fold and Monte Carlo crogging algorithms, we carry out a Monte Carlo simulation study to investigate the impact of the size of the forecast combination, and the time series length on the bias and variance of the prediction. With forecast performance measured by MSE, we decompose the MSE of the combined forecast into its bias and variance components (Geman, Bienenstock, and Doursat 1992), using the decomposition methodology of Taieb and Hyndman (2014) for multi-step-ahead forecasting.

The results are compared to bagging in order to allow a comparison to its better understood properties.

We adopt the same terminology as in Taieb and Hyndman (2014) and assuming that the process defined in Eq. (1) is stationary, we obtain the bias and variance components of the mean squared error of h-step-ahead combined forecast MSE_h as follows:

$$\begin{aligned}
&\mathrm{MSE}_h = \mathbb{E}_{\mathbf{x}_t}\left[\left(y_{t+h} - \widehat{M}^h(\mathbf{z}_{t-1})\right)^2 \middle| \mathbf{x}_t\right] \\
&= \mathbb{E}_{\mathbf{x}_t,\varepsilon}\left[\left(y_{t+h} - \mu_{t+h|t}\right)^2 \middle| \mathbf{x}_t\right] && \text{Noise} \\
&+\mathbb{E}_{\mathbf{x}_t}\left[\left(\mu_{t+h|t} - M^h(\mathbf{z}_{t-1})\right)^2\right] && \text{Squared Bias} \\
&+\mathbb{E}_{\mathbf{x}_t,\boldsymbol{y}_T}\left[\left(\widehat{M}^h(\mathbf{z}_{t-1}) - M^h(\mathbf{z}_{t-1})\right)^2 \middle| \mathbf{x}_t\right] && \text{Variance}
\end{aligned} \tag{5}$$

where $M^h(\mathbf{z}_{t-1}) = \mathbb{E}\left[\widehat{M}^h(\mathbf{z}_{t-1})\right]$, $\widehat{M}^h(\mathbf{z}_{t-1})$ is the combined h-step-ahead forecast for a given combination strategy, and $\mathbb{E}_x$ and $\mathbb{E}[\cdot|x]$ represent the expectation over x, and the expectation conditional on x, respectively. The resulting decomposition gives us a measure of the noise, the squared bias, and an estimate of the variance of the combination method.

4.2. Experimental design and data

For each combination strategy we estimate Eq.(5) via simulation considering a linear AR(6) and a nonlinear Smooth Transition Autoregressive (STAR) data generating process (DGP) also used by Ben Taieb and Hyndman (2014). The linear AR(6) process is given by:

$$\begin{aligned}
&y_t = 1.32y_{t-1} - 0.52y_{t-2} - 0.16y_{t-3} + 0.18y_{t-4} - 0.26y_{t-5} + \\
&0.19y_{t-6} + \varepsilon_t \quad .
\end{aligned} \tag{6}$$

where $\varepsilon_t \sim \mathrm{NID}(0,1)$. The STAR process has been used in several other studies (e.g. Terasvirta and Anderson 1992; Berardi and Zhang 2003) for understanding nonlinearities in the context of autoregressive forecast models and is given by:

$$y_t = 0.3y_{t-1} + 0.6y_{t-2} + (0.1 - 0.9y_{t-1} + 0.8y_{t-2})\left[1 + e^{(-10y_{t-1})}\right]^{-1} + \varepsilon_t \quad (7)$$

where$\varepsilon_t \sim \text{NID}(0, \sigma^2)$ and the error variance set to $\sigma^2 = 0.05^2$.

We generate time series of length $T \in \{50, 400\}$ in order to assess the impact of time series length on the bias and variance of each strategy. The number of forecasts included in the final combination is deemed critical to its performance (de Menezes, Bunn, and Taylor 2000). This is determined by the number of training sets or sampling size of each strategy which in turn determines the number of models which can be estimated. The sampling sizes evaluated are taken from the set $\{2, 5, 10,15, 20, 30\}$. For *k*-fold cross-validation this sample size is also equalivalent to the number of folds k, while for Monte Carlo crogging and bagging this is equivalent to the number of random splits and the number of bootstraps respectively. If for every sample bootstrap or cross-validation a single model is estimated, then the sampling size is also equal to the forecast combination size K.

4.3. Estimation

We generate for each DGP, a set of 1000 independent time series $S_i = \{y_1, \dots, y_T\}$ on which to train and estimate model parameters. To give an objective measure of the bias and variance components, we generate an independent time series from the same DGP serving as a test set. From this independent series, we obtain a set of 2000 input/output pairs $\{(\mathbf{y}_j, \mathbf{z}_j)\}_{j=1}^{2000}$ where $\mathbf{z}_j$ is the lagged vectors of inputs, and the vector $\mathbf{y}_j$ is the next *H* consecutive observations representing the lead time to be forecasted, in our case set to 10. The MSE is then calculated as follows:

$$\text{MSE}_h = \frac{1}{1000 \times 2000} \sum_{i=1}^{1000} \sum_{j=1}^{2000} \left(\mathbf{y}_j^h - \widehat{M}_{S_i}^h(\mathbf{z}_j)\right)^2$$

The three components can be estimated as follows:

$$\text{Noise}_h = \frac{1}{2000}\sum_{j=1}^{2000}\left(\mathbf{y}_j^h - \mathbb{E}\left[\mathbf{y}_j^h \mid \mathbf{z}_j\right]\right)^2$$

$$\text{Bias}_h^2 = \frac{1}{2000}\sum_{j=1}^{2000}\left(\mathbb{E}\left[\mathbf{y}_j^h \mid \mathbf{z}_j\right] - \bar{M}(\mathbf{z}_j)\right)^2$$

$$\text{Variance}_h = \frac{1}{1000 \times 2000}\sum_{i=1}^{1000}\sum_{j=1}^{2000}\left(\hat{M}_{S_i}^h(\mathbf{z}_j) - \bar{M}(\mathbf{z}_j)\right)^2$$

where $\bar{M}(\mathbf{z}_j) = \frac{1}{1000}\sum_{j=1}^{1000}\hat{M}_{S_i}^h(\mathbf{z}_j)$ is an estimate of $M^h(\mathbf{z}_j)$ and $\hat{M}_{S_i}^h(\mathbf{x}_j)$ is the combined forecast produced using dataset S_i. The variable $\mathbf{y}_j^h$ is the h^{th} element of the vector $\mathbf{y}_j$, representing the *h*-step-ahead forecast. Throughout this study, forecasts are produced using the recursive strategy due to its simplicty, intuition, widespread use in research and practice, and reduced computational load. This is in contrast to the direct strategy which although being immune to propagation of forecast errors, would require a different model combination for each forecast horizon, and becoming rather intensive computationally (Taieb et al. 2012). The conditional mean $\mathbb{E}\left[\mathbf{y}_j^h \mid \mathbf{z}_j\right]$ for the linear process is calculated analytically. In the case of the nonlinear process, we average over a large number of possible values for each future time point of the series using simulation. In all cases we use a Multilayer Perceptron (MLP), a feedforward neural network capable of approximating linear and nonlinear data generating processes. We employ the same MLP setup as described in Section 5.3 with $p = 6$. For Monte Carlo crogging we set the training-validation split to 70% - 30%.

4.4. Experimental Results

Results of the bias-variance decomposition including the MSE (first column), the bias (second column) and the variance (third column) for the linear and nonlinear DGP are shown in Figure 3 and Figure 4 respectively, for short series having length $T = 50$. The MSE provides a measure of the forecast error of the different combination approaches, Monte Carlo, k-fold, Bag_{Moob} and Bag_{Foob}, while the decomposed bias and variance allows an

examination of the strengths of the competing approaches across sampling sizes. For both DGPs, we can see that the largest of the three components is the bias which, as the number of samples and forecast horizon increases, is nearly two to three times as large as the variance. This suggests that the neural network base model structure with 2 hidden nodes and $p = 6$ autoregressive inputs, may not be sufficient to approximate the underlying DGP in the presence of the given noise level. Nevertheless, this scenario reflects the core challenge in real forecasting problems, where often the 'true' model stucture is not known in advance of model fitting and the data has significant levels of noise relative to the signal in the data. In contrast, where the model structure is known, then the DGP can trivally be estimated to high accuracy using NNs.

Considering the differences between the methods, Figure 3 shows that for the linear process, Bag_{Foob} and *k*-fold both have the highest variance while Monte Carlo consistently has the smallest variance on average across all horizons. This improvement has however not induced a large increase in bias. On the contrary, the bias of Monte Carlo is generally equal to, if not less than other methods, and consequently the forecasts for Monte Carlo outperform bagging and *k*-fold based on MSE.

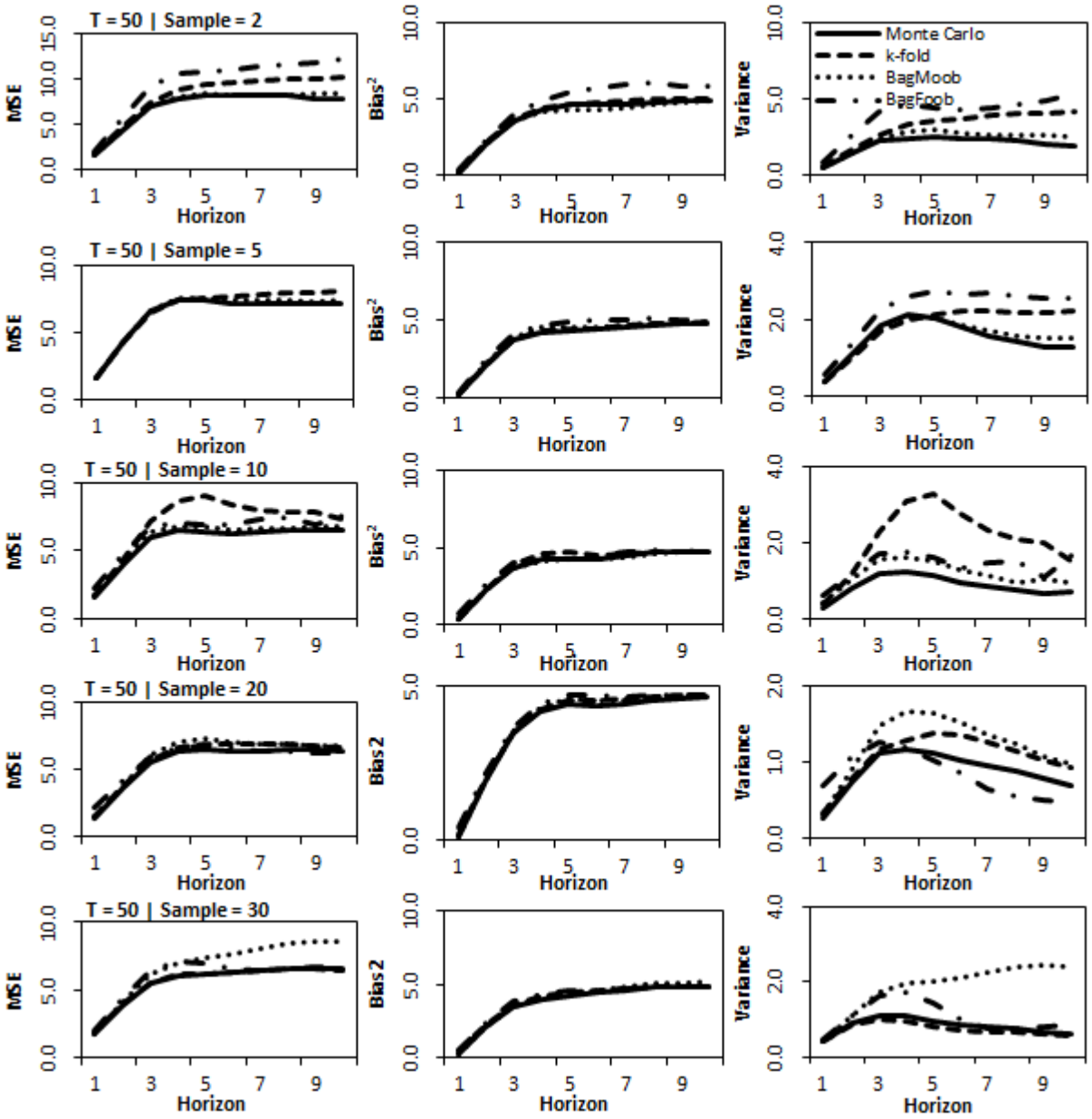

Figure 3.Decomposition for linear AR(6) processfor different sampling sizes showing MSE, bias and variance for time series length T=50 and forecast horizon of10.

Bag_{Moob} is often nearly as good as Monte Carlo particularly at small sampling sizes where it has similar performance in terms of variance and on average across horizons outperforms *k*-fold and Bag_{Foob}. As sampling size increases up to 30 samples, this relative performance in terms of variance is still present; however the difference between methods in terms of bias is smaller. The difference in performance among methods appear to come mainly from the reduction in the variance of the combined forecast.

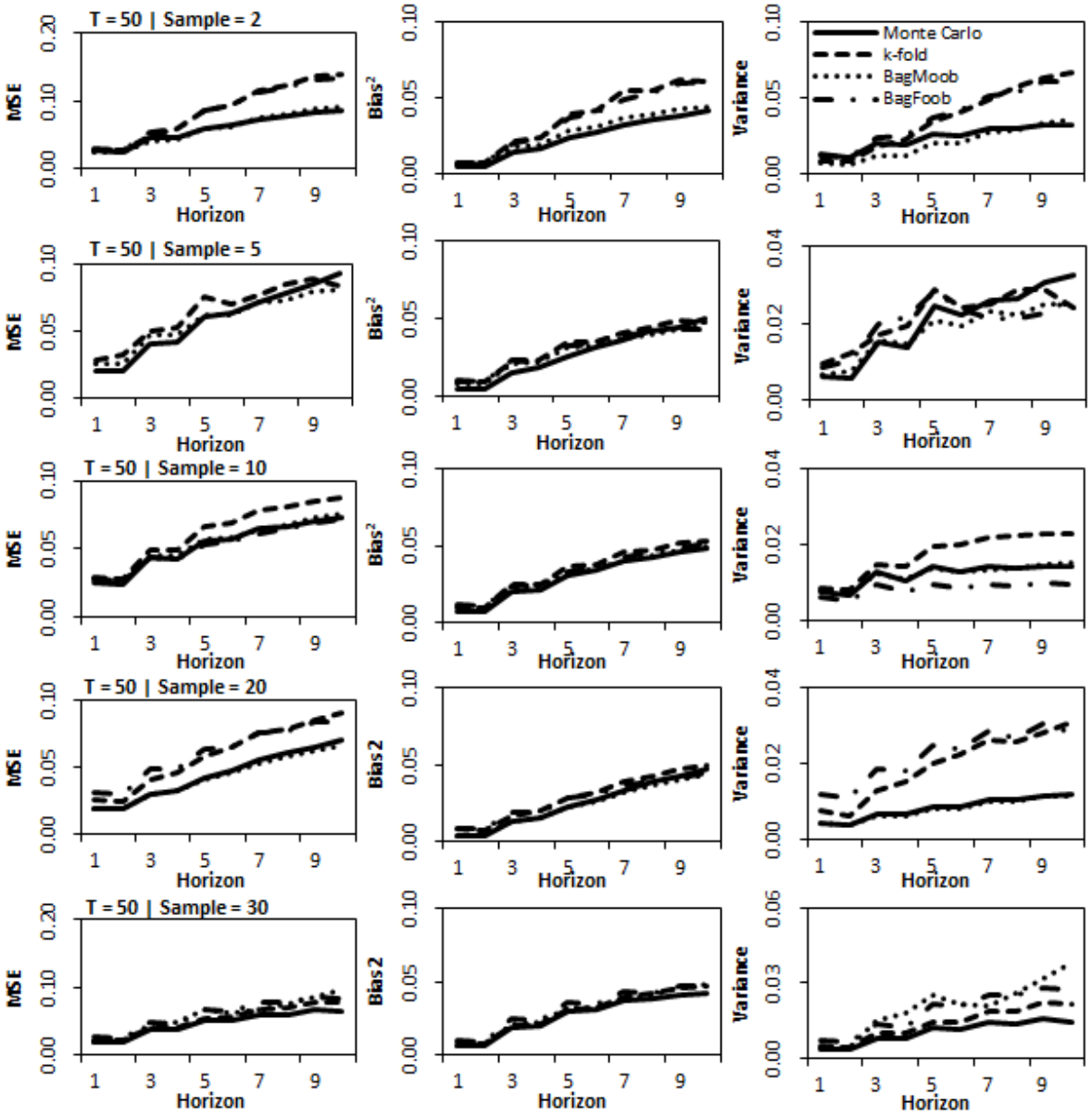

Figure 4.Decomposition for nonlinear STAR processfor different sampling sizes showing MSE, bias and variance for time series length T=50 and forecast horizon of 10.

Results of the nonlinear DGP shown in Figure 4 are similar to those obtained on the linear DGP, with Monte Carlo on average having the lowest variance and best forecasts in terms of MSE across sampling sizes. This is followed closely by Bag_{Moob} which at sampling sizes less than 20 performs similarly. As sample size increases, both Bag_{Foob} and *k*-fold improve in performance however no consistent difference is noted between the two methods. However with the exception of sampling size 10 where Bag_{Foob} performs well on variance, Monte Carlo is always better.

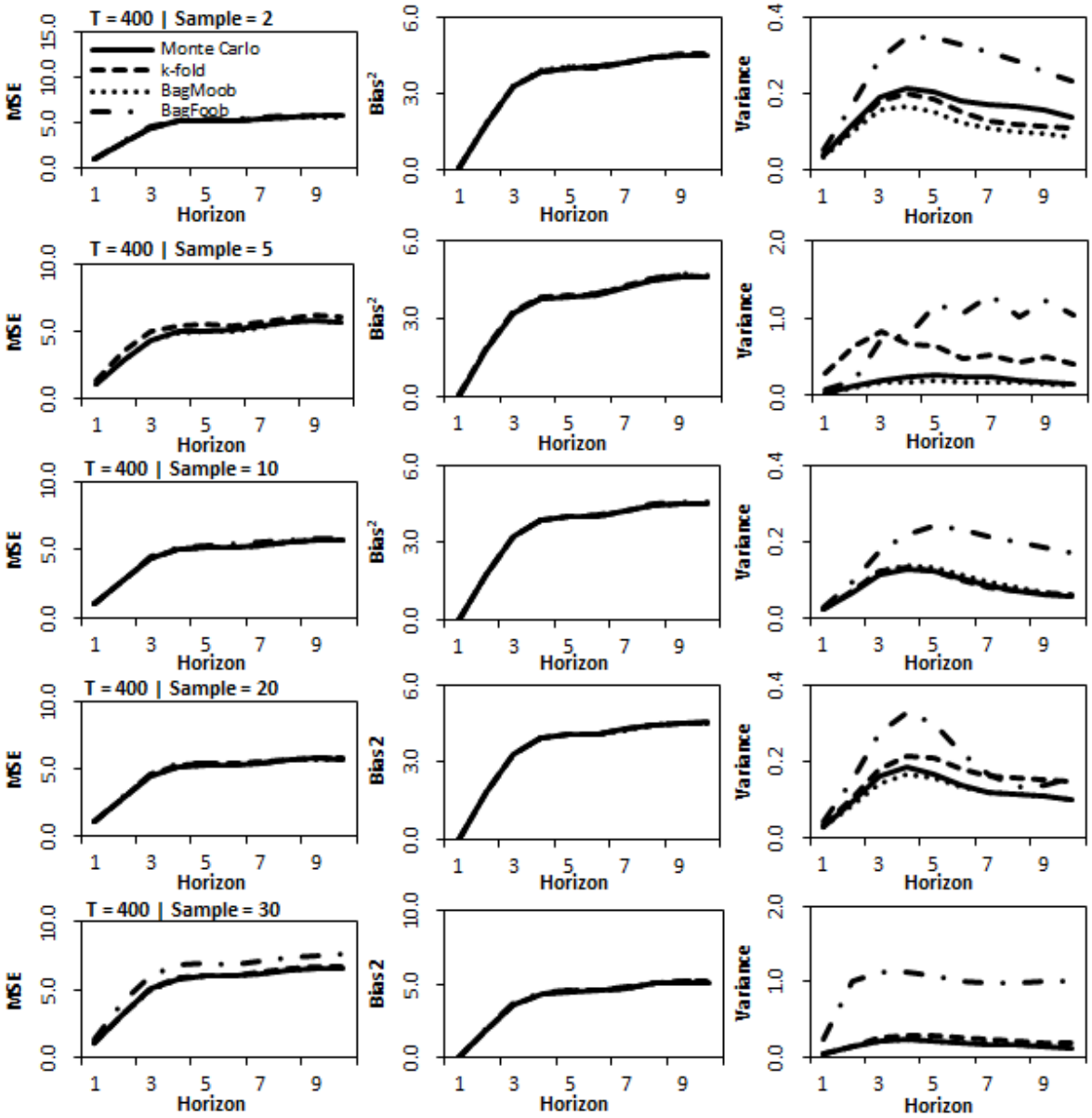

Figure 5.Decomposition for linear AR(6)processfor different sampling sizes showing MSE, bias and variance for time series length T=400 and forecast horizon of 10.

The evidence therefore indicates that Monte Carlo is best at reducing variance while not adversely increasing the bias of the combined forecast, and that $Bag_{\mathbf{Moob}}$ is on average always better than $Bag_{\mathbf{Foob}}$. In fact, for both the linear and nonlinear DGP and across all sampling sizes, Monte Carlo always produces a forecast having smaller bias than ***k***-fold and $Bag_{\mathbf{Foob}}$, and which consequently leads to improved accuracy overall. On linear time series, *k*-fold on average outperforms Bag_{Foob} across all sampling sizes and forecast horizons while for nonlinear time series at length 50, the difference in performance is less clear.

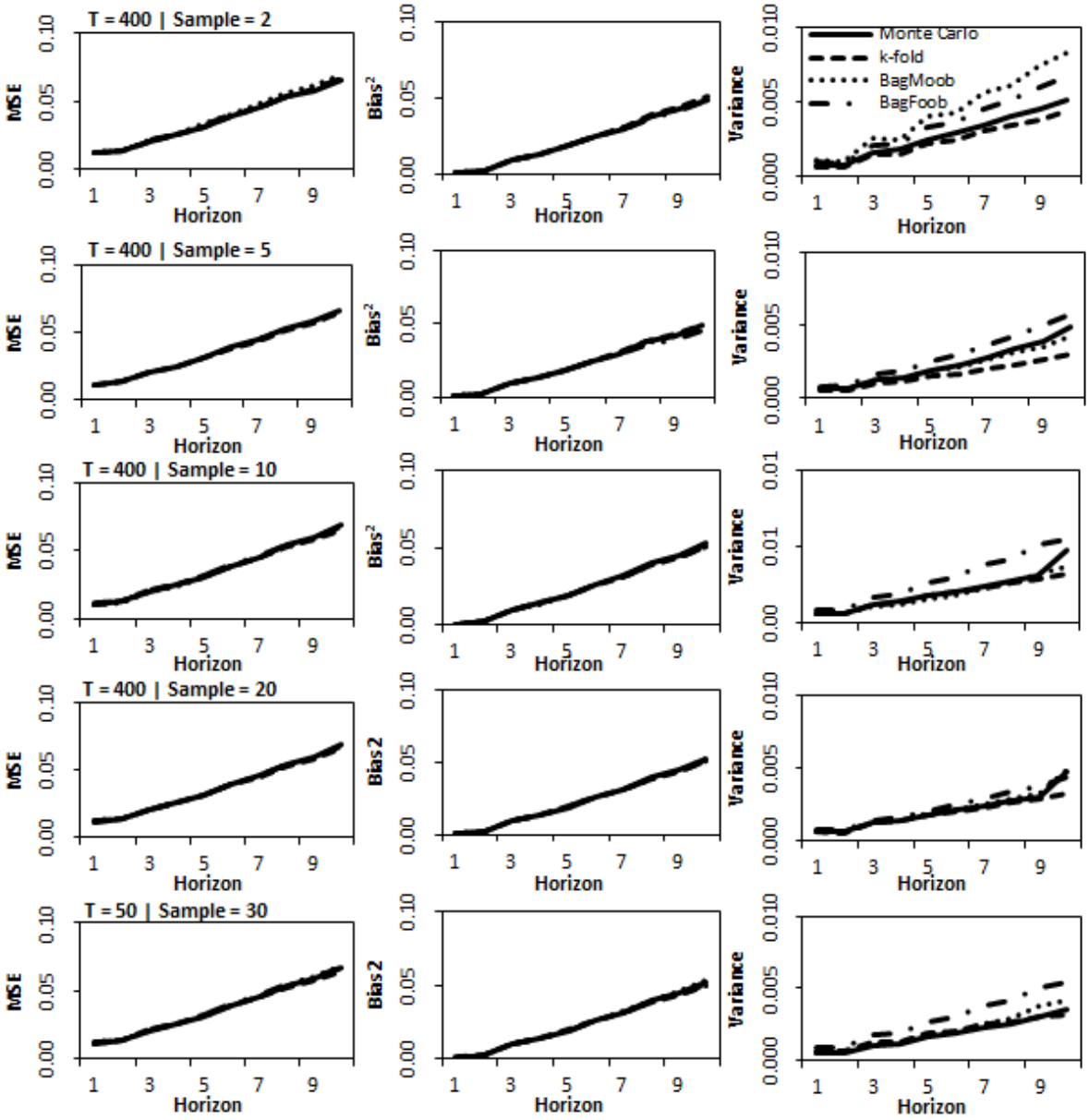

Figure 6.Decomposition for nonlinear STARprocessfor different sampling sizes showing MSE, bias and variance for time series length T=400 and forecast horizon of 10.

This is possibly due to the sampling scheme of cross-validation and Bag_{Moob} which uses a set of 'out of bag' observations which guarantees that all observations in the learning set are used for training. In contrast for Bag_{Foob} observations in the validation set are never used for training. Additionally for short time series, linear or nonlinear Monte Carlo which like bagging involves some random sampling appears to be much more effective at reducing variance than *k*-fold.

These results hold, if not much clearer, for the longer time series having 400 observations. For both the linear and nonlinear DGP, total forecast error is reduced with the significant reductions coming from the variance component. In fact the bias for all four methods, which is now considerably larger than the variance, is nearly equal across all sample sizes for all methods, while the variance of Monte Carlo and Bag_{Moob} is on average always lower than *k*-fold and Bag_{Foob}. Though small this results in a consistent improvement in MSE from using Monte Carlo over *k*-fold crogging, and Bag_{Moob} compared to Bag_{Foob}.

5. Empirical Evaluation Experiment

5.1. Design of combination methods

In this real-world experiment, we compare the forecasting accuracy of k-fold and Monte Carlo crogging to bagging, conventional neural network model averaging over multiple initialisations, and individual neural network model selection. One objective of this study is to determine which, if any, of the combination strategies is best for forecast combination, and under what conditions of sample size, and time series length. For each method we use the same number of samples to train, using the identical neural network set up and weight initialisation, to ensure that any differences in the forecast accuracy are attributable directly to the method of choice, allowing a fair and robust comparison. We train a total of 50 networks, each with different random starting weights to account for error variance from local minima in the network training.

For k-fold cross-validation we evaluate $k = 2, 5, 10, 15, 20, 25 \text{ and } 30$ producing a corresponding number of data splits for training. For example, in the case of 2-fold cross-validation, we obtain 2 splits of the learning set and for each split, train 50 similarly initialized networks producing altogether 100 forecasts. Similarly when k=10 we have 10-fold cross-validation, and a combination size of 10×50 forecasts. In order to provide a

direct comparison with *k*-fold crogging, we again create Monte Carlo cross-validation samples of size $2, 5, 10, 15, 20, 25$ and 30 using the same randomly initialized networks. We implement Bag_{Foob}, and Bag_{Moob} in the same manner allowing a fair and robust comparison.

5.2. NN3 competition Dataset

To empirically evaluate the performance of each method, we utilise the 111 time series from the NN3 competition dataset consisting of a representative set of long and short, seasonal and non-seasonal monthly time series drawn from a homogenous population of empirical business series (Crone, Hibon, and Nikolopoulos 2011). The time series contain between 68 and 144 observations. The reduced dataset contains a mixture of all time series types of which three are characterised as difficult to forecast, 4 as seasonal and the remaining 7 as non-seasonal containing also outliers and structural breaks. A summary of the characteristics of the time series is provided in Table 1.

Table 1: Summary description of NN3 competition time series dataset

	Complete Dataset				
			Reduced Dataset		
	Short	Long	Normal	Difficult	SUM
Non-Seasonal	25 (NS)	25 (NL)	4 (NN)	3 (ND)	57
Seasonal	25 (SS)	25 (SL)	4 (SN)	-	54
SUM	50	50	8	3	111

For each method using a fixed validation set the number of observations is set to 14 to allow for estimating of monthly seasonlity in the shortest possible series being 68 observations and with 18 needed for the test dataset. However these observations are not always fixed as with Monte Carlo crogging and Bag_{Moob} which sample different obervations for the validation set.

5.3. Design of benchmark Algorithms

For each of the benchmarks previously described, $\text{Holdout}_{\text{select}}$, $\text{Holdout}_{\text{avg}}$,

$\text{Noholdout}_{\text{avg}}$ and Bag_{Noob} we use the same identically initialized 50 MLPs as with the principal combination methods. In the case of $\text{Holdout}_{\text{select}}$ or individual model selection we select the network from the 50 differently initialized MLPs having the smallest MSE on the fixed size validation set. For $\text{Holdout}_{\text{avg}}$ we average over the predictions of these same networks while for the $\text{Noholdout}_{\text{avg}}$ method we average over each network this time trained without a validation set. This approach of averaging over multiple weight initializations of the same neural network architecuture is also known as neural network model averaging (Hansen and Salamon 1990). Both model averaging and model selection are two established methods of building neural network models for time series forecasting (Zhang and Berardi 2001; Naftaly, Intrator, and Horn 1997) both based on the Holdout method. Consequently they provide strong benchmarks for this study, and allow investigating the benefits of cross-validation versus ordinary validation (holdout). In the case of Bag_{Noob} we train each of the 50 networks on a bootstrap sample but unlike Bag_{Foob} and BagMoob no validation set is used therefore the entire learning set is bootstrapped.

5.4. MLP setup

For the implementation of crogging, bagging, neural network averaging and selection we use the same base learner of a univariate Multilayer Perceptron (MLP). MLPs are well researched, and their ability to approximate and generalize well any functional relationship to an arbitrary degree of accuracy has been proven (Hornik, Stinchcombe, and White 1989; Hornik 1991). In particular, they have been shown empirically to be able to forecast linear and nonlinear time series of different forms (Zhang, Patuwo, and Hu 1998). The functional form of these networks is given by:

$$\widehat{m}(\mathbf{z}_t) = \beta_0 + \sum_{k=1}^{H} \beta_h g\left(\gamma_{0i} + \sum_{i=0}^{I} \gamma_{hi}\, p_i\right) \tag{8}$$

with $I = 13$, inputs p_i, connected to each of H hidden nodes in a single hidden layer using the hyperbolic tangent transfer function, and a single output node with an identity function. This is sufficient to model monthly seasonality stochastically in addition to trends effectively combining an $AR(12)$ and $AR(1)$ process. To evaluate the impact of model complexity we consider neural networks having hidden nodes $H = 1, \dots, 5$. The architecture of the MLP is otherwise exactly the same allowing a fair assessment of the impact of the number of hidden nodes on performance of each combination method. Each time series is modelled directly without prior differencing or further data transformation to estimate level, seasonality, and potential trend directly in the network weights and the bias terms. Additionally all time series are linearly scaled into the interval of [-0.5, 0.5] to allow headroom for possible non-stationarity prior to training. We produce multistep forecasts using an iterative prediction, recursively generating one-step-ahead forecasts.

For parameter estimation the Levenberg-Marquardt algorithm (Hagan, Demuth, and Beale 1996) is used to minimise the MSEloss function up to a maximum of 1000 epochs. The algorithm requires setting a scalar μ_{LM} and its increase and decrease steps, using $\mu_{LM} = 10^{-3}$, with an increase factor of $\mu_{inc} = 10$ and a decrease factor of $\mu_{dec} = 10^{-1}$. All network training employ an early stopping criterion in order to avoid overfitting. This means that we track the MSE on the training and the validation set, and halt the training process and retain the network weights with the lowest error on the validation data after the error has not decrease for more than 50 epochs, or if μ_{LM} exceeds $\mu_{max} = 10^{10}$. For each new forecast model, we randomly initialize the starting weights for each MLP allowing for different solutions of the network to be achieved, taking care to ensure that the same starting weights are used for each method. This is in addition to the randomness introduced by the *k*-fold, Monte Carlo and bootstrap methods.

5.5. Evaluation

The forecast horizon for all methods is set to 12 months using a holdout sample of 1 to 18 months in the future, and a rolling origin evaluation to assess forecasting accuracy and performance (Tashman 2000). The size of the validation set during training depends on the method used, leaving the remaining observations for training. Where a fixed size validation set is used then it is set to 14 observations as explained in Section 5.2. In comparing results to that of the NN3 competitors (benchmarking), the forecast horizon will later be extended to 18 steps ahead forecasting from a fixed origin, as required by competition guidelines. We calculate the mean absolute scaled error (MASE) and the symmetric mean absolute error (SMAPE) for all methods in assessing forecast accuracy and performance. For a given actual X_t, and forecast F_t the SMAPE (Chen and Yang 2004) provides a scale independent measure that can be used to compare accuracy across time series . It is calculated as follows:

$$SMAPE = \frac{1}{N}\sum_{t=1}^{N}\left(\frac{|X_t - F_t|}{(|X_t| + |F_t|)/2}\right) \tag{9}$$

where N is the number of observations in the training set and H is the number of values being forecasted in the out-of-sample test set.

Hyndman and Koehler (2006) propose the use of the MASE as it is less sensitive to outliers and more easily interpretable than other scaled error measures. The MASE is defined as follows:

$$MASE = \frac{1}{H}\sum_{h=1}^{H}\left(\frac{|X_t - F_t|}{(N-1)^{-1}\sum_{i=2}^{N}|X_i - X_{i-1}|}\right) \tag{10}$$

To assess whether observed differences in MASE and SMAPE are statistically significant, the nonparametric Friedman test (Milton 1940, 1937) which requires no assumption about the distribution of forecast errors, and the post-hoc Nemenyi test (Nemenyi 1962) are employed. The Friedman test evaluates whether there is enough evidence that at least one method is

statistically different from the rest outputting a p-value. The Nemenyi test is based on the minimum distance between methods being compared such that methods are considered to be statistically different if the difference between the ranks of methods compared is larger than this 'critical' distance.

6. Results

6.1. Overall performance

The overall results on bagging and crogging are provided in Table 2. For ease of presentation, results are shown using only MASE as no statistically significant differences are noted between results based on MASE and SMAPE. It gives the results for the NN3 competition data for training, validation and test dataset, showing MASE averaged across all time series, sampling sizes and number of hidden nodes. Ranks based on the Nemenyi test are also shown for all datasets. Methods having no statistically significant difference in performance share the same ranking should in brackets. The method with the lowest average error is highlighted in bold, while those with the best model ranking is highlight in bold and underlined where the average ranking is best.

Table 2. Average errors and and rank of errors on the complete dataset by method. Ranking of errors is based on the Nemenyi Test.

Method	Average errors			Rank across all methods		
	Train	Validation	Test	Train	Validation	Test
Bag_{Moob}	0.80	0.77	1.10	2.81 (2)	2.83 (2)	**1.92 (1)**
Bag_{Foob}	0.97	0.92	1.15	3.67 (3)	3.46 (3)	3.23 (3)
k-fold	0.79	0.76	1.15	1.84 (1)	**1.85 (1)**	2.93 (2)
Monte-Carlo	**0.78**	**0.75**	**1.09**	**1.68 (1)**	1.86 (1)	**1.92 (1)**

The best method in each column is in boldface. The method with the best ranking is underlined. Methods with no statistically significant differences at the 0.05 level of significance share the same model ranking shown in brackets.

Across all series, Monte Carlo crogging, Monte-Carlo, has the lowest average MASE on the test set, as well as the best average ranking although results indicate no statistically significant difference in performance over bagging with moving 'out-of-bag', Bag_{Moob}. Both Monte-Carlo and Bag_{Moob} are however found to be significantly better than *k*-fold and

bagging with fixed ‘out-of-bag’, Bag_{Foob}. *k*-fold crogging outperforms and is statistically better than Bag_{Foob} suggesting that there are benefits in acurracy from having diffferent samples in the validation sets. For all methods out-of-sample results are consistent with those obtained in-sample on the training and validation datasets in terms of average errors, however results on rankings suggest that Monte-Carlo is more consistent. In-sample it ranks statistically better than Bag_{Moob} on training and validation datasets, and out-of-sample is just as good if not better on average error. This suggests that crogging is likely to be more robust to overfitting and issues of method selection particularly when selection is based on in-sample model fit. In the next section we consider performance based on the properties of the tiem series data.

6.2. Data properties

Table 3 shows the results of average MASE and ranking for long and short series across sample size and number of hidden nodes for all four methods. Results show no statistically significant differnces between Monte-Carlo and Bag_{Moob}. While Bag_{Moob} ranks best on the test set with an average rank of 2.04 for long series, Monte-Carlo is best on short series having an average ranking of 1.78. In-sample *k*-fold performs well ranked best on training and validation dataset for short time series. However unlike Monte-Carlo which also performs well on the test set, *k*-fold does not show similar performance out-of-sample indicating evidence of overfitting as previously discussed. The effect of overfitting seems reduced on long series where it is ranked joint first with no statistically significant difference in performance compared to Monte-Carlo and $Bag_{Moob.}$ Bag_{Foob} is always outranked by Monte-Carlo and Bag_{Moob} across all datasets although it does well in terms of average errors on test set for long series. This difference between ranks of errors and average errors suggests that there are several difficult time series, particularly affecting the less robust average of errors.

Table 3. Average errors and and rank of errors on the test dataset by method and time series length. Ranking of errors is based on the Nemenyi Test.

Type	Method	Average errors			Rank across all methods		
		Train	Validation	Test	Train	Validation	Test
Long	Bag_{Moob}	0.99	0.99	1.58	2.88 (2)	2.86 (2)	**2.04 (1)**
	Bag_{Foob}	1.22	1.15	**1.52**	3.80 (3)	3.52 (2)	3.24 (2)
	k-fold	0.98	0.98	1.56	1.96 (1)	2.02 (1)	2.64 (1,2)
	Monte-Carlo	**0.95**	**0.96**	1.55	**1.36 (1)**	**1.60 (1)**	2.08 (1)
Short	Bag_{Moob}	0.60	0.58	**0.69**	2.86 (2)	2.84 (2,3)	**1.82 (1)**
	Bag_{Foob}	0.68	0.66	0.79	3.46 (2)	3.30 (3)	3.18 (2)
	k-fold	**0.58**	**0.56**	0.81	**1.56 (1)**	**1.64 (1)**	3.22 (2)
	Monte-Carlo	0.59	0.57	**0.69**	2.12 (1)	2.22 (1,2)	**1.78 (1)**

The best method in each column is in boldface. The method with the best ranking is underlined. Methods with no statistically significant differences at the 0.05 level of significance share the same model ranking shown in brackets.

The order of rankings is similar when considering seasonal and non-seasonal data as shown in

Table 4. Again no statistically significant difference is noted in the ranking of Monte-Carlo and Bag_{Moob} although Monte-Carlo gives the best ranking on seasonal time seris while on non-seasonal Bag_{Moob} is ranked best. While both methods show no staistically significant differnces on the test set, on the training and validation datset, Monte-Carlo is always statistically better than Bag_{Moob}. This may potentially be explained by the structured nature of the sampling in Monte Carlo crogging, in that while observations are drawn at random, it is done without replacement ensuring that observations only appear in the trainining set once and therefore more distrinct observations can be sampled. In contrast, when Bag_{Moob} is used, observations in the training set may repeat offfering fewer distinct obsevations on which to train. The performance of Monte-Carlo and Bag_{Moob} would suggest that the feature of the randam sampling across both the training and validation set are important; coventional bagging which in contrasts involves random sampling but only of the training set appears to be inferior

Table 4. Average errors and and rank of errors on the test dataset by method and time series type. Ranking of errors is based on the Nemenyi Test.

Type	MASE	Average errors			Rank across all methods		
	Method	Train	Validation	Test	Train	Validation	Test
Non-seasonal	Bag_{Moob}	0.98	0.99	1.56	2.74 (2)	2.76 (2)	**1.96 (1)**
	Bag_{Foob}	1.21	1.15	**1.52**	3.56 (3)	3.40 (2)	3.20 (2)
	k-fold	0.98	0.98	1.56	1.86 (1)	**1.76 (1)**	2.86 (2)
	Monte-Carlo	**0.96**	**0.97**	1.53	**1.84 (1)**	2.08 (1)	1.98 (1)
Seasonal	Bag_{Moob}	0.61	0.58	**0.71**	3.00 (2)	2.94 (2)	1.90 (1)
	Bag_{Foob}	0.69	0.67	0.80	3.70 (3)	3.42 (2)	3.22 (2)
	k-fold	**0.58**	**0.56**	0.80	1.66 (1)	1.90 (1)	3.00 (2)
	Monte-Carlo	0.59	0.57	**0.71**	**1.64 (1)**	**1.74 (1)**	**1.88 (1)**

The best method in each column is in boldface. The method with the best ranking is underlined. Methods with no statistically significant differences at the 0.05 level of significance share the same model ranking shown in brackets.

While the above provides a good overview of the performance of the four methods, it is does not account for the potential difference in forecasting accuracy of each method as sampling size and number of time series observations change, discussed next.

6.3. Sampling size

We investigate how the choice of sampling size and the interaction with time series length affects forecasting accuracy. Recall that for *k*-fold cross-validation the number of training sets is equivalent to the number of folds k, while for Monte Carlo crogging and bagging this is equivalent to the number of random splits and the number of bootstraps respectively. Table 5 shows the results for the average MASE and ranking by time series length and sample size. The most accurate method on average for each sampling size (by row) is highlighted in bold, while the most accurate method across sampling sizes is underlined, for both long and short series. Ranking is done by method for each sample size.

Results of average error indicate that Monte-Carlo and Bag_{Moob} rank best on short series for nearly all sampling sizes. In contrast both methods perform comparatively poorly on long time series and are outperformed by Bag_{Foob} across nearly all sample sizes. Results based on rankings are somewhat different and show Monte-Carlo to be robust across time series length achieving good performance on both long and short time series. In contrast for

Table 5. Average errors and and rank of errors on test dataset by method and sampling size. Ranking of errors is based on the Nemenyi Test.

Type	Size	Average errors				Rank across all methods			
		Monte-Carlo	*k*-fold	Bag_{Moob}	Bag_{Foob}	Monte-Carlo	*k*-fold	Bag_{Moob}	Bag_{Foob}
Long	2	**1.56**	1.57	1.56	**1.56**	**2.15 (1)**	2.46 (1)	2.25 (1)	3.14 (2)
	5	1.59	1.53	1.57	**1.51**	2.36 (1,2)	**2.22 (1)**	2.44 (1,2)	2.98 (2)
	10	1.57	1.59	1.57	**1.44**	**2.29 (1)**	2.42 (1)	2.57 (1)	2.72 (1)
	15	**1.47**	1.54	1.56	1.49	**2.19 (1)**	2.78 (1)	2.37 (1)	2.66 (1)
	20	1.52	1.56	1.56	**1.50**	2.35 (1)	2.62 (1)	**2.33 (1)**	2.70 (1)
	25	1.59	1.54	1.56	**1.52**	2.39 (1)	2.63 (1)	**2.30 (1)**	2.68 (1)
	30	1.51	**1.50**	1.60	1.54	**2.15 (1)**	2.59 (1)	2.42 (1)	2.84 (1)
Short	2	**0.69**	**0.69**	0.70	0.78	**2.02 (1)**	2.10 (1)	2.38 (1)	3.50 (2)
	5	0.70	0.70	**0.69**	0.76	**2.02 (1)**	2.48 (1)	2.32 (1)	3.18 (2)
	10	**0.69**	0.78	**0.69**	0.79	**1.86 (1)**	3.04 (2)	2.06 (1)	3.04 (2)
	15	**0.69**	0.79	**0.69**	0.80	2.03 (1)	3.26 (2)	**2.01 (1)**	2.70 (2)
	20	**0.69**	0.86	**0.69**	0.76	**2.02 (1)**	3.16 (2)	2.06 (1)	2.76 (2)
	25	**0.69**	0.89	**0.69**	0.77	**1.78 (1)**	3.50 (3)	1.94 (1)	2.78 (2)
	30	**0.69**	1.01	**0.69**	0.76	**1.92 (1)**	3.44 (3)	2.00 (1,2)	2.64 (2)

The best method in each row is in boldface. The method with the best ranking is underlined. Methods with no statistically significant differences at the 0.05 level of significance share the same model ranking shown in brackets.

short series *k*-fold is observed to degrade rather quickly as sample size increases. This is because as sample size increases fewer observations are available for the validaiton set. For short series Bag_{Foob} which uses a fixed size validation set also performs poorly. This is explained by observing that while Monte-Carlo and Bag_{Moob} train on the entire learning set, with different observations used either as training or validation datasets which Bag_{Foob} uses a fixed size validation set of the same observations. For long series this sample size performance tradeoff is less noticable.

Results of average ranking however shows no statistically significant differences among any of the methods for long series at nearly all sample sizes suggesting that given a sufficient number of observations, each method is capable of performing similarly. Difference in reported performance using average errors and ranking is due to a few difficult to forecast series. For short series which are much harder to forecast, results of rankings remain consistent with those of average errors, and both Monte-Carlo and Bag_{Moob} outperform *k*-fold and Bag_{Foob} with statistical significance for sample sizes greater than 5. If

we consider average error, then Bag_{Foob} with 10 bootstraps has the lowest error for long time series, while several sizes of Monte-Carlo and Bag_{Moob} record the lowest error on short series.

6.4. Network parameters

Next, we seek to assess the sensitivity of each method to the complexity of the neural network base model. For each combination method we consider the impact of the number of hidden nodes on forecast accuracy. Table 6 summarises the performance of each combination method on the test dataset across different numbers of hidden nodes. The best performing method in each row corresponding to the number of hidden nodes is highlighted in boldface. Results based on rankings across all time series and sample sizes show that Monte-Carlo and Bag_{Moob} are always ranked better than *k*-fold or Bag_{Foob} and statistically better than Bag_{Foob} for all but hidden node size 4. Results are clear in showing that bagging with moving 'out-of-bag' should be preferred to a bagging with fixed 'out-of-bag' approach. In addition it suggests that Monte-Carlo should be preferred to *k*-fold.

Table 6. Average errors and rank of errors on test dataset by method and number of hidden nodes. Ranking of errors by method is based on the Nemenyi Test and done for each number of hidden nodes.

MASE	Average errors				Rank across all methods			
Hidden Nodes	Monte-Carlo	*k*-fold	Bag_{Moob}	Bag_{Foob}	Monte-Carlo	*k*-fold	Bag_{Moob}	Bag_{Foob}
1	1.11	1.14	**1.09**	1.28	2.16 (1)	2.44 (1)	**2.15 (1)**	3.26 (2)
2	1.14	1.28	**1.07**	1.18	2.13 (1)	2.84 (2)	**1.88 (1)**	3.15 (2)
3	**1.06**	1.11	1.08	1.12	**1.80 (1)**	2.98 (2)	2.19 (1)	3.03 (2)
4	**1.09**	1.14	1.15	1.10	**2.21 (1)**	2.85 (2)	2.29 (1)	2.65 (1,2)
5	1.09	1.09	1.10	**1.05**	2.39 (1,2)	2.55 (1,2)	**2.31 (1)**	2.76(2)

The best method in each row is in boldface. The method with the best ranking is underlined. Methods with no statistically significant differences at the 0.05 level of significance share the same model ranking shown in brackets.

Results of average errors suggests that for Bag_{Foob}, Monte-Carlo and *k*-fold forecast accuracy generally improves with the number of hidden nodes, showing reductions of 18%, 2% and 4% respectively from the use of 1 hidden node versus 5 hidden nodes. To further validate our finding, we perform for each method a ranking of the MASE across all time

series based on the number of hidden nodes in the neural network. Results of Nemenyi model ranking and Friedman test shown in Table 7 show no statistically significant differences in accuracy due to number of hidden nodes for Monte-Carlo and similarly for Bag_{Moob}. In contrast *k*-fold and Bag_{Foob} methods perform statistically better with larger networks.

Table 7. Rank of errors on test dataset by method and number of hidden nodes. Ranking of errors by number of hidden nodes is based on the Nemenyi Test and done for each method.

MASE	Mean rank				Model rank			
Hidden Nodes	Monte-Carlo	*k*-fold	Bag_{Moob}	Bag_{Foob}	Monte-Carlo	*k*-fold	Bag_{Moob}	Bag_{Foob}
1	3.20	2.82	3.41	3.68	1	1	2	3
2	**2.69**	3.60	2.94	3.50	1	2	1,2	2, 3
3	3.16	3.23	**2.80**	3.09	1	2	1	2
4	3.13	2.86	3.05	2.51	1	1	1,2	1
5	2.82	**2.50**	**2.80**	**2.21**	1	1	1	1

The best method in each column is in boldface. Methods with no statistically significant differences at the 0.05 level of significance share the same model ranking.

6.5. Select best versus Benchmarks

In this section we select for each time series and for each of the four methods previously evaluated, the best performing combination of sample size and number of hidden nodes. For each time series the sample size and hidden nodes having the producing lowest MASE on the validation set is selected. For each benchmark method, the number of hidden nodes is also selected based on the validation set accuracy. When the Holdout, Noholdout and Bag_{Noob} methods are used, the sample size is set to 50. Results of the average error and ranking of errors for each method aross all time series are presented in Table 8. Considering the average MASE across all time series, we see that the method with the lowest forecast error on the test set is Bag_{Moob} followed by *k*-fold. These results are however not statistically significant as observed by rankings which suggest no statistically significant difference between Monte-Carlo, *k*-fold and Bag_{Moob} when selecting based on in-sample accuracy on the validation set.

Table 8. Average errors and and rank of errors on test dataset by method and number of hidden nodes. Ranking of errors is based on the Nemenyi Test.

MASE	Average errors			Rank across all methods		
Method	Train	Validation	Test	Train	Validation	Test
Bag_{Moob}	0.77	0.63	**0.86**	6.34 (3,4)	5.37 (3,4)	4.73 (1)
Bag_{Foob}	0.80	0.60	0.94	5.42 (2,3)	3.79 (1,2)	4.90 (1,2)
k-fold	0.67	**0.55**	0.89	2.59 (1)	**2.75 (1)**	**4.51 (1)**
Monte-Carlo	0.73	0.60	0.98	4.42 (2)	4.32 (2,3)	4.78 (1)
$Holdout_{avg}$	0.79	0.72	1.05	5.56 (3,4)	7.22 (5)	5.17 (1,2)
$Noholdout_{avg}$	0.77	0.65	1.05	6.02 (3,4)	6.42 (3,4)	5.05 (1,2)
$Holdout_{select}$	0.91	0.65	1.15	6.65 (4)	5.66 (3,4)	5.99 (1,2)
Leave-one-out	**0.63**	0.58	0.93	**2.31 (1)**	3.03 (1)	4.86 (1,2)
Bag_{Noob}	0.79	0.68	0.94	5.68 (3,4)	6.04 (3,4)	5.02 (1,2)

The best method in each column is in boldface. The method with the best ranking is underlined. Methods with no statistically significant differences at the 0.05 level of significance share the same model ranking shown in brackets.

Leave-one-out and *k*-fold perform are stistically better than all other methods on the training and validation datasets which is not surprising given the tendency to overfitt. These resullts suggest that use of the valiation set for individual time series parameter (sample size and hidden nodes) selection is not particularly effective at discriminating among the methods.

6.6. Relative Ranking on results of NN3 competition

Table 9 reports the results obtained by the first eight participants of the NN3 competition and the benchmark model of the competition (AutomatANN). To these we compare Monte-Carlo crogging, *k*-fold crogging, both forms of bagging, as well as the set of benchmark approaches previously discussed. These are highlighted in boldface with the best method in each column underlined. For each method the sample size and number of hidden nodes is determined using in-sample mean forecast error on the validation set. To remain consistent with the reporting format of the competition, we report rankings using SMAPE and MASE. While these rankings provide evidence of each method's ability to accurately forecast relative to competition benchmark, it is not possible to performance tests of statistical significance having no access to the competition forecasts. Among all methods Bag_{Moob} and Monte-Carlo ranked 3rd and 5th overall according to SMAPE, and 1st and 2nd based on MASE across all time series. Monte-Carlo and Bag_{Moob} method outperforms well known statistical benchmarks including damped exponential smoothing (DES) and the simple

average of damped, single and holt exponential smoothing (Comb-S-H-D) shown to work well in other competitions (Makridakis and Hibon 2000). Most encouraging is the improvement in accruacy over model selection with Monte-Carlo and Bag_{Moob} showing a 19% and 21% improvement over $Holdout_{Select}$ according to SMAPE, and 15% and 16% according to MASE.

Table 9. Average errors and ranks of errors across all time series of the NN3 competition.

		Average errors		Ranking all methods		Ranking NN/CI	
		SMAPE	MASE	SMAPE	MASE	SMAPE	MASE
B09	Wildi	14.84	1.13	1	3	–	–
B07	Theta	14.89	1.13	2	3	–	–
**	**Bag_{Moob}**	**15.13**	**1.11**	**3**	**1**	**1**	**1**
C27	Illies	15.18	1.25	3	15	2	8
B03	ForecastPro	15.44	1.17	4	7	–	–
**	**Monte-Carlo**	**15.47**	**1.12**	**5**	**2**	**3**	**2**
B16	DES	15.9	1.17	6	7	–	–
B17	Comb S-H-D	15.93	1.21	7	13	–	–
**	**$Noholdout_{avg}$**	**15.94**	**1.19**	**8**	**10**	**4**	**5**
B05	Autobox	15.95	1.18	9	9	–	–
**	**Bag_{Foob}**	**15.95**	**1.17**	**10**	**6**	**5**	**4**
**	**Bag_{Noob}**	**15.99**	**1.13**	**11**	**5**	**6**	**3**
C03	Flores	16.31	1.20	12	12	7	6
**	***k*-fold**	**16.34**	**1.20**	**13**	**11**	**8**	**6**
B00	AutomatANN	16.81	1.21	14	13	9	7
**	**$Holdout_{avg}$**	**17.12**	**1.25**	**15**	**16**	**10**	**8**
**	**Leave-one-out**	**17.87**	**1.33**	**16**	**18**	**11**	**10**
**	**$Holdout_{select}$**	**19.10**	**1.32**	**17**	**17**	**12**	**9**

Methods implemented in this study are highlighted in bold. The best method in each column is underlined.

Among the computational intelligence (NN/CI) methods, Bag_{Moob} and Monte-Carlo ranked 1st and 3rd respectively according to SMAPE, and 1st and 2nd according to MASE. This reflects rather good performance by the proposed Monte-Carlo combination method and Bag_{Moob}. Bag_{Foob} ranks slight better than Neural network model averaging with no validation set ($NoHoldout_{Avg}$), while both the $Holdout_{Select}$ and leave-one-out are ranked worst than the competition benchmark method AutomatANN both in terms of SMAPE and MASE.While we do not show these results to be statistically significant relative to competition methods, they are not surprising given evidence already presenting showing Monte-Carlo and

BagMoob to perform better on average than the other methods evaluated in this study. Also this needs to be verified, results using SMAPE and the more robust MASE suggest that both methods outperform other neural network based methods of the competition. This includes the more complex approaches of Illies, Jäger, Kosuchinas, Rincon, Sakenas and Vaskevcius (C27) which is based on a combination of time series clustering, decomposition and the use of Echo State Networks (ESN), a type of recurrent NN, and the method of Flores et. al. (C03), which combines a self adaptive genetic algorithm to determine the terms of a seasonal ARIMA $(p,d,q)(P,D,Q)$ model.

7. Summary and conclusions

We have presented the first application and evaluation of Monte Carlo cross-validation as a method for producing combinations of univariate time series forecasts within a general framework of cross-validation aggregation (crogging). This general framework for combining autoregressive forecasts draws inspiration from bagging. Bagging which is based on bootstrap resampling is used for forecast combination as a relatively easy way to improve accuracy of an existing model and has grown in popularity over the last two decades. Several cross-validation sampling schemes are evaluated for combining forecasts including 2-fold, 10-fold, leave-one-out and Monte Carlo cross-validation.

Beyond Monte Carlo cross-validation, this study provides evidence that crogging which is relatively simple to implement, turns out to be even more effective at reducing the variance of the final forecast, and improving accuracy, in comparison to bagging with a fixed validation set, and neural network model averaging. Where bagging is implemented using a validation set, the 'out-of-bag' approach, then bagging performs just as good as Monte Carlo crogging. This is first shown theoretically through a decomposition of the mean squared error into its bias and variance components, which are both estimated through Monte and Bag_{Moob} are both effective at reducing variance, and in some cases, the bias of the combined forecast.

Secondly using competition data consisting of monthly real business time series, it was found that Monte-Carlo crogging and Bag_{Moob}, outperformed *k*-fold crogging and other benchmark methods, making Monte-Carlo a valid and attractive alternative for time series forecasting with autoregressive models.

With the exception of Bag_{Foob} the number of hidden nodes used was found to have no statistical impact on forecast accuracy. However for short series and large combination sizes, *k*-fold and Bag_{Foob} were outperformed by both Monte-Carlo crogging and Bag_{Moob}. In contrast for long time series little statistical difference was noted among the four principal methods evaluated. Our results show evidence of better in-sample performance on the validation set by Monte-Carlo versus $Bag_{Moob,}$ however this resulted in no improvement in terms of model selection of sample size and hidden node parameters even when using the validation set. This can be explained by the relative invariace of both methods to both paramaters. While Monte-Carlo crogging and Bag_{Moob} are most effective at larger sampling sizes, *k*-fold crogging performed poorly; attributable to the trade-off between the number of folds, and the number of time series observations available for training. As the number of folds increases, and likewise the number of forecasts combined, the size of the validation dataset decreaces. This suggests that with fewer observations available for early stopping, model estimation becomes poorer. This appears to be the biggest advantage that the Monte-Carlo and bagging with 'out-of-bag' methods afford, in that they decouples the number of samples available for forecast model estimation from the size of the training/validation datasets.

While this study has shown that crogging can be used to improve forecasting accuracy when applied to neural networks, further evidence is required using other forecasting methods e.g. regression to which crogging can be easily applied. Additionally, an obvious next step would be to investigate the application of crogging beyond autoregressive models to

moving average processes as proposed by Bergmeir and Hyndman (2014) for bagging exponential smoothing methods.

References

Aksu, C., and S. I. Gunter. 1992. An empirical analysis of the accuracy of SA, OLS, ERLS and NRLS combination forecasts. *International Journal of Forecasting* 8 (1):27-43.

Altman, NS. 1990. Kernel smoothing of data with correlated errors. *Journal of the American Statistical Association* 85 (411):749-759.

Arlot, Sylvain, and Alain Celisse. 2010. A survey of cross-validation procedures for model selection. *Statistics Surveys* 4 (0):40-79.

Assaad, M., R. Bone, and H. Cardot. 2008. A new boosting algorithm for improved time-series forecasting with recurrent neural networks. *Information Fusion* 9 (1):41-55.

Bates, J. M., and C. W. J. Granger. 1969. Combination of Forecasts. *Operational Research Quarterly* 20 (4):451-&.

Ben Taieb, Souhaib, and Rob Hyndman. 2014. Boosting multi-step autoregressive forecasts. Paper read at Proceedings of The 31st International Conference on Machine Learning.

Bengio, Yoshua, and Yves Grandvalet. 2004. No unbiased estimator of the variance of k-fold cross-validation. *The Journal of Machine Learning Research* 5:1089-1105.

Berardi, V. L., and G. P. Zhang. 2003. An empirical investigation of bias and variance in time series forecasting: Modeling considerations and error evaluation. *Ieee Transactions on Neural Networks* 14 (3):668-679.

Bergmeir, Christoph, and Rob J Hyndman. 2014. Bagging Exponential Smoothing Methods using STL Decomposition and Box-Cox Transformation. Monash University, Department of Econometrics and Business Statistics.

Birgé, Lucien, and Pascal Massart. 2007. Minimal penalties for Gaussian model selection. *Probability Theory and Related Fields* 138 (1-2):33-73.

Breiman, L. 1984. *Classification and regression trees*: Wadsworth International Group.

Breiman, L. 1996a. Bagging predictors. *Machine Learning* 24 (2):123-140.

Breiman, L. 1996b. *Out-of-bag estimation*. Technical report, Statistics Department, University of California Berkeley, Berkeley CA 94708, 1996b. 33, 34.

Breiman, L. 2001. Random forests. *Machine learning*, *45*(1): 5-32.

Brenning, A., J. Andrey, and B. Mills. 2011. Indirect modeling of hourly meteorological time series for winter road maintenance. *Environmetrics* 22 (3):398-408.

Brown, Gavin, Jeremy Wyatt, Rachel Harris, and Xin Yao. 2005. Diversity creation methods: a survey and categorisation. *Information Fusion* 6 (1):5-20.

Burman, Prabir. 1989. A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods. *Biometrika* 76 (3):503-514.

Burman, Prabir, Edmond Chow, and Deborah Nolan. 1994. A cross-validatory method for dependent data. *Biometrika* 81 (2):351-358.

Burman, Prabir, and Deborah Nolan. 1992. Data-dependent estimation of prediction functions. *Journal of Time Series Analysis* 13 (3):189-207.

Chen, Z., and Y. Yang. 2004. Assessing forecast accuracy measures. In *Working Paper*.

Clarida, R. H., L. Sarno, M. P. Taylor, and G. Valente. 2003. The out-of-sample success of term structure models as exchange rate predictors: a step beyond. *Journal of International Economics* 60 (1):61-83.

Clemen, R. T., and R. L. Winkler. 1986. Combining economic forecasts. *Journal of Business & Economic Statistics* 4 (1):39-46.

Clements, Michael P., and David F. Hendry. 2007. An Overview of Economic Forecasting. In *A Companion to Economic Forecasting*, edited by D. F. H. Michael P. Clements.

Cleveland, Robert B, William S Cleveland, Jean E McRae, and Irma Terpenning. 1990. STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics* 6 (1):3-73.

Crone, Sven F., Michèle Hibon, and Konstantinos Nikolopoulos. 2011. Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction. *International Journal of Forecasting* 27 (3):635-660.

de Menezes, L. M., D. W. Bunn, and J. W. Taylor. 2000. Review of guidelines for the use of combined forecasts. *European Journal of Operational Research* 120 (1):190-204.

Deng, Y. F., X. Jin, Y. X. Zhong, and Ieee. 2005. *Ensemble SVR for prediction of time series, Proceedings of 2005 International Conference on Machine Learning and Cybernetics, Vols 1-9*. New York: Ieee.

Dietterich, Thomas G. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation* 10 (7):1895-1923.

Dietterich, ThomasG. 2000. Ensemble Methods in Machine Learning. In *Multiple Classifier Systems*: Springer Berlin Heidelberg.

Donate, J. P., P. Cortez, G. G. Sanchez, and A. S. de Miguel. 2013. Time series forecasting using a weighted cross-validation evolutionary artificial neural network ensemble. *Neurocomputing* 109:27-32.

Efron. 1983. Estimating the Error Rate of a Prediction Rule - Improvement on Cross-Validation. *Journal of the American Statistical Association* 78 (382):316-331.

Efron, Bradley. 1979. 1977 Rietz Lecture - Bootstrap Methods - Another Look at the Jackknife. *Annals of Statistics* 7 (1):1-26.

Efron, Bradley, and Robert Tibshirani. 1986. Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. *Statistical Science*:54-75.

Efron, Bradley, and Robert Tibshirani. 1993. *An Introduction to the bootstrap*. London: Chapman and Hall.

Geman, S., E. Bienenstock, and R. Doursat. 1992. Neural networks and the bias variance dilemma. *Neural Computation* 4 (1):1-58.

Hagan, M.T., H.B. Demuth, and M.H. Beale. 1996. *Neural Network Design*: Pws Pub.

Hansen, Bruce E., and Jeffrey S. Racine. 2012. Jackknife model averaging. *Journal of Econometrics* 167 (1):38-46.

Hansen, L. K., and P. Salamon. 1990. Neural Network Ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12 (10):993-1001.

Hardle, Wolfgang, and James Stephen Marron. 1985. Optimal bandwidth selection in nonparametric regression function estimation. *The Annals of Statistics*:1465-1481.

Hart, Jeffrey D. 1991. Kernel regression estimation with time series errors. *Journal of the Royal Statistical Society. Series B (Methodological)*:173-187.

Hart, Jeffrey D, and Thomas E Wehrly. 1986. Kernel regression estimation using repeated measurements data. *Journal of the American Statistical Association* 81 (396):1080-1088.

Hillebrand, E., and M. C. Medeiros. 2010. The Benefits of Bagging for Forecast Models of Realized Volatility. *Econometric Reviews* 29 (5-6):571-593.

Hornik, K. 1991. Approximation capabilities of multilayer feedforward networks. *Neural Networks* 4 (2):251-257.

Hornik, K., M. Stinchcombe, and H. White. 1989. Multilayer Feedforward Networks Are Universal Approximators. *Neural Networks* 2 (5):359-366.

Hu, M. Y., G. Q. Zhang, C. Z. Jiang, and B. E. Patuwo. 1999. A cross-validation analysis of neural network out-of-sample performance in exchange rate forecasting. *Decision Sciences* 30 (1):197-216.

Hu, Michael Y., Guoqiang Zhang, Christine X. Jiang, and B. Eddy Patuwo. 1999. A Cross-Validation Analysis of Neural Network Out-of-Sample Performance in Exchange Rate Forecasting. *Decision Sciences* 30 (1):197-216.

Hyndman, Rob J., and Anne B. Koehler. 2006. Another look at measures of forecast accuracy. *International Journal of Forecasting* 22 (4):679-688.

Inoue, A., and L. Kilian. 2008. How useful is bagging in forecasting economic time series? A case study of US consumer price inflation. *Journal of the American Statistical Association* 103 (482):511-522.

Jose, V. R. R., and R. L. Winkler. 2008. Simple robust averages of forecasts: Some empirical results. *International Journal of Forecasting* 24 (1):163-169.

Kohavi, Ron. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2*. Montreal, Quebec, Canada: Morgan Kaufmann Publishers Inc.

Kourentzes, Nikolaos, Devon K. Barrow, and Sven F. Crone. 2014. Neural network ensemble operators for time series forecasting. *Expert Systems with Applications* 41 (9):4235-4244.

Krogh, Anders, and Jesper Vedelsby. 1995. Neural network ensembles, cross validation, and active learning. *Advances in neural information processing systems*:231-238.

Kunsch, H. R. 1989. The Jackknife and the Bootstrap for General Stationary Observations. *Annals of Statistics* 17 (3):1217-1241.

Macdonald, R., and I. W. Marsh. 1994. Combining exchange-rate forecasts - what is the optimal consensus measure. *Journal of Forecasting* 13 (3):313-332.

Makridakis, S., A. Andersen, R. Carbone, R. Fildes, M. Hibon, R. Lewandowski, J. Newton, E. Parzen, and R. Winkler. 1982. The accuracy of extrapolation (time series) methods: Results of a forecasting competition. *Journal of Forecasting* 1 (2):111-153.

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

Michaelsen, J. 1987. Cross-Validation in Statistical Climate Forecast Models. *Journal of Climate and Applied Meteorology* 26 (11):1589-1600.

Naftaly, U., N. Intrator, and D. Horn. 1997. Optimal ensemble averaging of neural networks. *Network-Computation in Neural Systems* 8 (3):283-296.

Opsomer, Jean, Yuedong Wang, and Yuhong Yang. 2001. Nonparametric regression with correlated errors. *Statistical Science*:134-153.

Perrone, Michael P., and Leon N. Cooper, eds. 1992. *When networks disagree: ensemble methods for hybrid neural networks*. Edited by R. J. Mammone, *Neural Networks for Speech and Image processing*: Chapman-Hall.

Picard, Richard R, and R Dennis Cook. 1984. Cross-validation of regression models. *Journal of the American Statistical Association* 79 (387):575-583.

Shao, J. 1997. An asymptotic theory for linear model selection. *Statistica Sinica* 7 (2):221-242.

Shao, L. 1993. Linear model selection via cross-validation. *Journal of the American Statistical Association* 88 (422):486-494.

Sorić, Petar, and Ivana Lolić. 2013. A note on forecasting euro area inflation: leave-h-out cross validation combination as an alternative to model selection. *Central European Journal of Operations Research*:1-10.

Stock, James H., and Mark W. Watson. 2004. Combination forecasts of output growth in a seven-country data set. *Journal of Forecasting* 23 (6):405-430.

Stone, M. 1974. Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society. Series B (Methodological)* 36 (2):111-147.

Stone, M. 1977. An Asymptotic Equivalence of Choice of Model by Cross-Validation and Akaike's Criterion. *Journal of the Royal Statistical Society. Series B (Methodological)* 39 (1):44-47.

Taieb, S. B., Bontempi, G., Atiya, A. F., & Sorjamaa, A. 2012. A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *Expert systems with applications*, *39*(8): 7067-7083.

Taieb, S. B., & Atiya, A. F. 2015. A Bias and Variance Analysis for Multistep-Ahead Time Series Forecasting. *IEEE transactions on neural networks and learning systems*.

Tashman, J. 2000. Out-of-sample tests of forecasting accuracy: an analysis and review. *International Journal of Forecasting* 16 (4):437-450.

Terasvirta, T., and H. M. Anderson. 1992. Characterizing nonlinearities in business cycles using smooth transition autoregressive models. *Journal of Applied Econometrics* 7 (S1):S119-S136.

Timmermann, A. 2006. Forecast combinations. In *Handbook Of Economic Forecasting*, edited by G. Elliott, C. W. J. Granger and A. Timmermann. Amsterdam Elsevier.

Watson, James H. Stock and Mark W. 2005. An empirical comparison of methods for forecasting using many predictors. Princeton University.

Wolff, C. C. P. 1987. Time-Varying Parameters and the out-of-Sample Forecasting Performance of Structural Exchange-Rate Models. *Journal of Business & Economic Statistics* 5 (1):87-97.

Zhang, G. P., and V. L. Berardi. 2001. Time series forecasting with neural network ensembles: an application for exchange rate prediction. *Journal of the Operational Research Society* 52 (6):652-664.

Zhang, G. Q., B. E. Patuwo, and M. Y. Hu. 1998. Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting* 14 (1):35-62.

Zhang, Ping. 1993. Model selection via multifold cross validation. *The Annals of Statistics*:299-313.

Zhou, Zhi-Hua, Jianxin Wu, and Wei Tang. 2002. Ensembling neural networks: Many could be better than all. *Artificial Intelligence* 137 (1–2):239-263