An Empirical Comparison of Machine Learning Models for Time Series Forecasting

Article *in* Econometric Reviews · August 2010

DOI: 10.1080/07474938.2010.481556 · Source: RePEc

CITATIONS
176

READS
11,585

4 authors:

Nesreen K. Ahmed
Intel
77 PUBLICATIONS 1,373 CITATIONS
SEE PROFILE

Amir Atiya
Cairo University
206 PUBLICATIONS 5,235 CITATIONS
SEE PROFILE

Neamat El Gayar
Cairo University
46 PUBLICATIONS 532 CITATIONS
SEE PROFILE

Hisham El-Shishiny
IBM
72 PUBLICATIONS 760 CITATIONS
SEE PROFILE

An Empirical Comparison of Machine Learning Models for Time Series Forecasting

Nesreen K. Ahmed
Faculty of Computers and Information
Cairo University, Giza, Egypt
n.kamel@gmail.com

Amir F. Atiya
Dept Computer Engineering
Cairo University, Giza, Egypt
amir@alumni.caltech.edu

Neamat El Gayar
Faculty of Computers and Information
Cairo University, Giza, Egypt
hmg@link.net

Hisham El-Shishiny
IBM Center for Advanced Studies in Cairo
IBM Cairo Technology Development Center
Giza, Egypt
shishiny@eg.ibm.com

Abstract

In this work we present a large scale comparison study for the major machine learning models for time series forecasting. Specifically, we apply the models on the monthly M3 time series competition data (around a thousand time series). There have been very few, if any, large scale comparison studies for machine learning models for the regression or the time series forecasting problems, so we hope this study would fill this gap. The models considered are multilayer perceptron, Bayesian neural networks, radial basis functions, generalized regression neural networks (also called kernel regression), K-nearest neighbor regression, CART regression trees, support vector regression, and Gaussian processes. The study reveals significant differences between the different methods. The best two methods turned out to be the multilayer perceptron and the Gaussian process regression. In addition to model comparisons, we have tested different preprocessing methods and have shown that they have different impacts on the performance.

1 Introduction

Machine learning models have established themselves in the last decade as serious contenders to classical statistical models in the area of forecasting. Research started in the eighties with the development of the neural network model. Subsequently, research extended the concept to other models, such as support vector machines, decision trees, and others, that are collectively called machine learning models (Hastie et al [22] and Alpaydin [2]). Some of these models trace their origins from the early statistics literature (see Hastie et al [22] for a discussion). There have been impressive advances in this field in the past few years, both in the theoretical understanding of the models and in the amount and variations of the models developed. In addition to model development and analysis there has to be a parallel effort of empirically validating the numerous existing models and comparing their performance. This would be of immense value to the practitioner, as it would narrow down his possible choices, and give him insight into the strong and weak points of the available models. In addition, this would also help to channel the research effort into the more promising tracks.

Large scale studies for comparing machine learning models have focused almost exclusively on the classification domain (Caruana and Niculescu-Mizil [10]). We have found no extensive study for regression problems. There have been numerous comparison studies that compare neural networks with traditional linear techniques for forecasting and other econometric problems. For example Sharda and Patil [47] have compared neural networks to ARIMA on the M-competition time series data. Hill et al [23] have also considered the M-competition data and have compared between

neural networks and traditional methods. Swanson and White [49] have applied their comparison on nine US macroeconomic series. Alon et al [1] have analyzed neural networks versus other traditional methods such as Winters exponential smoothing, Box-Jenkins ARIMA and multivariate regression, on retail sales data. Callen et al [8] have compared between neural networks and linear models for 296 quarterly earnings time series. Zhang et al [54] have considered a similar problem, but have added some additional accounting variables. Terasvirta et al [50] consider neural networks, smooth transition autoregressions, and linear models for 47 macroeconomic series. The outcome of all these studies has been somewhat mixed, but overall neural networks tended more to outperform classical linear techniques.

The problem is that these studies are confined to only the basic neural network model, and do not extend to the novel machine learning models. This, in fact, is the subject of this study. We conduct a large scale comparison study of a variety of machine learning models applied to the M3 competition data [29]. The M3 competition is the latest in a sequel of M forecasting competitions, organized by Makridakis and Hibon [34]. It consists of 3003 business-type time series, covering the types of micro, industry, finance, demographic, and others. There are yearly, quarterly, and monthly time series. In this study we consider only the monthly time series, as quarterly and yearly time series were generally too short. The M3 data has become an important benchmark for testing and comparing forecasting models. Having that many diverse time series gives confidence into comparison results. The training period for the considered time series ranges in length from 63 to 108 data points, so the comparison study will generally apply to the kinds of monthly business time series of this length range.

In this study we have compared the following models: multilayer perceptron, Bayesian neural networks, radial basis functions, generalized regression neural networks (also called kernel regression), K-nearest neighbor regression, CART regression trees, support vector regression, and Gaussian processes. Another focus of the study is to examine preprocessing methods, used in conjunction with the machine learning forecasting models. Preprocessing steps such as deseasonalization, taking the log transformation, and detrending have been studied before in the literature (Zhang et al [54], Ghysels et al [19], Zhang and Qi [55], Miller and Williams [39], and Balkin and Ord [4]). These methods will therefore not be the focus of the comparison study. The goal in this study is to compare other preprocessing strategies that are frequently used with machine learning models, such as differencing the time series, and taking moving averages.

To summarize the findings of the paper the overall ranking of the models from best to worse turned out to be about: multilayer perceptron, Gaussian processes, then Bayesian neural networks and support vector regression almost similar, then generalized regression neural networks and K-nearest neighbor regression about tied, CART

regression trees, then in the last spot radial basis functions. This ranking is generally broad-based and does not change much with different categories or features of the time series. The preprocessing method can also have a large impact on performance.

The paper is organized as follows. Next section we present a brief description of the compared models. Section 3 describes the tested preprocessing methods. Section 4 deals with the issue of how to set the model parameters. In Section 5 we describe the details of the simulations set-up. Following that, in Section 6 we present the comparison results. Section 7 gives some comments on the results, and finally Section 8 presents the conclusion of the paper.

2 Models

For each considered model there are myriads of variations proposed in the literature, and it would be a hopeless task to consider all existing varieties. Our strategy was therefore to consider the basic version of each model (without the additions, or the modifications proposed by so many researchers). The rationale is that most users will more likely opt (at least in their first try) to consider the basic form. For example for the K-nearest neighbor model (KNN), we used the basic form where the target outputs of the K-neighbors are equally weighted. There are many other variations, such as distance weighted KNN, flexible metric KNN, non-Euclidean distance-based, and so on, but we did not consider these variants.

The reason for generally selecting these considered eight models is that they are some of the most commonly used models. Below is a short description of the models considered.

2.1 Multilayer Perceptron (MLP)

The multilayer perceptron (often simply called neural network) is perhaps the most popular network architecture in use today both for classification and regression (Bishop [5]). The MLP is given as follows:

$$\hat{y} = v_0 + \sum_{j=1}^{NH} v_j g(w_j^T x') \tag{1}$$

where x' is the input vector x, augmented with 1, i.e. $x' = (1, x^T)^T$, w_j is the weight vector for j^{th} hidden node, $v_0, v_1, \ldots, v_{NH}$ are the weights for the output node, and $\hat{y}$ is the network output. The function g represents the hidden node output, and it is given in terms of a squashing function, for example (and that is what we used) the logistic function: $g(u) = 1/(1 + \exp(-u))$. A related model in the econometrics literature is

the smooth transition autoregression model that is also based on constructing linear functions and logistic function transitions [37], [51].

The MLP is a heavily parametrized model, and by selecting the number of hidden nodes NH we can control the complexity of the model. The breakthrough that lent credence to the capability of neural networks is the universal approximation property [14], [17], [28], [24]. Under certain mild conditions on the hidden node functions g, any given continuous function on a compact set can be approximated as close as arbitrarily given using a network with a finite number of hidden nodes. While this is a reassuring result, it is critical to avoid overparametrization, especially in forecasting applications which typically have a limited amount of highly noisy data. Model selection (via selecting the number of hidden nodes) has therefore attracted much interest in the neural networks literature (see Medeiros et al [38] and Anders and Korn [3] for example). We use a K-fold validation procedure to select the number of hidden nodes, but the details of this (and any parameter selection step) will be discussed next section.

To obtain the weights the mean square error is defined, and the weights are optimized using gradient techniques. The most well-known method, based on the steepest descent concept, is the backpropagation algorithm. A second order optimization method called Levenberg Marquardt is generally known to be more efficient than the basic backpropagation algorithm, and this is the one we used in our implementation (we use the Matlab function `trainlm`).

2.2 Bayesian Neural Network (BNN)

A Bayesian neural network (BNN) is a neural network designed based on a Bayesian probabilistic formulation (Mackay [30], [31]). As such BNN's are related to the classical statistics concept of Bayesian parameter estimation, and are also related to the concept of regularization such as in ridge regression. BNN's have enjoyed wide applicability in many areas such as economics/finance [18] and engineering [5]. The idea of BNN is to treat the network parameters or weights as random variables, obeying some a priori distribution. This distribution is designed so as to favor low complexity models, i.e. models producing smooth fits. Once data is observed the posterior distribution of the weights is evaluated and the network prediction can be computed. The predictions will then reflect both the smoothness aspect imposed through the prior and fitness accuracy aspect imposed by the observed data. A closely related concept is the regularization aspect, whereby the following objective function is constructed and minimized:

$$J = \alpha E_D + (1 - \alpha) E_W \tag{2}$$

where E_D is the sum of the square errors in the network outputs, E_W is the sum of the squares of the network parameters (i.e. weights), and α is the regularization parameter.

For the Bayesian approach, the typical choice of the prior is the following normal density that puts more weight onto smaller network parameter values:

$$p(w) = \left(\frac{1-\alpha}{\pi}\right)^{\frac{L}{2}} e^{-(1-\alpha)E_W} \tag{3}$$

where L denotes the number of parameters (weights). The posterior is then given by:

$$p(w|D,\alpha) = \frac{p(D|w,\alpha)p(w|\alpha)}{p(D|\alpha)} \tag{4}$$

where D represents the observed data. Assuming normaly distributed errors, the probability density of the data given the parameters can be evaluated as

$$p(D|w,\alpha) = \left(\frac{\alpha}{\pi}\right)^{\frac{M}{2}} e^{-\alpha E_D} \tag{5}$$

where M is the number of training data points. By substituting the expressions for the densities in (3) and (5) into (4), we get:

$$p(w|D,\alpha) = c\,\exp(-J) \tag{6}$$

where c is some normalizing constant. The regularization constant α is also determined using Bayesian concepts, from

$$p(\alpha|D) = \frac{p(D|\alpha)p(\alpha)}{p(D)} \tag{7}$$

Both expressions (6) and (7) should be maximized to obtain the optimal weights and α parameter, respectively. The term $p(D|\alpha)$ in (7) is obtained by a quadratic approximation of J in terms of the weights and then integrating out the weights. We used the Matlab version version `'trainbr'` for BNN (applied to a multilayer perceptron architecture). This routine is based on the algorithm proposed by Foresee and Hagan [16]. This algorithm utilizes the Hessian that is obtained any way in the Levenberg-Marquardt optimization algorithm in approximating (7).

2.3 Radial Basis Function Neural Network (RBF)

The radial basis function network is similar in architecture to the multilayer network except that the nodes have a localized activation function (Powell [44], and Moody

and Darken [40]). Most commonly, node functions are chosen as Gaussian functions, with the width of the Gaussian function controlling the smoothness of the fitted function. The outputs of the nodes are combined linearly to give the final network output. Specifically, the output is given by

$$y = \sum_{j=1}^{NB} w_j e^{-\frac{\|x-c_j\|^2}{\beta^2}} \tag{8}$$

where w_j, β and c_j denote the combining weight, the width of the node function, and the center of the node function for unit j. Because of the localized nature of the node functions, other simpler algorithms have been developed for training radial basis networks. The algorithm we used is the Matlab function (`newrb`). It is based on starting with a blank network, and sequentially adding nodes until an acceptable error in the training set is achieved. Specifically, we add a node centered around the training pattern giving the maximum error. Then we recompute all the output layer weights using the least squares formula. We continue this way until the error limit is reached or the number of nodes reaches a maximum predetermined value. While there are a variety of other RBF training algorithms, we opted for this one due to its availability in the Matlab suite and hence it will be more likely selected by the user interested in RBF's.

2.4 Generalized Regression Neural Network (GRNN)

Nadaraya and Watson developed this model (Nadaraya [43] and Watson [52]). It is commonly called the Nadaraya-Watson estimator or the kernel regression estimator. In the machine learning community the name generalized regression neural network (or GRNN) is typically used. We will use this latter term. The GRNN model is a nonparametric model where the prediction for a given data point x is given by the average of the target outputs of the training data points in the vicinity of the given point x [21]. The local average is constructed by weighting the points according to their distance from x, using some kernel function. The estimation is just the weighted sum of the observed responses (or target outputs) given by

$$\hat{y} = \sum_{m=1}^{M} w_m y_m \tag{9}$$

where the weights w_m are given by

$$w_m = \frac{\mathcal{K}(\frac{\|x-x_m\|}{h})}{\sum_{m'=1}^{M} \mathcal{K}(\frac{\|x-x_{m'}\|}{h})} \tag{10}$$

where y_m is the target output for training data point x_m, and $\mathcal{K}$ is the kernel function. We used the typical Gaussian kernel $\mathcal{K}(u) = e^{-u^2/2}/\sqrt{2\pi}$. The parameter h, called the bandwidth, is an important parameter as it determines the smoothness of the fit, since increasing it or decreasing it will control the size of the smoothing region.

2.5 K Nearest Neighbor Regression (KNN)

The K nearest neighbor regression method (KNN) is a nonparametric method that bases its prediction on the target outputs of the K nearest neighbors of the given query point (see Hastie et al [22]). Specifically, given a data point we compute the Euclidean distance between that point and all points in the training set. We then pick the closest K training data points and set the prediction as the average of the target output values for these K points. Quantitatively speaking, let $\mathcal{J}(x)$ be the set of K nearest neighbors of point x. Then the prediction is given by:

$$\hat{y} = \frac{1}{K} \sum_{m \in \mathcal{J}(x)} y_m \tag{11}$$

where again y_m is target output for training data point x_m.

Naturally K is a key parameter in this method, and has to be selected with care. A large K will lead to a smoother fit, and therefore a lower variance, of course at the expense of a higher bias, and vice versa for a small K.

2.6 Classification & Regression Trees (CART)

CART is a classification or regression model that is based on a hierarchical tree-like partition of the input space [7]. Specifically, the input space is divided into local regions identified in a sequence of recursive splits. The tree consists of internal decision nodes and terminal leaves. Given a test data point, a sequence of tests along the decision nodes starting from the root node will determine the path along the tree till reaching a leaf node. At the leaf node a prediction is made according to the local model associated with that node.

To construct a tree using the training set, we start at the root node. We select the variable (and its split threshold) whose splitting will lead to the largest reduction in mean square error. We continue these splits recursively, until the mean square error reaches an acceptable threshold. A typical practice is to perform some kind of pruning for the tree, once designed. This will eliminate ineffective nodes and to keep in check model complexity. To implement CART, we used the Matlab function (`treefit`).

2.7 Support Vector Regression (SVR)

Support vector regression (Scholkopf and Smola [46], [48]) is a successful method based on using a high-dimensional feature space (formed by transforming original variables), and penalizing the ensuing complexity using a penalty term added to the error function. Consider first for illustration a linear model. Then, the prediction is given by

$$f(x) = w^T x + b \tag{12}$$

where w is the weight vector, b is the bias and x is the input vector. Let x_m and y_m denote respectively the m^{th} training input vector and target output, $m = 1, \ldots, M$. The error function is given by

$$J = \frac{1}{2}\|w\|^2 + C\sum_{m=1}^{M} |y_m - f(x_m)|_\epsilon \tag{13}$$

The first term in the error function is a term that penalizes model complexity. The second term is the ϵ-insensitive loss function, defined as $|y_m - f(x_m)|_\epsilon = \max\{0, |y_m - f(x_m)| - \epsilon\}$. It does not penalize errors below ϵ, allowing it some wiggle room for the parameters to move to reduce model complexity. It can be shown that the solution that minimizes the error function is given by

$$f(x) = \sum_{m=1}^{M} (\alpha_m^* - \alpha_m) x_m^T x + b \tag{14}$$

where α_m and α_m^* are Lagrange multipliers. The training vectors giving non-zero Lagrange multipliers are called *support vectors*, and this is a key concept in SVR theory. Non-support vectors do not contribute directly to the solution, and the number of support vectors is some measure of model complexity (see Cherkassky and Ma [13], and Chalimourda et al [12]). This model is extended to the nonlinear case through the concept of *kernel* $\mathcal{K}$, giving a solution:

$$f(x) = \sum_{m=1}^{M} (\alpha_m^* - \alpha_m) \mathcal{K}(x_m^T x) + b \tag{15}$$

A common kernel is the Gaussian kernel. Assume its width is σ_K (the standard deviation of the Gaussian function). In our simulations we used the toolbox by Canu at al [9].

2.8 Gaussian Processes (GP)

Gaussian process regression is a nonparametric method based on modeling the observed responses of the different training data points (function values) as a multivariate normal random variable (see a detailed treatise in Rasmussen and Williams [45]). For these function values an a priori distribution is assumed that guarantees smoothness properties of the function. Specifically, the correlation between two function values is high if the corresponding input vectors are close (in Euclidean distance sense) and decays as they go farther from each other. The posterior distribution of a to-be-predicted function value can then be obtained using the assumed prior distribution by applying simple probability manipulations.

Let $V(X, X)$ denote the covariance matrix between the function values, where X is the matrix of input vectors of the training set (let the $(i, j)^{th}$ element of $V(X, X)$ be $V(x_i, x_j)$, where x_i denotes the i^{th} training input vector). A typical covariance matrix is the following:

$$V(x_i, x_j) = \sigma_f^2 e^{-\frac{\|x_i - x_j\|^2}{2\gamma^2}} \tag{16}$$

Thus, the vector of function values f (where f_i is the function value for training data point i) obeys the following multivariate Gaussian density:

$$f \sim \mathcal{N}_f(0, V(X, X)) \tag{17}$$

where $\mathcal{N}_f(\mu, \Sigma)$ denotes a multivariate normal density function in variable f with mean μ and covariance matrix Σ.

In addition, some independent zero-mean normally distributed noise having standard deviation σ_n is assumed to be added to the function values to produce the observed responses (target values), i.e.

$$y = f + \epsilon \tag{18}$$

where y is the vector of target outputs and ϵ is the vector of additive noise, whose components are assumed to be independent. The terms f_i represent the inherent function values, and these are the values we would like to predict, particularly for the test data.

Then, for a given input vector x_*, the prediction $\hat{f}_*$ is derived as

$$\hat{f}_* = E(f_*|X, y, x_*) = V(x_*, X)[V(X, X) + \sigma_n^2 I]^{-1} y \tag{19}$$

This equation is obtained by standard manipulations using Bayes rule applied on the given normal densities.

3 Preprocessing Methods

Preprocessing the time series can have a big impact on the subsequent forecasting performance. It is as if we are "making it easier" for the forecasting model by transforming the time series based on information we have on some of its features. For example, a large number of empirical studies advocate the deseasonalization of data possessing seasonalities for neural networks (Zhang and Qi [55], Miller and Williams [39], and Balkin and Ord [4]). Other preprocessing methods such as taking a log transformation and detrending have also been studied in the literature. We will use a deseasonalization step (if needed) and a log-step, however, these are not the object of this comparison study. What would be more novel is to compare some of the other preprocessing typically used for machine learning forecasting models, as there are very few comparison studies (if any) that consider these. The methods considered are (let z_t denote the time series):

1. No special preprocessing (LAGGED-VAL): the input variables to the machine learning model are the lagged time series values (say $z_{t-N+1}, \ldots, z_t$), and the value to be predicted (target output) is the next value (for one-step ahead forecasting).

2. Time series differencing (DIFF): We take the first backward difference and apply the forecasting model on this differenced series.

3. Taking moving averages (MOV-AVG): We compute moving averages with different-sized smoothing windows, for example:

$$u_i(t) = \frac{1}{J_i} \sum_{j=t-J_i+1}^{t} z_j, \qquad , i = 1, \ldots, I \tag{20}$$

where J_i is the size of the averaging window (the J_i's in our case take the values 1,2,4,8). The new input variables for the forecasting model would then be $u_i(t)$ and the target output is still z_{t+1}. The possible advantage of this preprocessing method is that moving averages smooth out the noise in the series, allowing the forecasting model to focus on the global properties of the time series. The existence of several moving averages with different smoothing levels is important so as to preserve different levels of time series detail in the set of inputs.

A note on the differencing issue (Point 2. above), is that while differencing for linear models is a well-understood operation having principled tests to determine its need, this is not the case for nonlinear models. The situation there is more ad hoc, as our experience and studies on other time series have demonstrated that differencing is not always a good strategy for nonstationary time series, and the converse is true for stationary time series. Here we attempt to shed some light on the utility of differencing.

4 Parameter Determination

For every considered method there are typically a number of parameters, some of them are key parameters and have to be determined with care. The key parameters are the ones that control the complexity of the model (or in short model selection parameters). These are the number of input variables given to the machine learning model (for example the number of lagged variables N for LAGGED-VAL and DIFF), the size of the network (for MLP and for BNN), the width of the radial bases β for RBF, the number of neighbors K for KNN, the width of the kernels h for GRNN, and for each of SVR and GP we have three parameters (we will discuss them later).

For linear models, the typical approach for model selection is to use an information criterion such as Akaike's criterion, the Bayesian information criterion, or others, which consider a criterion consisting of the estimation error added to it a term penalizing model complexity. For machine learning approaches such criteria are not well-developed yet. Even though some theoretical analyses have obtained some formulas relating expected prediction error with the estimation error (training error) and model complexity (Magdon-Ismail et al [32], Murata et al [42], and Moody [41]), these formulas are mostly bounds and have not been tested enough to gain acceptance in practical applications. The dominant approach in the machine learning literature has been to use the K-fold validation approach for model selection. Empirical comparisons indicate its superiority for performance accuracy estimation and for model selection (Kohavi [25]) over other procedures such as the hold out, the leave one out and the bootstrap methods. In the K-fold validation approach the training set is divided into K equal parts (or folds). We train our model using the data in the $K-1$ folds and validate on the remaining K^{th} fold. Then we rotate the validation fold and repeat with the same procedure again. We perform this training and validation K times, and compute the sum of the validation error obtained in the K experiments. This will be the validation error that will be used as a criterion for selecting the key parameters (we used $K = 10$ in our implementation, as this is generally the best choice according to Kohavi [25]).

For each method there are generally two parameters (or more) that have to be determined using the K-fold validation: the number of input variables (i.e. the number of lagged variables N), and the parameter determining the complexity (for example the number of hidden nodes for MLP, say NH). We consider a suitable range for each parameter, so for N we consider the possible values $[1, 2, \ldots, 5]$, and for NH we consider the candidate values $NH = [0, 1, 3, 5, 7, 9]$ (0 means a linear model). First, we fix NH as the median of the candidate NH values, and perform 10-fold validation to select N. Then we fix this selected N and perform a 10-fold validation to select NH. Note that for MLP we have the possibility of "having zero hidden nodes" ($NH = 0$), meaning simply a linear model. Balkin and Ord [4] have shown that the possibility

of switching to a linear model improved performance. We perform a similar tuning procedure as above for all models (except GP as we will see later). The following are the ranges of the parameters of the other models. For BNN the number of hidden nodes is selected from the candidate values $NH = [1, 3, 5, 7, 9]$. Note that BNN does not encompass a linear model like MLP, because as developed in [16] and [31] it applies only to a multilayer network. For a linear model it has to be rederived, leading to some form of a shrinkage-type linear model. For RBF the width of the radial bases β is selected from the possible values $[2.5, 5, 7.5, 10, 12.5, 15, 17.5, 20]$. For GRNN the bandwidth parameter h is selected from $[0.05, 0.1, 0.2, 0.3, 0.5, 0.6, 0.7]$. The number of neighbors K in the KNN model is tested from the possibilities $[2, 4, 6, 8, 12, 16, 20]$.

For the case of SVM the key parameters that control the complexity of the model are ϵ, C, and σ_K. In Cherkassky and Ma [13] and in Chalimourda et al [12] a comprehensive analysis of these parameters and their effects on the out of sample prediction performance and model complexity is presented. Chalimourda et al. [12] and Mattera [35] argue that C should be set as the maximum of the target output values y_{max}, and that the prediction performance should not be very sensitive to the value of C. Using theoretical as well as experimental analysis Kwok [27] suggests that ϵ should be set equal to the noise level in the data. Chalimourda et al. [12] argued that the prediction performance is sensitive to σ_K, and that this parameter has to be carefully selected. We followed the lead, and fixed C as y_{max}. We allowed ϵ and σ_K to be set using 10-fold validation. We consider a two-dimensional grid of ϵ and σ_K using the values $\sigma_y * [0.5, 0.75, 1, 1.25, 1.5] \times \sigma_0 * [0.05, 0.1, 0.25, 0.5, 1, 2, 4]$. The term σ_y is the estimated noise level in the time series, and it is estimated by subtracting a centered moving average of the time series from the time series itself, and obtaining the standard deviation of the resulting residual time series. This is of course only a rough estimate of the noise standard deviation, as an accurate level is not needed at this point. For the K-fold validation we need only a ballpark estimate to determine the search range. The term σ_0^2 gives a measure of the spread of the input variables, and is measured as the sum of the variances of the individual input variables.

For Gaussian processes there are three key parameters: σ_f, σ_n, and γ. It will be prohibitive to use a three-dimensional 10-fold validation approach on these parameters. We opted for the model selection algorithm proposed by Rasmussen and Williams [45]. It is an algorithm that maximizes the marginal likelihood function. The authors make a point that such criterion function does not favor complex models, and overfitting will therefore be unlikely (unless there is a very large number of hyperparameters).

Concerning the other less key parameters and model details, we selected them as follows. For MLP and BNN, we have used the logistic function activation functions for the hidden layer, and a linear output layer. Training is performed for 500 epochs for MLP and for 1000 epochs for BNN (some initial experimentation indicated BNN

needs more training iterations). Training also utilized a momentum term of value 0.2, and an adaptive learning rate with initial value 0.01, an increase step of 1.05 and a decrease step of 0.7. For RBF's the maximum number of radial bases is selected as 25% of the size of the training set (this number was obtained by experimenting on the training set, keeping a few points for validation). For GRNN, we used Gaussian kernels. For SVR, we used the more commonly used Gaussian kernel.

5 Experimental Setup

The benchmark data that we have used for the comparison are the M3 competition data [29]. The M3 is a dataset consisting of 3003 monthly, quarterly and annual time series. The competition was organized by the International Journal of Forecasting (Makridakis and Hibon [34]), and has attracted a lot of attention. A lot of follow-up studies have come out analyzing its results, up to the current year. We have considered all monthly data in the M3 benchmark that have more than 80 data points. The range of lengths of the time series considered has turned out to be between 81 and 126 and the number of considered time series has turned out to be 1045. From each time series, we held out the last 18 points as an out of sample set. All performance comparisons are based on these 18x1045 out of sample points. We considered only *one-step-ahead* forecasting.

The time series considered have a variety of features. Some possess seasonality, some exhibit a trend (exponential or linear), and some are trendless, just fluctuating around some level. Some preprocessing needs to be done to handle these features. Many papers consider the issues of preprocessing (Zhang et al [54], Zhang and Qi [55], Miller and Williams [39], and Balkin and Ord [4]), and so they are beyond the scope of this work. Based on some experimentation on the training set (withholding some points for validation), we settled on choosing the following preprocessing. We perform the following transformation, in the following order:

1. Log transformation
2. Deseasonalization
3. Scaling

For the log transformation, we simply take the *log* of the time series. Concerning deseasonalization, a seasonality test is performed first to determine whether the time series contains a seasonal component or not. The test is performed by taking the autocorrelation with lag 12 months, to test the hypothesis "no seasonality" with using Bartlett's formula for the confidence interval (Box and Jenkins [6]). If the test indicates the presence of seasonality, then we use the classical additive decomposition

approach [33]. In this approach a centered moving average is applied, and then a month-by-month average is computed on the smoothed series. This average will then be the seasonal average. We subtract that from the original series to create the deseasonalized series. The scaling step is essential to get the time series in a suitable range, especially for MLP and BNN where scaling is necessary. We have used linear scaling computed using the training set, to scale the time series to be between -1, and 1.

After these transformations are performed we extract the input variables (LAGGED-VAL, DIFF, or MOV-AVG) from the transformed time series. Then the forecasting model is applied. Once we perform the forecasting, we unwind all these transformations of course in reverse order.

We used as error measure the symmetric mean absolute percentage error, defined as

$$SMAPE = \frac{1}{M}\sum_{m=1}^{M}\frac{|\hat{y}_m - y_m|}{(|\hat{y}_m| + |y_m|)/2} \tag{21}$$

where y_m is the target output and $\hat{y}_m$ is the prediction. This is the main measure used in the M3 competition. Since it is a relative error measure it is possible to combine the errors for the different time series into one number.

To even out the fluctuations due to the random initial weights (for MLP and BNN) and the differences in the parameter estimation (for all methods) due to the specific partition of the K-fold validation procedure, we repeat running each model ten times. Each time we start with different random initial weights and we shuffle the training data randomly so that the K-fold partitions are different. Then we perform training and out of sample prediction. Each of the ten runs will produce a specific SMAPE. We then average these ten SMAPE's to obtain the overall SMAPE for the considered time series and the forecasting method. The overall SMAPE (SMAPE-TOT) for all 1045 time series is then computed. This is the main performance measure we have used to compare between the different methods.

We also obtained the following rank-based performance measure. Koning et al [26] proposed a significance test based on the rankings of the different models, and applied it to the M3 competition models. It is termed multiple comparisons with the best (MCB), and is based on the work of McDonald and Thompson [36]. It essentially tests whether some methods perform significantly worse than the best method. In this method we compute the rank of each method q on each time series p, say $R_q(p)$, with 1 being the best and 8 being the worst. Let $Q \equiv 8$ denote the number of compared models, and let P be the number of time series (in our case $P = 1045$). The average rank of each model q, or $\bar{R}_q$, is computed by averaging $R_q(p)$ over all time series. The

$\alpha\%$ confidence limits (we used $\alpha = 95\%$) will then be

$$\bar{R}_q \pm 0.5 q_{\alpha Q} \sqrt{\frac{Q(Q+1)}{12P}} \tag{22}$$

where $q_{\alpha Q}$ is the upper α percentile of the range of Q independent standard normal variables. Further details of this test can be found in [26].

We also performed another statistical significance test, proposed by Giacomini and White [20]. It is a recently developed predictive ability test that applies to very general conditions. For example it applies to arbitrary performance measures, and can handle the finite sample effects on the forecast performance estimates. It is also a conditional test, i.e. "can we predict if Model i will be better than Model j at a future date given the information we have so far". The basic idea is as follows. Consider two forecasting models where we measure the difference in the "loss function" (for example the SMAPE) for the two models. Let that difference be ΔL_{t+1}. That represents the SMAPE for Model 1 for the time series value to be forecasted of time $t+1$ minus that of Model 2. Considering ΔL_{t+1} as a stochastic process, under the null hypothesis of equal conditional predictive ability for the two models we have

$$E(\Delta L_{t+1} | \mathcal{F}_t) = 0 \tag{23}$$

for any given σ-field $\mathcal{F}_t$. This means that the out of sample loss difference is a martingale difference sequence, and $E(h_t \Delta L_{t+1}) = 0$ for any $\mathcal{F}_t$ measurable function h_t. The function h_t is called a test function and it should be chosen as any function of $\Delta L_{t'},\ t' \leq t$ that could possibly predict ΔL_{t+1} in some way. For our situation we used two test functions: $h_t = 1$ and $h_t = \Delta L_t$ (we considered the SMAPE as the loss function).

Another rank-based measure is the "fraction-best" (or in short FRAC-BEST). It is defined as the fraction of time series for which a specific model beats all other models. We used the SMAPE as a basis for computing this measure. The reason why this measure could be of interest is that a model that has a high FRAC-BEST, even if it has average overall SMAPE, is deemed worth testing for a new problem, as it has a shot at being the best.

6 Results

Tables 1, 2, and 3 show the overall performance of the compared forecasting models on all the 1045 time series, for respectively the LAGGED-VAL, DIFF, and MOV-AVG preprocessing methods. Figures 1, 2, and 3 show the average ranks with confidence bands for the eight methods for respectively the LAGGED-VAL, DIFF, and MOV-AVG preprocessing methods.

Table 1: The Overall Performance of the Compared Methods on All the 1045 Time Series for the LAGGED-VAL Preprocessing Method

Model	**SMAPE-TOT**	**Mean Rank**	**Rank Interval**	**FRAC-BEST**
MLP	0.0857	2.78	(2.62,2.94)	35.6
BNN	0.1027	3.42	(3.26,3.58)	16.9
RBF	0.1245	5.33	(5.17,5.49)	6.7
GRNN	0.1041	5.24	(5.08,5.40)	6.3
KNN	0.1035	5.10	(4.94,5.26)	7.3
CART	0.1205	6.77	(6.61,6.93)	3.2
SVR	0.0996	4.19	(4.03,4.35)	8.3
GP	0.0947	3.17	(3.01,3.33)	15.7

Table 2: The Overall Performance of the Compared Methods on All the 1045 Time Series for the DIFF Preprocessing Method

Model	**SMAPE-TOT**	**Mean Rank**	**Rank Interval**	**FRAC-BEST**
MLP	0.1788	5.02	(4.86,5.18)	6.5
BNN	0.1749	4.58	(4.42,4.74)	14.7
RBF	0.2129	5.24	(5.08,5.40)	8.8
GRNN	0.1577	3.92	(3.76,4.08)	14.7
KNN	0.1685	4.57	(4.41,4.73)	11.7
CART	0.1529	3.93	(3.77,4.09)	26.1
SVR	0.1709	4.59	(4.43,4.75)	6.7
GP	0.1654	4.15	(3.99,4.31)	10.7

One can observe from the obtained results the following:

- The ranking of the models is very similar for both the LAGGED-VAL and the MOV-AVG preprocessing methods, being overall MLP, GP, then BNN and SVR approximately similar, then KNN and GRNN almost tied, CART, then RBF. Both preprocessing methods yield very similar performance for all methods. The exception is for BNN, which ranks fourth for LAGGED-VAL but second for MOV-AVG (in terms of the SMAPE).

- The DIFF preprocessing, on the other hand, gives much worse performance than the other two preprocessing methods. The ranks are also different from

Table 3: The Overall Performance of the Compared Methods on All the 1045 Time Series for the MOV-AVG Preprocessing Method

Model	**SMAPE-TOT**	**Mean Rank**	**Rank Interval**	**FRAC-BEST**
MLP	0.0834	2.88	(2.72,3.04)	35.4
BNN	0.0858	2.94	(2.78,3.10)	15.9
RBF	0.1579	6.28	(6.12,6.44)	4.9
GRNN	0.1033	4.99	(4.83,5.15)	4.7
KNN	0.1034	4.95	(4.79,5.11)	8.4
CART	0.1172	6.31	(6.15,6.47)	3.8
SVR	0.1040	4.41	(4.25,4.57)	6.7
GP	0.0962	3.26	(3.09,3.42)	20.0

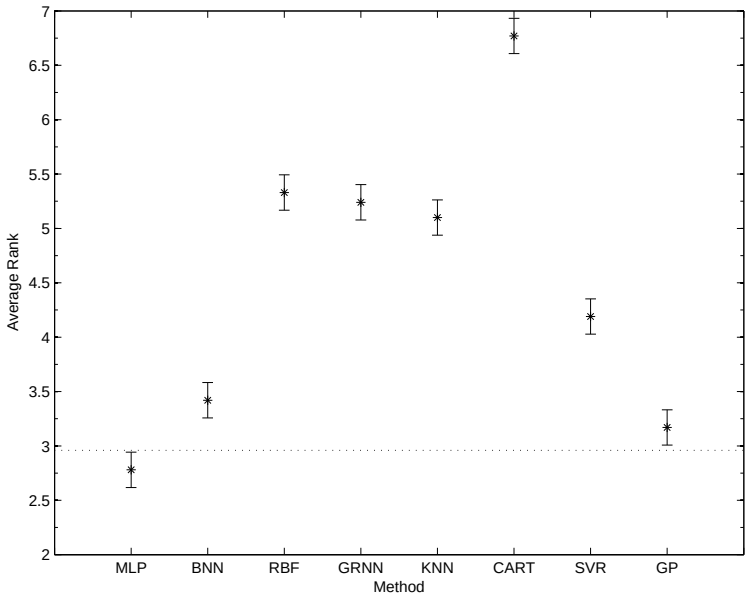

Figure 1: The average ranks with 95% confidence limits for the multiple comparison with the best test for the LAGGED-VAL preprocessing method. The dashed line indicates that any method with confidence interval above this line is significantly worse than the best (as described in Section 5 immediately before Eq. (22)).

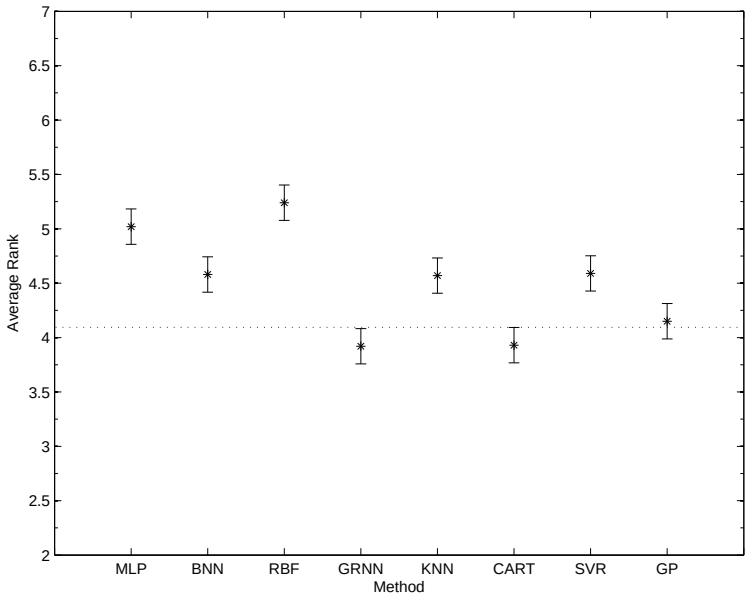

Figure 2: The average ranks with 95% confidence limits for the multiple comparison with the best test for the DIFF preprocessing method. The dashed line indicates that any method with confidence interval above this line is significantly worse than the best (as described in Section 5 immediately before Eq. (22)).

the ranks obtained for the other two preprocessing methods. They are: CART and GRNN almost equal, GP, KNN, SVR, BNN, MLP, then RBF. Since DIFF is an inferior preprocessing method, we would give more weight to the rankings obtained for the other preprocessing procedures, when judging the overall performance of the machine learning models.

Table 4 shows the results of the Giacomini-White predictive ability test at the 95% confidence level for the LAGGED-VAL preprocessing method. We would like to stress the fact that this is a conditional test. This means it asks the question whether averages of and lagged values of the difference in SMAPE between two models can predict future differences. To shed light into this issue, we have computed the correlation coefficient between ΔL_t and ΔL_{t+1} where ΔL_t is the difference in SMAPEs at time t of Models i and j. Table 5 shows these correlation coefficients for the LAGGED-VAL preprocessing method. One can see the very interesting phenomenon that the correlation coefficients are very high. This means that the outperformance of one model over another is positively serially correlated and therefore a persistent phenomenon. It would be interesting to explore the efficiency of a model based on

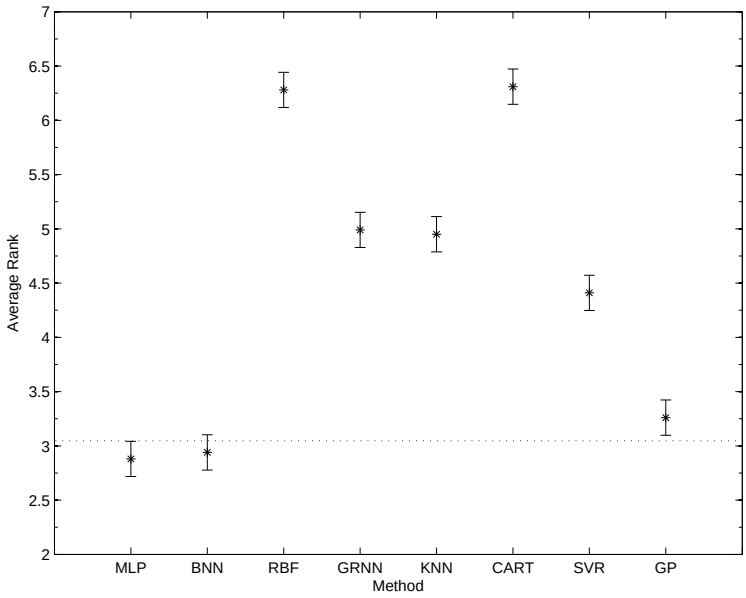

Figure 3: The average ranks with 95% confidence limits for the multiple comparison with the best test for the MOV-AVG preprocessing method. The dashed line indicates that any method with confidence interval above this line is significantly worse than the best (as described in Section 5 immediately before Eq. (22)).

Table 4: The results of the Giacomini-White test [20] for conditional predictive ability, at the 95% confidence level for the LAGGED-VAL preprocessing method. The plus sign means that it is possible to predict a statistically significant difference between the SMAPEs of the model in the corresponding column and the model in the corresponding row, conditional on past available information

Models	MLP	GP	SVR	BNN	KNN	GRNN	CART	RBF
MLP								
GP	+							
SVR	+	+						
BNN	+	+	+					
KNN	+	+	+	+				
GRNN	+	+	+	+	+			
CART	+	+	+	+	+	+		
RBF	+	+	+	+	+	+	+	

Table 5: The correlation coefficient between the difference in SMAPEs at time t and that at time $t+1$ for any two models. For example the entry in Row 2 and Column 3 represents $corr(SMAPE_t(MLP) - SMAPE_t(GP), SMAPE_{t+1}(MLP) - SMAPE_{t+1}(GP))$.

Models	MLP	GP	SVR	BNN	KNN	GRNN	CART	RBF
MLP	1	0.47	0.53	0.48	0.45	0.46	0.49	0.63
GP	0.47	1	0.38	0.52	0.4	0.38	0.42	0.57
SVR	0.53	0.38	1	0.54	0.43	0.43	0.42	0.57
BNN	0.48	0.52	0.54	1	0.52	0.52	0.52	0.59
KNN	0.45	0.4	0.43	0.52	1	0.24	0.34	0.56
GRNN	0.46	0.38	0.43	0.52	0.24	1	0.38	0.56
CART	0.49	0.42	0.42	0.52	0.34	0.38	1	0.53
RBF	0.63	0.57	0.57	0.59	0.56	0.56	0.53	1

Table 6: The Overall symmetric MAPE (SMAPE-TOT) by Data Category on All the 1045 Time Series for the LAGGED-VAL Preprocessing Method

Model	**Micro**	**Macro**	**Industry**	**Finance**	**Demographic**
MLP	0.182	0.035	0.091	0.086	0.024
BNN	0.229	0.040	0.103	0.101	0.032
RBF	0.232	0.072	0.123	0.144	0.045
GRNN	0.199	0.053	0.102	0.122	0.046
KNN	0.195	0.053	0.104	0.121	0.044
CART	0.231	0.062	0.118	0.144	0.049
SVR	0.187	0.051	0.098	0.123	0.041
GP	0.189	0.044	0.096	0.109	0.030

Table 7: The Overall symmetric MAPE (SMAPE-TOT) by Other Categories on All the 1045 Time Series for the LAGGED-VAL Preprocessing Method

Model	**Trend**	**No-Trend**	**Seasonal**	**Non-Seasonal**	**Zero-Hid**	**Non-Zero-Hid**
MLP	0.1366	0.0597	0.1081	0.0811	0.0848	0.0869
BNN	0.1634	0.0716	0.1978	0.0831	0.1078	0.0937
RBF	0.2109	0.0803	0.1511	0.1191	0.1304	0.114
GRNN	0.1729	0.0689	0.1169	0.1015	0.1087	0.0958
KNN	0.1724	0.0683	0.1154	0.1011	0.1084	0.0944
CART	0.2043	0.0776	0.1354	0.1175	0.126	0.1104
SVR	0.1655	0.0658	0.1113	0.0971	0.1029	0.0935
GP	0.1565	0.0631	0.1126	0.0911	0.0981	0.0888

regressing past lags of the SMAPE differences in predicting future SMAPE difference. This could be the basis of a dynamical model selection framework, that switches from one model to another based on the forecast of the delta loss function.

To seek more insight into the comparison results we have tested the relative performance of the different categories of time series. The M3 time series can be categorized into the categories: macro, micro, industry, finance, and demographic. Each category exhibits certain features that might favor one model over the other. Table 6 shows the SMAPE-TOT results for the LAGGED-VAL preprocessing procedure. The table clearly shows that the ranks are almost the same as the general ranks obtained over all the time series combined (with the BNN model somewhat an exception as it shows a larger deviation from the general rank). We have applied a similar experiment for the other two preprocessing methods. For lack of space the corresponding tables are not shown, but we have observed the exact same conclusion as the case of the LAGGED-VAL preprocessing. This indicates that the relative outperformance of certain models is a general phenomenon, not the outcome of some categories favoring some methods.

We have also tested how the models would fare according to some other categories as to the nature of the time series. For example we have categorized the time series as to whether they possess a trend or are non-trending. The existence of a trend stretches the range of the time series values and it might therefore be harder to train a model. Another categorization is seasonality versus nonseasonality. Even though we deseasonalize the time series, any deseasonalization step is usually far from ideal, and some residual seasonality remains. It would be interesting to examine which models can handle such imperfections better than others. Another caregorization is

Table 8: SMAPE for the Eight Models for the Time Series Example Considered in Figure 4

Model	**SMAPE**
MLP	0.0384
BNN	0.0411
RBF	0.0442
GRNN	0.0513
KNN	0.0511
CART	0.0566
SVR	0.0468
GP	0.0397

the following. The MLP model encompasses the case of a zero hidden node network, which essentially means a linear model. It might therefore be surmised that the superiority of MLP is inherited from its ability to reduce to a linear model. To test this hypothesis we have categorized the time series into a group where a zero hidden node network is selected (the "Zero-Hid" group) and a group where more hidden nodes are selected (the "Non-Zero-Hid" group), as determined by the parameter estimation step. Table 7 shows the SMAPE-TOT results for all the aforementioned categorizations for the LAGGED-VAL preprocessing procedures.

One can see that the category-restricted ranking of the models is quite similar to the overall ranking (with the exception of BNN in the seasonal/non-seasonal categorization). This again indicates that the relative performance is broad-based. No specific feature of the time series favors a particular model. One can observe how the existence of a trend negatively affects the performance for all models, and in addition how it accentuates the performance differences among the models. One can see that the rank of MLP for the Non-Zero-Hid group is still number one, suggesting that the superiority of MLP is genuine and not merely due to its ability to switch to a linear model. However, the relative differences in performance between the models in the Non-Zero-Hid group get smaller, indicating that the other models could possibly benefit if such a switching mechanism is incorporated into them.

To illustrate the presented ideas in a concrete manner, Figure 4 shows an example of a time series together with the forecasts of the two models that achieved the top two spots for this time series, namely MLP and GP. Table 8 shows the SMAPE of all the different models for this particular time series.

As it turns out, the computational demands of the different models vary significantly. Table 9 shows the computation time per time series for each model. All

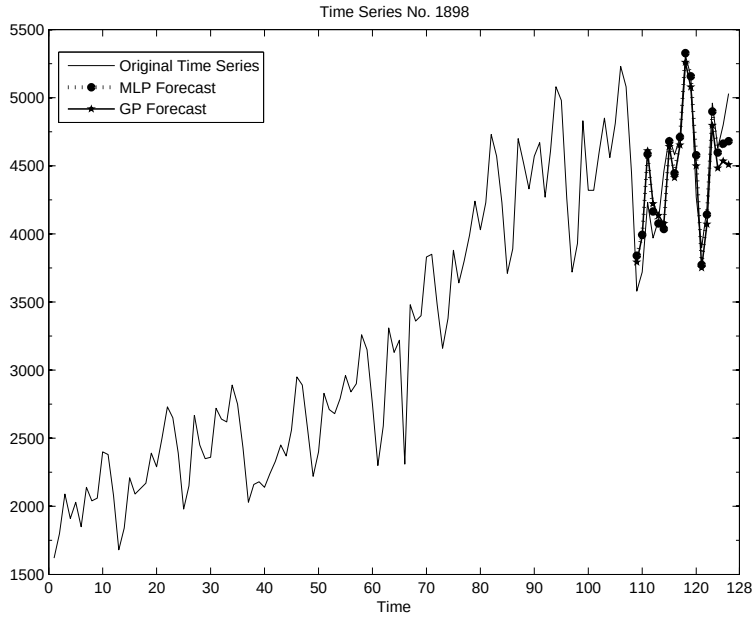

Figure 4: The forecasts of the top two models (MLP and GP) for one of the M3 time series. Shown is the training period, followed by the out of sample period (the last 18 points). The forecasts are shown only for the out of sample period.

measurements are based on running the methods on an Intel Centrino Duo 1.83 GHZ machine, and using Matlab's `tic` and `toc` functions. One can see that the most computationally demanding model is BNN. Following that comes MLP (as if this is the price one has to pay for its superior peformance). Following this comes RBF, GP, SVR, CART, then GRNN and KNN, both of which need very little computation time. At the level of time series lengths we are dealing with here, it seems that the computational issue is not significant enough to be a deciding factor for the modeler dealing with only one or few time series. However, it could be an important consideration if one decides to experiment with many models or do a very demanding parameter tuning step.

7 Comments on the Results

From the presented results it is clear that the different machine learning models differ significantly and there is an unambiguous ranking. This ranking seems to be broad-based in the sense that it is not dictated by the time series having certain features that favor some models. However, we must mention that this ranking applies to

Table 9: The Computation Time for Every Method Per Time Series

Model	**Time (Min/Series)**
MLP	1.6076
BNN	2.8251
RBF	0.6917
GRNN	0.0885
KNN	0.0068
CART	0.0803
SVR	0.3099
GP	0.6268

business-type time series. Some other time series, for example related to physics, chemistry, or other phenomena tend to exhibit very different features concerning the size of the series, the level of the noise, etc. It would be interesting to see how the tested models would compare on these. Also it would be interesting to find out if the obtained conclusions apply for quarterly or yearly business-type time series. We must also mention that the ranking obtained is a general ranking and still in any business forecasting problem several models have to be tested. Different business type time series have different stylized facts and hence the performance ranks can be different. However, the general ranking can still be a very valuable guide for the modeler in choosing which models to test.

The other observation is that preprocessing can have a large effect on model performance. Differencing the time series resulted in a different ranking, and generally much worse performance. As mentioned, we put more weight on the ranking of the LAGGED-VAL and MOV-AVG preprocessing results, because these preprocessing methods are the ones that the user will most probably use, due to their effectiveness. The reason why DIFF leads to worse results than the other two preprocessing methods could possibly be the following. It is likely due to the fact that for many of the considered time series the absolute level is a useful piece of information. For example there could possibly be some mean reverting behavior that dictates a more likely reversal direction when high or low levels are reached. When choosing between LAGGED-VAL and MOV-AVG, one has to consider the features of the considered time series. For example if there is a long term average that is known to have some significance, then moving average preprocessing could be a more advantageous method. If one would have used lagged preprocessing, then the number of lags could be excessive.

The MLP and the GP models are the best two models, with MLP being the best for the LAGGED-VAL and MOV-AVG preprocessing methods and GP being more robust

because it did also very well for the DIFF preprocessing. This is a very interesting result, because GP only very recently caught the attention of the machine learning community. It has been developed long time ago, but was never really widely studied or widely applied until very recently. The MLP model yielded very good results partly (but not wholly) because of its ability to reduce to a linear model. This is consistent with one of the conclusions of the M3 competition, which states that simple models tend to outperform more complex models. While support vector machines (SVM) is one of the top models in classification, in regression it does not seem to keep up the top spot. Generally similar in performance as SVR is BNN. Surprisingly it is in the second spot in terms of the fraction-best measure, and for MOV-AVG preprocessing it does very well. Apparently its performance is a bit erratic. For some time series it is in the top position while for others it has a high SMAPE. After this comes GRNN and KNN. Their performance is in general average, but they tend to be quite robust, rarely giving bad surprises. GRNN outperforms KNN a little, in general. It is interesting that the three methods KNN, GRNN and GP are all based on constructing a locally-based weighted average of the target outputs of the training set. Yet, GP outperforms the other two by so much. Perhaps, this is because GP is based on a solid probabilistic model. CART gave erratic performance, almost worst performance for the LAGGED-VAL and MOV-AVG preprocessing methods and almost best performance for the DIFF preprocessing method. Perhaps this is because it does not have enough flexibility as some of the other models with respect to linearly transforming the input space. The worst model overall is RBF. It consistently gave bad errors. Perhaps the fact that the centers and the kernel widths are not tunable lead it to be too inflexible.

8 Conclusion

We have presented in this work a large scale comparison of eight machine learning models on the M3 monthly time series of lengths ranging from 63 to 108 points (for the training period). The machine learning models are considered in their basic forms without the modifications and the additions proposed by so many researchers. The study shows that there are significant differences between the models, and also that preprocessing can have a significant impact on performance. The two best models turned out to be MLP and GP. This is an interesting result, as GP up until few years ago has not been a widely used or studied method. We believe that there is still room for improving GP in a way that may positively reflect on its performance.

A study that would also be of interest is to extend such a comparison to more recently developed machine learning models (for example Costillo and Hadi's functional networks [11] and White's Quicknet [53]). We believe comparison studies can

guide not only the practitioner in selecting appropriate models, but also the research community in focusing the research effort to more feasible or more promising directions.

Acknowledgement

We would like to acknowledge the help of Athanasius Youhanna of Cairo University, who has developed the seasonality test for this work. We also would like to acknowledge the useful discussions with Professor Ali Hadi of the American University of Cairo and Cornell University and with Professor Halbert White of UCSD. This work is part of the *Data Mining for Improving Tourism Revenue in Egypt* research project within the Egyptian Data Mining and Computer Modeling Center of Excellence.

References

[1] I. Alon, M. Qi, and R. J. Sadowski, *Forecasting aggregate retail sales: a comparison of artificial neural networks and traditional methods*, Journal of Retailing and Consumer Services, 8, 147–156, 2001.

[2] E. Alpaydin, *Introduction to Machine Learning*, MIT Press, 2004.

[3] U. Anders, and O. Korn, *Model selection in neural networks*, Neural Networks, 12, 309-323, 1999.

[4] S.D. Balkin and J.K. Ord, *Automatic neural network modeling for univariate time series*, International Journal of Forecasting, 16(4), 509-15, 2000.

[5] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxfoed University Press, 1995.

[6] G. Box and G. Jenkins, *Time Series Analysis, Forecasting and Control*, Holden-Day Inc., 1976.

[7] L. Breiman, *Classification and Regression Trees*, Chapman & Hall, Boca Raton, 1993.

[8] L. J. Callen, C. C.Y. Kwan, P. C. Y. Yip, and Y. Yuan, *Neural network forecasting of quarterly accounting earnings*, International Journal of Forecasting, 12, 475-482, 1996.

[9] S. Canu and Y. Grandvalet and V. Guigue and A. Rakotomamonjy, *SVM and Kernel Methods Matlab Toolbox*, Perception Systmes et Information, INSA de Rouen, Rouen, France, 2005.

[10] R. Caruana, A. Niculescu-Mizil, *An Empirical Comparison of Supervised Learning Algorithms*, Proceedings of the 23rd International Conference on Machine Learning (ICML2006), June 2006, 161-168.

[11] E. Castillo and A. S. Hadi, *Functional Networks*, in Encyclopedia of Statistical Sciences, (Samuel Kotz, N. Balakrishnan, Campbell B. Read and Brani Vidakovic, eds.), 4, 25732583, 2006.

[12] A. Chalimourda, B. Scholkopf, and A. J. Smola, *Experimentally optimal ν in support vector regression for different noise models and parameter settings*, Neural Networks, 17(1), 127-141, 2004.

[13] V. Cherkassky, and Y. Ma, *Practical Selection of SVM Parameters and Noise Estimation for SVM Regression*, Neural Networks, 17(1), 113-126, 2004.

[14] G. Cybenko, *Approximation by superposition of sigmoidal functions*, Mathematics of Control, Signals and Systems, 2, 303-314, 1989.

[15] J. G. De Gooijer and R. J. Hyndman, *25 years of IIF time series forecasting, a selective review*, Tinbergen Institute Discussion Paper, TI 2005-068/4, 2005.

[16] F. D. Foresee and M. T. Hagan, *Gauss-Newton approximation to Bayesian learning*, in Proceedings IEEE Int. Conference Neural Networks, 1930-1935, 1997.

[17] K. Funahashi, *On the approximate realization of continuous mappings by neural networks*, Neural Networks, 2, 183-192, 1989.

[18] R. Gencay and M. Qi, *Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping and bagging*, IEEE Transactions on Neural Networks, 12, 726-734, 2001.

[19] E. Ghysels, C. W. J. Granger, and P. L. Siklos, *Is seasonal adjustment a linear or nonlinear data filtering process?*, Journal of Business and Economics Statistics, 14, 374-386, 1996.

[20] R. Giacomini and H. White, *Tests of Conditional Predictive Ability*, Econometrica, 74, 1545-1578, 2006.

[21] W. Hardle, *Applied Nonparametric Regression, Econometric Society Monographs*, 19, Cambridge University Press, 1990.

[22] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer Series in Statistics. Springer-Verlag, 2001.

[23] T. Hill, M. OConnor, and W. Remus, *Neural network models for time series forecasts*, Management Science, 42, 1082-1092, 1996.

[24] K. Hornik, M. Stinchombe, and H. White, *Multi-layer Feedforward networks are universal approximators*, Neural Networks, 2, 359-366, 1989.

[25] R. Kohavi, *A study of cross-validation and bootstrap for accuracy estimation and model selection*, Proceedings International Joint Conference on Artificial Intelligence, IJCAI, 1995.

[26] A. J. Koning, P. H. Franses, M. Hibon, and H. O. Stekler, *The M3 competition: statistical tests of the results*, International Journal of Forecasting, 21, 397-409, 2005.

[27] J.T. Kwok, *Linear dependency between and the input noise in support vector regression*, Proceedings of ICANN 2001, LNCS 2130, 405-410, 2001.

[28] M. Leshno, V. Lin, A. Pinkus, and S. Schocken, *Multilayer feedforward networks with a nonpolynomial activation function can approximate any function*, Neural Networks, 6, 861-867, 1993.

[29] M3 Competition, http://www.forecasters.org/data/m3comp/m3comp.htm.

[30] D. J. C. MacKay, *Bayesian interpolation*, Neural Computation, 4, 415-447, 1992.

[31] D. J. C. MacKay, *A Practical Bayesian Framework for Backpropagation Networks*, Neural Computation, 4, 448-472, 1992.

[32] M. Magdon-Ismail, A. Nicholson and Y. Abu-Mostafa, *Financial markets, very noisy information processing*, Proceedings of the IEEE, 86(11), 2184-2195, 1998.

[33] S. Makridakis, S. C. Wheelwright, and R. J. Hyndman, *Forecasting: Methods & Applications*, 3rd Eddition, Ch. 3, Wiley, 1998.

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

[35] D. Mattera and S. Haykin, *Support vector machines for dynamic reconstruction of a chaotic system*, in *Advances in Kernel Methods: Support Vector Learning*, B. Scholkopf, C. Burges, and A. Smola (Eds.), 211241, MIT Press, 1999.

[36] B. J. McDonald and W. A. Thompson, *Rank sum multiple comparisons in one and two way classifications*, Biometrika, 54, 487-497, 1967.

[37] M. C. Medeiros, and A. Veiga,*A hybrid linear-neural model for time series forecasting*, IEEE Transactions on Neural Networks 11, 1402-1412, 2000.

[38] M. C. Medeiros, T. Terasvirta, and G. Rech, *Building neural network models for time series: A statistical approach*, Journal of Forecasting, 25, 49-75, 2006.

[39] D. M. Miller and D. Williams, *Damping seasonal factors: Shrinkage estimators for the X-12-ARIMA program*, International Journal of Forecasting, 20, 529-549, 2004.

[40] J. E. Moody and C. Darken, *Fast learning in networks of locally-tuned processing units*, Neural Computation 1, 281-294, 1989.

[41] J. Moody, *The effective number of parameters: an analysis of generalization and regularization in nonlinear learning systems*, Advances in Neural Information Processing Systems 4, 847-854, 1992.

[42] N. Murata, S. Yoshizawa, and S. Amari, *Learning curves, model selection, and complexity of neural networks*, Advances in Neural Information Processing Systems 5, 607-614, 1993.

[43] E. A. Nadaraya, *On estimating regression*, Theory of Probability and its Applications 10, 186-190, 1964.

[44] M. J. D. Powell, *Radial basis functions for multivariable interpolation: A review* in *Algorithms for Approximation*, J. C. Mason and M. G. Cox, Eds. Oxford: Clarendon, 143-168, 1987.

[45] C. E. Rasmussen, and C. K. L. Williams, *Gaussian Processes for Machine Learning*, MIT Press, 2006.

[46] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, 2001.

[47] R. Sharda, R.B. Patil, *Connectionist approach to time series prediction: An empirical test*, Journal of Intelligent Manufacturing, 3, 317323, 1992.

[48] A. J. Smola, and B. Scholkopf, *A Tutorial on Support Vector Regression*, NeuroCOLT Technical Report, TR-98-030, 2003.

[49] N.R. Swanson, and H.White, *A model-selection approach to assessing the information in the term structure using linear models and artificial neural networks*, Journal of Business and Economic Statistics, 13, 265-75, 1995.

[50] T. Terasvirta, D. van Dijk, and M. C. Medeiros, *Linear models, smooth transition autoregressions, and neural networks for forecasting macroeconomic time series: A reexamination*, International Journal of Forecasting, 21, 755-774, 2005.

[51] D. van Dijk, T. Terasvirta, and P. H. Franses, *Smooth transition autoregressive models - a survey of recent developments*, Econometric Reviews, 21, 1-47, 2002.

[52] G. S. Watson, *Smooth regression analysis*, Sankhy, Series A 26, 359-372, 1964.

[53] H. White, *Approximate nonlinear forecasting methods*, in G. Elliott, C.W.J. Granger, and A. Timmermann, eds., Handbook of Economics Forecasting. New York: Elsevier, 460-512, 2006.

[54] W. Zhang, Q. Cao, and M. J. Schniederjans, *Neural network earning per share forecasting models: A comparative analysis of alternative methods*, Decision Sciences, 35 (2), 205-237, 2004.

[55] G. P. Zhang and M. Qi, *Neural network forecasting for seasonal and trend time series*, European Journal of Operational Research, 160, 501-514, 2005.